

THE GREAT WAR: Western Front

Map Editor Documentation

Version 1.1

Petroglyph Games, Inc.

Frontier Foundry

1. Map Editor Manual and Tutorials	4
1.1 Map Editor	5
1.1.1 00_Menu Bar	11
1.1.2 01_Height Tab	15
1.1.3 02_Texture Tab	17
1.1.4 03_Props Tab	18
1.1.5 04_Color Tab	19
1.1.6 05_Splines Tab	20
1.1.7 06_Decals Tab	21
1.1.8 07_Grass Tab	22
1.1.9 08_Water Tab	23
1.1.10 09_Enviro Tab	24
1.1.11 10_Lighting Tab	26
1.1.12 11_Post FX Tab	27
1.1.13 12_Objects Tab	28
1.1.14 13_Pass Tab	30
1.1.15 14_Regions Tab	35
1.1.16 15_Audio Tab	39
1.1.17 16_Cinematic Tab	40
1.2 TUTORIAL: Creating a "New Skirmish Map" Mod	46
1.2.1 01: Setting up and running your Skirmish Map Mod	47
1.2.2 02: Using the Map Editor to customize your new skirmish map	54
1.2.3 03: Advance Map customization for your new Skirmish Map	80
1.3 TUTORIAL: Creating a Cinematic	96
1.4 INFORMATION: LUA Debug Tools	99
1.5 LUA Modules	101
1.5.1 Misc	102
1.5.2 PG	108
1.5.3 AIRPC	109
1.5.4 ClientCamera	110
1.5.5 ClientCinematic	113
1.5.6 ClientGUIComponent	114
1.5.7 ClientGUIUtils	115
1.5.8 ClientObjectManipulation	129
1.5.9 ClientObjectivesDisplay	131
1.5.10 ClientPGGUI	135
1.5.11 ClientPlayers	136
1.5.12 GUIAdvancedTicker	137
1.5.13 GUIAnimation	143
1.5.14 GUIButtonBase	144
1.5.15 GUICarousel	146
1.5.16 GUIClockProgress	150
1.5.17 GUIComboBox	152
1.5.18 GUIDecoratedIconButton	155
1.5.19 GUIEditBox	158
1.5.20 GUIGeneric	161
1.5.21 GUIIconButton	177
1.5.22 GUILine	179
1.5.23 GUIListBox	181
1.5.24 GUIMPEGMovieQuad	193
1.5.25 GUIMenu	196
1.5.26 GUIModel	197
1.5.27 GUIMovieQuad	199
1.5.28 GUIProgressBar	202
1.5.29 GUIScene	204
1.5.30 GUISceneReference	205
1.5.31 GUISliderBar	207
1.5.32 GUITextBlock	210
1.5.33 GUITextButton	212
1.5.34 GUITexturedQuad	213
1.5.35 GUITicker	216
1.5.36 GUITree	221
1.5.37 FrameUpdate	225
1.5.38 ServerCinematic	226
1.5.39 ServerEffectSystem	229
1.5.40 ServerGUIUtils	232
1.5.41 ServerInstanceManagement	234
1.5.42 ServerObjectManipulation	237
1.5.43 ServerObjectSelection	241
1.5.44 ServerObjectives	243
1.5.45 ServerPlayers	246
1.5.46 ServerTGWUtils	248
1.5.47 ServerUnitOrders	256
1.5.48 ServerVictoryConditions	258
1.5.49 Localization	259
1.5.50 ObjectManipulation	260
1.5.51 Players	267
1.5.52 TGWUtils	272
1.5.53 TerrainRegions	276

1.5.54 UnitOrders	278
1.5.55 PGServer	280
1.5.56 Timers	282
1.5.57 GameSignals	285
1.5.58 StateSignals	299
1.5.59 EXAMPLES	300
1.5.59.1 emptyscript.lua	301
1.5.59.2 rpcplayerselector.lua	302

Map Editor Manual and Tutorials

INTRODUCTION

This document provides players with information needed to use the Map Editor to build new map content. This document is separated into four sections:

- **Map Editor Specifics** - The different tabs contained in the tool and how they work
- **Skirmish Map Tutorial** - How to build your own skirmish map using our provided example
- **Cinematic Tutorial** - How to create your own video that can be played with the map you just created (if you wish)
- **LUA Debug Tools** - Some debug commands you can use to help debug the map and the scripting instructions therein to create an interesting scenario for players
- **LUA Modules** - An index of LUA modules available for The Great War: Western Front.

You can check on the official modding site <https://modding.playthegreatwar.com> for the latest version of this document.

Map Editor

- The Map Editor consists of several smaller editors, documented below.
- Through the Map Editor the user will be able to generate *Map Data* that can be used in the development of Mods.
 - For reference, the chart below notes the file extension and a brief description of each file type.

File Type	Brief Description
.ted	Saves the visual aspects of the Map such as: Height Map, Terrain Textures, Props, Vertex Color, Splines, Decals, Grass, Water, Environment settings, Lighting, and Post FX data.
.sob	Saves the logical aspects of the map such (such as location) for: Objects. Passability, and Region data are all saved in its corresponding .cpd/.gpd.
.cpd/.gpd	Saves the logical aspects of the map such as Passability and Region data used in conjunction with its respective .sob
.tga	Generates a map preview that makes up the Players Minimap.
.tec	Saves All Cinematic information such as Camera/Target/Frustum/Roll/Shake/Lighting. Primarily on Begin Battle of Historical Missions.

Getting Started

How to Launch the Map Editor


- The Map Editor can be launched one of 2 ways:
 - **Through the Mod Manager Menu**
 - Library Play Main Menu Mod Manager Menu Map Editor Button
 - This method forces you to have the Game and the Map Editor running at the same time.
 - Closing closing the game will close both the Game and Map Editor regardless if you have saved or not.
 - **Launch Options via Steam Library Game Properties**
 - Library The Great War: Western Front Properties Launch Options
 - Use the following command:
 - ClientLauncherG64.exe RUN_EDITOR
 - Followed by launching the game normally.
 - Library Play
 - This method will allow you to just have the Map Editor Open, but prevents you from being able to launch the game.

The chart below, provides an example of the files needed for each type of Map Mod.

SKIRMISH MAP		CAMPAIGN MAP		HISTORICAL MAP
PUB_20x25_MAP_01_00.ted		PUB_20x25_MAP_01_00.ted		PUB_20x25_MAP_01_00.ted
PUB_20x25_MAP_01_00_SHARED.sob (USE AS MAIN .cpd/.gpd)		PUB_20x25_MAP_01_00_SHARED.sob (USE AS MAIN .cpd/.gpd)		PUB_20x25_MAP_01_00.sob (USE AS MAIN .cpd/.gpd)
PUB_20x25_MAP_01_00_ALLIED_CTRL.sob (INCLUDE .cpd/.gpd)		PUB_20x25_MAP_01_00_ALLIED_CTRL.sob (INCLUDE .cpd/.gpd)		PUB_20x25_MAP_01_00_SKIRMISH.sob (INCLUDE .cpd/.gpd)
PUB_20x25_MAP_01_00.tga		PUB_20x25_MAP_01_00_GERMAN_CTRL.sob		PUB_20x25_MAP_01_00.tga
		PUB_20x25_MAP_01_00.tga		PUB_20x25_MAP_01_CIN_00.tec

NOTE ⚠

- Maps can be comprised of several .sob files but there can only be one that dictates the Passability (.cpd/.gpd).
- In the list above, that .sob file is marked as "USE AS MAIN".

 **NOTE**

- In order for a Mod to run, the data above must be placed inside its respective folder at the following location : **C:\Users\USER_NAME\Documents\Petroglyph\TheGreatWar\Mods**
 - **HOWEVER**, please be aware that the Map Editor saves **ALL** its files by default at the following location: **C:\Users\USER_NAME\Documents\Petroglyph\TheGreatWar\Custom_Maps**
 - Its not possible to save/load **.sob** at different locations without that **.sobs** corresponding **.cpd/.gpd** not being saved properly.
 - **THEREFORE**, when developing and testing new map content the user will need to move files back and forth between the two locations when working on the Map Editor and wanting to test the Mod in game.

- [Getting Started](#)
 - [How to Launch the Map Editor](#)
 - [Useful Hotkeys](#)
 - [Map Editor Anatomy](#)
- [Editors Tabs](#)
- [Map Anatomy and Sizes](#)

Useful Hotkeys

- Spacebar – Toggles gameplay camera view states.
- [and] keys – Alters the currently active brush to be smaller or larger.
- Holding Shift and selecting an area will instantly paint that area with the selected brush.
- Holding Control will activate the rectangle tool and instantly paint that area with the selected brush in a rectangular shape.
- Alt + Left Click – Sample the height, color, or texture that is currently under the cursor.
- Holding Ctrl + the Right Mouse Button - Allows you to quickly rotate your camera.
- Holding Alt + Right Mouse Button - Allows you to quickly drag your camera view around the map.
- Ctrl + D – Places the camera in a top-down view of the map.

Map Editor Anatomy

 **Note**

General Map Editor layout, with brief explanation of each section. Provided Map Editor Documentation will references names given to editor below.



Menu Bar (1)

- Located near the top of the Map Editor, File, Edit, View, etc. provide a series of settings, tools, and options. (See [00_Menu Bar](#) for more information)

Main Tool Bar (2)

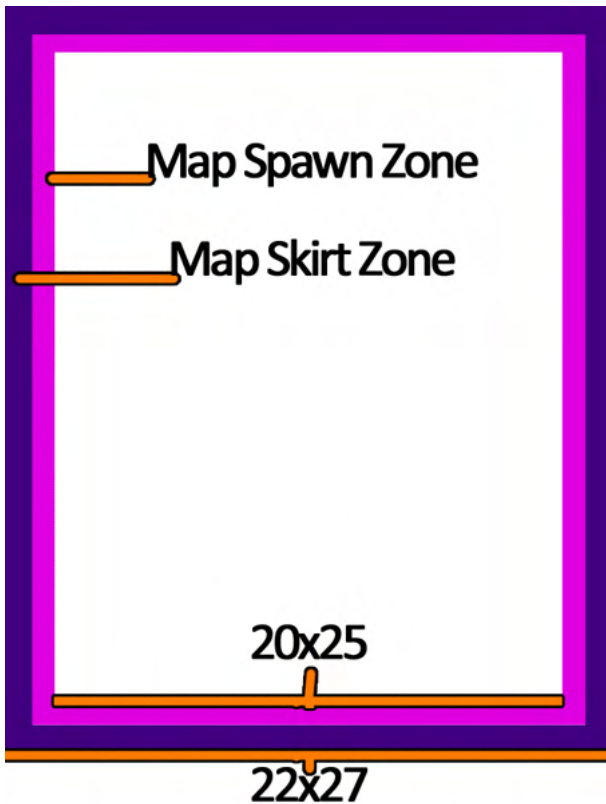
- Located directly beneath it, the Main Tool Bar is comprised of the following Tool Bars, each with their own specific use (written below as they are seen in Editor from **Left to Right**).
 - Standard Tool Bar
 - New/Open/Save Icons
 - Gadget Tool Bar
 - Comprised mainly of context sensitive tools (each becoming active or grayed out when inactive) depending on the Editor Tab currently Selected.
 - Normal Brush
 - Sample Mode
 - Fill Mode
 - Line Brush Mode
 - Ruler Mode
 - Can only be used in the Props/Objects Tabs
 - Locks the camera from moving, can only zoom in and out.
 - Distance is displayed in units and cells.
 - Results displayed in **Status Bar**
 - Snap To Grid Mode
 - **Not Functional**
 - View Tool Bar
 - Show/Hide Map Objects (Ctrl+H)
 - Show/Hide Water
 - Show/Hide Fog
 - Show/Hide Selection Boxes
 - Show/Hide Spawner Connection Lines
 - Show/Hide Brush
 - Can only be used when in Pass Tab.
 - Performs the same function as the Passability Category drop down.
 - Show/Hide Targeting
 - Can only be used when in Pass Tab.
 - Performs the same function as the Passability Category drop down.
 - Show/Hide Visibility
 - Can only be used when in Pass Tab.
 - Performs the same function as the Passability Category drop down.
 - Show/Hide Movement
 - Can only be used when in Pass Tab.

- Performs the same function as the Passability Category drop down.
- Show/Hide Passability (Ctrl+P)
- Show/Hide Regions (Ctrl+R)
 - Must open Regions Tab once in order to initialize.
- Show/Hide Nav Mesh
- Enable/Disable Vertex Lock Mask
- Show/Hide Vertex Lock Mask
- Enable/Disable Height Guide Polygon
- Editor Tool Bar
 - The Map Editor is comprised of several different editors. Each editor is accessible via the "Tabs" below the Menu Bar. Each respective Terrain Editor Tab has its corresponding Editor Panes, with properties respective to it.
- Editor Panes (3)
 - These are context sensitive panes. Each editor has a different number of panes available, each pane having settings that can be adjusted when working in that editor.
- 3D Space (4)
 - 3D Game space
- Status Bar (5)
 - Provides information such as Mouse location, Object Type selected, Objects selected, etc.

Editors Tabs

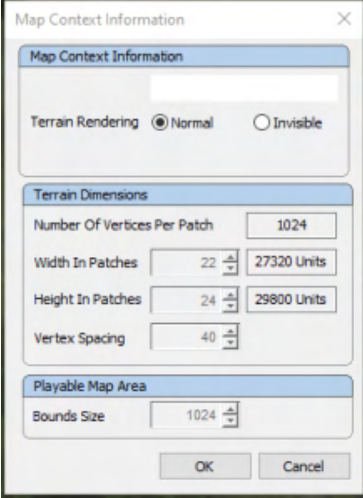
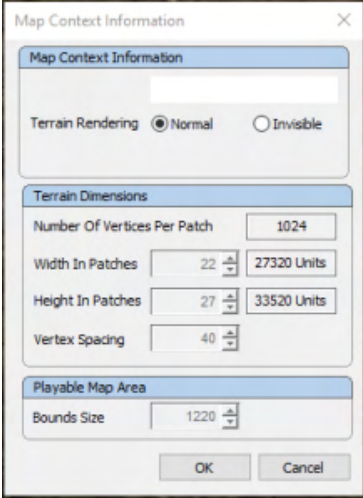
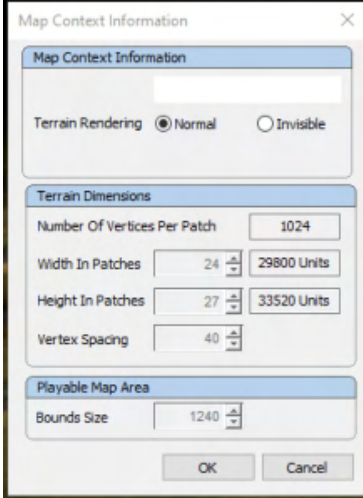
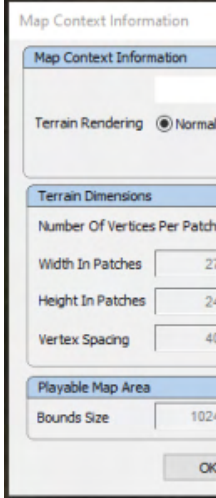
- [00_Menu Bar](#)
- [01_Height Tab](#)
- [02_Texture Tab](#)
- [03_Props Tab](#)
- [04_Color Tab](#)
- [05_Splines Tab](#)
- [06_Decals Tab](#)
- [07_Grass Tab](#)
- [08_Water Tab](#)
- [09_Enviro Tab](#)
- [10_Lighting Tab](#)
- [11_Post FX Tab](#)
- [12_Objects Tab](#)
- [13_Pass Tab](#)
- [14_Regions Tab](#)
- [15_Audio Tab](#)
- [16_Cinematic Tab](#)

Map Anatomy and Sizes



- Map size refers to the Width and Height needed to create the map in the Map Editor .
 - **HOWEVER**, the actual play space of the map is actually 2 less on both axis, as each map allows for both a Skirt and a Map Spawn Zone, which are technically unplayable areas.
 - The Map Skirt is a boundary of impassable that is applied automatically by the Map Editor and cannot be removed. Prevents the player camera from looking at the skybox.
 - The Map Spawn Zone, which allows an object to spawn in via the reserves system or script. The MapSpawnZone Passability is an inner section of playable terrain which allows reinforcements to be spawned in properly while in game, yet prevents players from entering this map area or using it to traverse the map in actual gameplay, furthermore the MapSpawnZone passability protects the spawning Company by blocking targeting.
 - When creating your own maps, MapSpawnZones are roughly a distance of 1043 units (26 cells) starting where the Map Skirt ends.
 - Distances can be measured in the editor via the Ruler Mode (See Main Tool Bar in [Map Editor](#))

22x24	22x27	24x27	27x24
-------	-------	-------	-------

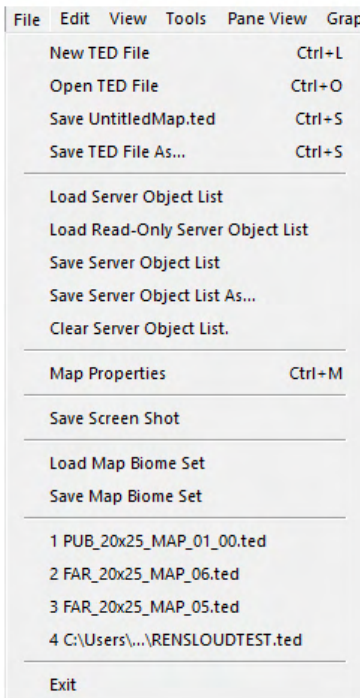
<ul style="list-style-type: none"> • HILLS OF GREEN • RURAL WOODLAND • UNDERMINED • PASTORAL FARMLAND 	<ul style="list-style-type: none"> • TOWN AND COUNTRY • SECLUDED COPSE • FORT OVERLOOK • WOOD AND RUIN 	<ul style="list-style-type: none"> • FORTIFIED HILLS • THE DEAD WOODS • RIDGELINE • - 	<ul style="list-style-type: none"> • CROSSROADS TOW • - • - • - 
<p>BIOMES</p> <ul style="list-style-type: none"> • Biome Set: A Biome is a file containing a map's settings which can include texture settings, lighting settings, environment settings, <ul style="list-style-type: none"> • The Following Biomes have been provided. 			
<ul style="list-style-type: none"> • TGW_Hills_of_Green_Biome.mms • TGW_Rural_Woodland_Biome.mms • TGW_Undermined_Biome.mms • TGW_Pastoral_Farmland_Biome.mms 	<ul style="list-style-type: none"> • TGW_Town_and_Country_Biome.mms • TGW_Secluded_Copse_Biome.mms • TGW_Fort_Overlook_Biome.mms • TGW_Wood_Ruin_Biome.mms 	<ul style="list-style-type: none"> • TGW_Fortified_Hills_Biome.mms • TGW_Dead_Woods_Biome.mms • TGW_RidgeLine_Biome.mms 	<ul style="list-style-type: none"> • TGW_Crossroads_To

00_Menu Bar

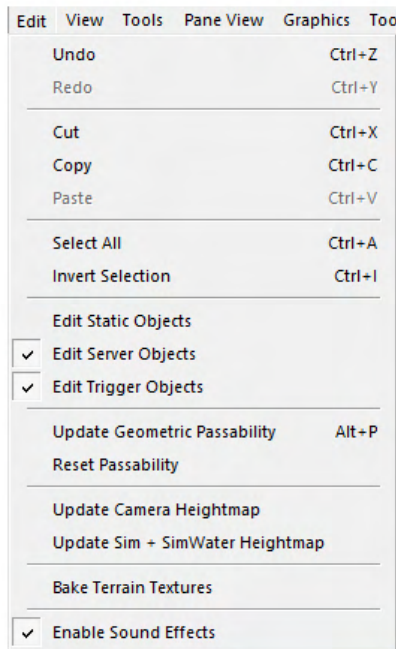
Getting Started

The Menu Bar contains several useful tools and settings. Those not exactly obvious or specific to the Map Editor will be commented here.

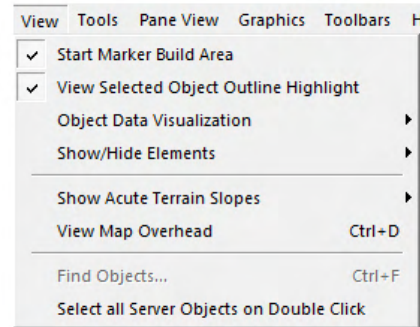
A



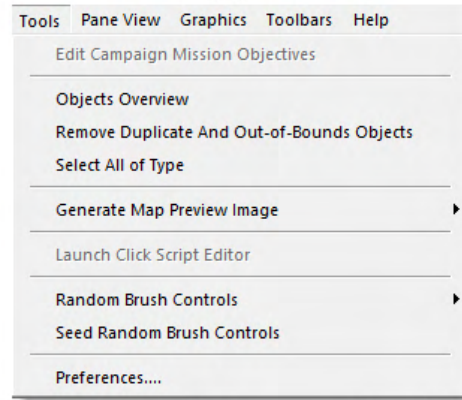
B



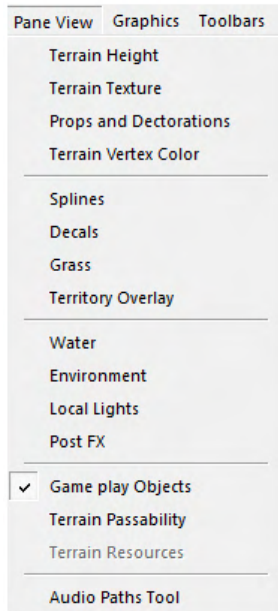
C



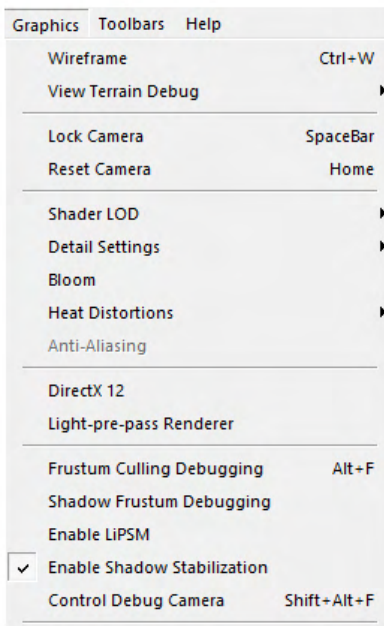
D



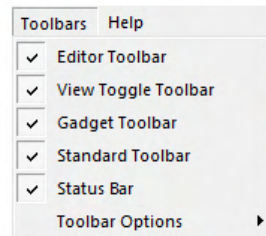
E



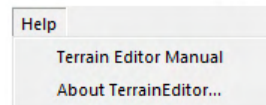
F



G



H



- File (A)
 - Load Read-Only Server Object List
 - Since TGW data uses multiple .sobs, its sometimes necessary to view them without editing them. This "loads" the contents of a sob, but prevents any editing done to it.
 - Can load Multiple sobs.
 - Must be removed when doing a **File Clear Server Object List**
 - Map Properties
 - Brings up the Map Context Information dialogue.

- Load Map Biome Set
 - A Biome is a file containing a map's settings which can include texture settings, lighting settings, environment settings, and other user-selected map options.
 - For more information on Map Sizes or the available Biomes please see the **Map Anatomy and Sizes** in the [Map Editor](#) section.
- Edit (B)
 - Reset Passability
 - **NOT USED IN TGW**
- View (C)
 - Start Marker Build Area
 - **NOT USED IN TGW**
 - Select All Server Objects On Double Click
 - **NOT USED IN TGW**
- Tools (D)
 - Objects Overview
 - Objects from both Prop and Objects Tab are shown in a list.
 - Allows for a quick delete of groups of objects.
 - Allows for "jump to" location of objects.
 - Remove Duplicate and Out-of-Bounds Objects
 - **NOT USED IN TGW**
 - Generate Map Preview Image
 - Random Brush Controls
 - **NOT USED IN TGW**
 - Seed Random Brush Controls
 - **NOT USED IN TGW**
 - Preferences.....
- Pane View (E)
 - Terrain Height
 - Opens Height Tab
 - Terrain Texture
 - Opens Texture Tab
 - Props and Decorations
 - Opens Props Tab
 - Terrain Vertex Color
 - Opens Color Tab
 - Splines
 - Opens Splines Tab
 - Decals
 - Opens Decals Tab
 - Grass
 - Opens Grass Tab
 - Territory Overlay
 - **NOT USED IN TGW**
 - Water
 - Opens Water Tab
 - Environment
 - Opens Enviro Tab
 - Local Lights
 - Opens Lighting Tab
 - Post FX
 - Opens Post FX Tab
 - Game play Objects
 - Opens Objects Tab
 - Terrain Passability
 - Opens Pass. Tab
 - Audio Path Tool
 - Opens Audio Tab
- Graphics (F)
 - Wireframe
 - View Terrain Debug
 - Lock Camera
 - Locks the camera to a "view" that's roughly closer to what it would be in game (Spacebar toggles between Lock Camera and Reset Camera).
 - Reset Camera
 - (Spacebar toggles between Lock Camera and Reset Camera)
 - Shader LOD
 - Used to lower the settings seen in the terrain editor. May improve performance on low spec machines.
 - Detail Settings
 - Used to lower the settings seen in the terrain editor. May improve performance on low spec machines.
 - Bloom
 - Used to lower the settings seen in the terrain editor. May improve performance on low spec machines.
 - Heat Distortions
 - Used to lower the settings seen in the terrain editor. May improve performance on low spec machines.
 - DirectX12
 - **NOT USED IN TGW**
 - Light-pre-pass Renderer
 - **NOT USED IN TGW**
 - Frustum Culling Debugging
 - **NOT USED IN TGW**

- Shadow Frustum Debugging
 - **NOT USED IN TGW**
- Enable LiPSM
 - **NOT USED IN TGW**
- Enable Shadow Stabilization
 - **NOT USED IN TGW**
- Control Debug Camera
 - **NOT USED IN TGW**
- Toolbars (G)
 - Allows user to hide elements within the Main Tool Bar, as well change the size of the Entire Tool Bar.
- Help (H)

01_Height Tab

Getting Started

The “Height” tab will allow you to change the topography and structure of your terrain in both the passable play area and in the border area that creates the bounds of your map.

To alter the terrain, select one of the brushes under the Brush Controls section and Left-Click the terrain. To reverse the direction of the brush, hold down the Control key and Ctrl+Left Click the terrain.

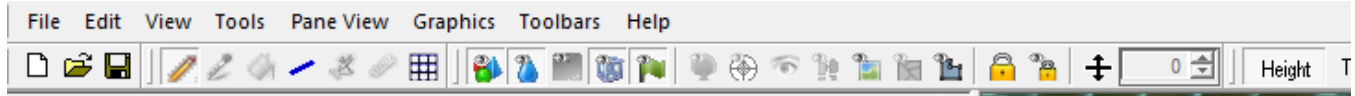
Tip: When using tools such as the surface noise brush which can rapidly change the terrain, be sure to repair or smooth out any visual tears in the terrain before saving and uploading your map.

Tip: If you wish to save a set of Copy Paste brushes for use in another session or map file, simply click the blue diskette icon to save the file. Similarly, if you wish to import a set of Copy Paste brushes, place the file in the same folder as your other Copy Paste Brush Files and then select that file from the drop-down list when in the terrain editor.

Tip: Since the Height map is nothing more than elevation information represented through colors (black - Low Elevation and white- High Elevation), its possible to save out the height map from the editor and bring it into a program like photoshop. This allows for quick blocking out of interesting terrain shapes for your maps.

Editor Pane Tools

- Under **Brush Controls**, select the brush type you would like to use.
 - **Additive Brushes (Conical/Spherical)** – These brushes will add or subtract height from the terrain in their respective shape.
 - **Leveling Brush** – This brush instantly sets the terrain to the height entered in the height field.
 - **Ramp Brush** – Ramps can be made from one area of terrain to another by clicking the start point of the ramp, clicking the end point of the ramp, then clicking one last time to confirm ramp creation.
 - **Clamp Height Brush** – Brush that pulls terrain toward a preferred height with adjustable intensity and variation.
 - **Surface Noise Brush** – Height adjustment brush which is able to paint variable patterns into the terrain.
 - **Copy Paste Brush** – Brush used to select an area of altered terrain and copy that terrain for placement onto other areas of the map. Copy Paste controls will be activated once a section of terrain has been selected for later use with the Copy Paste function.
 - **Terrain Mask Brush** – Terrain Mask Brush – Brush used to paint a “mask” onto the terrain. Areas covered by a Terrain Mask are protected from alteration by tools which have that Terrain Mask selected.
 - Terrain Masks created here and can also be used in the following Tabs: Texture, Props, Color, Pass, and Regions. Furthermore, they can be toggled on and off via the Main Tool Bar View Tool Bar



- There are various settings available under each brush control, some of which are only applicable for certain brush types:
 - **Radius** – Brush size.
 - **Blend Radius** – Size of area within the brush closest to the preferred height; areas between the blend radius and the brush radius will gradually change to the specified height.
 - **Intensity** – Rate/speed at which the brush alters the terrain. (not useful until the user switches tools to actually deform the terrain.)
 - **Smoothing** – Strength of the smoothing effect applied when the brush alters the terrain height. (not useful until the user switches tools to actually deform the terrain.)
 - **Clamp Height** – Height at which terrain will be adjusted towards.
 - **Clamp Range** – Height that any terrain variations are able to occupy when terrain is clamped towards the Clamp Height.
 - **Terrain Mask** – Selects the Terrain Mask that will be used to protect terrain areas from alteration by this brush.
 - **Freeze Objects** – Check to disable automatic adjustment of objects placed on the terrain when terrain height changes occur. (not useful until the user switches tools to actually deform the terrain.)
 - **Round/Square/Line** – Select the shape of the brush.
- The **Copy Paste Controls** section will allow you to manage and edit areas you have saved for use with the Copy Paste brush.
 - To use the Copy Paste brush, select it from the Brush Controls menu and Drag-select while holding the left mouse button to select the area on the terrain that you wish to add to a Copy Paste brush. When you are satisfied with your selection, release the left mouse button.
 - Under the Copy Paste Controls, you will see both the Copy Paste Brushes that you have previously created that session in addition to any other Copy Paste brushes you have saved as a previous Copy Paste Brush File.
 - There are various settings in the Copy Paste Controls section which will alter the application of the Copy Paste brush you have selected:
 - **Height** – The absolute position value that the brush will paint onto the terrain.
 - **Offset** – The amount that the terrain features contained within the brush will be offset from the Height value.
 - **Z Ang.** – Angle position of brush in relation to original copied terrain features.
 - **90° (R)** – Rotates the selected brush at a 90 degree angle. Uses “R” hotkey.
 - **Scale** – Amplifies the copied terrain’s features by the indicated magnitude.
 - **Drop** – Returns the Copy Paste brush to its original Height setting. Uses the “D” hotkey.
 - **Flip Horizontal/Vertical** – Flip the direction of the brush horizontally or vertically.
 - **Greater/Replace/Less** – The mode by which the Copy Paste brush will apply the new terrain to the currently existing terrain. The “Greater” option will only apply the brush to areas of current terrain which would be increased in height by the brush. The “Replace” option will completely replace the terrain which the brush is applied to with the contents of the Copy Paste brush. The

"Less" option will only replace areas of current terrain which have a height that is lower than the height of the Copy Paste brush that is applied to them.

- **Color/Mats/Props/Splines/Decals** – Checkmarks indicate which aspects of the selected Copy Paste brush that will be applied upon use to the current terrain.
- **Mask Color to Height** – Color present on the copy paste terrain will only be applied to height areas which meet the placement style requirement.
- **Mask Mats to Height** – Textures present on the copy paste terrain will only be applied to height areas which meet the placement style requirement.
- **Track Terrain** – Disables the Height function of the Copy Paste brush and automatically adjusts the Copy Paste brush to the current height of the terrain under the mouse cursor.
- **Show Patch Grid** – Displays the patch grid on the terrain.
- **Height Map** – A visual representation of the height variations in the map. Here you can import and export PNG files which contain the height information of your map for later use.
 - **Min/Max** – Sets the minimum and maximum height values that will be used when applying the currently loaded height map to the terrain.
 - **Min Camera** – Game camera will never focus below the specified height.
 - **Water Ht** – Sets the Height of the water table.
- **Playable Bounds** – Allows the user to resize and reposition the map's playable bounds.
 - **Center** – The position of the map's center point and playable area.
 - **Size** – The absolute size of the map.
 - *Tip: When editing the Playable Bounds section, the green outline displayed on the terrain will indicate what changes will be made if you apply the alterations to the map.*
- **Terrain Resize Tools** – Changes the size of the map using absolute value or scaling. Allows the user to alter the alignment of the map's playable area.
 - **Center** – Adjusts the alignment of the map playable area using the center point as reference.
 - **Size** – Alters the size of the map in the "patches" unit.
 - **Resize** – Increases or decreases the terrain size by the indicated value.
 - **Rescale** – Scales the currently existing terrain size to fit the indicated value.
 - *Tip: When using the Terrain Resize Tools, the green outline displayed on the terrain will indicate what changes will be made if you apply the alterations to the map.*
- **Symmetry Tool** – Allows the user to mirror the terrain of their map along a central line.
 - **Terrain, Splines, Decals, Props, Game Objects, Passability, Paths** – Includes the checked items in the application of the mirror tool.
 - **Mirror Reflect** – Mirrors the terrain directly across the indicated line.
 - **Mirror Rotate** – Mirrors and flips the terrain across the indicated line.
 - **Angle** – Alters the angle of the central line along which the terrain will be mirrored.

02_Texture Tab

Getting Started

The texture tab allows the user to paint textures onto the environmental terrain. Texture sets, specific materials, brushes, and cliff texture settings can be altered here.

To place a texture material on the map, select the desired texture from the Texture Set window and Left-Click the Terrain. To use the Cliff Brush, select it from the dropdown menu in the "Brush Style" section and select the first texture in a series of matching textures, then Left-click the the terrain to apply the material(s).

Tip: When attempting to create natural looking terrain, use a small radius leveling brush with small height variations throughout an area prior to using the texture brush.

Tip: The Cliff Healing Brush is useful for situations where preexisting cliff terrain has been re-adjusted or re-shaped and requires its cliff terrain textures to be recalculated without disrupting any surrounding or layered textures that are not part of that cliff brush.

Tip: Beginners may benefit from loading a Biome, as it includes pre-made Texture Sets to play around with. (Menu Bar File Load Map Biome Set)

Editor Pane Tools

- **Texture Set** – Displays the currently loaded textures, with the selected texture outlined in green. These textures can be saved with the Biome, but the user can alter them individually in the Material settings below this section.
 - **Load/Save** – Loads and saves a texture set for use.
 - **Replace** – Upon activation, prompts the user for confirmation. The first texture selected (must not be currently selected) after the prompt will be the texture you wish to replace. The second texture you select will be the texture which replaces all instances of the first texture currently on the terrain. Replaced textures are still preserved in the Texture palette for later use.
 - **Terrain Diffuse Intensity** – Alters the intensity with which the diffuse settings impact the terrain.
 - **Terrain Specular Intensity** – Alters the intensity with which the terrain expresses its specular settings.
 - **PBR** – Used to activate Noisy Normals, used to break up repetitive textures and are adjustable per material. Using the checkbox offsets /scales, and noisy normals don't influence the visual result.
 - Creating a PBR: The specular texture is interpreted as an MRA texture. Red channel is interpreted as the 'metalness' of the surface. Green is the 'roughness' of the surface and the blue is the 'ambient occlusion'.
- **Material Settings**
 - **Texture Path** – When selected, allows the user to specify which texture file will be used for that material.
 - **Bump Texture** – When selected, allows the user to specify which file will be used to render the material's bump texture.
 - **Spec Texture** – When selected, allows the user to specify which file will be used to render the material's specular settings.
 - **Surface Type** – Sets the logical type of surface represented by the material.
 - **Diffuse** – Changes the diffuse coloration of the texture.
 - **Specular** – Changes the specular tint of the texture.
 - **Material Shininess** – Controls the size of the specular highlight.
 - **Blend Hardness** – Adjusts the hardness of the texture's edge blending with surrounding textures.
 - **Z Angle** – Rotates the texture.
 - **Scale** – Sets the magnification of the base texture file that will be displayed within the material.
 - **U Offset** – Offsets the texture horizontally.
 - **Tilt Angle** – Stretches the perspective angle of the texture.
 - **V Offset** – Offsets the texture vertically.
 - **Damage Material** – (Not used in the Great War Western Front project.)
 - **Lock Settings** – Locks the selected material's settings so that they cannot be altered.
 - **Hard Edge** – Prevents the material from blending with other textures placed on the terrain.
 - **Hole** – Disables the display of the currently selected material and displays a blank space in its place.
- **Brush Style** – Controls the brush shape and settings for texture application.
 - **Additive Conical Brush** – Used to paint a section of texture onto the terrain.
 - **Cliff Brush** – Paints the sides of angled environments such as steep, cliffs, mountains, and other variable terrain. This tool utilizes a set of materials to apply various textures to different terrain angles to generate a cohesive texture application.
 - **Cliff Healing Brush** – Reapplies the currently selected Cliff Brush set to any previously placed Cliff Terrain texture areas, and only to those areas.
 - The Brush style dialog has various settings that alter the aspects of certain brushes:
 - **Brush Radius** – Adjusts the brush size.
 - **Generate Material Normals** – This modifies the Z and Tilt Angles of the selected materials to make them suitable for use with the cliff brush.
 - **Material Count** – Number of textures used to generate the Cliff Brush.
 - **180°/360°** – The total rotation value that is used for the Generate Material Normals function. Recommended setting is 180°.
 - **Starting Angle** – The starting angle to use for the tilt angles. Generally can be left at 0°.
 - **Tilt Angle** – Tilt angle to apply to the current set of cliff materials. 90° is usually the best value for this setting.
 - **Line Mode** – Turns the brush into a line-style brush. To begin painting, click the desired start point for the line, and then drag and release the mouse button at the desired end point.

03_Props Tab

Getting Started

Decorative props can be placed onto the terrain to further enhance your map. Trees, bushes, roots, decorative rocks, structures, special effects, and more can be placed and edited in the Props tab.

To place a prop, select it from the list and Shift+Left Click the terrain. Props can be rotated using the circular "Rotate Gizmo" and can be moved around the terrain by either dragging the movement arrows surrounding the prop, or by Left Clicking the prop and dragging it to the desired location on the map.

Tip: Although there is no method to "scale" the size of an existing prop, there is a method to add a color to tint to each in the Object Control Pane.

Tip: If you are looking for destroyable props, you won't find them here as they are considered Objects and can be found in the Objects Pane. Props are in general mainly used for decoration that the game does not need to logically keep track of.

Editor Pane Tools

- **Add Object** – Lists all of the available props that can be placed on the map.
 - **Search** – Searches the list of available Props.
 - **Map Objects Only** – Only displays Props in the list which have already been placed on the map.
 - **View Options** – Displays a dialog that allows the user to toggle active selection of different categories of Props and Objects. Props and objects toggled off with this dialog still exist on the map.
 - **Disable/Enable Selection** – Toggles whether the current object or prop type is able to be selected when editing the terrain.
 - **Hide/Show Objects** – Toggles visual display of the currently selected object or prop type on the terrain.
 - **Enable All/Show All** – Toggles all props and object types to be selectable or visible.
 - **Preview** – Shows a visual preview of the currently selected prop or object.
 - **Object Control** – Displays the currently selected prop's settings. Settings are able to be changed once the prop has been placed onto the terrain.
 - **Name** – Displays the specific name of the selected object, which will be used to differentiate it from other instances of the same prop which exist on the map.
 - **Type** – Lists the original filename of the selected prop's model.
 - **Pos X, Y, Z** – Sets the absolute X, Y, and Z position values of the prop or object.
 - **Z Offset** – Sets the Z value in relation to terrain height.
 - **Ang X, Y, Z** – Adjusts the X, Y, and Z angles of the prop, its move gizmo, and its rotate gizmo.
 - **Ignore Height** – When checked, the prop will ignore changes to the terrain height.
 - **World Rotation** – Selected props can be rotated around a central point in the terrain.
 - **Color** – Sets the prop's diffuse/tint color.
 - **+ –** Adds a new color set to the color list.
 - **Select All of Color** – Will either select all props of the same color that is chosen in the dropdown list, or will select all props that are the same color as the prop that is currently selected on the terrain.
 - **Season Preview:** Although this is presented in the Objects Control Pane, implying that it is specific to the selected object, it is **ACTUALLY** a global setting that changes **ALL** the tree models on the map to their alt models for each respective season.
 - **Alignment** – Used to transfer and apply the alignment aspects of one prop or object to another.
 - **X/Y/ Z Pos** – Transfers the X, Y and Z alignment of the first prop or object.
 - **Selected Min** – The minimum alignment value of the first selected prop.
 - **Selected Center** – The center alignment value of the first selected prop.
 - **Selected Max** – The maximum alignment value of the first selected prop.
 - **Dest Min** – The minimum alignment value of the destination prop.
 - **Dest Center** – The center alignment value of the destination prop.
 - **Dest Max** – The maximum alignment value of the destination prop.
 - **Align Height** – Aligns multiple objects to a similar height as each other.
 - **Drop to Terrain** – Places the object against the terrain.
 - **Apply** – Applies the changes entered.
 - **Alignment Mode** – Enables Alignment mode and unlocks the editing of Alignment Rotation settings section for the currently selected prop.
 - **Map Objects** – Displays a list of props that are currently placed onto the map. Selecting a prop from the list will select the prop for editing and jump to the prop on the map.
 - **Search** – Searches the list of usable map objects.
 - **Align Rotation** – Prop alignment settings which are enabled for editing when "Alignment Mode" under the Alignment section is checked.
 - **X, Y, & Z to X, 90°/180°** – Aligns the prop in the X, Y and Z axis with respect to X.
 - **X, Y, and Z to Y, 90°/180°** – Aligns the prop in the X, Y and Z axis with respect to Y.
 - **X, Y, and Z to Z, 90°/180°** – Aligns the prop in the X, Y and Z axis with respect to Z.
 - **Graphics** – Toggles specific graphics settings for the selected props or objects.
 - **Cast Shadow** – Toggles whether the selected prop or object can create a shadow.
 - **LOD** – Level of detail that is required for the prop or object to appear.
 - **Reflect** – Prop can reflect in water plane.
 - **Near Camera Fade** – Allows the object to fade the closer the user views it.
 - **Far Cull** – Object is able to be culled from the map at a particular viewing distance.
 - **Refract** – Object is able to refract within the water plane.

04_Color Tab

Getting Started

The color tab can be used to paint and blend solid colors over terrain textures. This can help add complexity to map art and improve the visual depth.

To paint color onto the map, select the desired color under the Vertex Color setting and Left-Click the brush onto the terrain.

Tip: Holding shift and selecting an area will instantly paint that area with the selected color. Holding Control will activate the rectangle tool and instantly paint that area with the selected color in a rectangular shape.

Tip: The Color Tab is also useful when blocking out a map. Users can bring in visual guides created in a 3rd party image editing software load them in. For example, you can export out the blank Color Map, draw out your regions, then import them back in in order to draw them in.

Editor Pane Tools

- **Terrain Vertex Color Painting** – Displays a map preview of the colors currently painted onto the map.
 - **Recompute Color Map from Terrain** – Reevaluates the color map and updates the map displayed to reflect a more accurate color map.
- **Brush Controls**
 - **Brush Radius** – Alters brush size.
 - **Round/Square/Line/Fill** – Determines the brush type and shape.
 - **Tolerance** – Adjusts the color tolerance for the Fill brush type. Fill must be selected to enable this field.
 - **Vertex Color** – Color which will be painted onto terrain textures by the color tool.
 - **Opacity** – Alters the opacity with which the color is applied to map textures.
 - **Soft Edge** – If enabled, this setting will cause the brush to paint onto the terrain with a diffused, rounded edge.
- **Import/Export** – Imports a previously created TGA file as a color map or Exports the currently loaded Color map as a TGA file.

05_Splines Tab

Getting Started

Splines can be used to create roads, rivers, waterfalls, or variety of other similar line objects with animated textures on it.

To make a new spline, select the desired Spline style from the Presets section then Shift-Left click on the terrain. Subsequent spline sections can be added by Ctrl+Left clicking on the terrain once a start point has been placed. A Spline must have at least 3 total points to display on the map. Spline shape and angle can be adjusted by dragging the spline's points along the terrain.

Editor Pane Tools

- **Presets**
 - Search – Search the current list of spline presets.
 - Save/Load – Save your current spline presets or load a previously saved collection of spline presets.
 - New – Create a new spline in the spline presets list.
 - Apply to Selection – Apply the attributes of the preset currently chosen in the spline presets list to the spline that you have selected on the terrain.
- **Preset Parameters** – Displays the parameters of the selected spline preset. Certain parameters will only be displayed for certain spline types (such as river or road).
 - Surface - Determines the types of VFX emitted by units which walk upon these terrain items.
 - Dense Water – Texture that will be displayed around the outer edges of the spline.
 - White Water – Texture that will be displayed in the center of the spline.
 - Water Bump – Select the bump texture that will be used for the spline's water surface.
 - Water Distortion – Alters the magnitude of visual distortion displayed on and under the spline.
 - Water Mask – Red channel of the selected texture will act as foam in the deep areas of the water.
 - Base Texture – Base texture that will be used for the road spline.
 - Bump Map – Bump map texture that will be used for the road spline.
 - Road Damage – Decorative texture that will be displayed on top of the road spline's base texture.
 - Width – Width of the spline that will be created.
 - Scale – The scale at which the chosen water texture will be displayed upon the spline.
 - LOD – Level of Detail within which the spline is displayed.
 - Diffuse – Displays the tint/diffuse color of the spline.
 - Opacity – Opacity of the spline.
 - Start – Opacity for start of spline.
 - End – Opacity for end of spline.
 - Spec – The tint/Specular lighting setting for light reflected off of the spline.
 - Specular Intensity – Represents the intensity of light reflection off of the spline.
 - Gloss – Represents how shiny/glossy the spline appears.
 - Scroll U – Speed at which the texture travels towards the start point of the spline.
 - Scroll V – Speed at which the texture travels towards the end point of the spline.
 - Clamp V – Adjusts V position of texture and bump map within the spline.
 - Clamp U – Adjusts U position of texture and bump map within the spline.
- **River Preset Parameters** – Adjusts the aspects of the River Preset spline's animation.
 - Dense Water Scroll Scale – Rate at which the Dense lower water will scroll.
 - White Water Scroll Scale –
 - Normal Scroll Scale 1 – Rate at which the first layer of upper water will scroll.
 - Normal Scroll Scale 2 – Rate at which the second layer of upper water will scroll.
 - Distortion Scroll Scale – Magnitude of the distortion displayed upon the spline's textures.
 - White Water Texture Scale U – Changes the display scale of the upper water layer texture.
- **Track Parameters** – Specific settings related to the scrolling of the spline's textures across the spline path.
 - Preset Name – Name of the spline preset that is used on the currently selected spline.
 - Opacity Edge 1 – Opacity of spline's edge 1.
 - Middle – Opacity of spline's center section.
 - Edge 2 – Opacity of spline's edge 2.
 - Fade Distance – Depth of river spline or similar at which opacity settings begin.
 - Width – Size of the individual spline segments upon which the texture resides. Similar to spline angle smoothness.
 - Adj Width – Amount to adjust the spline segment width.
- **Control Point** – Adjusts the aspects of the currently selected point on the spline and the attached spline section.
 - Z Adjust – Adjust the Z axis location of the selected control point.
 - Width - Adjust the width of only the segment that lies on the currently selected control point.
 - Delete – Deletes the currently selected control point.
- **Misc**
 - Remove Track – Remove the currently selected spline.

06_Decals Tab

Getting Started

Decals can be used to add interesting features, imperfections, or aspects to the ground of the terrain.

Shift+Left click on the terrain to place a new decal. The settings of the new decal can be adjusted after placement in the left side of the editor.

Editor Pane Tools

- **Decals** – Currently loaded decals are displayed here for selection and use.
 - Save/Load – Saves the currently loaded set of Decals as a Decal File or Loads a set of Decal files into the 'Decals' window for use.
 - Apply to Selection – Applies the attributes of the decal currently selected in the 'Decals' window to the currently selected decal present on the terrain.
- **Material Properties** – Adjusts specific aspects of the decal's material settings.
 - Render Mode – The type of mode used to render the decal. This will determine the appearance and environmental interaction exhibited by the decal.
 - Diffuse – Selects the texture that will be used on the decal.
 - Normal – Selects the texture that will be used on the decal.
 - Specular Intensity – Magnitude of the specular lighting effect that is being emitted from the Decal.
 - Gloss – Controls the size of the specular highlight.
 - LOD – Sets the level of detail at which the decal will be rendered. (use of this tool is decremented for The Great War: Western Front)
- **Decal Properties**
 - Lock Button – When selected, locks the X and Y Scale values to the same value.
 - Scale X – Value that represents the size of the decal along the X-axis.
 - Scale Y – Value that represents the size of the decal along the Y-axis.
 - Angle – Rotates the decal.
 - Intensity – Sets the opacity with which the texture will be applied to the terrain.
 - Tint Color – Selects a solid color with which to tint the decal.
 - Project on Cliffs – (Not used in GreyGoo project.)
 - Randomize – Randomizes the angle at which the next decal will be placed.

07_Grass Tab

Getting Started

The Grass tool allows you to place animated small-scale vegetation onto the map to help enhance the terrain environment.

You can paint grass onto the terrain by selecting the desired grass in the selection window, and left clicking on the terrain. To remove already placed grass, select the "Clear" or "Subtract" brush modes from the dropdown menu.

Editor Pane Tools

- **Grass Materials**
 - Save/Load – Saves the currently loaded grass materials as a Material file or loads a set of Grass settings into the Grass Materials window.
- **Brush Settings**
 - Brush Style – Selects the type of action that will be taken when the user applies the brush to the terrain.
 - Add – Increases the amount of grass objects on the terrain.
 - Subtract – Decreases the amount of grass objects on the terrain.
 - Clear – Removes grass objects from the terrain.
 - Brush Radius – Brush size.
- **Grass Material Settings**
 - Texture – Selects the texture that will be used for the grass objects.
 - Color 0 – Selects the main/upper color of the grass objects.
 - Color 1 – Selects the accent/lower color of the grass objects.
 - Lock Settings – Locks and prevents the Grass Material Settings from being altered until the "Lock Settings" setting has been unchecked.
 - Density – Adjusts the amount of grass objects per terrain square for the selected grass material.
 - Wind Bend Scale – Adjusts the intensity of grass sway in response to simulated wind.
 - Width/Height – Adjusts the width and height of the grass objects.
 - Random Size – Adjusts the frequency that variable size grass object patches will appear for the currently selected Grass Material.
 - Tilt Angle – Tilt angle that the grass objects align in relation to the terrain.
 - Anchor X/Y – Adjusts the grass material's anchor in relation to the X or Y axis.

Note: "Noise Strength" is no longer in use as the size and density brush is utilized to determine the amount of grass that is scattered.

08_Water Tab

Getting Started

The Water tab is where you to control what your map's water appearance, how easily it becomes visible, and many other aspects of the water table.

The water table is visible when the terrain is lowered in height enough for it to be visible. If you want to see the water table on your map, either lower some areas of the terrain enough for it to show or increase the height of the water table itself.

Editor Pane Tools

- **Load/Save Settings**
 - Save/Load Settings – Save current water settings as a .wtr file or load a previously saved Water Settings file.
- **Generic Water Settings**
 - Water Rable Ht – Adjusts the height at which the water table renders. Once terrain falls below this value, the water table will start to render over it.
 - Opacity – Represents the overall clarity of the water and the visibility of terrain/objects beneath it.
 - Edge Fade Depth – Adjusts the length of distance that the edges of water areas will fade into terrain.
 - Fade Opacity – Clarity/Opacity of water that is rendered over edge fade areas.
 - Reflection Dis. – Amount of reflection distortion applied to prop reflections. Must have reflection enabled for each prop.
 - Infinite Water – Makes water near edges appear to extend into the horizon.
- **Textures Settings**
 - Wave Bump – Bump texture that will be used to create the surface waves of the water.
- **Layers Settings**
 - Water Shallow – Color of the shallow water, if set by the water depth and height.
 - Shallow Layer Opacity – Opacity of the shallow water layer, if set by water depth and height.
 - Water Deep Color – used to control how the player is able to see thru the water and to give off the affect of deep and shallow water.
 - Blend Power – used in conjunction with Water Deep Color above.
 - Dark Compensation – similar to Water Deep Color
 - Color Extinction Depth Settings – Used with Water Deep Color, Blend Power, and Dark Compensation to give the effect of deep and shallow water.
- **Behavior Settings** (Wave Bump Scrolling)
 - Size – Changes the size of the texture used to render the waves on the surface of the water.
 - Velocity – Represents the speed at which waves in the water surface are rendered and displayed.
 - Direction – Alters the angle or direction in which waves on the water's surface render.

09_Enviro Tab

Getting Started

Environmental settings control the settings for the map sun, fog, wind, and other environmental effects. You can set whether your map is a night or day map, in addition to altering how Fog of War appears at different levels of reveal as well.

Before proceeding with your environment settings, be sure to select a Post FX Chain from the dropdown list so your settings will display as they would ingame. Changing settings such as the map's brightness can be done by changing the Intensity under Sun. Other environmental aspects like Fog and Clouds can be enabled from this tab as well.

Editor Pane Tools

- **Weather Scenario**
 - Current – The Environment Scenario that is currently selected 1-100. (Each Scenario contains its own parameters example: position of sun, color of diffused colors, etc.)
 - Available – the number of Environment Scenarios to select from. You can adjust the amount available by hitting (Add) and (Remove)
 - Weather – Setting the type of weather set for each Environment.
 - Scenario – Choosing the type of weather pattern that is to be used for each weather setting. This is mainly used to select what type of fx to be used. example: rain, snowfall, etc.
- **Environment Set Operations**
 - Save/Load – Save the current Enviro settings to a file for later use or select a set of previously used Enviro settings for use.
 - Merge – Combines the current Enviro settings with those in the file selected by listing the additional Enviro settings in the “Current” dropdown menu.
 - Post FX Chain – Selects which Post FX chain to use when interpreting Enviro settings.
- **Sun Settings**
 - Intensity – Alters the brightness of the environment sunlight.
 - Z Angle – Rotates the sunlight source around the terrain.
 - Tilt Angle – Moves the sunlight source lower or higher in relation to the terrain.
 - Ambient Color – Color of the ambient environment.
 - Diffuse – Color of the light diffusing off the terrain.
 - Spec Intensity – Intensity with which light reflects off of surfaces.
 - Shadow Bias – Opacity of shadows displayed on the map.
 - Day/Night – Denotes the map as a day or night cycle; value of 0 indicates day while a value of 1 indicates nighttime. This is used to allow lights placed in editor or lights on props to give off light. the value 0-1 anything in between will determine how bright the light source affects the surrounding area.
 - Force Fill Light Alignment – Assigns the additional fill lights to automatic preset locations in relation to the primary sun light.
 - Update Sun from Cubemap Lighting – Analyzes the lighting cubemap and automatically sets the sun light parameters based on the cubemap.
- **Fog of War** – Allows editing of the tint and alpha values for unexplored, previously explored, and currently visible Fog of War. View – When checked, allows the display and editing of the Fog of War in various states on the editor's map. If the
 - “View” option is unchecked, terrain will display as if there is no active Fog of War.
 - Full – Display settings for unexplored Fog of War.
 - Explored – Display settings for previously explored Fog of War.
 - None – Display settings for for Currently Visible Fog of War.
- **Particle Light Settings** – Alters how particle light is displayed in the map's environment.
 - Alpha Diffuse Multiplier – Changes the diffuse color multiplier in Alpha particles.
 - Alpha Additive Multiplier – Changes the additive color multiplier in Alpha particles.
 - Bump Emissive Adder – Adds to the emissive bump color for Alpha particles.
 - Copy Sun Diffuse – Copy the selected Diffuse color under Sun Settings.
- **Sky Dome** – Background texture or image that will be displayed in the sky area.
 - Dome1
 - Z Angle – Angle of the texture displayed on the lower section of the sky area.
 - Z Offset – Adjusts the lower skydome further or closer from the ground.
 - Dome2
 - Z Angle – Angle of texture displayed on the ceiling/upper section of the sky area.
 - Z Offset – Adjusts the ceiling/upper section of the sky area further or closer from the ground.
- **Fill Light Settings** – Allows the use of two alternate fill lighting sources on the map.
 - Intensity 1 – Sets the brightness/strength of Fill Light 1.
 - Z Ang – Sets the Z angle of Fill Light 1.
 - Tilt Ang – Sets the tilt angle of Fill Light 1.
 - Diffuse – Sets the Diffuse color of Fill Light 1.
 - Intensity 2 – Sets the brightness/strength of Fill Light 2.
 - Z Ang – Sets the Z angle of Fill Light 2.
 - Tilt Ang – Sets the tilt angle of Fill Light 2.
 - Diffuse – Sets the Diffuse color of Fill Light 2.
 - Mirror Sun - Used to set the backlight. It adjusts how strong the shadowed areas are on the overall map.
- **Cubemap Lighting** – Manages settings related to surface reflection lighting.
 - Use Level's Cubemap – Estimates and generates a cubemap from the Skydome settings.
 - Intensity – Alters the strength of the cubemap lighting.
 - Reset – Reverts cubemap lighting settings to default/none.
 - Sky Cube – Selects a file containing a group of cubemap settings.
 - Sky Cube with Sun – Selects a file containing a group of cubemap settings that include settings for sun cubemap lighting.
- **Weather Settings**
 - Wind Dir – Wind Direction. Represents the direction that the environmental wind blows.

- Environment/Grass Wind Speed – Represents the intensity/speed of the wind for environmental objects which animate based on this setting.
- Cloud Size – Adjusts the size/scale of the cloud shadow projections onto the terrain.
- Cloud Wind Speed – Alters the drift speed of shadows projected by cloud objects onto the terrain.
- Cloud Texture – Selects the texture which is used to render the cloud shadows onto the terrain.
- **Fog Settings** – Toggles the display of mist/fog in the environment.
 - Start – The view distance at which fog begins to display into view.
 - End – The view distance at which fog begins to fade out of view.
 - Sky % – The visibility of fog in the sky area.
 - Color – Tint color of the environmental fog.
- **ASSAO Settings**
 - **NO LONGER USED.**
- **SSAO Settings** – Settings for the shading and display of dynamically created shadows.
 - Radius – Amount that the shader will extend out when generating its effects.
 - Intensity – Opacity of ambient lighting occlusion.
 - Max Occlusion – Maximum allowable occlusion that will be rendered on the terrain itself.
- **Terrain Modification** – Alters major weather settings for the terrain surface (and some objects as well).
 - Noisy Normals- scalars for this environment set.
 - Wet
 - Shine Strength- a multiplier on the shininess of the terrain.
 - Wet Brightness- a multiplier on the underlying terrain brightness.
 - Normalize- set to zero to have terrain normals be 'maximally wet' (ie forgotten) or 'minimally wet' (ie maintained).
 - Snow
 - Normalize- the intensity of terrain noisiness.
 - Snow Color- the color replacing the terrain surface.
 - Scattered Snow
 - Primarily used for the strategic map, don't use!

10_Lighting Tab

Getting Started

The Lighting tab is where you can add individual decorative light points and objects to your terrain.

Individual lights can have their own tint, brightness, display frequency, and other settings which can be altered in the left-hand settings panel for each light. To place a light object, Shift+Left Click on the terrain. To move a light object, Left Click and drag the light to the desired location.

Editor Pane Tools

- **Light Type**
 - Omni – Creates an Omnidirectional light.
 - Projected – Projects a light-based image onto a surface.
 - Spot – Projects light in one direction from its source.
 - Box – Creates a rectangular light.
- **Light Properties**
 - Diffuse – Selects the color which the light source will diffuse onto the terrain around it. Applicable to all light sources.
 - Range – Selects the distance which the light source will emit light.
 - Intensity – Selects the brightness of the light source. Applicable to all light sources.
 - Far/Near Attenuation – Applicable to projected light sources. Adjusts the size of the projected light source.
 - Texture – The texture which will be displayed in/on the light source. Applicable to Projected and Cookie light types.
 - Night Only – If this checkbox is enabled, the light source will alter its state around the night/day cycle.
 - Ignore on Low-end Renderer – Indicates that the light object will not be rendered if the client is set to only display necessary render objects in low quality setting.
- **Time Properties**
 - Noise – Variable frequency of interruption in the display of the light source.
 - Noise time – Variable duration of interruption in the display of the light source.
- **Light Orientation**
 - Position Z – Alters the height at which the light is displayed.
 - Angle X, Y, Z – Used to adjust the angle at which the light object is displayed in the X, Y, and Z axis. Applicable to Projected, Spot, and Box lighting objects.
- **Misc**
 - Delete ALL Light Sources – Deletes every light source present on the map.

11_Post FX Tab

Getting Started

The PostFX editor will allow you to specify the different graphical settings you wish to utilize on your map. Effects like bloom, color correction, and Depth of Field can be adjusted here.

Color tint of the entire map can be altered at once by adjusting the saturation and color correction of the PostFX settings. Default settings generally work best with a wide variety of maps if you wish to retain the same graphical settings as the game's base maps.

Tip: Beginners may benefit from loading a pre made PostFX, Post FX Chains to play around with. Use the Import button in the Post FX Chains Pane. Although pre-made chains are named by the size of the map (So first time users wouldn't have to mess with the Map Edge setting below), there is no issue using them on any size map as long as you adjust the Map Edge.

Editor Pane Tools

- **PostFx Editor**
- **HDRBloom** – Allows the user to control the display of the bloom effect.
 - Enable – When checked, enables the bloom setting for clients that meet the requirements.
 - Bloom Cutoff –
 - Bloom De-saturation –
 - Bloom Strength –
- **ToneMap** – Allows the user to control the display of the HDR Tone Mapping effect.
 - Exposure – Adjusts the exposure level of the entire map.
 - MaxBright – Alters relative brightness of map.
- **HeatDistortionHDR** – Enables heat distortion effects for the map.
- **DepthOfField** – Allows the user to control the display of the Depth of Field effect.
 - Focal Distance – Distance away from the main area of the screen that focus will be applied.
 - Focal Range – Range of focus area around the point of focus.
- **ColorCorrection** – Allows the user to control the display of the ColorCorrection effect.
 - Black Point
 - Red/Green/Blue – Individually adjusts the Red, Green, or Blue settings of the Black Point for the entire map.
 - Lock RGB – Lock the relative RGB values for the black point.
 - White Point
 - Red/Green/Blue – Individually adjusts the Red, Green, or Blue settings of the White Point for the entire map.
 - Lock RGB – Lock the relative RGB values for the white point.
 - SaturationControl – Allows the user to control the display of the Saturation Control effect.
 - De-saturate – Reverses the Saturation setting and removes color intensity from the map.
 - Intensity – Magnitude at which the saturation effect is applied.
- **TempChromaLuma**
 - Temperature- a color to tint the screen by, written in Kelvin.
 - Chroma Scale- a control of how 'strong' colors are (similar to saturation).
 - Chroma Power- a control of the distribution of color strength (eg higher means a collapse of saturation values down to zero).
 - Luma Scale- a control of pixel brightnesses.
 - Luma Power- a control of the distribution of pixel brightness (eg lower means a collapse of pixel brightnesses to one).
 - Vignette Radius- the radius in screen units of the vignette.
 - Vignette Falloff- the rate of the vignette fades in.
 - Vignette Height- the two camera heights between which the vignette can change. The right-hand values (alts) are interpolated to as the camera height tends toward the Vignette Height's right-hand value (alt).
 - Colorize- a color to tint the screen by, maintaining the pixel brightness.
 - Scattering Scale- generally, don't use this parameter.
 - Scattering Color- generally, don't use this parameter.
- **Map Edge**
 - Minimum X- the world x-position of the left-hand map edge.
 - Minimum Y- the world y-position of the bottom map edge.
 - Maximum X- the world x-position of the right-hand map edge.
 - Maximum Y- the world y-position of the top map edge.
 - Falloff Radius- the radius the edge falloff takes to go from none to full.
 - Falloff Power- a control of the falloff rate as a power curve.
 - Edge Color- the color demarking the beginning of the edge.
 - Edge Radius- the width of the color demarkation.
 - Height- the lower camera height at which the northern map edge fully disappears.
 - Height Alt- the upper camera height at which the northern map edge fully appears.

12_Objects Tab

Getting Starte

Gameplay objects like player spawn points, AI functionality markers, and destructible blocking objects can be placed in the Objects tab.

- To place a gameplay object onto the terrain, Shift+Left Click on the terrain where you wish to place it.
- To alter specific settings about the object(s) you have placed, select their associated spawners, which appear as green flags on the terrain.
 - Spawner Flags can be moved around on the terrain to a convenient location without impacting the location of their associated objects.
 - **HOWEVER**, if no flag is available, you should still be able to click on the object and edit the Object Properties.

Editor Panes

- **Objects**
 - Search – Searches the list of Objects that are available for map placement.
 - Map Objects Only – Displays a list of Objects that are currently placed onto the map.
 - View Options – Displays the selection & view options panel so the user can toggle the display of various types of objects placed onto the map.
- **Preview** – Displays a preview of the Object the user has selected in the list.
- **SOB Objects** – Displays a list of the currently placed objects on the terrain.
 - Search - Search a list of currently placed objects on the terrain.
- **Object Properties** – Displays settings and information for the currently selected object on the terrain. The spawn flag for the desired object must be selected to edit certain settings.
 - Name – Displays the name given to the specific object that is selected. The user is able to rename objects by typing in this field.
 - ID – Lists a specific number identifier for the selected object on the terrain.
 - Type – Displays the base object type for the selected object on the terrain.
 - Allegiance – Represents the 'team' that the object will be placed on upon spawning. HUMANPLAYER is the default setting. The allegiance setting refers to all players on the map and is only able to be changed on the Spawner Properties flag of an object.
 - Pos XYZ – The specific X, Y, and Z position of the currently selected object on the terrain.
 - Z Off – The distance of the object relative to terrain height.
 - Ang XYZ – The X, Y, and Z angle of the currently selected object on the terrain.
 - Color– Sets the object's tint color and group.
 - + – Adds a new color to the color list.
 - Select All of Color – Selects all objects that are of the same color selected in the dropdown list or of the currently selected object.
 - Ignore Height – Object ignores adjustments to terrain height.
 - World Rotation – Objects selected can be rotated around a central point.
 - Spawner Properties– Allows the user to edit the properties of the currently selected spawner Flag object on the terrain (Not used The Great War).
 - Selected Spawner – Displays which spawner flag is currently selected for editing.
 - Edit All Spawners – When checked, all edits made to the current spawner will carry over to all selected spawners.
 - Undo Changes – Revert recent changes made to spawners.
 - Initially Active – When checked, the spawner activates when the instance has launched.
 - Max Subgroup Size – Maximum number of objects per spawn marker that can be present at one time.
 - Spawn Count – Total number of spawner activations per instance before the object will no longer spawn. A designation of -1 means the object will continuously respawn.
 - First Activation Delay – Amount of time that must pass before the spawner activates for the first time in the instance.
 - Respawn Delay – Amount of time that must pass before the spawner activates again.
 - Respawn at % dead – Percentage of health missing on current object(s) required before the spawner will attempt to respawn a new object. A 0 value will cause constant spawn trigger.
 - Spawned Objects – Lists the objects that will be spawned by the currently selected spawner.
 - Object Properties – Displays the name of the object that will be spawned by the currently selected spawner.
 - Is Factory – (Not used The Great War.)
 - Spawn Type – (Not used The Great War.)
 - Spawn Chance – (Not used The Great War.)
 - Level 1/2/3/4 – (Not used The Great War.)
 - Aggro Enabled (unit can attack) – (Not used The Great War.)
 - Targetable – When checked, the object/unit can be targeted by direct attack (Not used The Great War.)
 - Invincible (non-attackable) – When checked, the object/unit cannot be attacked (Not used The Great War.)
- **Alignment Tools** – Used to transfer and apply the alignment aspects of one object to another.
 - X, Y, and Z Pos – Transfers the X, Y and Z alignment of an object.
 - Selected Min – The minimum alignment value of the first selected object.
 - Selected Center – The center alignment value of the first selected object.
 - Selected Max – The maximum alignment value of the first selected object.
 - Dest Min – The minimum alignment value of the destination object.
 - Dest Center – The center alignment value of the destination object.
 - Dest Max – The maximum alignment value of the destination object.
 - Align Height – Aligns multiple objects to a similar height as each other.
 - Drop to Terrain – Places the object against the terrain.
 - Alignment Mode – Enables Alignment mode and unlocks the editing of Alignment Rotation settings section for the currently selected object.
 - Hide Move Gizmo – When checked, hides the movement arrows that display when an object is selected.
 - Hide Rotate Gizmo – When checked, hides the rotation ring that display when an object is selected.
 - Use World Space – Uses exact world coordinates rather than relative coordinates to dictate position.
- **Align Rotation** – Object alignment settings which are enabled for editing when "Alignment Mode" under the Alignment section is checked.
 - X, Y, & Z to X, 90°/180° – Aligns the object in the X, Y and Z axis with respect to X.

- X, Y, and Z to Y, 90°/180° – Aligns the object in the X, Y and Z axis with respect to Y.
- X, Y, and Z to Z, 90°/180° – Aligns the object in the X, Y and Z axis with respect to Z.

13_Pass Tab

Getting Started

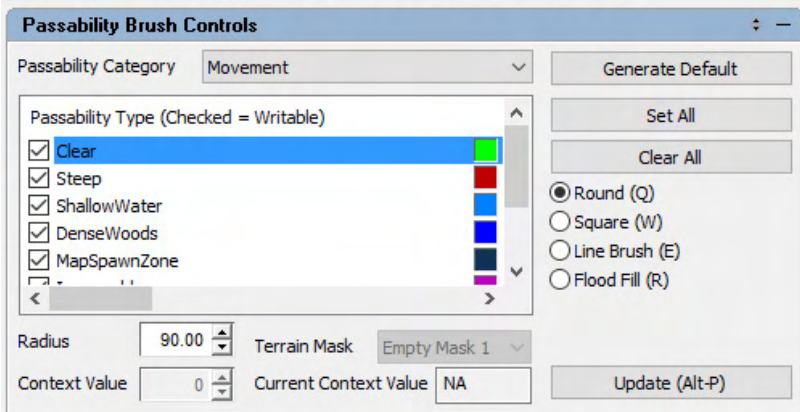
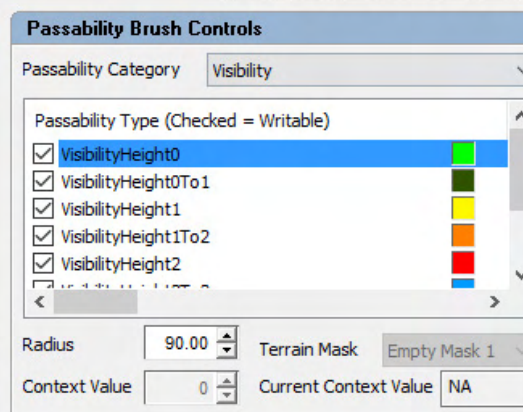
The Passability tool will allow you to alter the pathing aspects of different terrain types, add brush areas, and change visibility/targeting behavior on the map.

If you wish to have passability automatically calculated for your map, you can simply click the "Generate Default" button to have the map editor generate a rough passability map. To manually alter the passability, select the desired passability type from the Passability brush controls section and Left Click the terrain. To change the type of passability you are applying (movement, brush, etc), select the desired category from the dropdown menu.

Tip: When creating brush areas on your map, the context value is important. Brush areas with the same context value will share vision, so be sure to give a new context value to every brush area on your map if you want them to each grant their own individual vision areas.

Editor Pane Tools

- **Passability Brush Controls** – Allows selection and editing of different movement, brush, visibility, and targeting areas.
 - Passability Category – Selects the category of movement types desired for editing.
 - The number keys can be used to select a specific passability (i.e. clear = 1, steep = 2, pit = 6, etc.)
 - Generate Default – Directs the Map Editor to attempt to automatically apply the appropriate Movement and Visibility passability types based on terrain height and shape.
 - Passability Type – Lists different kinds of passability for editing that fall under the Passability Category selected. A checkmark next to the name of the Passability type indicates that it is shown on the map and ready for editing. The color displayed after the type name indicates the overlay which is used to represent that type.
 - Radius – Brush size.
 - The brackets keys can be used to shrink ([) or grow (]) the brush size.
 - Set All – Checks all listed Passability Types for editing and display.
 - Clear All – Unchecks all listed Passability Types for editing and display.
 - Context Value – The group designation which the next brush area will be painted as.
 - Current Context Value – The group designation of the brush area that is currently under the mouse cursor.
 - Round/Square/Line Brush/Flood Fill – Shape of the brush used to paint passability onto the terrain.
 - Height Map – Visual representation of the current passability painted onto the map.
 - **Debug Display Tools** – Allows the user to select different modes of movement, visibility, and pathing for visualization on the current passability map.
-
- Below is a description of each Passability Category and how the paints are used in regards to The Great War: Western Front (TGW).
 - Movement is the main and first type of Paint used in map development, ALL other passabilities Categories although editable in order to achieve a particular style, should be left as default.
 - For The most part, PAINT MOVEMENT and accept all defaults for the other categories.

<ul style="list-style-type: none"> • As the name implies, this control Movement for Objects (such as Companies). <ul style="list-style-type: none"> • Some Movement types affect the other categories. • Used extensively through map development. 	<ul style="list-style-type: none"> • As the name implies, this control Visibility for Objects. <ul style="list-style-type: none"> • For The Great War, we mainly leave visibility up to
<h2 style="text-align: center;">MOVEMENT</h2> 	<h2 style="text-align: center;">VISIBILITY</h2> 

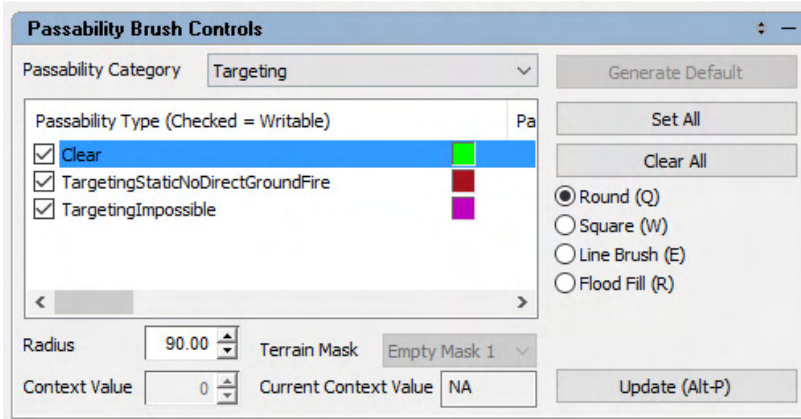
- **Clear**
- **Steep (SHOULD BE AVOIDED IN TGW)**
 - Prevents Company Movement into area.
 - Prevent visibility into area (Visibility set to **VisibilityHeight3**).
 - Prevents Direct fire into the area (Visibility set to **TargetingStaticNoDirectGroundFire**),
 - Overuse can cause Company Pathing issues.
- **ShallowWater (NOT USED IN TGW)**
- **DenseWoods**
 - Used to Create Dense Forest game mechanic on a Map.
 - Prevents Vehicle Movement into the area.
 - Is painted automatically when BRUSH is painted.
- **MapSpawnZone**
 - Although it appears to have no effect at all, that is not the case.
 - It prevents other units from entering this map area and using it to traverse the map in actual gameplay.
 - It also protects the spawning Company by blocking targeting of them while they are in area.
 - When creating your own maps, MapSpawnZones are roughly a distance of 1043 units (26 cells) starting where the Impassable Skirt ends (for more on the Map Skirt See Map Anatomy and Sizes in [Map Editor](#)).
- **Impassable (SHOULD BE AVOIDED IN TGW)**
 - Prevents Company Movement into area.
 - Prevent visibility into area (Visibility set to VisibilityImpossible).
 - Should not be used in plays space.
 - Overuse can cause Company Pathing issues.
- **Cliff (SHOULD BE AVOIDED IN TGW)**
 - Basically same properties as Pit. Try not to use.
- **Pit**
 - Prevents Company Movement into area.
 - Allows visibility into area.

- **VisibilityHeight0-3 (NOT USED IN TGW)**
- **Visibility Impossible (NOT USED IN TGW)**
 - For the TGW, visibility is handled by the game in a

- As the name implies, this control Targeting for Objects.
 - For The Great War, we mainly leave targeting up to the system.

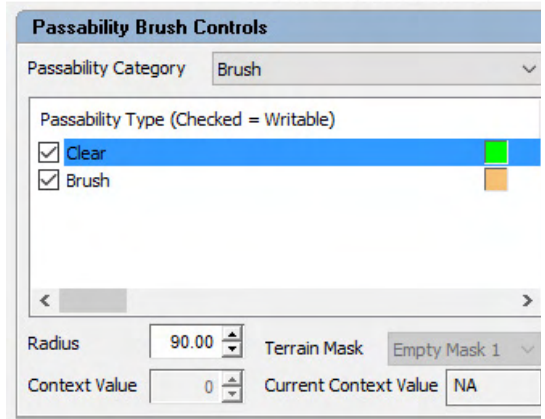
- As the name implies, sets up Brush (Dense Forest Mec)
 - Used extensively through map development.

TARGETING



- Clear
- **TargetingStaticNoDirectGroundFire (SHOULD BE AVOIDED IN TGW)**
 - Placed by Steep Movement.
- **TargetingImpossible (SHOULD BE AVOIDED IN TGW)**
 - Placed by Impassable Movement.

BRUSH



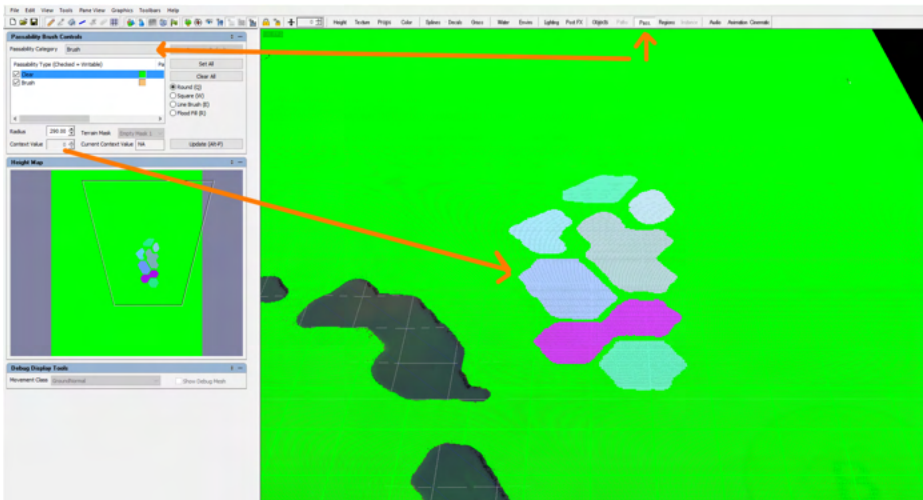
- Clear
- Brush
 - Used to Create Dense Forest game mechanic on a map
 - Vehicle Movement is handled by DenseWoods mechanic
 - Paints DenseWoods movement automatically when painted
 - Painting Clear does not delete the DenseWoods
 - Uses a Context Value that when different, creates different maps.
 - See Below for more information.

The Great War: Western Front Dense Forests

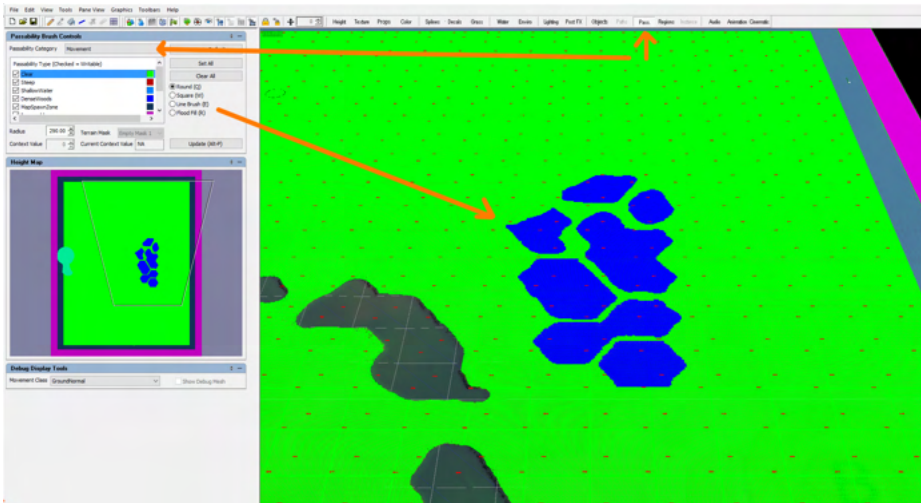


- Dense Forest are a gameplay element that is intended to:
 - Prevent Trench Placement if in a region.
 - Prevent Tank Movement.
 - Stealth Infantry Companies, as long as the enemy is not in the same forest.
- The following is a brief explanation as to how to create a Dense Forest using the Pass Tab.

Creating Brush



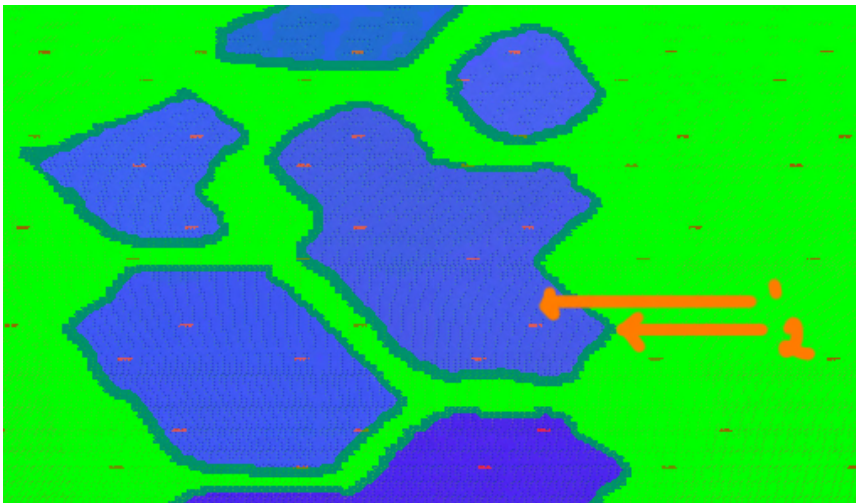
- While in the Pass Tab change the Passability Category to **Brush**.
 - Brush comes in two types: Clear and Brush.
 - Clear is essential our way to delete any Brush.
 - Brush
- Brush is essentially what causes the stealth behavior in Companies.
- When drawn it starts at a Context Value of 0
 - The **Context Value** is important as a different value lets the system know that this should be considered a different Dense Forest.
 - When a Context Value is shared across different brush blobs, the system considers them the same Dense Forest. Generally we give separate brush blobs a different value, so enemies companies are forced inside to get vision.
 - **Current Context Value**, will let you know what the current context is of whatever brush blob you hover your mouse over.
- **HOWEVER**, Brush on its own does not prevent Movement, therefore when you draw brush, the system automatically draws DenseWoods Passability.
 - Switching Passability Category to **Movement** we can see that in action.



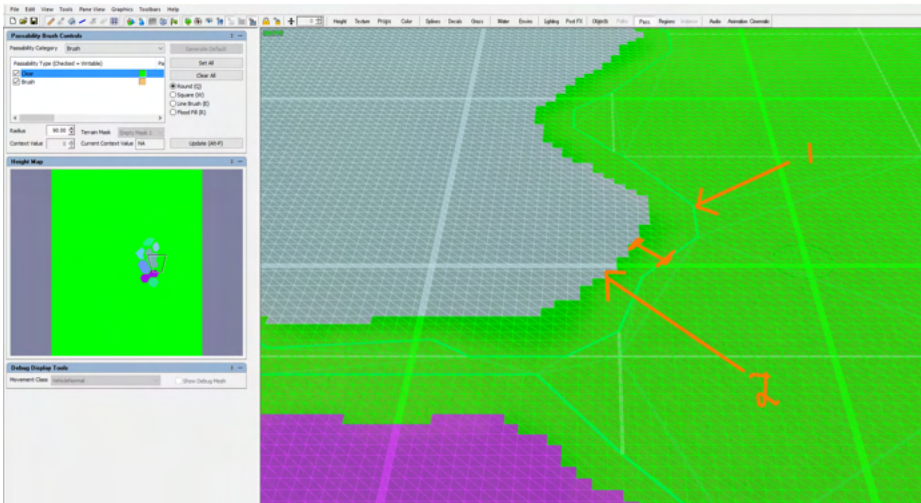
- Now, its best practice to paint brush first, since painting DenseWoods Passability does not work the same as when you paint brush, meaning painting Dense Wood **WILL NOT PAINT BRUSH**.
 - **Furthermore**, erasing Brush DOES NOT ERASE THE CREATED DENSEWOOD PASSABILITY, so Keep that in mind.

Refining Brush

- Once your Brush is painted, a different Context added, the last thing we need to do is refine our Brush.
- If we left our Brush as is, we would find that Tanks moving around the edges of our Brush, tend to stealth.
 - This is caused due to tank movement which will have them skirt forest as CLOSELY as possible, using the tanks center as reference instead of its extents.
- **THEREFORE**, in order to fix this, we generally take our Bush and paint Clear around the edges of our Brush blob.
 - In the image below, its an image of the concept. As you can see we took our Brush (1) and we painted Clear around it in order to shrink it. In the end, we should push the Brush far enough from the DenseWood Passability (2) to keep edge hugging tanks from getting stealth.



- While in the Movement Pasability Category, you can use the Debug Displays Tools to change the Movement Class from **Ground Normal** to **Vehicle Normal**.
 - Doing this will allow you to see the Tank Movement(1) while painting Brush (2)



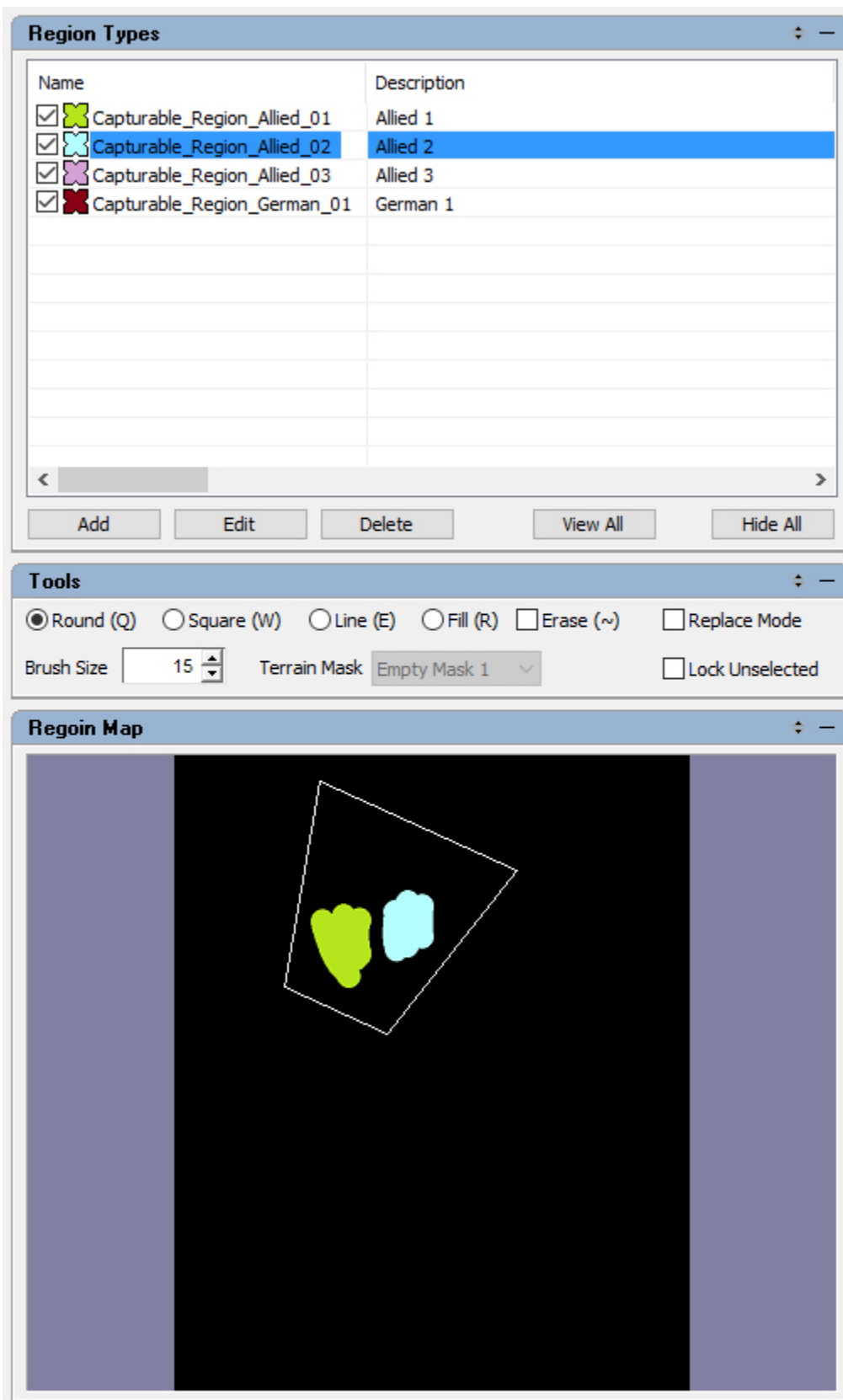
- Once we are happy and tested our Brush, we can then proceed to Prop, Color, and Texture accordingly to convey the Dense Forest.



14_Regions Tab

Getting Started

Regions are layers which can be affiliated with an effect generator define in xml. When an object enters or exits a region, the affiliated effect generator is activated.



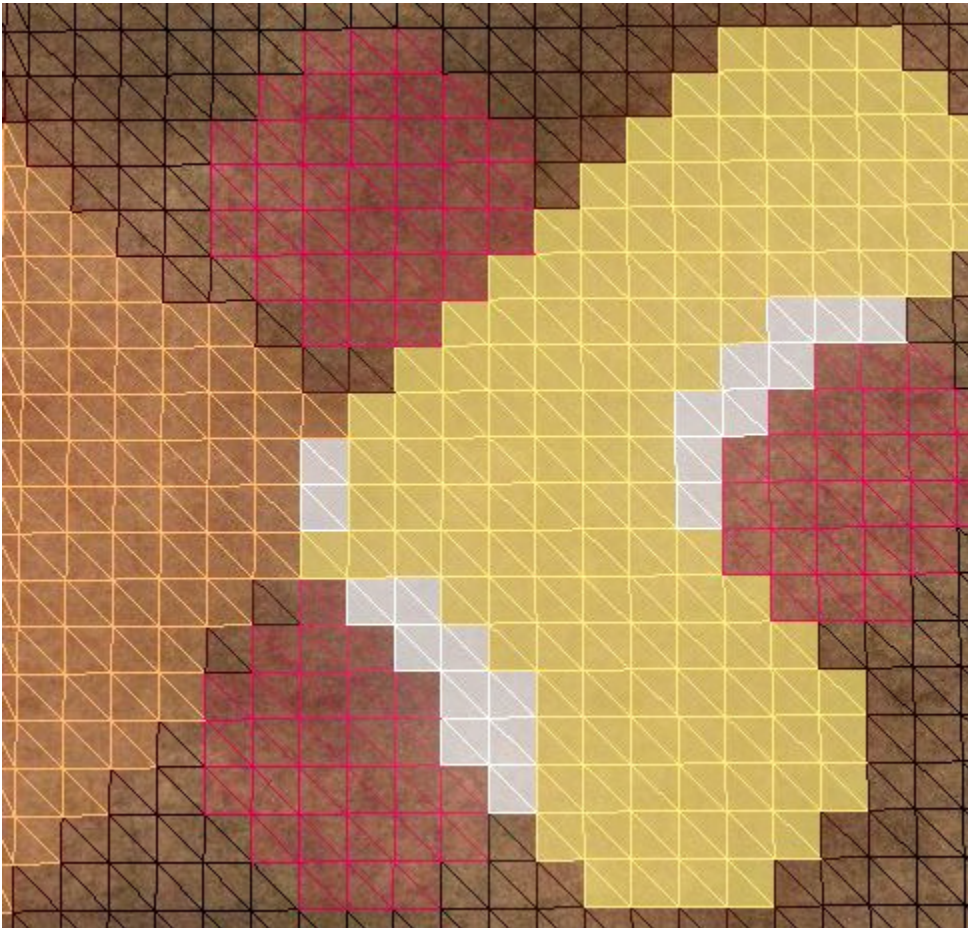
Editor Pane Tools

Region Types:

- **Name:** Name of the region MUST match the name of the Effect Generator in order to function.
- **Description:** description for future users.
- **Add:** Allows the creation of new Regions.
- **Edit:** Allows you to change the name or the description of an existing region.
- **Delete**
- **View All:** Checks the checkbox to the left of the Name. Region is visible.
- **Hide All:** UnChecks the checkbox to the left of the Name. Region is Hidden.

Creating a Region

- To create a new trigger region, select the Add option in the **Region Types** editing pane.
- A dialog allowing you to name the region will appear. The Name of the region MUST match the name of the Effect Generator in order to function.
- By default, when created a Region will be visible. The checkbox to the left of the name denotes this as it starts checked.
- To paint regions on the heightfield, select an existing region or create a new one by selecting the Add button in the Region Types editing pane.
 - In the main editing view, hold the left mouse button and drag to paint the region in the desired location.
 - Regions can be painted atop each other as they are considered separate layers. Further, they can cover irregular spaces and multiple areas across the heightfield independent of each other.
 - Objects which are created within a region are considered to be entering the region, and will activate any associated Effect Generator if defined so.
- Options are available to Add, Edit, and Delete regions at the bottom of the Region Types editing pane.
- Region colors given in order and cannot be changed, as they are assigned on creation in the same order.



Tools

- The Tools editing pane allows you to modify the brush used to paint the trigger region. The following options are available:
 - **Round (Q):** The default selection, round shaped Brush. This option causes the currently selected region to be painted on the heightfield. Hotkey Q to switch.
 - **Square (W):** Square shaped Brush. Works same as default, but square. Hotkey W to switch.

- **Line (E):** The line draw tool allows you to paint the region in a line, at the brush width. To paint in a line, left-click the starting and ending positions of the line on the heightfield in the main editing view. Hotkey E to switch.
- **Fill (R):** Uses the fill brush to fill pre-painted trigger region areas for any layer able to be manipulated (the checkbox is checked in the **Region Types** editing pane). Clicking outside the areas will fill the remaining space. Hotkey R to switch.
- **Erase (~):** Turns any of the above Brushes into a delete Brush. Hotkey ~ to switch.
- **Brush Size:** The radius of the brush in cells.
- **Terrain Mask:** A Terrain Mask is created via the Height Tab, and can be loaded here. The Terrain Mask prevents paint from being applied where the mask is.
- **Replace Mode:** This option allows you to replace currently painted region areas with the currently selected region when painting. Prevents painted regions from overlapping.
- **Lock Unselected:** Prevents unselected regions from being replaced when painting with your current paint. Works kind of like a Terrain Mask.

Region Map

- Provides you with a visual of the Region currently painted. If a Region is hidden, it will show up without color.

Available Regions

- Since a Region is capable of triggering an effect generator when something enters it or exits it, for The Great War: Western Front the following Name will get you the specific results.

Name	Description	NOTES
Capturable_Region_Allied_01	Allied 1	Used Primarily on the South side of the map, in skirmish associated with Object type CAPTURE_POINT_ALLIED_01 (A)
Capturable_Region_Allied_02	Allied 2	Used Primarily on the South side of the map, in skirmish associated with Object type CAPTURE_POINT_ALLIED_02 (B)
Capturable_Region_Allied_03	Allied 3	In skirmish associated Object type CAPTURE_POINT_ALLIED_03 (C), but not necessary for this map.
Capturable_Region_German_01	German 1	Used Primarily on the North side of the map, in skirmish associated with Object type CAPTURE_POINT_GERMAN_01 (X)
Capturable_Region_German_02	German 2	Used Primarily on the North side of the map, in skirmish associated with Object type CAPTURE_POINT_GERMAN_02 (Y)
Capturable_Region_German_03	German 3	In skirmish associated Object type CAPTURE_POINT_GERMAN_03 (Z), but not necessary for this map.
Capturable_Region_Allied_Home	Allied Home Region	Used Primarily on the South side of the map, in skirmish associated with Object type STRUCTURE_FACTION1_HQ_00_SPAWN_OBJECT (Allied Command Trench)
Capturable_Region_German_Home	German Home Region	Used Primarily on the North side of the map, in skirmish associated with Object type STRUCTURE_FACTION2_HQ_00_SPAWN_OBJECT (Central Powers Command Trench)
Region_Water_00	Map Terrain Effect: Slow Company	Used to denote the "shallow" water sections of the map that slow down Companies moving through them. Addressed later in the tutorial.

15_Audio Tab

Getting Started

The Audio tab allows you to set audio events which will determine which audio files play in certain areas. It will allow your map to have a unique combination of ambient sounds.

To start a new Audio path, Shift+Left click on the terrain. Hold the Shift button and Left-Click on subsequent areas that you wish to denote the borders of the audio activation/origination area.

Editor Pane Tools

- **Audio Paths**
 - Current Audio Paths List
 - Path – Indicates the specific Audio Path denoted by the entry.
 - Looping Event – Lists the Looping Audio Event present in the indicated audio path.
 - Random Event – Lists the Intermittent Audio event present in the indicated audio path.
 - Edges – Number of edges or borders present on the Audio Path.
 - Looping Event – Event which will constantly play in the next audio path that is created.
 - Intermittent Event – Event which will periodically play in the next audio path that is created.
- **Mini Map** – Visual representation of the current Audio Paths present on the map terrain.

16_Cinematic Tab

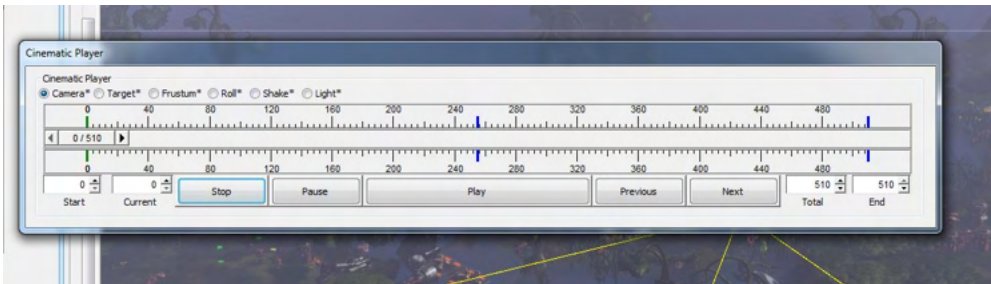
Getting Started

- Cinematics are generally used in Historical Missions and called via lua script.
- This system allows the designers to fly the camera along a path to show the player locations, or enemy activity that's relevant to the story or the mission. The camera can either fade to black at the end of a sequence, or transition directly into the game camera. This is a nice effect as the player is put directly into the action at the end of the cinematic.
- Cinematic sequences are created / edited in the Map Editor and then saved out as a .tec file (Located ..\Project\Art\CINEMATICS). A sequence is then initiated via the debug command 'cinplay FileName' and user can then record using video capture software during the game.
 - Although it is best to create a Cinematic on the map you wish to record on, Cinematics files can be used on any map, issues only being if you somehow place your camera markers outside of playable bounds.

Anatomy of the Cinematic Editor

Cinematic Player

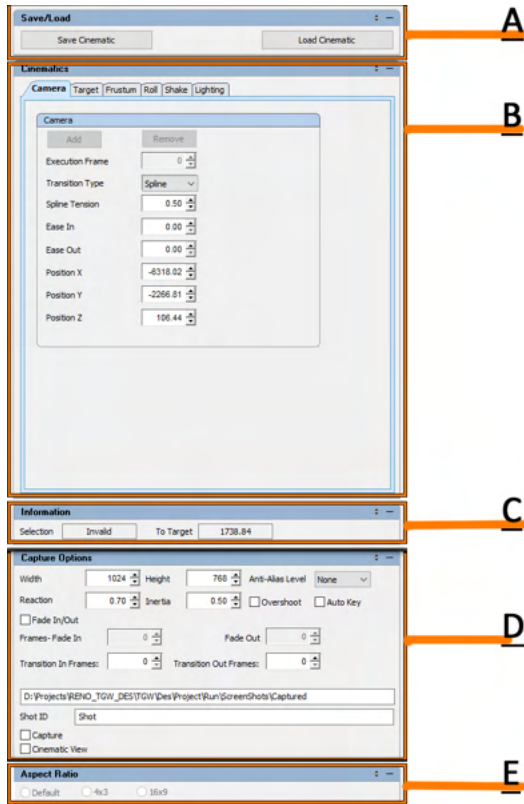
The Cinematic Player (a.k.a. the time line) provides the user with features to view and edit their cinematic.



- **Time Line Slider**
 - Displays the Current frame and the End frame. Drag slider in order to "scrub" through cinematic.
- **Radio Buttons**
 - **Frame Types** are normally shown as a colored line on the time line (Camera shown as blue, Target shown as red, etc). The Radio Button allows you to see one **OTHER** frame type than the one you currently have selected in the Cinematic Pane.
- **Input Boxes**
 - **Start**
 - Displays the current Start Frame and allows change to this value. On change will visually adjust the time line in order to reflect the new Start Frame.
 - **NOTE:** your first camera is always placed at Frame 0 and can NEVER be deleted, however by changing the Start you force the cinematic to start at your selected frame, but there are no camera settings available there since there technically no Camera Frame at that point.
 - **Current**
 - Displays the Current Frame and change to this value will jump to that specific frame.
 - **Total**
 - Displays the cinematic's Total Frames and allows change this number.
 - Total has a hard max of 600 frames.
 - If the Total is made lower than the End, Total and End take the lower number and become equal.
 - Changes to this value visually adjusts the time line.
 - **End**
 - Displays current End frame and allows change to this value. Changes to this value visually adjusts the time line.
- **Buttons**
 - **Stop**
 - Cancels Play/Paused commands and jumps the Slider to the Start Frame.
 - **Pause**
 - While playing the cinematic, pauses the Time Line at the current frame.
 - **Play**
 - Jumps the Slider to the Start Frame and plays the cinematic.
 - **Previous**
 - When multiple Frame types of a single type are present on the time line, this causes the Time Slider to jump to them, moving towards the Start Frame.
 - **Next**
 - When multiple Frame types of a single type are present on the time line, this causes the Time Slider to jump to them, moving towards the End Frame.

Editor Pane Tools

Refers to the set of vertical aligned tools located to the Left hand side of the editor. Each will be explained in brief with an in depth explanation to follow.



- **A) Save/Load Pane**
 - Save and Loads .tec files, please ensure files are saved and loaded from the following location '..\Des\Project\Art\CINEMATICS'
- **B) Cinematics Pane**
 - Works in conjunction with the Cinematic Player to Add new Frames Types and Edit/Remove existing ones (**See Section Below**).
 - Furthermore, by selecting one of the 6 tabs it will cause the Time Line to display All Frames of that type. Frame Types are shown as a Specific Color.
 - There are 6 types of frame types that can be added to the Cinematic by selecting a blank frame from the Time Line.
 - Although It is possible to Add multiple Frame Types to a frame, the SAME Frame Type cannot be added to a single frame.
 - To Edit an existing Frame, first select one of the 6 tabs that represent each type of frame, as this will make them visible on the time line. Then on the time line select one of the frames. The Settings for that frame will now be available in the Cinematics Pane.
 - To Remove an existing Frame, first select one of the 6 tabs that represent each type of frame, as this will make them visible on the time line. Then on the time line select one of the frames. Then in the Cinematics Pane, hit the Remove Button.
- **C) Information**
 - When "scrubbing" the time line, will provide you with the distance of the camera from the Target at that frame. Does not work when Playing the cinematic via the time line.
- **D) Capture Options**
 - Mainly provides settings that can be used to "Capture" the cinematic from **WITHIN** the Terrain Editor, HOWEVER does have a few settings that can be useful in creating cinematic (**See Section Below**).
 - When Capturing from the Terrain Editor, .bmp files of each frame are created at the specified location.
- **E) Aspect Ratio**
 - Not used at this time.

Keyframes and Animation

Each camera configuration can be used as a keyframe in an animation. The camera moves between keyframes with a transition. Transitions can be hard cuts, rotations, linear movement or specified with a smoothed spline curve. The path of the camera is displayed in the tool.

Cinematics Pane : Camera



The camera is positioned and oriented in space. Via the CAMERA TAB, the designer can also set the field of view and clipping plane distances in order to control what is revealed to the player.

Camera animations can be reviewed by use of a timeline that allows the designer to play, pause, or set an animation at any given frame.

- **Camera Tab**

- **Add/Remove**
 - In order to Add any other Camera other than the first. select a blank Frame on the Cinematic Player then press Add.
 - In order to Remove, select any existing Camera on the time line, then press Remove.
 - **NOTE:**
 - Adding and Removing will not work if **Cinematic View** is checked on.
 - You cannot use the Remove button on the Camera on Frame 0.
- **Execution Frame**
 - Allows you to take an existing camera and move it to a different location on the time line.
- **Transition Type**
 - How to transition to the next keyframe:
 - Spline - Depending on your Spline Tension, this type will allow line (Camera path) to bend and arc.
 - Linear - This will prevent your line (Camera path) from bending at all, always maintains a rigid path.
 - Cut - This prevents the camera from advancing down the line (Camera Path) waiting for the next Camera Frame in the sequence before moving jumping there.
 - Rotate - The effect is best seen using 2 Cameras, but essential will move the camera around a target, using the first frame as the start of the circle and the second as the end.
- **Spline Tension**
 - The lower the tension, the straighter the A to B line becomes.
- **Ease-In/Ease-Out**
 - How smoothly the camera arrives or departs this location.
- **Position X,X,Z**
 - Allows you to set camera position numerically.

Cinematics Pane : Target

The camera will always orient itself toward the Target. Via the Target TAB, the designer can have the Camera look in specific direction by placing down a target.

Target animations can be reviewed by use of a timeline that allows the designer to play, pause, or set an animation at any given frame.

- **Target Tab**

- (Same as Settings for Camera Tab above)

Cinematics Pane : Frustum

The frustum tab allows the designer to set various

- **Target Tab**

- **Execution Frame**
 - Allows you to take an existing Frame and move it to a different location on the time line.
- **Transition Type**
 - How to transition to the next keyframe:
 - Linear - Basically tweens settings between 2 Frustum Frames, as the Camera moves down its path, providing a smooth transitions between settings.
 - Cut- instead of tweening settings between 2 Frustum Frames, will immediately change settings on arriving at the next Frustum Frame.
- **Camera Field of View**
 - The Camera field of view is the extent of the observable world that is seen at any given moment, the higher the number the more it sees and the SMALLER the target looks.
- **Near Clip**
 - Clips objects shorter than a given distance, can also affect z-buffer accuracy.
- **Far Clip**
 - Clips geometry farther than a given distance, can also affect z-buffer accuracy.
- **Ease-In/Ease-Out**
 - How smoothly the camera transitions between multiple frustum frames as it moves down the path.
- **Object Fade Distance**
 - distance at which objects fade out.
- **Object Rejection Distance**
 - distance at which objects are hidden.
 - **NOTE:** It may seem like the "Far Clip" and "Object Rejection Distance" control the same thing, and they are similar except that the object rejection distance is specifically for objects. This was needed to fine tune which objects were shown.

Cinematics Pane : Roll/Shake

The camera can be told to roll or shake at a given keyframe, and these effects have ease-in / ease-out.

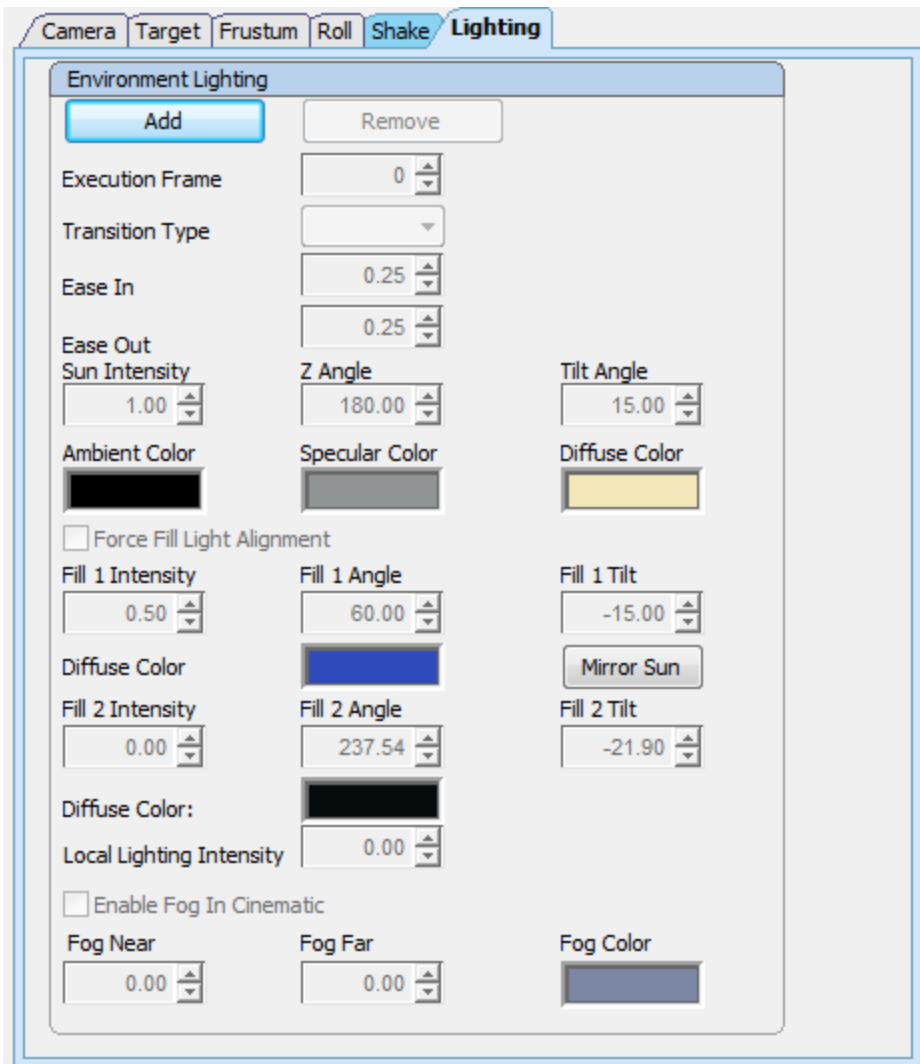
- **Roll Tab**

- **Execution Frame**
 - Allows you to take an existing Frame and move it to a different location on the time line.
- **Transition Type**
 - How to transition to the next keyframe:
 - Linear - Basically tweens settings between 2 Roll Frames, as the Camera moves down its path, providing a smooth transitions between settings.
 - Cut- instead of tweening settings between 2 Roll Frames, will immediately change settings on arriving at the next Frustum Frame.
- **Camera Roll**
 - A value ranging from -90 to 90. Negative Numbers will cause it to Roll Left and Positive Right.
- **Ease-In/Ease-Out**
 - How smoothly the camera transitions between multiple Roll frames as it moves down the path.

- **Shake Tab**

- **Execution Frame**
 - Allows you to take an existing Frame and move it to a different location on the time line.
- **Duration**
 - How many frames the shake should exist.
- **Frequency**
 - How often it should transition between the Up-Down, Left Right settings.
- **Up-Down**
 - The higher the number the more pronounced the effect.
- **Left-Right**
 - The higher the number the more pronounced the effect.

Cinematics Pane : Lighting



For a given camera key frame location the environment lighting can be adjusted. Changes in lighting are eased in and eased out as the camera arrives or departs respectively.

NOTE: When creating your first lighting Frame, the settings will default to those set in the Environment Tab. **Please be careful in** Adjusting the lighting via the Lighting Frame will create permanent lighting effects in a map. Normally, A designer should create their First Frame at 0, then immediately create one at the end of the time line. Therefore, any changes made, should be reverted by the original lighting settings placed at the last frame.

Cinematic Options

There are options for the cinematic as a whole. **HOWEVER**, most settings are not currently used since we record cinematics using screen capture software instead of the game recording for us.

- **Overshoot**
 - The camera will miss the end point a bit before stopping eventually. The Reaction and Inertia settings below can be tweaked to achieve a more pronounced effect.
 - Reaction
 - Inertia
- **AutoKey**
 - When checked, the system will automatically create a key frame on the time line when a Camera or Target is adjusted numerically of by mouse.
- **Fade In/Out**
 - When checked, using the settings below can add a Fade in or Fade out to the cinematic.
 - **Frames Fade In**
 - Will Fade in from black at the Start of the Cinematic. The lower the number the quicker the effect.
 - **Fade Out**
 - Will Fade entirely into black by the last frame of the cinematic. The lower the number the quicker the effect.
- **Transition in Frames**
 - In Game will take the in game cameras location and settings will start to transition into the Start of the Cinematic. The lower the number the quicker the effect.

- **Transition Out Frames**
 - In Game will transition entirely into the in Game Cameras location and settings by the last frame of the cinematic. The lower the number the quicker the effect.
- **Cinematic View**
 - When checked, will allow the Designer to see what the Cinematic Camera sees in the terrain editor.
 - **NOTE:**
 - Designers will not be able to add/remove cameras or targets while in this mode.
 - Designers as the camera can use Terrain Editor Movements to move the camera they are currently looking through, HOWEVER its almost negligible.
- **Misc.**
 - Settings primarily used in order to "record" the cinematic WITHIN the terrain editor. Generates a series of .bmp images one for each frame of the cinematic. This folder can then be opened by video editing software and the user can see their "movie" play there.
 - **Width/Height/Anti-Alias Level**
 - Settings for .bmp
 - **Shot ID**
 - prefix added to the name of the .bmp
 - **Capture**
 - When checked will generate a single .bmp for each frame of the cinematic in a specified folder.

Aspect Ratio

Setting is no longer used, as it used to apply back when the game was able to capture cinematics played in game.

TUTORIAL: Creating a "New Skirmish Map" Mod

- The goal of this tutorial is to set up the basic file structure for a "New Skirmish Map" Mod and then to use the Map Editor to get a brief understanding of how to create what files are needed and get them to be playable in game.
- While we will focus on the process for a skirmish map mod, most of what is learned here can be applied to creating the maps and setting up Historical and Campaign Mods.

01: Setting up and running your Skirmish Map Mod

- RECOMMEND STARTING HERE
 - Learn how to create the basic file structure for your new "Skirmish Map" mod using the provided SampleSkirmishMap Folder and get a high-level understanding of what files are needed.
 - Once completed, you will have a working "Skirmish Map" Mod running with placeholder data that you will be able to load in the game.

Using the Map Editor to customize your new Skirmish Map

- REQUIRES THE PREVIOUS STEP
 - With a location now set up to test your work from the previous step, we cover some basics of the Map Editor Tool and begin the process of creating and customizing the map files.
 - Once completed, you will have a working Custom Skirmish Mod running with your own custom map.

Using the Map Editor to Advance customize your new Skirmish Map

- REQUIRES THE PREVIOUS STEP
 - With your basic Map Completed above the following gives you a few more advanced customizations with Custom Skirmish Mod running with your own custom map.

01: Setting up and running your Skirmish Map Mod

- The goal of this tutorial is to set up the file structure for your new "Skirmish Map" mod using the included sample "SkirmishTestMap Folder", familiarize yourself with the high-level data, and finally launch the Mod in game.
- When completed, you will have a working "Skirmish Map" Mod running with placeholder data that you will be able to load in the game.
 - This newly created file structure will serve as the proper location to load and test your content going forward.

Getting Started

- [Setting up the initial MyNewMap_01 Folder](#)
- [Customizing the newly created MyNewMap_01 folder](#)
 - [Modify TheGreatWar_Mod.xml](#)
 - [Rename Contents of the \Data\Art\Maps Folder](#)
 - [Rename Contents of the \Data\Art\Textures\SRGB Folder](#)
 - [Rename Contents of the \Data\luaversions\current\lua Folder](#)
 - [Modify MYNEWMAP_01_00.lua](#)
 - [Rename Contents of the \Data\Text Folder](#)
 - [Modify MYNEWMAP_01_00.csv](#)
 - [Rename Contents of the \Data\xml Folder](#)
 - [Modify MYNEWMAP_01_00_Instances.xml](#)
 - [Modify ModConfig.xml](#)
 - [Launch MYNEWMAP_01 as a Mod](#)



NOTE

- While all of these files/folders can be created by hand, the fastest/easiest way to create your own custom Skirmish mod is to just copy the entire directory structure of the provided "SkirmishTestMap Folder" into your mod and rename and edit files to match your custom skirmish map.
- You can use whatever editor you prefer to open XML and LUA files (for example: Notepad++, Visual Studio, ZeroBrane Studio, etc)

Setting up the initial MyNewMap_01 Folder

- For this tutorial a "SkirmishTestMap" folder has been provided in the "SampleMods" folder where you have the game installed
 - While in your Steam Library, right click on **The Great War: Western Front** Click *Properties* Open a dialog Click *Installed Files* Click *Browse*
 - This will open a window showing the installed game's file structure:
 - Open "SampleMods" and you will see the "SkirmishTestMap" folder we want to copy.
- Copy and paste the entire "SkirmishTestMap" folder into the following location: C:\Users\USER_NAME\Documents\Petroglyph\TheGreatWar\Mods
 - USER_NAME is your windows username.
- Rename the newly pasted folder from "SkirmishTestMap" to "MyNewMap_01"
- Now that you have a working copy of the folder created and renamed, we need to rename the contents before trying it in game.

Customizing the newly created MyNewMap_01 folder

Modify TheGreatWar_Mod.xml

- Located here: ..\MYNEWMAP_01\TheGreatWar_Mod.xml
- Open the MyNewMap_01 folder and open TheGreatWar_Mod.xml
 - Look for the tag `<Name>Petroglyph: Skirmish Test Map</Name>`
 - This is the name shown in the MOD manager and is used to identify your mod.
 - Change: `<Name>Petroglyph: Skirmish Test Map</Name>` to `<Name>My New Skirmish Map 01</Name>`
 - This name can be whatever you prefer and has nothing to do with any other files we will be renaming later.
 - Next find the `<LocalizedNames>` section of tags
 - These tags are used to localize your mod name into other languages.
 - Look for the tag `<Name>Petroglyph: Skirmish Test Map</Name>`
 - Change `<Name>Petroglyph: Skirmish Test Map</Name>` to `<Name> My New Skirmish Map 01 </Name>`
 - Again, this name can be whatever you prefer and has nothing to do with any other files we will be renaming later.
 - Make sure to save your file!

```
<?xml version="1.0" encoding="utf-8">
<!--
This file contains info about the mod used by the game when loading/displaying the mod
-->
<!-- ModInfo -->
<!--
Mod name displayed in-game. The mod name and version must be unique across all installed mods
-->
<Name>My New Skirmish Map 01</Name>
<!--
Mod name displayed in-game (localized)
-->
<LocalizedNames>
  <Entry>
    <Locale>en-us</Locale>
    <Name>My New Skirmish Map 01</Name>
  </Entry>
</LocalizedNames>
<!--
Description displayed in-game (default)
-->
<Description>A sample mod that adds a skirmish map that can be used as a starting point for creating new skirmish maps. Accessible via "Skirmish" from the main menu.</Description>
<!--
Description displayed in-game (localized)
-->
<LocalizedDescriptions>
  <Entry>
    <Locale>en-us</Locale>
    <Name>A sample mod that adds a skirmish map that can be used as a starting point for creating new skirmish maps. Accessible via "Skirmish" from the main menu.</Name>
  </Entry>
</LocalizedDescriptions>
<!--
Author displayed in-game
-->
<Author>Petroglyph</Author>
<!--
Version of mod
-->
<VersionLow>0</VersionLow>
<VersionHigh>0</VersionHigh>
<!--
Can be one of TEXT_MOD_MANAGER_MOD_TYPE_CAMPAIGN, TEXT_MOD_MANAGER_MOD_TYPE_GENERAL, TEXT_MOD_MANAGER_MOD_TYPE_SKIRMISH, TEXT_MOD_MANAGER_MOD_TYPE_HISTORIC, or use non-localized text
-->
<ModTypeText>TEXT_MOD_MANAGER_MOD_TYPE_SKIRMISH</ModTypeText>
</ModInfo>
</ModInfoMod>
```

NOTE

- Although beyond the scope of this tutorial, *TheGreatWar_Mod.xml* is further defined here as the other tags may prove useful such as *d*escription and *author* tags: [Mod Structure](#)
- Before continuing, you can load this via the Mod Manager in order to see it in action.
 - Check Status Box to load it and click *Confirm* (game will restart)
 - Also make sure *Petroglyph: Sample Skirmish Map* is unchecked since they still have the same data at this point and will conflict.



- After restarting, go to the Skirmish Menu and you can see the map:



- Make sure to unload (uncheck in Mod Manager and confirm) before starting the process below, as the game will potentially crash due to missing data depending on your progress in this tutorial when you try to run it.

Rename Contents of the \Data\Art\Maps Folder

- Located here: ..\MYNEWMAP_01\Data\Art\Maps
- Later when working with the Map Editor, these files will be the ones that will be replaced with our new versions but for now we will just be renaming the provided samples.
- Replace "PUB_20x25_MAP_01" with "MYNEWMAP_01_00" for all file types .ted/.sob/.gpd/.cpd in all of the sub-directories
 - This should be done to all files in the maps directory, including the files inside the Passability and ServerObjects folders.
 - Change **PUB_20x25_MAP_01_00.ted** to **MYNEWMAP_01_00.ted**
 - \Passability\
 - Change **PUB_20x25_MAP_01_00_ALLIED_CTRL.cpd** to **MYNEWMAP_01_00_ALLIED_CTRL.cpd**
 - Change **PUB_20x25_MAP_01_00_ALLIED_CTRL.gpd** to **MYNEWMAP_01_00_ALLIED_CTRL.gpd**
 - Change **PUB_20x25_MAP_01_00_SHARED.cpd** to **MYNEWMAP_01_00_SHARED.cpd**
 - Change **PUB_20x25_MAP_01_00_SHARED.gpd** to **MYNEWMAP_01_00_SHARED.gpd**
 - \ServerObjects\
 - Change **PUB_20x25_MAP_01_00_ALLIED_CTRL.sob** to **MYNEWMAP_01_00_ALLIED_CTRL.sob**
 - Change **PUB_20x25_MAP_01_00_SHARED.sob** to **MYNEWMAP_01_00_SHARED.sob**

Rename Contents of the \Data\Art\Textures\SRGB Folder

- Located here: ..\MYNEWMAP_01\Data\Art\Textures\SRGB
- This folder is mainly used for any images and textures that you want to add or modify especially for unit and historic mods.
 - Skirmish maps mainly just have the mini map image here that shows in the Skirmish UI and is used in battle.
- This is another file that we will be updating with the Map Editor but for now we will just be renaming the provided sample.
 - Change **PUB_20x25_MAP_01_00.tga** change to **MYNEWMAP_01_00.tga**

Rename Contents of the \Data\luaversions\current\lua Folder

- Located here: \MYNEWMAP_01\Data\luaversions\current\lua\
- First, we will start with the trench templates which control the trench layout provided for the player and the AI. These are created using the Trench Configurator tool but again for now, we will just be renaming the provided samples.
 - \TrenchTemplates\
 - Change **PUB_20x25_MAP_01_00_TMP_NORTH_000.lua** to **MYNEWMAP_01_00_TMP_NORTH_000.lua**
 - Change **PUB_20x25_MAP_01_00_TMP_NORTH_000_PLAYER.lua** to **MYNEWMAP_01_00_TMP_NORTH_000_PLAYER.lua**
 - Change **PUB_20x25_MAP_01_00_TMP_SOUTH_000.lua** to **MYNEWMAP_01_00_TMP_SOUTH_000.lua**
 - Change **PUB_20x25_MAP_01_00_TMP_SOUTH_000_PLAYER.lua** to **MYNEWMAP_01_00_TMP_SOUTH_000_PLAYER.lua**
- Now that we have the trench templates renamed, we will rename the map script.
 - \MapScripts\
 - Change **PUB_20x25_MAP_01_00.lua** to **MYNEWMAP_01_00.lua**

NOTE

- Although beyond the scope of this tutorial **MYNEWMAP_01_00.lua** is filled with several functions and variables needed for a Skirmish Map to run while in Tactical.
- While almost all of this can be left alone for skirmish maps, we will update a few of the lines to reference the new trench template names.
- You can use whatever editor you prefer to open XML and LUA files (for example: Notepad++, Visual Studio, ZeroBrane Studio, etc)

Modify **MYNEWMAP_01_00.lua**

- Located here: `..\MYNEWMAP_01\Data\luaversions\current\lua\MapScripts\MYNEWMAP_01_00.lua`
- With the files now renamed, we will modify the trench template references within the file in order to allow our Mod to run properly.
- Open the file and update the trench template names with our new names from above, making sure to keep the "TrenchTemplates." part of the name and keep the name in " " .

```
MapData = {
  MainTimerValue = 1200, -- Choose a Default Value for Campaign (1200 = 20:00) :: In Skirmish int is updated with Time Slider value (See 'Start()' Below)
  CurrentModeStringCheck = nil,
  InstanceName = "Default", -- value is overridden with correct instance name

  -- Store all the Player Information for Both Sides:
  SideAInfo = {}, -- Human Player
  SideBInfo = {}, -- AI player
  EnvironInfo = {}, -- the Environment Player has the SpawnPoint Objects

  CurrentPhase = "PHASE_01",
  HQVictoryOnly = false, -- True == Taking Command Trench ends the match :: False == HOLDING ALL Control Points ends the match

  -- Controls when the AI stops offering surrender (idea is not enough time left so although AI feels will lose, the player probably won't win in time so risk it)
  AI_Stop_Surrender_Time = 210, -- Remaining Timer (3:30) left when AI stops being allowed to surrender (referenced in Misc and NOT below)
  AI_Told_Stop_Surrender = false, -- Is set to true in Misc.lua when AI is told to stop (referenced in Misc and NOT below)

  -- Used to check if we need to override the normal victory result screens
  -- Timer Expires, AI Surrenders, AI Cease Fire...
  Override_Victory = true, -- default is true to handle timer-expiring, will change to false when we hit a early win via control points
  Override_Victory_Value = 3, -- see: https://jira.corp.frontier.co.uk/browse/FARA-3319 for values

  -- Handles the Trench Data for AI and the Player (In skirmish, the allies are usually the south and the germans are usually the north)
  UsePlayerTrenchData = true,
  TrenchDataSouthPlayer = "TrenchTemplates.PUB_20x25_MAP_01_00_TMP_SOUTH_000_PLAYER", -- Player Trench Data
  TrenchDataNorthPlayer = "TrenchTemplates.PUB_20x25_MAP_01_00_TMP_NORTH_000_PLAYER", -- Player Trench Data
  TrenchDataNorth = "TrenchTemplates.PUB_20x25_MAP_01_00_TMP_NORTH_000", -- AI Trench Data
  TrenchDataSouth = "TrenchTemplates.PUB_20x25_MAP_01_00_TMP_SOUTH_000", -- AI Trench Data

  -- Which AI Data file to send to the AI for the mission:
  -- These are currently the generic data files
  AIDataFile_Faction1 = "AI_data.ATTACK_ALLIES",
  AIDataFile_Faction2 = "AI_data.ATTACK",
}
```

- Change "TrenchTemplates.PUB_20x25_MAP_01_00_TMP_SOUTH_000_PLAYER" to "TrenchTemplates.MYNEWMAP_01_00_TMP_SOUTH_000_PLAYER"
- Change "TrenchTemplates.PUB_20x25_MAP_01_00_TMP_NORTH_000_PLAYER" to "TrenchTemplates.MYNEWMAP_01_00_TMP_NORTH_000_PLAYER"
- Change "TrenchTemplates.PUB_20x25_MAP_01_00_TMP_NORTH_000" to "TrenchTemplates.MYNEWMAP_01_00_TMP_NORTH_000"
- Change "TrenchTemplates.PUB_20x25_MAP_01_00_TMP_SOUTH_000" to "TrenchTemplates.MYNEWMAP_01_00_TMP_SOUTH_000"
- Make sure to save your file!

Rename Contents of the `\Data\Text` Folder

- Located here: `..\MYNEWMAP_01\Data\Text\``PUB_20x25_MAP_01_00.csv`
- This is the text file used to setup the text tags for all of your in-game text such as map name and descriptions.
- This file can also be used to localize your text into different languages, but the tags will default to ENGLISH column if no matching text is found in the other columns.
 - Change `PUB_20x25_MAP_01_00.csv` to `MYNEWMAP_01_00.csv`

Modify **MYNEWMAP_01_00.csv**

- Open **MYNEWMAP_01.csv** and update each text tag. You can also change the associated text, but we will be skipping for now (for image consistency in skirmish menus)
 - Change `TEXT_NAME_PUB_20X25_MAP_01_00_CLN` to `TEXT_NAME_MYNEWMAP_01_00_CLN`
 - Change `TEXT_DESC_PUB_20X25_MAP_01_00_CLN` to `TEXT_DESC_MYNEWMAP_01_00_CLN`
 - Change `TEXT_NAME_PUB_20X25_MAP_01_00_DMG` to `TEXT_NAME_MYNEWMAP_01_00_DMG`
 - Change `TEXT_DESC_PUB_20X25_MAP_01_00_DMG` to `TEXT_DESC_MYNEWMAP_01_00_DMG`
 - Change `TEXT_NAME_PUB_20X25_MAP_01_00_SCR` to `TEXT_NAME_MYNEWMAP_01_00_SCR`
 - Change `TEXT_DESC_PUB_20X25_MAP_01_00_SCR` to `TEXT_DESC_MYNEWMAP_01_00_SCR`
- Make sure to save your file!

NOTE

- When Opening a .csv file for the first time (and when saving), please ensure the following settings are set in your .csv editor of choice (See [4. Changing Text](#))

- Having the instance name in the text tags helps prevent issue when creating multiple Mods as it lessens the chance of duplicate tags across mods.
- The different versions of the tags `_CLN`, `_DMG`, and `_SCR` are used to allow different text for the different versions of the map. (Clean, Damaged, Scarred) or (Early War, Mid War, and Late War) on the skirmish screen.

Rename Contents of the \Data\XML Folder

- Located here: `..\MYNEWMAP_01\Data\XML\`
- The `ModConfig.xml` file is used by the mod to know what XML files the mod includes so its name should be left alone so the game can correctly load it.
- The `_Instance.xml` file contains all the hook ups for the map files, lua scripts, and text tags so needs to be unique to the mod.
 - Change `PUB_20x25_MAP_01_00_Instances.xml` to `MYNEWMAP_01_00_Instances.xml`

Modify MYNEWMAP_01_00_Instances.xml

- Located here: `..\MYNEWMAP_01\Data\XML\MYNEWMAP_01_00_Instances.xml`
- We are now going to open the newly renamed file and will be updating tags based on everything we have renamed above.

```
<!-- Early War Version -->
<Instance Name="PUB_20x25_MAP_01_00" Variant="Base_SimpleRTS_Skirmish_Instance">
  <IsDeveloper> false </IsDeveloper>

  <MapFileName> PUB_20x25_MAP_01_00.ted </MapFileName>
  <ServerObjectFileName> PUB_20x25_MAP_01_00_ALLIED_CTRL.sob </ServerObjectFileName>
  <ArtServerObjectFileName> PUB_20x25_MAP_01_00_SHARED.sob </ArtServerObjectFileName>
  <PassabilityFilePrefix> PUB_20x25_MAP_01_00_SHARED </PassabilityFilePrefix>

  <LuaMapScriptName> lua\MapScripts\PUB_20x25_MAP_01_00.lua </LuaMapScriptName>

  <!-- Even Map so this doesn't mean anything -->
  <MFSkirmishAttackerFaction> Faction1 </MFSkirmishAttackerFaction>
  <MFSkirmishDefenderFaction> Faction2 </MFSkirmishDefenderFaction>

  <!-- Tier 2 is all toys and no bonuses : Should be overridden by Skirmish UI -->
  <MFSkirmishTechLevel> 2 </MFSkirmishTechLevel>

  <LocationNameTextID> TEXT_NAME_PUB_20x25_MAP_01_00_CLN </LocationNameTextID>
  <LocationDescriptionTextID network="client"> TEXT_DESC_PUB_20x25_MAP_01_00_CLN </LocationDescriptionTextID>

  <LoadingTextureName>
    <Entry> Empty.tga </Entry>
  </LoadingTextureName>
  <PreviewLoadingTextureName> PUB_20x25_MAP_01_00.tga </PreviewLoadingTextureName>

  <RandomLoadingTextID>
    <Entry> TEXT_CAMPAIGNS_MAP_PVE </Entry>
  </RandomLoadingTextID>

  <MapPlayPosition network="client">
    <X> 243 </X>
    <Y> 243 </Y>
  </MapPlayPosition>

  <AlternatesByDate>
    <Entry>
      <StartYear> 1915 </StartYear>
      <StartMonth> 0 </StartMonth>
      <StartDay> 0 </StartDay>
      <AlternateInstance> PUB_20x25_MAP_01_00_1916 </AlternateInstance>
    </Entry>
    <Entry>
      <StartYear> 1917 </StartYear>
      <StartMonth> 0 </StartMonth>
      <StartDay> 0 </StartDay>
      <AlternateInstance> PUB_20x25_MAP_01_00_1918 </AlternateInstance>
    </Entry>
  </AlternatesByDate>
</Instance>

<!-- Mid-War Version -->
<!-- Replace the MapFileName, ArtServerObjectFileName, and PreviewLoadingTextureName with alternate damage versions if wanted -->
<Instance Name="PUB_20x25_MAP_01_00_1916" Variant="PUB_20x25_MAP_01_00">
  <MapFileName> PUB_20x25_MAP_01_00.ted </MapFileName>
  <ArtServerObjectFileName> PUB_20x25_MAP_01_00_SHARED.sob </ArtServerObjectFileName>
  <LocationNameTextID> TEXT_NAME_PUB_20x25_MAP_01_00_DMG </LocationNameTextID>
  <LocationDescriptionTextID network="client"> TEXT_DESC_PUB_20x25_MAP_01_00_DMG </LocationDescriptionTextID>
  <PreviewLoadingTextureName> PUB_20x25_MAP_01_00.tga </PreviewLoadingTextureName>
  <IsDateVariation> true </IsDateVariation>
</Instance>

<!-- Late-War Version -->
<!-- Replace the MapFileName, ArtServerObjectFileName, and PreviewLoadingTextureName with alternate damage versions if wanted -->
<Instance Name="PUB_20x25_MAP_01_00_1918" Variant="PUB_20x25_MAP_01_00">
  <MapFileName> PUB_20x25_MAP_01_00.ted </MapFileName>
  <ArtServerObjectFileName> PUB_20x25_MAP_01_00_SHARED.sob </ArtServerObjectFileName>
  <LocationNameTextID> TEXT_NAME_PUB_20x25_MAP_01_00_SCR </LocationNameTextID>
  <LocationDescriptionTextID network="client"> TEXT_DESC_PUB_20x25_MAP_01_00_SCR </LocationDescriptionTextID>
  <PreviewLoadingTextureName> PUB_20x25_MAP_01_00.tga </PreviewLoadingTextureName>
  <IsDateVariation> true </IsDateVariation>
</Instance>
```

- `<Instance Name="PUB_20x25_MAP_01_00" Variant="Base_SimpleRTS_Skirmish_Instance">`
 - This is the unique name of the instance that the game loads with the variant containing some of the shared skirmish data. (historical, multiplayer, and campaigns use different `Base_` instances
 - Change `PUB_20x25_MAP_01_00` to `MYNEWMAP_01_00`
- `<MapFileName>`, `<ServerObjectFileName>`, `<ArtServerObjectFileName>`, and `<PassabilityFilePrefix>`
 - These are the `\Data\Art\Maps` files renamed above
 - Change `PUB_20x25_MAP_01_00.ted` to `MYNEWMAP_01_00.ted`
 - Change `PUB_20x25_MAP_01_00_ALLIED_CTRL.sob` to `MYNEWMAP_01_00_ALLIED_CTRL.sob`
 - Change `PUB_20x25_MAP_01_00_SHARED.sob` to `MYNEWMAP_01_00_SHARED.sob`
 - Change `PUB_20x25_MAP_01_00_SHARED` to `MYNEWMAP_01_00_SHARED`
 - Note: The file extension is missing here since this just tells the game which `.cpd` and `.gpd` to use for passability
- `<LuaMapScriptName>`
 - This is the map script used to control the instance and includes the file path.
 - Change `lua\MapScripts\PUB_20x25_MAP_01_00.lua` to `lua\MapScripts\MYNEWMAP_01_00.lua`
- `<LocationNameTextID>` and `<LocationDescriptionTextID network="client">`
 - These are used to hook up the map name and description text tags from the `.csv` file we created.
 - Change `TEXT_NAME_PUB_20x25_MAP_01_00_CLN` to `TEXT_NAME_MYNEWMAP_01_00_CLN`
 - Change `TEXT_DESC_PUB_20x25_MAP_01_00_CLN` to `TEXT_DESC_MYNEWMAP_01_00_CLN`
- `<PreviewLoadingTextureName>`
 - This is used to hook up the minimap for use on the skirmish selection screen and the minimap used in game.
 - Change `PUB_20x25_MAP_01_00.tga` to `MYNEWMAP_01_00.tga`
- `<AlternateInstance>`

- These TWO tags are used to setup which instance to use for the different destruction states (Mid War and Late War) which are setup below.
- Change **PUB_20x25_MAP_01_00_1916** to **MYNEWMAP_01_00_1916**
- Change **PUB_20x25_MAP_01_00_1918** to **MYNEWMAP_01_00_1918**
- Note: The <StartYear> tag doesn't match the names due to shifts in development and naming conventions along with how skirmish UI works - Leave the <StartYear> as 1915 and 1917
- <Instance Name="PUB_20x25_MAP_01_00_1916" Variant="PUB_20x25_MAP_01_00">
 - This is instance used for the mid war version, with the variant telling it to pull missing tags from the original version (such as the script name)
 - Change **PUB_20x25_MAP_01_00_1916** to **MYNEWMAP_01_00_1916**
 - Change **PUB_20x25_MAP_01_00** to **MYNEWMAP_01_00**
 - For the purpose of this tutorial, we are just going to plug in the same <MapFileName>, <ArtServerObjectFileName>, and <PreviewLoadingTextureName> but as you create your own maps, you can create alternate versions with additional damage and plug in accordingly.
 - Change **PUB_20x25_MAP_01_00.ted** to **MYNEWMAP_01_00.ted**
 - Change **PUB_20x25_MAP_01_00_SHARED.sob** to **MYNEWMAP_01_00_SHARED.sob**
 - Change **PUB_20x25_MAP_01_00.tga** to **MYNEWMAP_01_00.tga**
 - <LocationNameTextID> and <LocationDescriptionTextID network="client"> are again the text tags for the name and description of the mid war version that show on the loading screen
 - Change **TEXT_NAME_PUB_20X25_MAP_01_00_DMG** to **TEXT_NAME_MYNEWMAP_01_00_DMG**
 - Change **TEXT_DESC_PUB_20X25_MAP_01_00_DMG** to **TEXT_DESC_MYNEWMAP_01_00_DMG**
- <Instance Name="PUB_20x25_MAP_01_00_1918" Variant="PUB_20x25_MAP_01_00">
 - This is instance used for the late war version, with the variant telling it to pull missing tags from the original version (such as the script name)
 - Change **PUB_20x25_MAP_01_00_1918** to **MYNEWMAP_01_00_1918**
 - Change **PUB_20x25_MAP_01_00** to **MYNEWMAP_01_00**
 - Change **PUB_20x25_MAP_01_00.ted** to **MYNEWMAP_01_00.ted**
 - Change **PUB_20x25_MAP_01_00_SHARED.sob** to **MYNEWMAP_01_00_SHARED.sob**
 - Change **PUB_20x25_MAP_01_00.tga** to **MYNEWMAP_01_00.tga**
 - <LocationNameTextID> and <LocationDescriptionTextID network="client"> are again the text tags for the name and description of the late war version that show on the loading screen
 - Change **TEXT_NAME_PUB_20X25_MAP_01_00_SCR** to **TEXT_NAME_MYNEWMAP_01_00_SCR**
 - Change **TEXT_DESC_PUB_20X25_MAP_01_00_SCR** to **TEXT_DESC_MYNEWMAP_01_00_SCR**
- Make sure to check for extra _ or incorrect spaces in middle of names due to copy and paste errors and then save your file!



NOTE

- This file defines the data the game needs to bring in in order to load the Instance properly - errors in this file, such as incorrect names or formatting issues are a common cause of crashes at game start.
- We are only covering the tags related to the data we modified here.

Modify ModConfig.xml

- Located here: ..\MYNEWMAP_01\Data\XML\ModConfig.xml
- Open ModConfig.xml
 - Remember, the ModConfig.xml file is used by the mod to know what XML files the mod includes and can get more complicated in MODs that modify things such as units and historical missions.
- Since we are only using a single instance XML file for a new map, we only need to update a single entry under <AdditionalInstancesXMLFiles> to have the name of our new Instances.xml file.

```
<ModConfigs>
  <ModConfig>
    <AdditionalGameObjectXMLFiles>
    </AdditionalGameObjectXMLFiles>
    <AdditionalGameConstantsXMLFiles>
    </AdditionalGameConstantsXMLFiles>
    <AdditionalInstancesXMLFiles>
      <Entry>PUB_20x25_MAP_01_00_Instances.xml</Entry>
    </AdditionalInstancesXMLFiles>
    <AdditionalSFXEventXMLFiles>
    </AdditionalSFXEventXMLFiles>
  </ModConfig>
</ModConfigs>
```

- Change **PUB_20x25_MAP_01_00_Instances.xml** to **MYNEWMAP_01_00_Instances.xml**
- Make sure to save your file!

Launch MYNEWMAP_01 as a Mod

- With all the steps taken above, it is now possible to load the Mod in game again with everything looking exactly the same but now using the updated file names.
 - If any issues arise, at this point they are most likely a typo in one of the previous steps (either in the files names or XML hookups most likely).
- Remember to go to the MOD manager and *Check the Status Box* to load it and click *Confirm* (game will restart)
- After restarting, go to the Skirmish Menu and you should see the map in the list and be able to load into it (including the Mid and Late War versions).
 - At this point, you should also be able to activate the "Petroglyph: Sample Skirmish Map" mod and have both running - although unless you rename your text tag you will see two *SAMPLE SKIRMISH MAP* in the list.
- Congratulations - if all went well, you are now ready to move on to the next tutorial and actually modify the map data you have setup using the Map Editor.

NOTE

- A good way to make sure you are seeing your new version in the skirmish list is change the text in the .csv that goes with **TEXT_NAME_MYNEWMAP_01_00_CLN**
- For example, I changed the text from **SAMPLE SKIRMISH MAP** to **MY NEW SKIRMISH MAP** and also have the "*Petroglyph: Sample Skirmish Map*" mod activated so I see both maps in the Skirmish map list.



02: Using the Map Editor to customize your new skirmish map

- The goal of this tutorial is to familiarize the user with the Map Editor as well as the process of map creation.
 - It builds on the work done in the previous one ([01: Setting up and running your Skirmish Map Mod](#)) so it is recommend doing that tutorial first.
- When completed, you will have created new versions of the needed .ted/.sob/.cpd/.gpd/.tga files and have updated your new Skirmish Mod, **My New Skirmish Map 01**



NOTE

While the focus is on a skirmish map, most of what is learned here can be applied to creating maps for the other type of Mods: Historical and Campaign.

SKIRMISH MAP		HISTORICAL MAP		CAMPAIGN MAP
PUB_20x25_MAP_01_00.ted		PUB_20x25_MAP_01_00.ted		PUB_20x25_MAP_01_00.ted
PUB_20x25_MAP_01_00_SHARED.sob (USE AS MAIN .cpd/.gpd)		PUB_20x25_MAP_01_00.sob (USE AS MAIN .cpd/.gpd)		PUB_20x25_MAP_01_00_SHARED.sob (USE AS MAIN .cpd/.gpd)
PUB_20x25_MAP_01_00_ALLIED_CTRL.sob (INCLUDE .cpd/.gpd)		PUB_20x25_MAP_01_00_SKIRMISH.sob (INCLUDE .cpd/.gpd)		PUB_20x25_MAP_01_00_ALLIED_CTRL.sob (INCLUDE .cpd/.gpd)
PUB_20x25_MAP_01_00.tga		PUB_20x25_MAP_01_00.tga		PUB_20x25_MAP_01_00_GERMAN_CTRL.sob (INCLUDE .cpd/.gpd)
		PUB_20x25_MAP_01_CIN_00.tec		PUB_20x25_MAP_01_00.tga

Getting Started

- Since this tutorial will not go into detail for all the Map Editor features, users can familiarize themselves on tools and features via the [Map Editor](#) page, as well as terms which will be used throughout this tutorial.
 - For more information on each editor within the Map Editor, please see the **Editor Tabs** Section in [Map Editor](#) for more in depth details on each one.
 - Also since there are elements that are needed that are out of scope for this tutorial, they will be provided for the user at this location: **C:\Users\USER_DATA\Documents\Petroglyph\TheGreatWar\UserData\ModSampleFiles\MyNewMap_01_00** and will be referenced throughout this tutorial.
- Below is a list of the files we will be creating:

SKIRMISH MAP	STATUS	
MyNewMap_01_00.ted	NEED	The visual aspects of our Map such as: Height Map, Terrain Textures, Props, Vertex Color, Splines, Decals, Grass, Water, Environment settings, Lighting, and Post FX data.
MyNewMap_01_00_SHARED.sob (USE AS MAIN .cpd/.gpd)	NEED	The "Main" logical aspects of our map such as: Spawn Markers and Destroyable Props, along with all of the Passability and Region data in their corresponding .cpd/.gpd files.
MyNewMap_01_00_ALLIED_CTRL.sob (INCLUDE .cpd/.gpd)	NEED	"Supporting" logical aspects of our map including Start Markers and Control Points. Although .cpd/.gpd files are created, they are not used for passability and region data.
MyNewMap_01_00.tga	NEED	The mini map image created in editor and adjusted in a photo editing program to account for distortion. This is used for the mini map and other UI locations.

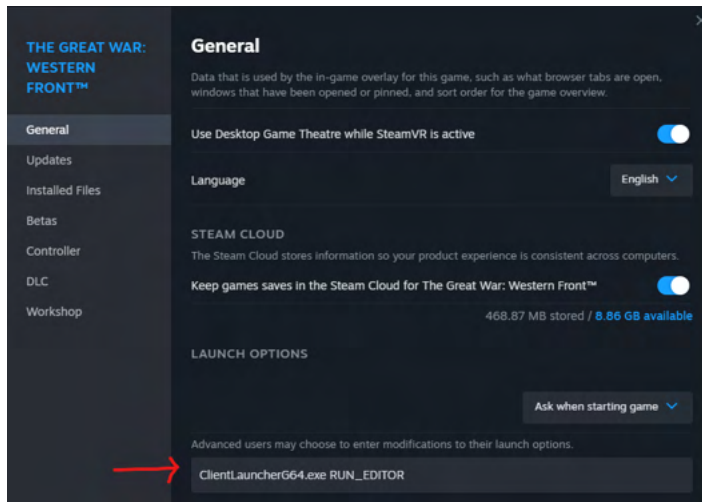
How to Launch the Map Editor

- The Map Editor can be launched one of 2 ways:
 - (Recommended) Through the "Map Editor" Button on the Mod Manager Menu in Game**



- Advantages:
 - Fast and Easy access to both the Map Editor and the game
 - Allows you to have both the game and Map Editor running at the same time.
 - Easier to test your work since can see the map running in the game at the same time while you edit things.
 - Especially useful when creating cinematics and trench layouts.
 - Easy to stop working in the Map Editor and return to playing the game.
- Disadvantages:
 - This method has the Game and the Map Editor running at the same time.
 - Can be PC intensive (although can close the game and leave map editor still running)
 - Exiting the game while Map Editor is up will not allow you to restart the game until close the Map Editor.

- **Modify Launch Options via Steam Library Game Properties**



- While in your Steam Library, right click on **The Great War: Western Front** Click *Properties* Open a dialog
 - Enter the following under launch options: ClientLauncherG64.exe RUN_EDITOR
 - Close dialog and Launch the Game via clicking Play in steam. Instead of the Game starting, the Map Editor will start.
- Advantages:
 - Able to launch directly into the Map Editor without running the game first.
 - Can only have the Map Editor running, so less PC intensive.
- Disadvantages:
 - You are unable to run the game until you close the Map Editor and remove the launch options.

- Initial File Set Up
 - Creating a new (.ted) file: MyNewMap_01_00.ted
 - Creating a new (.sob) file: MyNewMap_01_00_ALLIED_CTRL.sob
- Adding Functionality to MyNewMap_01_00_ALLIED_CTRL.sob
 - Setting up Start Markers for the Command Trenches
 - Visualizing where the Command Trenches should go
 - Moving the Start Markers
 - Setting Up Control Points

- Visualizing where to place Control Points
 - Placing Control Point Objects around the Map
- Creating and Adding Functionality for our Main SOB: MyNewMap_01_00_SHARED.sob
 - Create New (MAIN SOB): MyNewMap_01_00_SHARED.sob
 - Loading the Proper PostFX
- MyNewMap_01: Adding Functionality to MyNewMap_01_00_SHARED.sob
 - Setting Up MapSpawnZone Passability
 - Visualizing where to draw MapSpawnZone Passability
 - Drawing MapSpawnZone Passability
 - Setting Up Spawn Points
 - Visualizing where to place the Spawn Point Objects
 - Placing Spawn Point Objects around the map
 - Setting Up Regions
 - Visualizing where to paint the Regions
 - Creating and Painting the Regions
 - Creating a new (.tga) file: MyNewMap_01_00.tga
 - Visualizing Game Elements for Mini Map Creation
 - Creating the Mini Map
- Moving Files from Map Editor directory to our Mod directory
- Seeing your new map in game

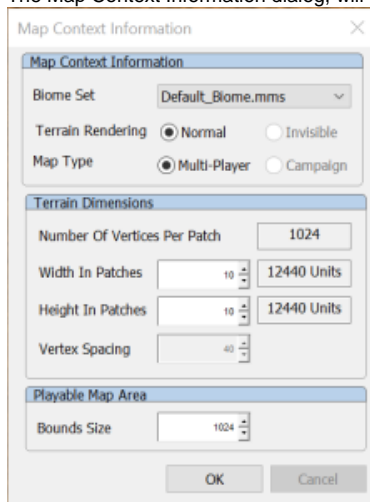
Initial File Set Up

Creating a new (.ted) file: MyNewMap_01_00.ted

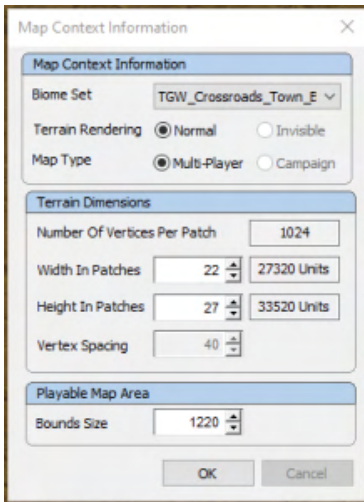
File Type	Brief Description
.ted	Saves the visual aspects of our Map such as Height Map, Terrain Textures, Props, Vertex Color, Splines, Decals, Grass, Water, Environment settings, Lighting, and Post FX data.
.sob	Saves the logical aspects of the map such as start markers, control points, spawners, and destroyable props.
.cpd/.gpd	Saves the passability and region data that is used in conjunction with its matching .sob
.tga	Generates a map preview that makes up the Players Minimap and is used in the other UI locations.

Our first step is to create and save a new ted file.

- Click: Menu Bar File **New TED File**
- Clicking **New TED File** will bring up the following **Map Context Information Dialogue**.
 - The Map Context Information dialog, will have you define 3 different settings for your new map



- Biome Set: Initial map settings which can include texture settings, lighting settings, environment settings, and other user-selected map options.
 - Width and Height In Patches: The width and height of your overall map in "patches", a rough unit of measurement.
 - Bounds Size: The size of the surrounding impassible, decorative area which lies on the borders of the map. This area cannot be traversed by any unit type and exists to allow visual continuity.
- For this Tutorial we will be making a 22x27 map, using the TGW_Crossroad_Town_Biome with a Bounds Size of 1220.



- Enter the following settings into the **Map Context Information Dialogue** and click Ok.
 - Biome: (Using Dropdown): TGW_Crossroad_Town_Biome
 - Width in Patches: 22
 - Height In Patches: 27
 - Bounds Size: 1220
- Click "OK" once the selections look like the ones below and click ok on the warning.
 - Upon hitting "OK" you will receive a warning that reads "You will have a non-square map after this operation...", please feel free to ignore and click "OK".
- The map will update to your new settings and refresh to include two **MARKER_PLAYER_START** objects, which we will cover in the next step but first let's save our new .ted file.



- Click Menu Bar **File Save TED file as...**
 - In the Pop-Up Dialogue, name the file: **MyNewMap_01_00** and hit the **Save button**.
 - Upon hitting "OK" you might receive a warning that reads "The SOB has not been saved..." ignore it for now as this will be the next section.

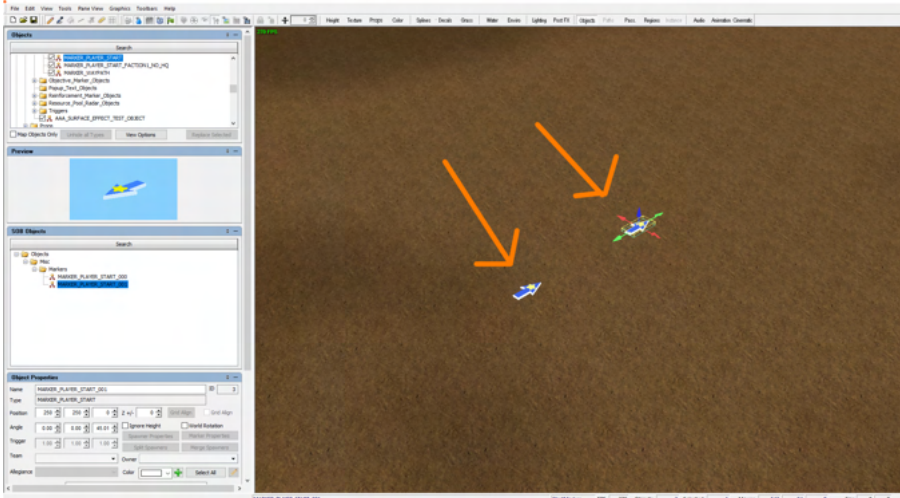
NOTE

- Remember that for this tutorial, it is important we stick to the same naming convention that we followed when setting up the Data for our MOD in the previous tutorial: [01: Setting up and running your Skirmish Map Mod](#).
 - **OTHERWISE** we will have to update that data (especially the XML hookups) with any name changes.
- NOTE: Map Editor Data (saved and auto generated) needs to be stored in the `..\Documents\Petroglyph\TheGreatWar\Custom_Maps` folder
 - **HOWEVER**, mod content runs out of their corresponding Mod folder location (`C:\Users\USER_NAME\Documents\Petroglyph\TheGreatWar\Mods`) which we created in the previous step.
 - Since Map Editor generates and uses data only in the correct location, we will handle moving it over to the mod at the end of tutorial.
- For more information on Map Sizes or the available Biomes please see the **Map Anatomy and Sizes** in the [Map Editor](#) section.

Creating a new (.sob) file: MyNewMap_01_00_ALLIED_CTRL.sob

File Type	Brief Description
.ted	Saves the visual aspects of our Map such as: Height Map, Terrain Textures, Props, Vertex Color, Splines, Decals, Grass, Water, Environment settings, Lighting, and Post FX data.
.sob	Saves the logical aspects of the map for Objects and Decorations.
.cpd/.gpd	Saves the Passability and Region data that is used in conjunction with its respective .sob
.tga	Generates a map preview that makes up the Players Minimap.

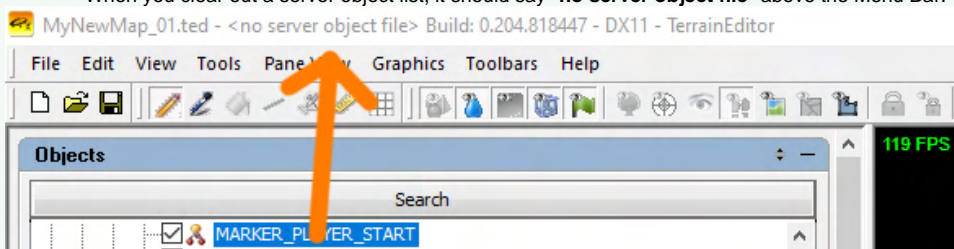
The first-time a .ted is created in the Map Editor, **MARKER_PLAYER_START** object types will automatically be added to the map. These objects (or **Start Markers**) denote the location where each Player's Command Trenched will spawn. Since these objects are part of the support .sob, we will save and create that first.



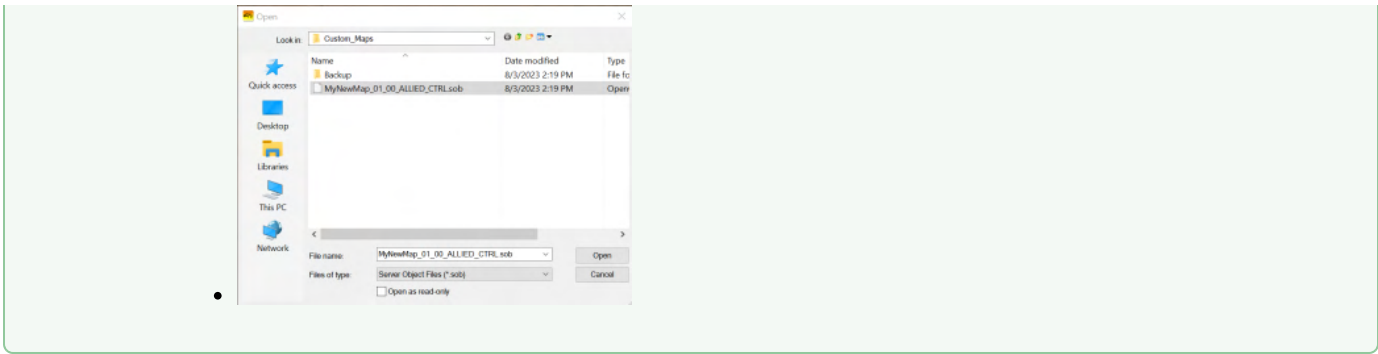
- Click: Menu Bar **File Save Server Object List As...**
 - In the Pop-Up Save Dialogue, name the file: **MyNewMap_01_00_ALLIED_CTRL.sob** and hit **Save button**.
 - Since this **SPECIFIC SOB** will only hold Start Markers and Control Points (and will have the Allied CTRL points at the bottom) we use **_ALLIED_CTRL** to name it.

NOTE

- Note: We will be saving .SOB files throughout this tutorial (and swapping between them) and as a general rule, whenever changing between sob's it is important that we first **CLEAR OUT ALL** current .sob data first
 - Not doing so could cause time consuming issues in the .sob that's being loaded in, such as object names changing and region and passability data being messed up.
 - This should even be done when opening a previously saved version of the same sob that is already loaded.
- To Clear an existing SOB: Click: Menu Bar **File Clear Server Object List**
 - When you clear out a server object list, it should say "**no server object file**" above the Menu Bar.



- Feel free to try this now, and then just load the sob **MyNewMap_01_00_ALLIED_CTRL.sob** again:
 - Click: Menu Bar **File Load Server Object List**
 - Make sure you click **Load Server Object List** and not **Load Read-Only Server Object List**
 - And then select your sob in the Pop-Up Dialogue and click **OPEN**




Adding Functionality to MyNewMap_01_00_ALLIED_CTRL.sob

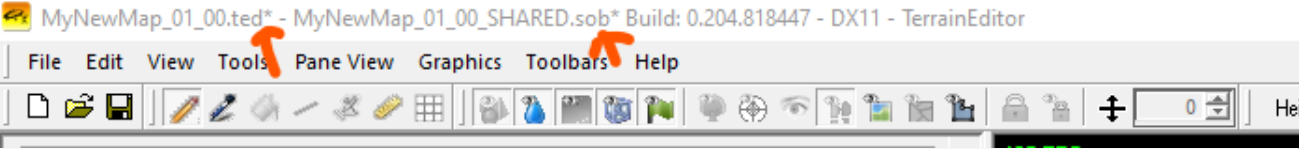
 MyNewMap_01_00.ted - MyNewMap_01_00_ALLIED_CTRL.sob

File Edit View Tools Pane View Graphics Toolbars Help

Now that we have our **MyNewMap_01_00.ted** and **MyNewMap_01_00_ALLIED_CTRL.sob** saved, we can start creating the gameplay.

 **NOTE**

- If the system detects changes and you have yet to save the files, the name of the files at the top of the screen will have an asterix above the name.

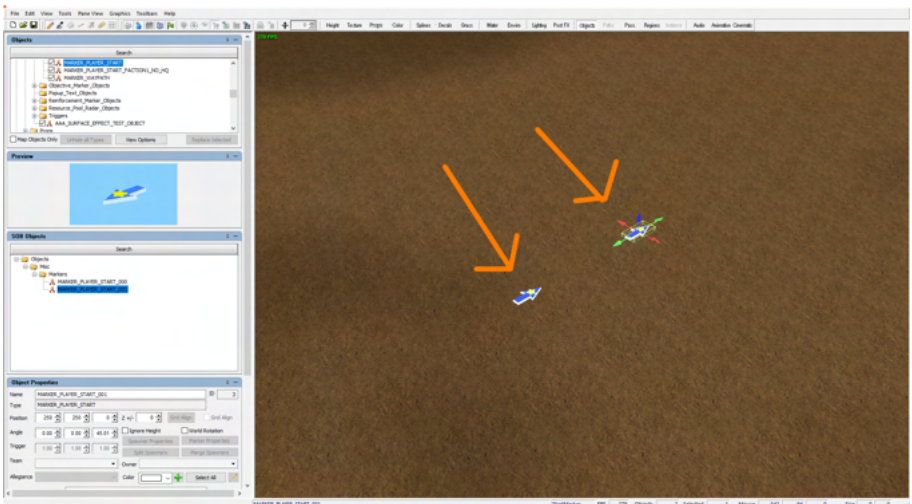


MyNewMap_01_00.ted* - MyNewMap_01_00_SHARED.sob* Build: 0.204.818447 - DX11 - TerrainEditor

File Edit View Tools Pane View Graphics Toolbars Help

- The commonly used hot key Control+ S will only save the .ted file and not the .sob file so recommend using the Options under File.

Setting up Start Markers for the Command Trenches



- As mentioned before when a .ted is created, two start markers are automatically generated and placed in the Map Editor (represented by the blue arrows objects).
 - The locations of these Start Markers will denote the position of both players Command Trench.
 - For Skirmish and Campaign Maps, **MARKER_PLAYER_START_000** should **ALWAYS** be placed in the **South**, while **MARKER_PLAYER_START_001** should always be placed in the **North**.
 - Historical Maps swap locations depending on what side the player is on, with the player using **MARKER_PLAYER_START_000** and the AI using **MARKER_PLAYER_START_001**.

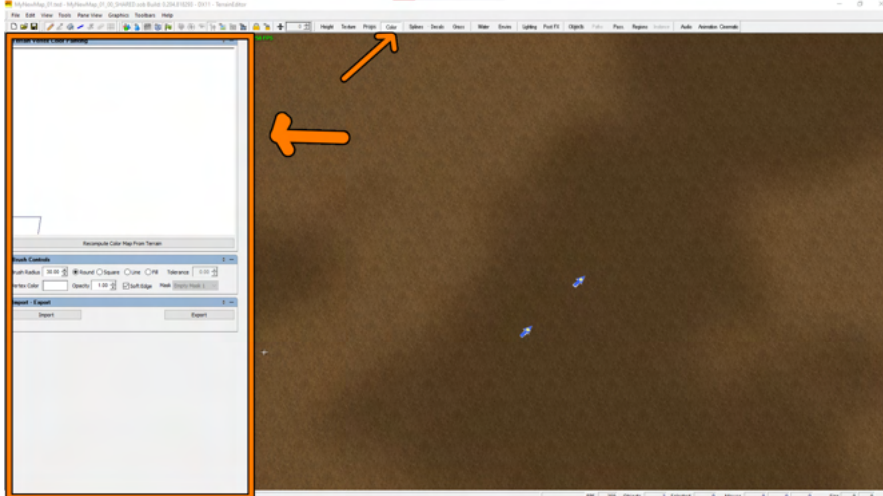
- For most objects in the map editor, we would just select and move the markers, but since we are dealing with trench and trench connectivity, we need exact coordinates when placing Start Markers since they will bring in their corresponding Command Trenches.

NOTE

- In order for Trenches of any kind (including Start Markers and Command Trenches) to work properly, they must be placed in the exact spot on the triangle grid. This means that just "eyeballing" it into place is not recommended as it could prevent them from attaching to other trenches and objects in game properly.
 - The locations are normally figured out by use the **Trench Configurator** in game via the debug tools (See [Lua Debug Tools](#)) to get the exact coordinates.
 - However, for this tutorial the coordinates of the Command Trenches will be provided for you in order to save time.

Visualizing where the Command Trenches should go

- Switch over to the **Color Tab** via Main Tool Bar Editor Tool Bar **Color Tab**
- When done correctly you should see the **Editor Panes** windows change to something like the image below:



NOTE

Color Tab

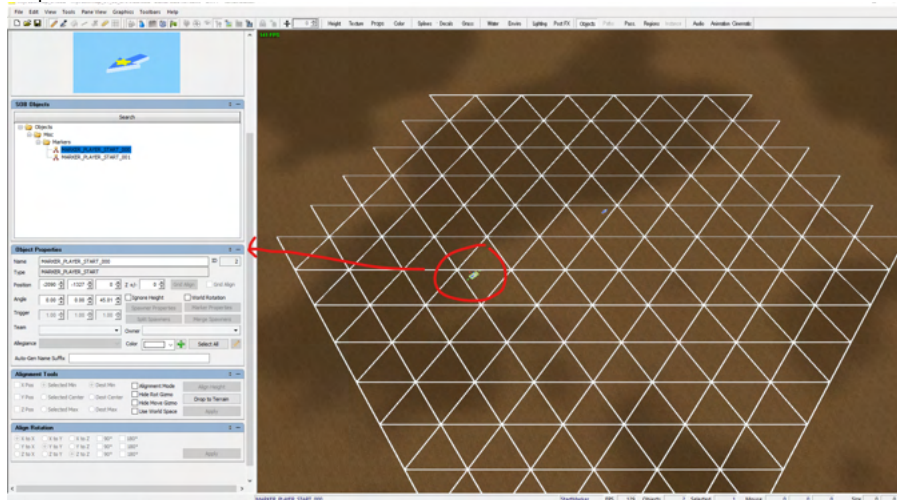
- The Color Tab is generally used as a means to vertex paint the Terrain, adding visual interest to areas and breaking up the "tiled" look of the terrain or to add fake shadows under large objects.
- **HOWEVER**, for this tutorial we will be using the ability to **Import** .tga files as vertex color in order to bring in "visual guides", in order to make it easier to convey instructions.
- Guides will be found in a sub directory of where you have the game installed via Steam, and each will be called out during its respective steps and can be loaded using the **Import** button:
 - **\\TerrainEditorData\\ModSampleFiles\\MyNewMap_01_00**
 - Remember to get the full path: while in your Steam Library, right click on **The Great War: Western Front** Click **Properties** Open a dialog Click **Installed Files** Click **Browse**
 - This will open a window showing the installed game's file structure:
 - Probably a variation of C:\Program Files (x86)\Steam\steamapps\common\TheGreatWar\TerrainEditorData\ModSampleFiles\MyNewMap_01_00\

- Use the Import Button to bring up the Open dialogue.
 - Navigate to where Steam has the game installed (See above note for details) and **\\TheGreatWar\TerrainEditorData\ModSampleFiles\MyNewMap_01_00**
 - Select **Color_Trenches.tga** and click **OPEN**
- Once done loading, you will see images of the Allies and Central Powers flags drawn on the ground as rough guidelines.

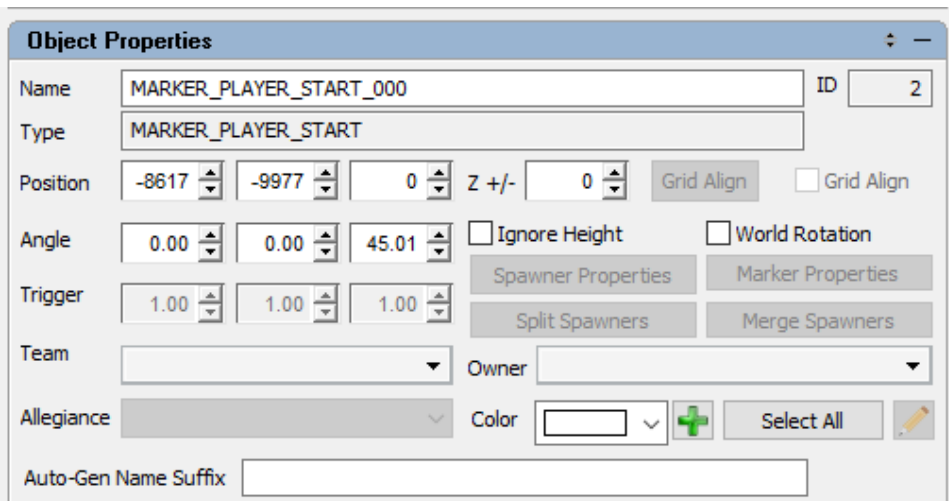
Moving the Start Markers

- Start Markers are Server Objects. Therefore, in order to Move them, you must open the Objects Tab
 - **Main Tool Bar Editor Tool Bar Objects Tab**
- Once in the Objects Tab, it is possible to select the objects via Mouse click.

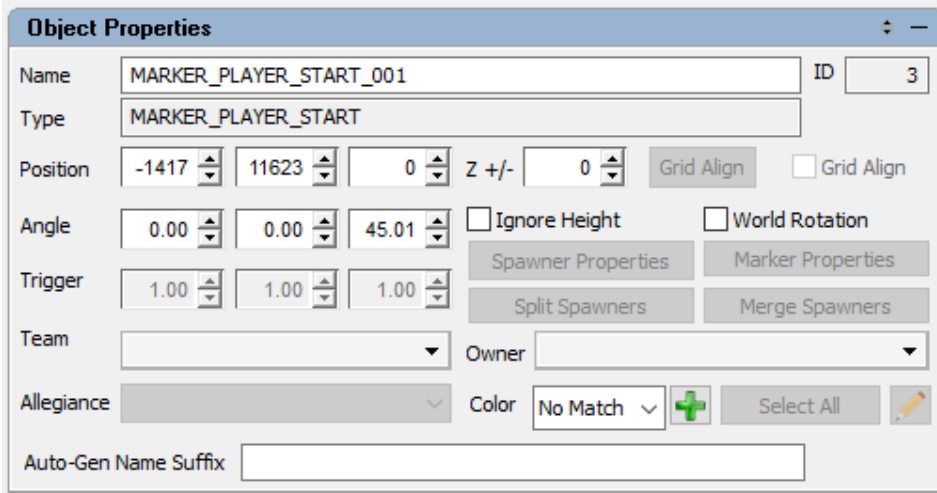
- Note: If you moved the camera, you can also directly select the objectives from the Editor Pane via Expanding the Folder View under SOB Objects
- Once **MARKER_PLAYER_START_000** is selected notice how the Editor Panes section of the screen (See Map Editor Anatomy in [Map Editor](#)) will fill up with information pertinent to that object.
- Now, although the Start Marker object can be moved via a Left Click + Hold + drag, we will move it directly by using the Objects Properties Pane.



- Using the Objects Properties Pane, make sure you have the Object named **MARKER_PLAYER_START_000** and change **Position** and **Angle** to the following:
 - Values are updated via clicking in the window and typing directly and clicking enter (or selecting something else)
 - Angle should already be by 45.01 but make sure.



- Then repeat the same process, Select the **MARKER_PLAYER_START_001** and change its **Position** and **Angle** to the following:



- Once both Start Markers are placed in their correct position they should be near the center of the flags.
- Recommend Saving before moving to next step.
 - Menu Bar File **Save SOB MyNewMap_01_00_ALLIED_CTRL.sob**

NOTE

Triangle Placement Grid

- Although "eyeballing" can't be used for trench locations, a good way to visualize and to get an idea of spacing is to turn on the Triangle Placement Grid.
 - Click: Menu Bar View Show/Hide Elements Triangle Placement Grid
- This allows you to toggle on visibility of the grid within the Map Editor in order to make better decisions when blocking out terrain, and painting passability and regions.
 - This is the same grid sizing that is used in game when the players are placing trenches in pre-battle.
 - For basic sizing - manned trenches are 2 segments and communication trenches are 1 segment.
- If you entered the coordinates correctly the Start Markers should be lined up directly on the grid.



- Feel free to toggle this grid when needed, but this tutorial will call it out during its respective steps.

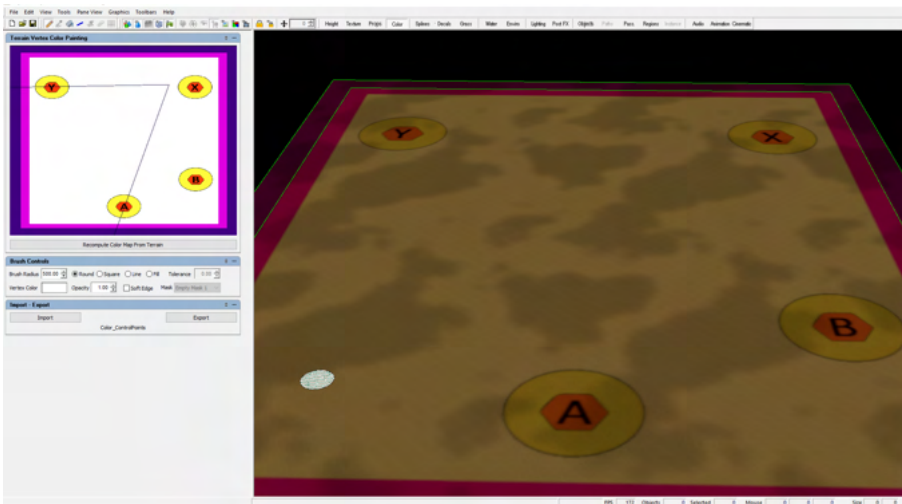
Setting Up Control Points

Now that we have our start markers in the correct locations, we need to add the Control Points to our map.

- While there is a total of 6 different control points available (3 for each of the two factions), you don't need them all of every map.

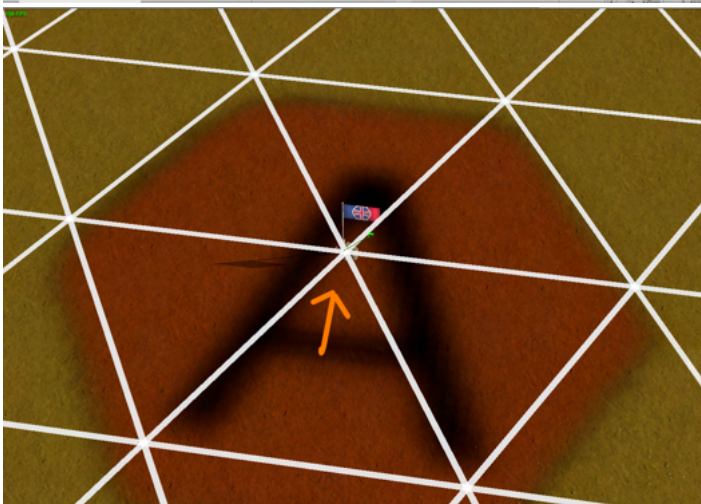
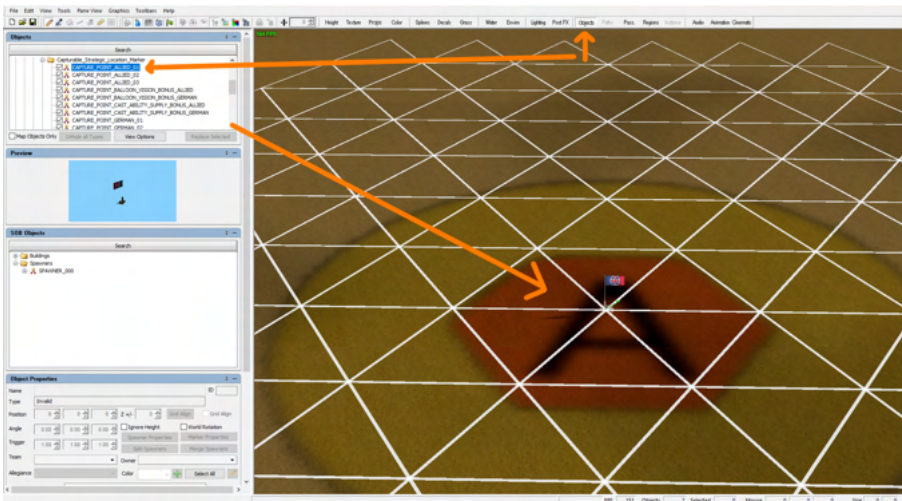
Visualizing where to place Control Points


- Switch over to the **Color Tab** via Main Tool Bar Editor Tool Bar **Color Tab**
- Use the Import Button to bring up the Open dialogue.
 - Navigate to where Steam has the game installed (See above note for details) and `\TheGreatWar\TerrainEditorData\ModSampleFiles\MyNewMap_01_00\`
 - Select **Color_ControlPoints.tga** and click **OPEN**



Placing Control Point Objects around the Map

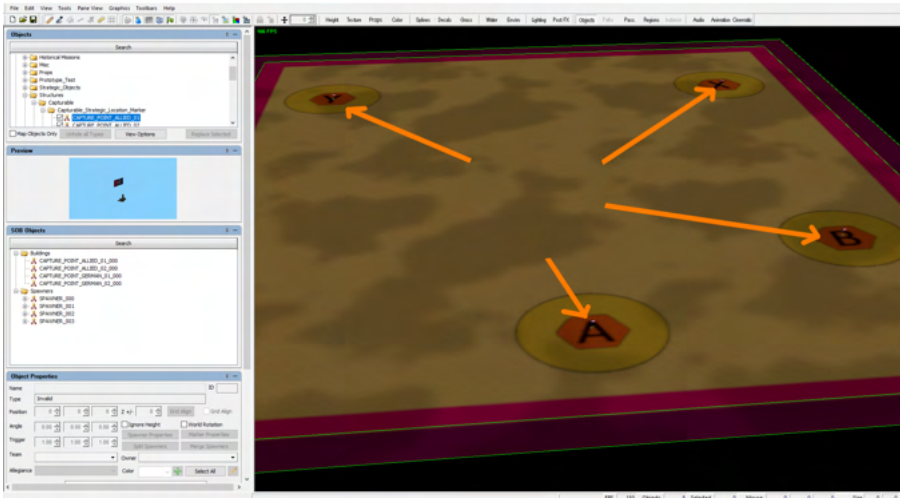
- With the guide in place, we now have an idea as to where to place our first Control Point Object, CAPTURE_POINT_ALLIED_01 (A).
- Switch over to the **Objects Tab** via Main Tool Bar Editor Tool Bar **Objects Tab**
- The first we need to do is locate our object (**CAPTURE_POINT_ALLIED_01**) within the folder trees in the object list and select it via Left-Click
 - Objects Structures Capturable Capturable_Strategic_Location_Marker **CAPTURE_POINT_ALLIED_01**
- Once the Object is Selected, Move the mouse over the location where you would like to place (the triangle intersection closest to the Guide letter (A)) and press Shift + Left Click to place in the 3D world.



 **NOTE**

- Unlike trenches, Control Points don't require the Triangle Grid as rigidly since they don't have any technical reasons they can't be placed anywhere.
- We generally still use the Triangle Grid to see how the player will build trenches around said points.
- It is still preferred to place Command trenches on top of a vertex since this allows control points to work best with player trench placement around it
- In our color grid, you will notice both a yellow circle and an orange/red hex for each control point
 - The circle is help visualize the capture radius for each control point
 - The hex is to visualize the no-build area around the control point if it is placed at the center vertex. (a player can build on the edges)

- Repeat the same process with the following objects:
 - Place **CAPTURE_POINT_ALLIED_02** at (B)
 - Place **CAPTURE_POINT_GERMAN_01** at (X)
 - Place **CAPTURE_POINT_GERMAN_02** at (Y)



With our 4 Control Points added and our command trenches, we now have everything we need on this sob: **MyNewMap_01_00_ALLIED_CTRL.sob**

- Save: Menu Bar File **Save SOB MyNewMap_01_00_ALLIED_CTRL.sob**
- Remember to Clear the SOB before moving on: Menu Bar File **Clear Server Object List**
 - If done correctly, all of the Control Points will unload but you can still see the color map on the .ted file.

NOTE

- Remember: As a general rule, whenever creating a new or opening any existing .sobs, it is important that we first **CLEAR OUT ALL** current .sob data.

Creating and Adding Functionality for our Main SOB: MyNewMap_01_00_SHARED.sob

Create New (MAIN SOB): MyNewMap_01_00_SHARED.sob

- With **MyNewMap_01_00_ALLIED_CTRL.sob** now cleared from the Map Editor (and the object list empty), we can proceed to create and save our "main" .sob.
- Click: Menu Bar File **Save Server Object List As...**
 - In the Pop-Up Save Dialogue, name the file: **MyNewMap_01_00_SHARED.sob** and hit **Save button**.
 - Since this **SPECIFIC SOB** will only hold things like Passability, Region Data, and Spawn Markers we use **_SHARED** to name it.
 - This is a hold-over naming from campaign maps, where we have things like GERMAN_CTRL and ALLIED_CTRL sobs and they "Shared" the same data but had different control point placements.
- Now for our next step we will be loading the proper PostFX data.

Loading the Proper PostFX

- The PostFX Tab will allow you to specify the different graphical settings you wish to utilize on your map. Effects like bloom, color correction, and Depth of Field can be adjusted here.
- Click: Main Tool Bar Editor Tool Bar **PostFX Tab**

- In the Post FX Chains Pane press the **Import Set** Button to bring up the Open dialogue:
 - Navigate to where Steam has the game installed (See above note for details) and `\TheGreatWar\TerrainEditorData\ModSampleFiles\MyNewMap_01_00\`
 - Select **MapFXChain_22x27.pcs** and click **OPEN**
 - If you get a warning, click ok to replace the existing PostFX.



NOTE

- Normally, you would setup PostFX Chains later in the process as one of the last steps of map building but loading the premade chain set will allow us to continue our tutorial with the proper settings (specifically MapEdge).
- The default and provided Post FX Chains are composed of 4 different chains:
 - **Colortest** should be used when you need to see the map clearly as it has most of Post FX turned off.
 - `_Rain/_Summer/_Winter` are required that allows the map to have different "Seasons" in skirmish and campaign.
- Also sometimes having the PostFX on can make the map look too dark to properly work in.
 - This can be changed using the **Enviro Tab**
 - `Main Tool Bar Editor Tool Bar Enviro Tab (Editor Panes section) Environment Set Operations`
 - Change the Post FX Chain drop down to **Color Test**, which has a lot of the Post FX turned off.
- For more info see [11_Post FX Tab](#).

- After loading the correct Post FX in the previous step, we have now finished our set up and are ready to continue adding functionality to our new map.
- At this point we make sure we save our current .ted and .sob before proceeding.
 - Menu Bar File **Save SOB MyNewMap_01_00_SHARED.sob**
 - Menu Bar File **Save MyNewMap_01.ted**

MyNewMap_01: Adding Functionality to MyNewMap_01_00_SHARED.sob

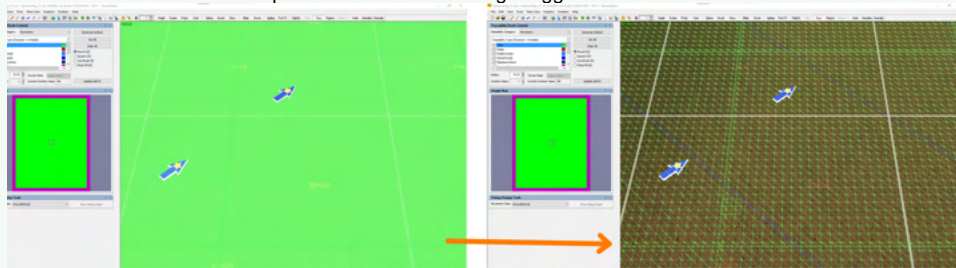
SKIRMISH MAP	STATUS
MyNewMap_01_00.ted	
MyNewMap_01_00_SHARED.sob (USE AS MAIN .cpd/.gpd)	CREATED
MyNewMap_01_00_ALLIED_CTRL.sob (INCLUDE .cpd/.gpd)	
MyNewMap_01_00.tga	NEED



NOTE

Toggle Solid Cell

- Using certain Tabs (like the **Passability and Region Tabs**) will make it very difficult to actually see the terrain, let alone the visual guide we load in.
- Toggle Solid Cell can be used to see through the grid making it easier to see the terrain.
 - Click: Menu Bar Graphics View Terrain Debug Toggle Solid Cell



- Usually, we will toggle the grid to allow us to see through it, but sometime is easier to toggle between the modes when actually painting Regions and Passability.

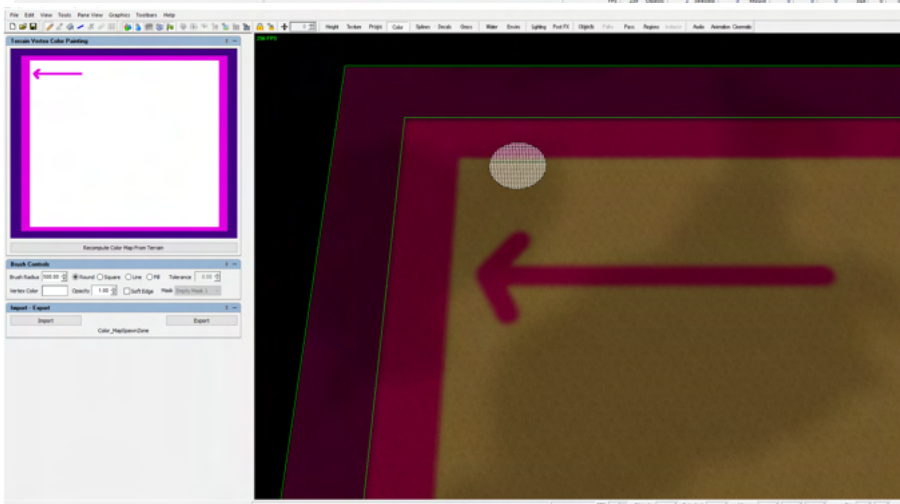
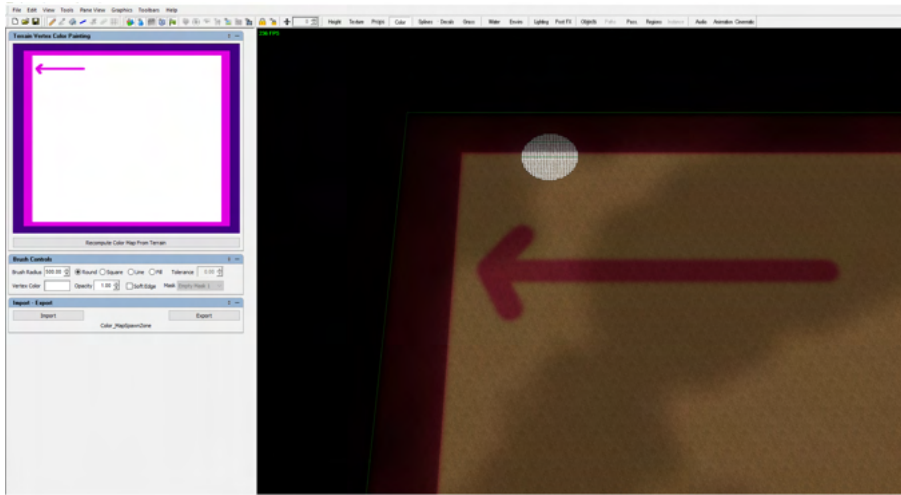
Setting Up MapSpawnZone Passability

- The **MapSpawnZone Passability** is an inner section of playable terrain which allows reinforcements (via the reserves system or script) to be spawned in properly and serves multiple purposes.
 - It prevents other units from entering this map area and using it to traverse the map in actual gameplay.

- It also protects the spawning Company by blocking targeting of them while they are in area.

Visualizing where to draw MapSpawnZone Passability

- Switch over to the **Color Tab** via Main Tool Bar Editor Tool Bar **Color Tab**
- Use the Import Button to bring up the Open dialogue.
 - Navigate to where Steam has the game installed (See above note for details) and `\TheGreatWar\TerrainEditorData\ModSampleFiles\MyNewMap_01_00\`
 - Select `Color_MapSpawnZone.tga` and click **OPEN**

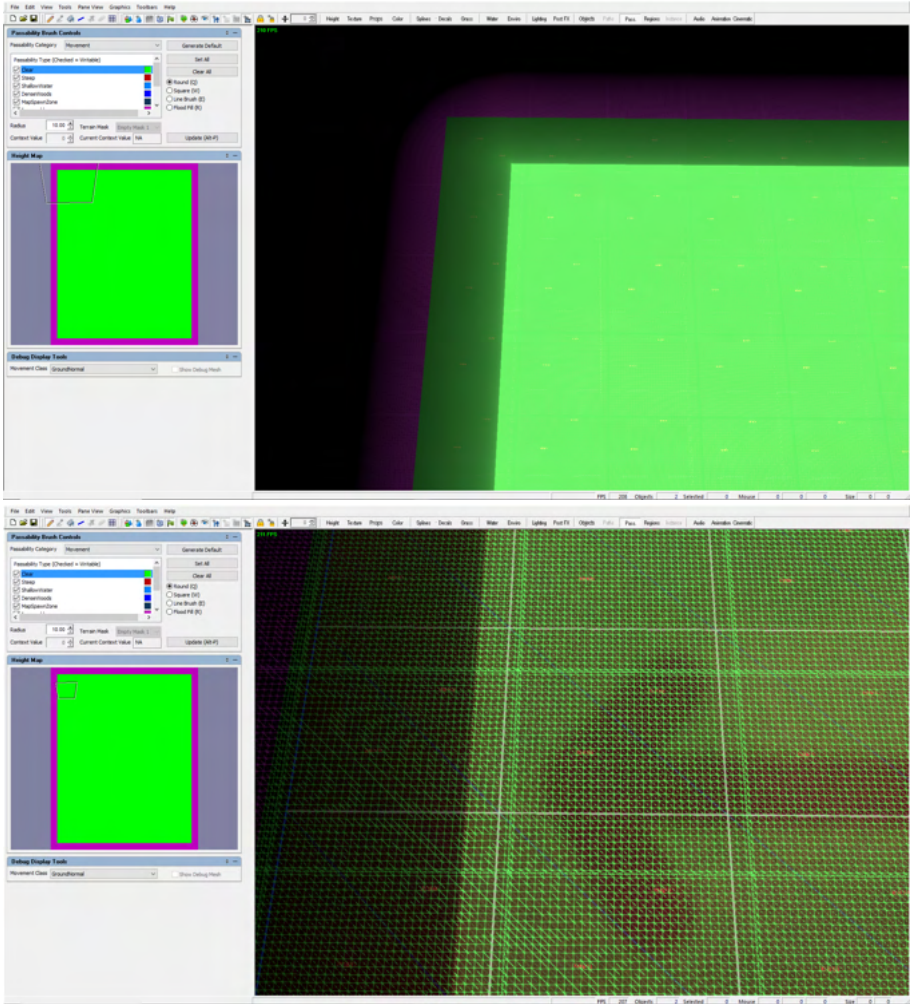


NOTE

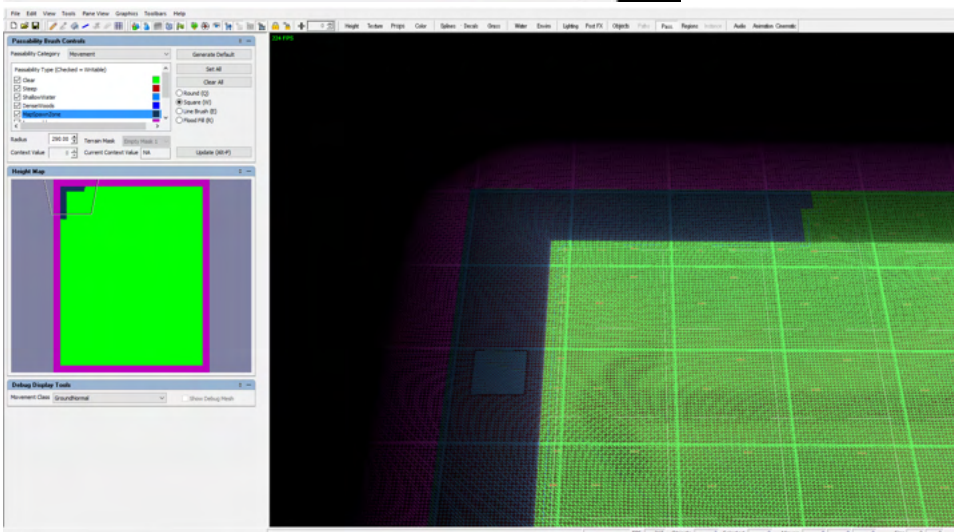
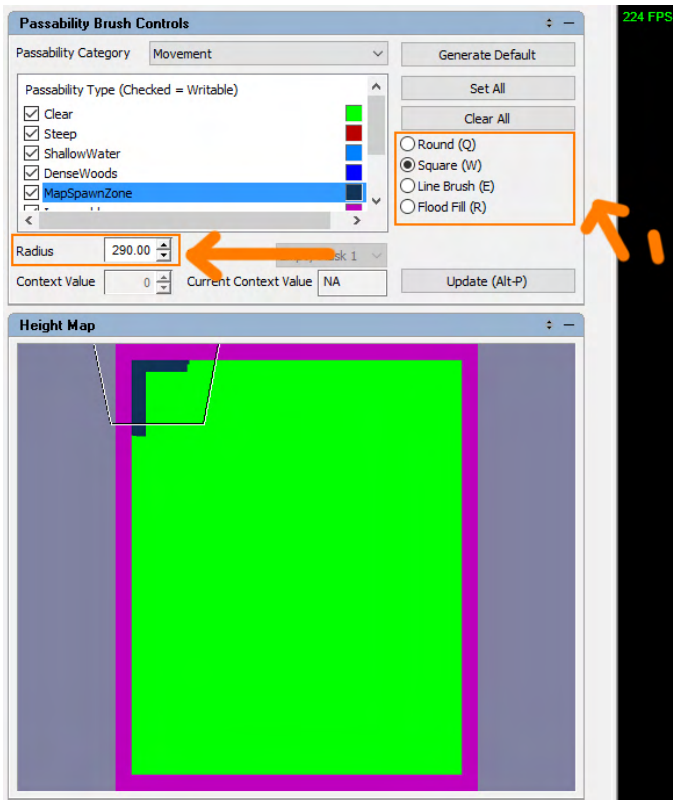
- In game, the MapSpawnZone is generally hidden underneath the MapEdge which is defined and can be changed in PostFX Post FX Editor MapEdge
 - See [11_Post FX Tab](#) for more info of Making Post FX.
 - In the above image on the left, you can see the MapSpawnZone hidden underneath the Map Edge, while in the image on the right, you can see it with the Map Edge hidden.

Drawing MapSpawnZone Passability

- With the guide in place, we now have an idea as to where to draw the MapSpawnZone Passability.
- Switch over to the **Pass Tab**
 - Main Tool Bar Editor Tool Bar **Pass Tab**
- Once in the Pass Tab, it is possible to "paint" various types of Passabilities with the mouse.
 - If can't see the guide, remember to "Toggle Solid Cell" via Menu Bar Graphics View Terrain Debug Toggle Solid Cell



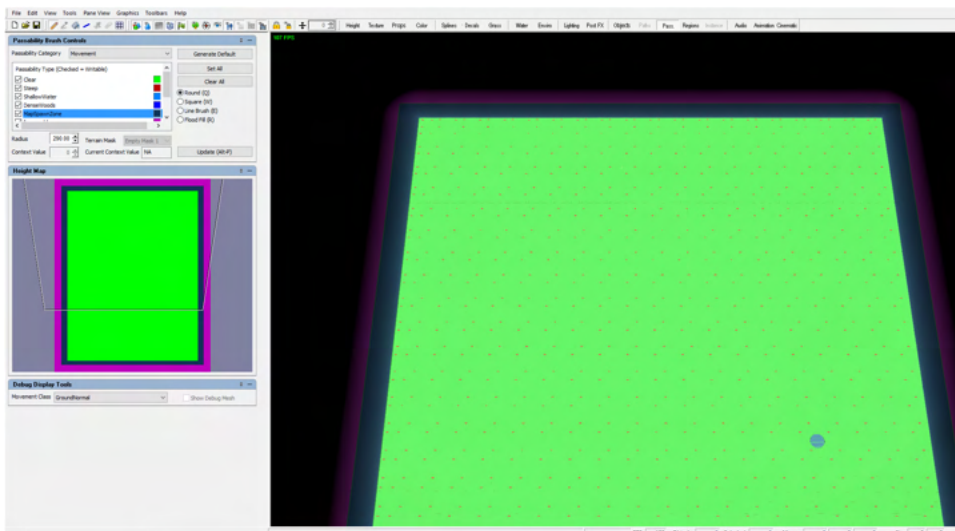
- For now, left click on the MapSpawnZone in the Passability Brush Controls pane to select it.
- From here, we will work with the settings in the Passability Brush Controls in order to paint over our guide using left click to paint.
 - Using the square brush will make this easier to paint (and you can paint "Clear" passability again to fix any mistakes)



NOTE

- When a map .sob is first created it will always have the outer edge of it painted Impassable. This is also known as the Map Skirt and it cannot be removed.
- Since it cannot be removed, you can be more forgiving when painting MapSpawnZone and your brush slips into the impassable paint as it cannot do anything to it, making painting a little less tedious on that side.
- For more information on Pass Tab (See [13_Pass Tab](#)).

- We will continue this process until we have painted the MapSpawnZone Passability all around the outside of our playable space.
- When finished, it should look something like this.



NOTE

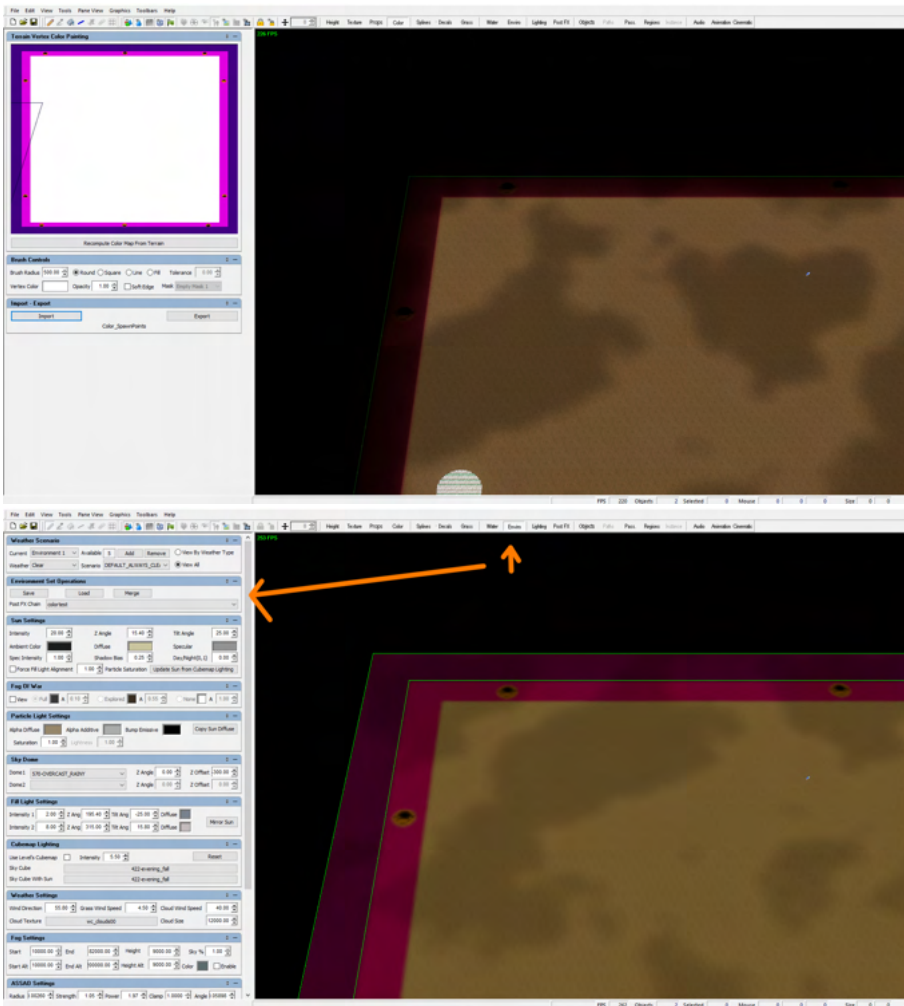
- When creating your own maps, MapSpawnZones are roughly a distance of 1043 units (26 cells) starting where the Impassable Skirt ends (for more on the Map Skirt See Map Anatomy and Sizes in [Map Editor](#)).
- Distances can be measured in the editor via the Ruler Mode (NOTE: Need to be on object tab for Ruler Button to work)
 - [Main Tool Bar Editor Tool Bar Objects Tab](#)
 - [Main Tool Bar Gadget Tool Bar Ruler Mode](#)
 - Measurements are shown on the left-hand side of the Status Bar (for more info on the Status Bar, see Map Editor Anatomy in [Map Editor](#))

Setting Up Spawn Points

- Now that the MapSpawnZone Passability has been drawn in, we need to place objects called **REINFORCEMENT_SPAWN_POINT_MAGNET_OBJECT**, around the edges of the map (within the MapSpawnZones).
- Recommend Saving your TED and SOB before moving on.

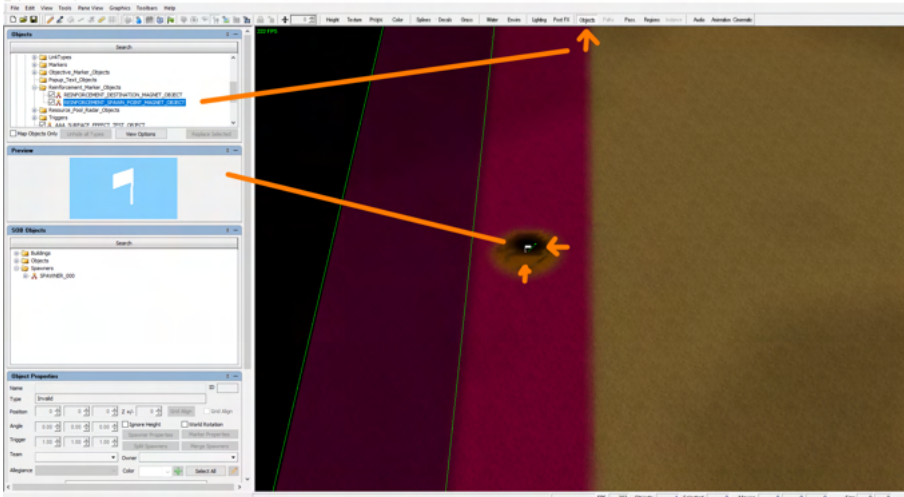
Visualizing where to place the Spawn Point Objects

- Switch over to the **Color Tab** via Main Tool Bar Editor Tool Bar **Color Tab**
- Use the Import Button to bring up the Open dialogue.
 - Navigate to where Steam has the game installed (See above note for details) and `\TheGreatWar\TerrainEditorData\ModSampleFiles\MyNewMap_01_00\`
 - Select **Color_SpawnPoints.tga** and click **OPEN**
- When loaded should look like the images below. (Remember to change the Post FX chain to color test if seeing the left image)

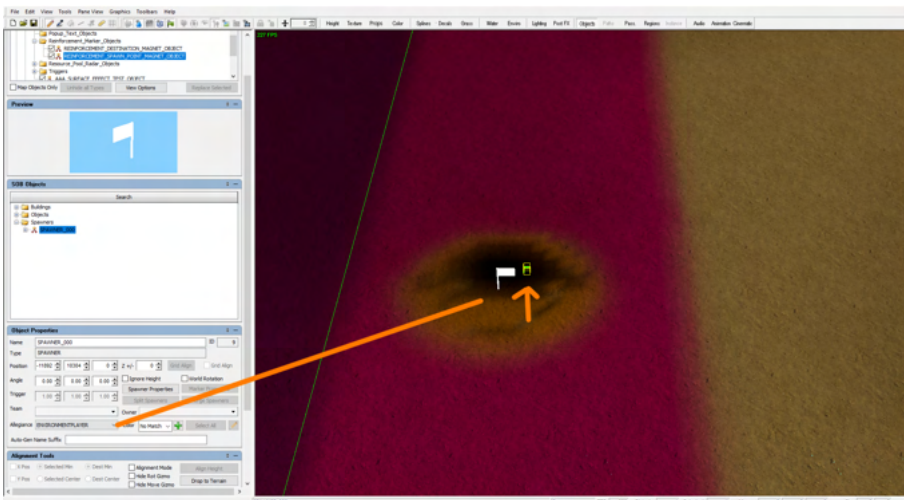


Placing Spawn Point Objects around the map

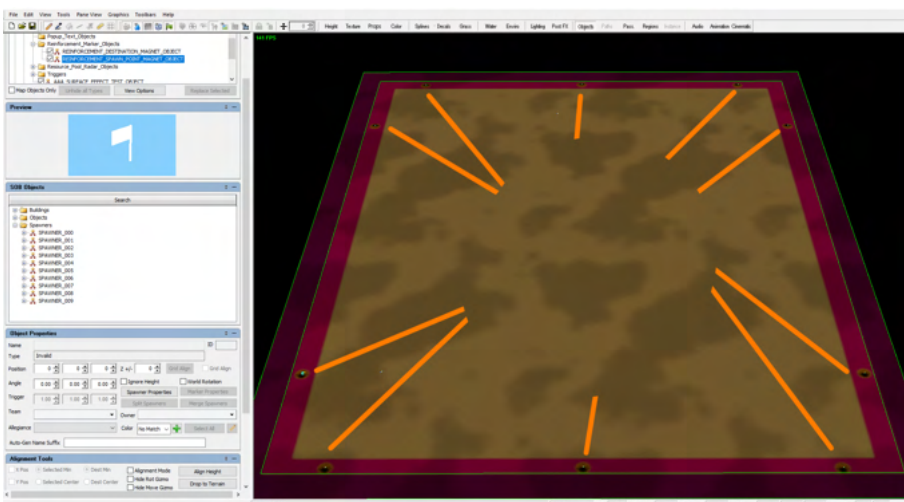
- With the guide in place, we now have an idea as to where to place the **REINFORCEMENT_SPAWN_POINT_MAGNET_OBJECT**.
- Switch over to the **Objects Tab**
 - Main Tool Bar Editor Tool Bar **Objects Tab**
- Look for and Select the following object **REINFORCEMENT_SPAWN_POINT_MAGNET_OBJECT** within the folder trees located in the Objects Pane.
 - Objects Misc Reinforcement_Marker_Objects
 - Make sure you select the **REINFORCEMENT_SPAWN_POINT_MAGNET_OBJECT** (white flag) and NOT the **REINFORCEMENT_DES_TINATION_MAGNET_OBJECT** (black flag)
- Once the Object is selected, move the mouse over the location where you would like to place (the center of the orange icons on the guide) and press **Shift + Left Click** to place in the 3D world.



- Once placed note how the **REINFORCEMENT_SPAWN_POINT_MAGNET_OBJECT** has a small green flag attached to it. This is called a **Spawner Flag**.
- In order for our objects to work properly, we need to change the Allegiance on them from the Default **HUMANPLAYER** to **ENVIRONMENTPLAYE R**.
 - This is what will allow the Spawn Points to properly swap sides when their respective Control Point is taken.
 - Left Click on the green flag object to select it .
 - In the Editor Pane **Object Properties**, look for Allegiance drop down.
 - Change it from the Default **HUMANPLAYER** to **ENVIRONMENTPLAYE R**



- Once that is done, we will repeat this same set of actions action until all 10 Spawn Points have been placed around the map at the center of the visual guides.



NOTE

- When designing your maps, remember that Spawn Points placed at the top and the bottom of the map, DO NOT switch over when their associated Control Point is taken but those on the left and right side DO switch.
- When trouble shooting spawn points in your mod, forgetting to set the allegiance flag is one of the most common mistakes.
 - Having the point not within the correct region is the second most common.

Setting Up Regions

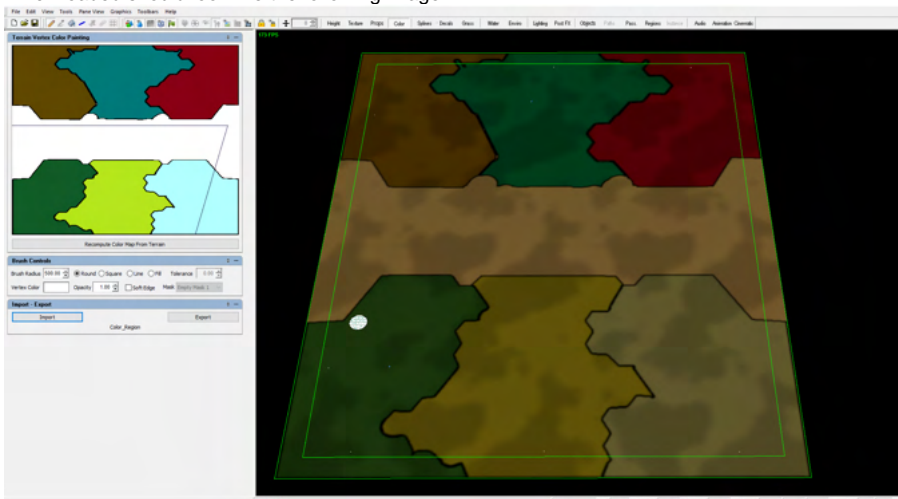
- Now that the MapSpawnZone and the Spawn Objects have been placed, we set up the Regions that will link the Control Points to their Respective Spawn Points.
 - **Furthermore**, Regions also create the valid space where players are able to place their Trenches.
- Recommend Saving your TED and SOB before moving on.

NOTE

- A Trench of any kind requires a "Friendly" Region in order to have valid placement, "friendly" being a region that belongs to a Control Point that player owns.
 - Otherwise, players will not be able to place a trench in that location during pre-battle (or have one created one via script).
- Furthermore in order for a company to move through a series of trenches, trenches in that network must be built on top of a region, otherwise companies will travel outside of the safety of the trench network believing it is no mans land.
 - With that in mind, interesting layouts can be achieved by creating gaps within the Regions, as players will need to consider placement.
- Trench placement is also affected by Passability types so when trouble shooting an area where trench construction should/shouldn't be possible, make sure to check both regions and passability at the location.

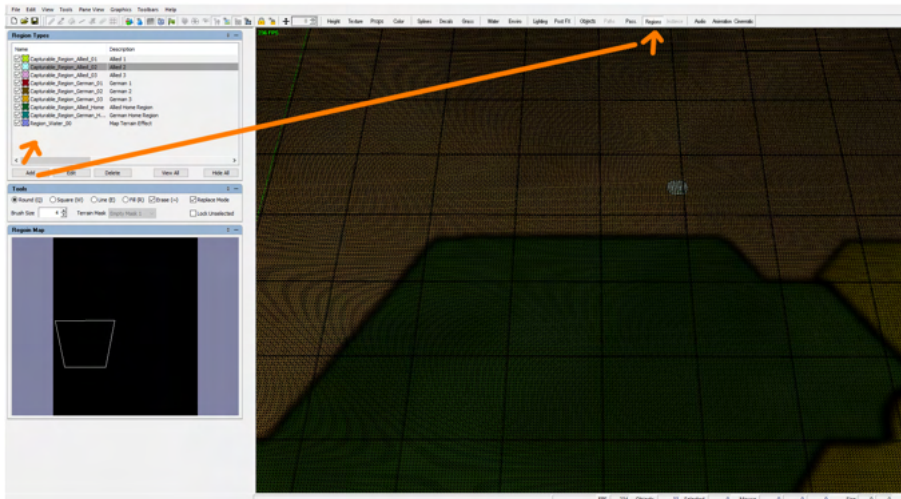
Visualizing where to paint the Regions

- Switch over to the **Color Tab** via Main Tool Bar Editor Tool Bar **Color Tab**
- Use the Import Button to bring up the Open dialogue.
 - Navigate to where Steam has the game installed (See above note for details) and `\TheGreatWar\TerrainEditorData\ModSampleFiles\MyNewMap_01_00\`
 - Select **Color_Region.tga** and click **OPEN**
- When loaded should look like the following image.



Creating and Painting the Regions

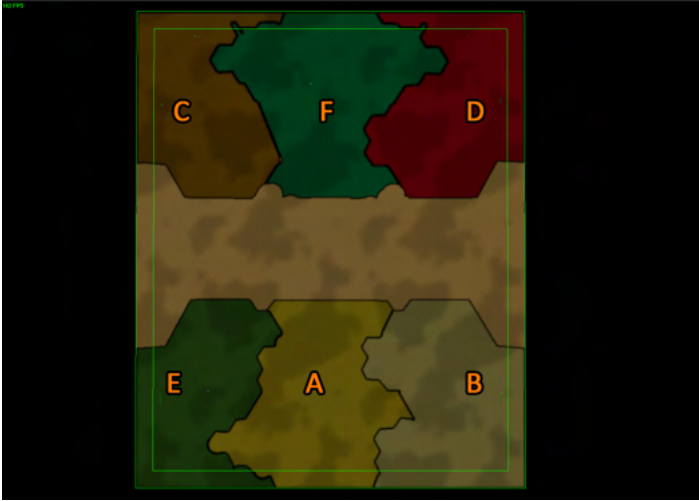
- With the guide in place, we now have an idea as to where to draw each of our six regions.
- Switch over to the **Regions Tab** via Main Tool Bar Editor Tool Bar **Regions Tab**
- Once in the Regions Tab, it is possible to "paint" with the mouse (similar to how we painted passability), however we will need to first create Region Types by naming them.
 - Editor Panes (See Map Editor Anatomy in [Map Editor](#)) Region Types.
- Using the Table below and the Add button in the Region Types Pane, we begin to create our new Region Types.



NOTE

- When it comes to the "Name" of the Region Type it **HAS TO BE EXACT** as it is spelled on the chart below. In game the given "Name" is what links it to its corresponding .xml file and a typo or different name would break that link.
- **FURTHERMORE**, it is recommended that you create them in the same order as the chart below, since it will make the tutorial easier, as color is assigned by the order of when the region type is created.
 - It is also recommended adding all of them to every map, even if you originally intend to now have 3 control points per side.

Name	Description	Key	NOTES
Capturable_Region_Allied_01	Allied 1	A	Used Primarily on the South side of the map, in skirmish associated with Object type CAPTURE_POINT_ALLIED_01 (A)
Capturable_Region_Allied_02	Allied 2	B	Used Primarily on the South side of the map, in skirmish associated with Object type CAPTURE_POINT_ALLIED_02 (B)
Capturable_Region_Allied_03	Allied 3	N/A	In skirmish associated Object type CAPTURE_POINT_ALLIED_03 (C), but not necessary for this map.
Capturable_Region_German_01	German 1	C	Used Primarily on the North side of the map, in skirmish associated with Object type CAPTURE_POINT_GERMAN_01 (X)
Capturable_Region_German_02	German 2	D	Used Primarily on the North side of the map, in skirmish associated with Object type CAPTURE_POINT_GERMAN_02 (Y)
Capturable_Region_German_03	German 3	N/A	In skirmish associated Object type CAPTURE_POINT_GERMAN_03 (Z), but not necessary for this map.
Capturable_Region_Allied_Home	Allied Home Region	E	Used Primarily on the South side of the map, in skirmish associated with Object type STRUCTURE_FACTION1_HQ_00_SPAWN_OBJECT (Allied Command Trench)
Capturable_Region_German_Home	German Home Region	F	Used Primarily on the North side of the map, in skirmish associated with Object type STRUCTURE_FACTION2_HQ_00_SPAWN_OBJECT (Central Powers Command Trench)
Region_Water_00	Map Terrain Effect: Slow Company	N/A	Used to denote the "shallow" water sections of the map that slow down Companies moving through them. Addressed later in the tutorial.



- With all of the regions created, it is now time to actually paint the regions.
- In order to paint, select a type in the Region Types Pane and in the **Tools Pane** ensure that the **Replace Mode** is checked.
 - If can't see the guide, remember to "Toggle Solid Cell" via Menu Bar Graphics View Terrain Debug Toggle Solid Cell

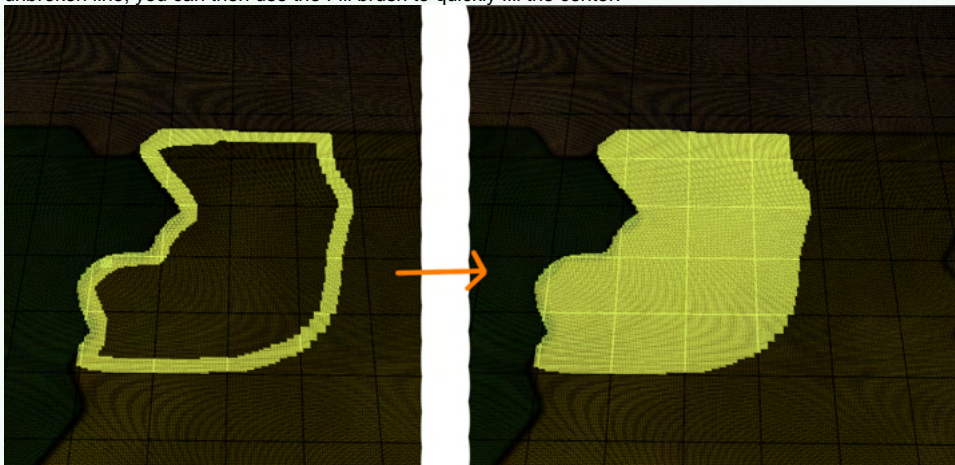
✔ NOTE

- Although it is possible to have **Region Types** overlap and exist at the same location, for this product we avoid this, since there is no visual element to convey the overlap to the player and it could create some strange confusing behavior with the control points and spawn points tech.
 - ALWAYS use **Replaced Mode** when painting, as it will replace any Region type already there with the one currently being used to paint.
- For more information on how to use the Region Tab See: [14_Regions Tab](#)

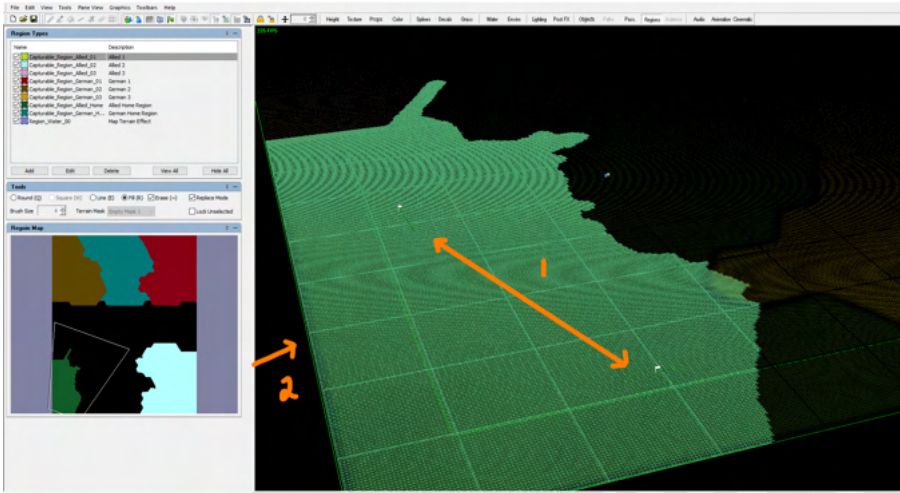
- After selecting the Region, you can use left click to paint the region (similar to have you painted regions).
 - Mistakes can be fixed either by painting a different region, or via selecting erase and painting over the area.

✔ NOTE

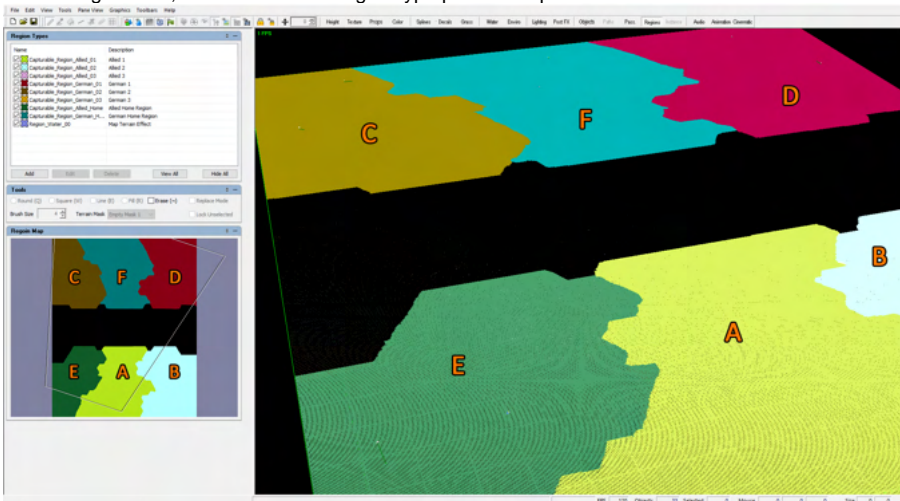
- The process of Painting Regions can be expedited a bit by careful use of the Fill brush. Essentially, as long as you can create an unbroken line, you can then use the Fill brush to quickly fill the center.



- Regions are important to spawn points as they create the link that lets them know who owns them.
 - As seen in the image below, we need to ensure Spawn Points are sitting on top of a Region Type (1).
 - Also, its best practice to paint Region Type ALL the way to the edge of the Map (2).



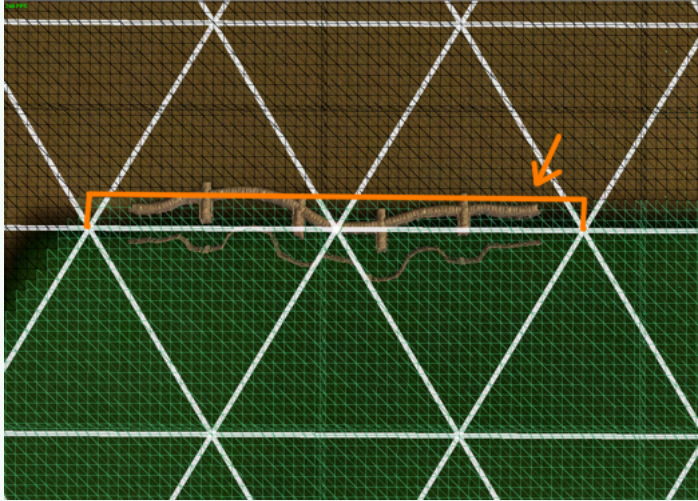
- Continue with this process until all 6 Region Types are painted.
- In the image below, we can see each region type painted as per the chart above.



NOTE

Painting regions is an extremely iterative process, so don't expect to get it perfect the first time. It's best practice to block it in and through various playthroughs refine it in order to give the player an interesting play space to build their trench layouts.

- When a Region is next to No Man's Land or a Dense Forest, it's important to use the grid to paint beyond what would be considered the center of the trench (as seen in the image below) **IF** you want the player to be able to place a trench at that location.
- In the image below, we have a firing trench, which is generally two triangles long. Since we want the player to place a trench facing into No Man's Land, we paint our region slightly beyond what would be the center of the trench. Without this, the player would have to place their trench on the line below. Also remember that valid placement of a trench is the midpoint of the trench.



With our map edge, spawn points, and regions painted, we now have everything we need on this sob: **MyNewMap_01_00_Shared.sob**

- Before we move on to create the Mini Map image, make sure to save your **MyNewMap_01_00.ted** and **MyNewMap_01_00_Shared.sob**

Creating a new (.tga) file: MyNewMap_01_00.tga

With the functionality added to both **MyNewMap_01_00_SHARED.sob** and **MyNewMap_01_00_ALLIED_CTRL.sob** the last thing we need to do is create a Mini Map image (.tga), before moving the files over to our Mod Folder for in game testing.

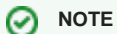
SKIRMISH MAP	STATUS
MyNewMap_01_00.ted	
MyNewMap_01_00_SHARED.sob (USE AS MAIN .cpd/.gpd)	
MyNewMap_01_00_ALLIED_CTRL.sob (INCLUDE .cpd/.gpd)	
MyNewMap_01.tga	NEED

Visualizing Game Elements for Mini Map Creation

- Since our map does not have any visuals on it yet, we are going to rely on our Color tab to provide some boundaries while testing.
- Switch over to the **Color Tab** via Main Tool Bar Editor Tool Bar **Color Tab**
- Use the Import Button to bring up the Open dialogue.
 - Navigate to where Steam has the game installed (See above note for details) and `\TheGreatWar\TerrainEditorData\ModSampleFiles\MyNewMap_01_00\`
 - Select **Color_All_Phase01.tga** and click **OPEN**

Creating the Mini Map

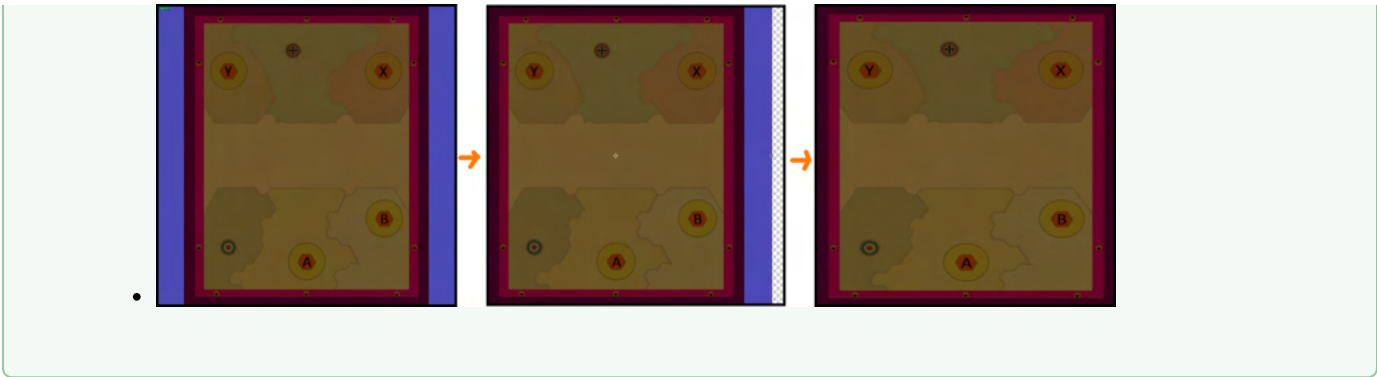
- With our placeholder visuals in place, we can generate our mini map via the **Generate Map Preview Image**
 - Menu Bar Tools Generate Map Preview Image **1024x1024**
 - Make sure "Use Playable Bounds" is unchecked.
- Once its generated, the image can be found here:
 - `C:\Users\USER_DATA\Documents\Petroglyph\TheGreatWar\Custom_Maps\MyNewMap_01.tga`



NOTE

Optionally customizing the Mini Map

- Once the mini map is created, it tends to be "thinner" than normal when run in game.
 - This means the location of Companies moving around the map will be a bit distorted.
- In order to account for this, it's best to take the original image with the purple borders into an image editing program and stretch the image from left to right so as that the "purple" borders are no longer seen (and then save).



- Now with our all of our data created and functionality added, we can finally move our files over to MOD directory from where the Map Editor has been saving files.

SKIRMISH MAP	STATUS
MyNewMap_01_00.ted	
MyNewMap_01_00_SHARED.sob (USE AS MAIN .cpd/.gpd)	
MyNewMap_01_00_ALLIED_CTRL.sob (INCLUDE .cpd/.gpd)	
MyNewMap_01_00.tga	

Moving Files from Map Editor directory to our Mod directory

- This final step of the tutorial builds on the work done in the previous tutorial, since you must have a functional Mod to move the new data to.
 - See [01: Setting up and running your Skirmish Map Mod](#)
 - Before continuing it is recommended you copy your entire skirmish map mod directory elsewhere to have a backup in case so you don't need to redo the entire first tutorial if something is wrong
 - **C:\Users\USER_NAME\Documents\Petroglyph\TheGreatWar\Mods\MyNewMap_01**
- We now need to move all of our new data from the Map Editor folder to the Mod folder we created that the game uses.:
 - Remember the Map Editor saves **ALL** its files at the following location: **C:\Users\USER_NAME\Documents\Petroglyph\TheGreatWar\Custom_Maps**
 - While the game requires the mod data to be located at: **C:\Users\USER_NAME\Documents\Petroglyph\TheGreatWar\Mods**
- The chart below shows where to place the Map Editor Data and move it over to its respective location in our **MyNewMap_01** mod folder.

Map Editor Data	Mod Data
C: \\Users\USER_NAME\Documents\Petroglyph\TheGreatWar\Custom_Maps	C: \\Users\USER_NAME\Documents\Petroglyph\TheGreatWar\Mods
MyNewMap_01_00.ted	C: \\Users\USER_DATA\Documents\Petroglyph\TheGreatWar\Mods\MyNewMap_01\Data\Art\Maps
MyNewMap_01_00_SHARED.sob	C: \\Users\USER_DATA\Documents\Petroglyph\TheGreatWar\Mods\MyNewMap_01\Data\Art\Maps\ServerObjects
MyNewMap_01_00_SHARED.cpd	C: \\Users\USER_DATA\Documents\Petroglyph\TheGreatWar\Mods\MyNewMap_01\Data\Art\Maps\Passability
MyNewMap_01_00_SHARED.gpd	C: \\Users\USER_DATA\Documents\Petroglyph\TheGreatWar\Mods\MyNewMap_01\Data\Art\Maps\Passability
MyNewMap_01_00_ALLIED_CTRL.sob	C: \\Users\USER_DATA\Documents\Petroglyph\TheGreatWar\Mods\MyNewMap_01\Data\Art\Maps\ServerObjects
MyNewMap_01_00_ALLIED_CTRL.cpd	C: \\Users\USER_DATA\Documents\Petroglyph\TheGreatWar\Mods\MyNewMap_01\Data\Art\Maps\Passability
MyNewMap_01_00_ALLIED_CTRL.gpd	C: \\Users\USER_DATA\Documents\Petroglyph\TheGreatWar\Mods\MyNewMap_01\Data\Art\Maps\Passability

- Once everything is moved over, we can run the game and try it out.

Seeing your new map in game

- Make sure the Mod is loaded via the Mod manager and then you can find your new map in the Skirmish Map UI.



- Select the Map, add some units, and Click Start
- If everything was created correctly (including the lua changes for the previous tutorial), the map will load and present you with the choice about the pre-building trenches



- Since the trench layouts went with the sample map, they won't be perfect - but you will be able to play the map verse the AI.

Congrats and you are ready for the more advanced elements of height map, forests, and water.

03: Advance Map customization for your new Skirmish Map

- The goal of this tutorial is to take the Basic Skirmish Map created in the previous step and add more advanced elements such as Dense Forest and Water Region to the map.
 - It builds on the work done in the previous one (02: Using the Map Editor to customize your new skirmish map) so it is recommend doing that tutorial first.
- When completed, you will have updated your new Skirmish Mod, **My New Skirmish Map 01**

Getting Started

- Since this tutorial will not go into detail for all the Map Editor features, users can familiarize themselves on tools and features via the [Map Editor](#) page, as well as terms which will be used throughout this tutorial.
 - For more information on each editor within the Map Editor, please see the **Editor Tabs** Section in [Map Editor](#) for more in depth details on each one.
- Below is a list of the files we will have already created and should be ready for the next step:

SKIRMISH MAP	STATUS	
MyNewMap_01_00.ted		The visual aspects of our Map such as: Height Map, Terrain Textures, Props, Vertex Color, Splines, Decals, Grass, Water, Environment settings, Lighting, and Post FX data.
MyNewMap_01_00_SHARED.sob (USE AS MAIN .cpd/.gpd)		The "Main" logical aspects of our map such as: Spawn Markers and Destroyable Props, along with all of the Passability and Region data in their corresponding .cpd/.gpd files.
MyNewMap_01_00_ALLIED_CTL.sob (INCLUDE .cpd/.gpd)		"Supporting" logical aspects of our map including Start Markers and Control Points. Although .cpd/.gpd files are created, they are not used for passability and region data.
MyNewMap_01_00.tga		The minimap image created in editor and adjusted in a photo editing program to account for distortion. This is used for the minimap and other UI locations.

- [Adding Advanced Functionality to MyNewMap_01_00_SHARED.sob](#)
 - [Setting Up Pit Passability](#)
 - [Visualizing where to draw Pit Passability](#)
 - [Drawing Pit Passability](#)
 - [Setting Up DenseWoods Passability](#)
 - [Visualizing where to paint the DenseWoods Passability](#)
 - [Drawing Dense Woods Passability](#)
 - [Refining Brush](#)
 - [Setting Up Water Regions](#)
 - [Visualizing Where to paint the regions](#)
 - [Painting Water Regions](#)
 - [Adjusting Terrain Height to visualize our new map elements](#)
 - [Visualizing Height Map for our new elements](#)
 - [Loading a Height Map](#)
 - [Updating the Mini Map image: MyNewMap_01.tga](#)
 - [Visualizing Game Elements for MiniMap Creation](#)
 - [Creating the Mini Map](#)
- [Moving Updated Files from Map Editor directory to our Mod directory](#)
- [Seeing your updated map in game](#)

Adding Advanced Functionality to MyNewMap_01_00_SHARED.sob

- Load .ted: Click "Open TED File" and Select: MyNewMap_01.ted
- Load .sob: Click "Load Server Object List" and Select: MyNewMap_01_00_SHARED.sob



MyNewMap_01_00.ted - MyNewMap_01_00_SHARED.sob

File Edit View Tools Pane View Graphics Toolbars Help

Setting Up Pit Passability

- The **Pit** passability is mainly used when you want to prevent movement to a location of the map but you still want the player to be able to look into it and shoot over it.
 - In TGW this is especially great for visualizing old undermine pits, which make great visual points of interest and if added inside a player's region can add tactical variety to a map, as players can't build or walk through the pit.

NOTE

- Although called **Pit** Passability, it doesn't necessarily have to just be used or actual pits.
 - It is also used cliffs that force players to go around but be careful with the height/angle since both sides can shoot through it with their rifles.
 - It is also used for deeper water that you want players to not be able to enter (or build on), but still see over.
 - Shallow water uses **Clear** passability (so units can move through it) and the **Region_Water_00** as covered below.
 - For more information on Passability see [13_Pass Tab](#).

Visualizing where to draw Pit Passability

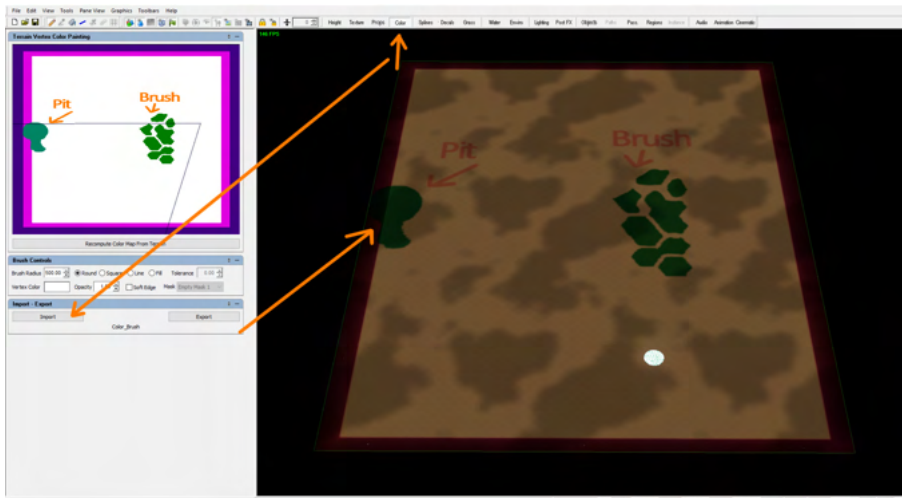
- Switch over to the **Color Tab** via Main Tool Bar Editor Tool Bar **Color Tab**

NOTE

Color Tab

- The Color Tab is generally used as a means to vertex paint the Terrain, adding visual interest to areas and breaking up the "tiled" look of the terrain or to add fake shadows under large objects.
- **HOWEVER**, for this tutorial we will be using the ability to **Import** .tga files as vertex color in order to bring in "visual guides", in order to make it easier to convey instructions.
- Guides will be found in a sub directory of where you have the game installed via Steam, and each will be called out during its respective steps and can be loaded using the **Import** button:
 - **\TerrainEditorData\ModSampleFiles\MyNewMap_01_00**
 - Remember to get the full path: while in your Steam Library, right click on **The Great War: Western Front** Click **Properties** **Open a dialog** Click **Installed Files** Click **Browse**
 - This will open a window showing the installed game's file structure:
 - Probably a variation of C:\Program Files (x86)\Steam\steamapps\common\TheGreatWar\TerrainEditorData\ModSampleFiles\MyNewMap_01_00\

- Use the Import Button to bring up the Open dialogue.
 - Navigate to where Steam has the game installed (See above note for details) and **\TheGreatWar\TerrainEditorData\ModSampleFiles\MyNewMap_01_00**
 - Select **Color_Brush.tga** and click **OPEN**



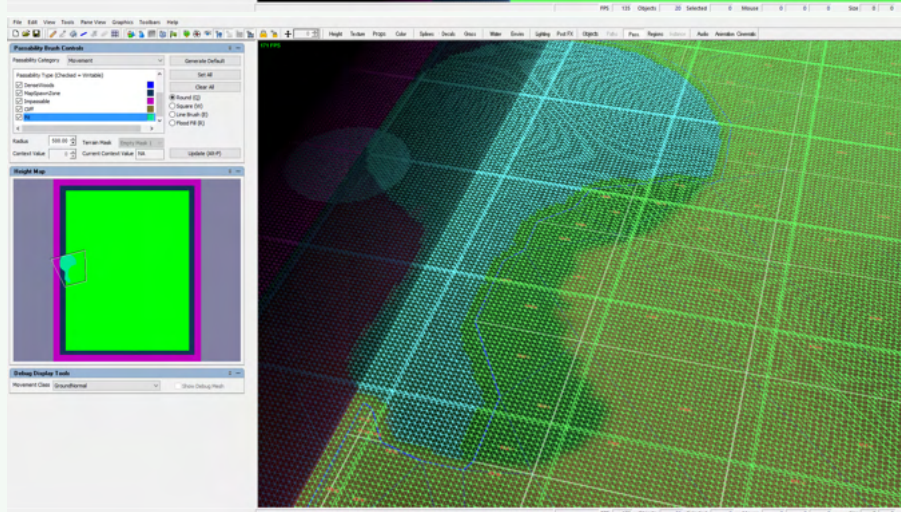
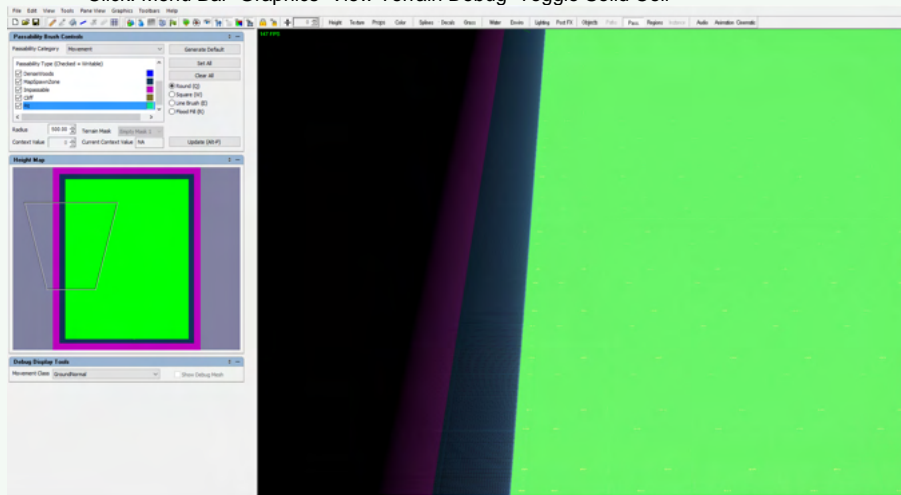
Drawing Pit Passability

- With the guide in place, we now have an idea as to where to draw the **Pit** Passability.
- Switch over to the **Pass Tab**
 - Main Tool Bar Editor Tool Bar **Pass Tab**
- Once in the **Pass Tab**, it is possible to "paint" various types of Passabilities with the mouse.
 - We will be focusing on the one labeled **Pit** as seen in the Passability Brush Controls Pane.
 - For more information on Pass Tab (See [13_Pass Tab](#) for more info).

NOTE

Toggle Solid Cell

- Using certain Tabs (like the **Passability and Region Tabs**) will make it very difficult to actually see the terrain, let alone the visual guide we load in.
- Toggle Solid Cell can be used to see through the grid making it easier to see the terrain.
 - Click: Menu Bar Graphics View Terrain Debug Toggle Solid Cell

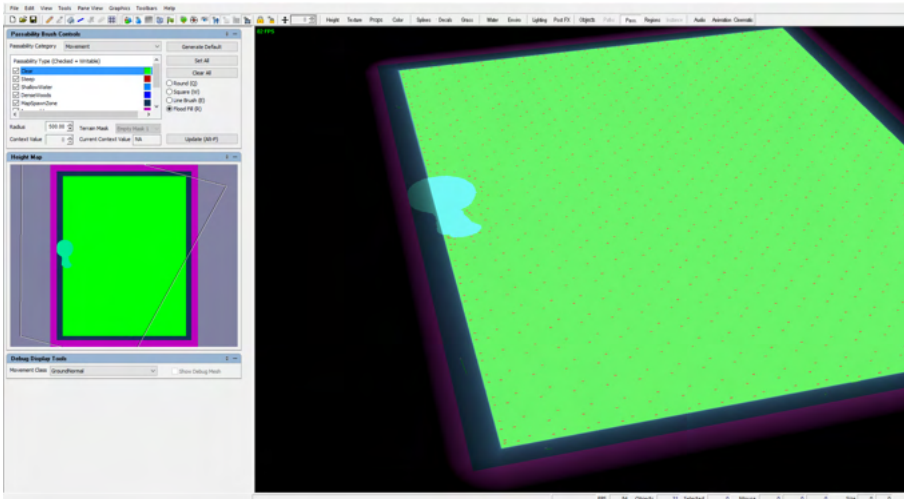


- Usually, we will toggle the grid to allow us to see through it, but sometime is easier to toggle between the modes when actually painting Regions and Passability.

POST FX

- Sometimes having the PostFX on can make the map look too dark to properly work in.
 - This can be changed using the **Enviro Tab**
 - Main Tool Bar Editor Tool Bar Enviro Tab (Editor Panes section) Environment Set Operations
 - Change the Post FX Chain drop down to **Color Test**, which has a lot of the Post FX turned off.

- For now, left click on **Pit** in the Passability Brush Controls Pane to select it.
- From here, we will work with the settings in the Passability Brush Controls in order to paint over our guide using left click to paint.
 - Depending on where you make mistakes, painting either **Clear** or the **MapSpawnZone** will allow you to fix it.
- When finished, it should look something like this.



NOTE

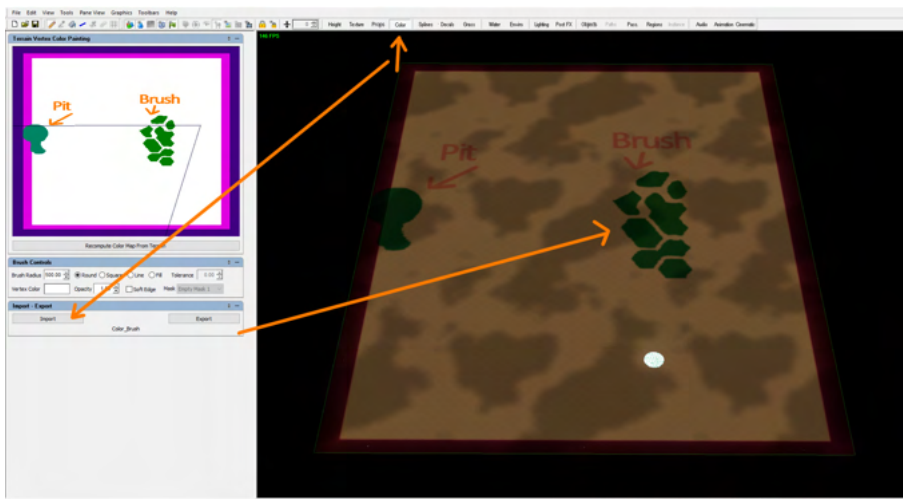
- Keyboard Shortcuts (when in the Pass Tab):
 - The number keys can be used to select a specific passability (i.e. clear = 1, steep = 2, pit = 6, etc.).
 - The brackets keys [] can be used to shrink ([) or grow (]) the brush size.

Setting Up DenseWoods Passability

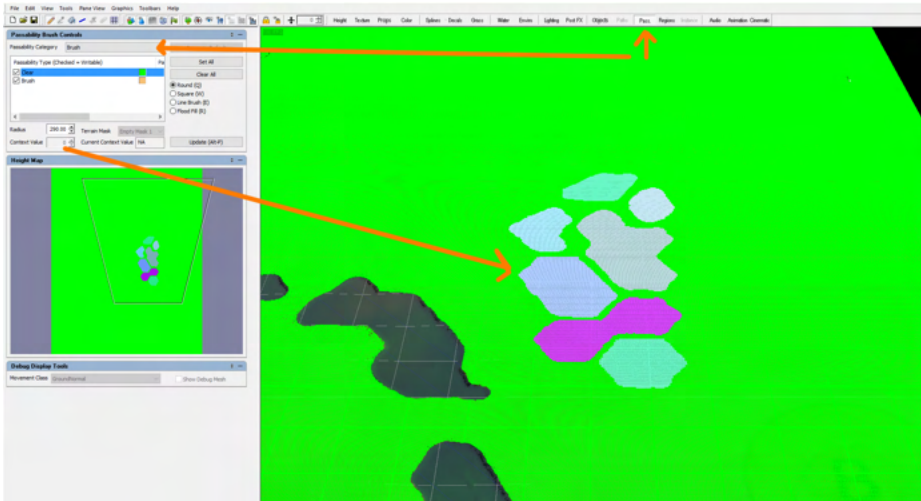
- Now that we have drawn in our **Pit** passability, we will create some forests using **DenseWoods** Passability.
 - Forests are a gameplay element that is intended to:
 - Prevent Trench Placement if in a region.
 - Prevent Tank Movement.
 - Stealth Infantry Companies, as long as the enemy is not in the same forest.

Visualizing where to paint the DenseWoods Passability

- As we have already loaded **Color_Brush.tga** in our previous step we will continue using the visual guide we have.



Drawing Dense Woods Passability



- While in the **Pass** Tab change the Passability Category from **Movement** to **Brush** via the dropdown
 - The Brush category comes in two types: **Clear** and **Brush**.
 - Clear (No Brush, so is used to delete any Brush)
 - Brush

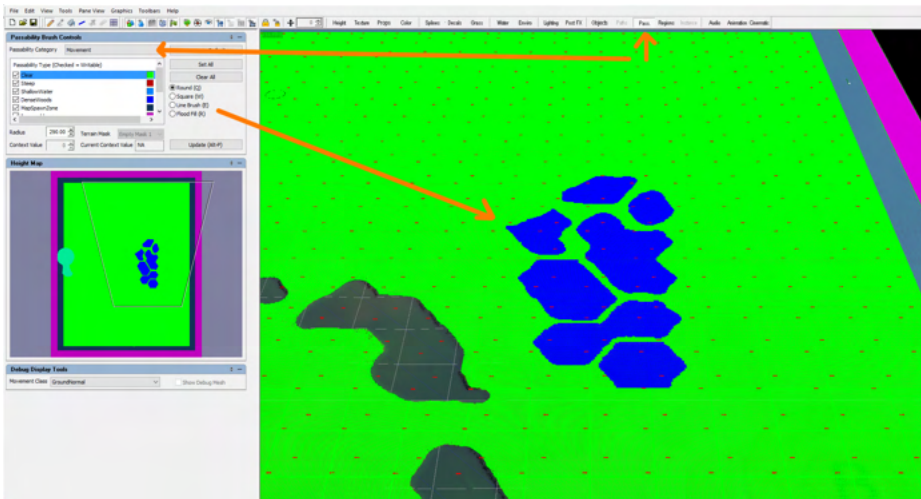


NOTE

Context Value

- One of the key gameplay elements of Brush is that it provides the stealth behavior to companies inside it. This requires enemy units to enemy brush in order to actually see the units.
- **Context Value** is important as a different value lets the system know that this should be considered a different Dense Forest which allows you to create separate sections close together forcing units to move between the different sections in order to see units inside.
 - When a **Context Value** is shared across different brush blobs, the system considers them the same Dense Forest.
 - This means that is all of your brush (forests) had the same **Context Value**, as long as you had a unit in a brush somewhere on the map, all of your units in brush could see into adjacent brush patches with their normal vision radius.
 - Generally, we give separate brush blobs a different value, so enemies' companies are forced inside the same brush patch to get vision.
 - **Current Context Value**, will let you know what the current context is of whatever brush blob you hover your mouse over.
- The easiest way to create the different patches is to create everything with same **Context Value** (recommend using 1, but the default is 0) and then when all of your brush is done, go back through and change the value for each wanted section and flood fill that section.

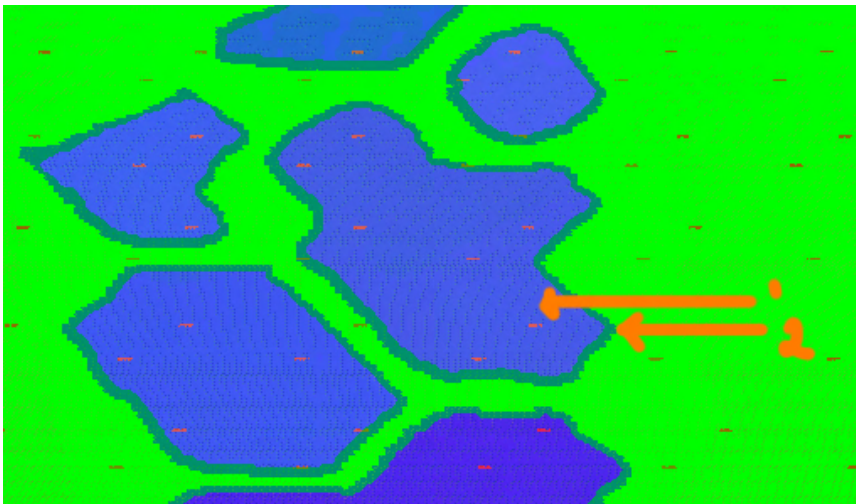
- Select the **Brush** Passability Type (and changed the context value to 1) and left click to paint the different brush patches.
 - **NOTE:** Brush on its own does not prevent Movement, therefore when you draw brush at Context Value 0, the system automatically draws **DenseWoods** Passability at the same locations.
 - Switching Passability Category to **Movement** after painting down brush using "0" allows us to see the automatically painted **DenseWoods** Passability.
 - Note: The other context values don't automatically draw the Passability type (which is why we recommend using 1 at first since makes it easier to adjust things without worrying about the automatic passability)



- Now, its best practice to paint brush first, since painting **DenseWoods** Passability directly does not work the same as when you paint brush, meaning painting **Dense Wood** will **NOT** automatically paint **BRUSH**.
 - Furthermore, erasing Brush **DOES NOT ERASE** the auto painted **DENSEWOOD PASSABILITY**, so Keep that in mind.
 - A good way to create brush is to **FIRST** create everything using **Context Value 1** (so NO movement passability is auto painted) until you have it looking the way you want, using clear to erase mistakes.
 - **Second:** Switch to the Movement Passability Category and clear out all of the **DenseWood** Passability that might have been auto painted during your work (in case had 0 selected for some of it).
 - **Third:** Flood fill all of you brush patches with **Context Value 0** (so it paints the matching **DenseWood** Passability)
 - **Finally,** go back and flood fill the different sections with the different **Context Values** that you want each to have.

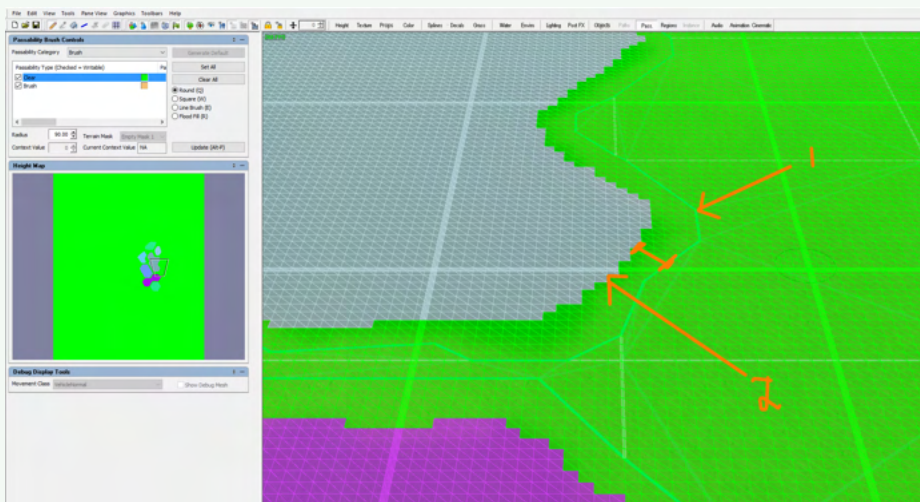
Refining Brush

- Once your Brush is painted and a different Context added, the last thing we need to do is refine our Brush.
- If we left our Brush as is, we would find that Tanks moving around the edges of our Brush, tend to stealth.
 - This is caused due to tank movement which will have them skirt forest as **CLOSELY** as possible, using the tanks center as reference instead of its extents.
- **THEREFORE,** in order to fix this, we generally take our Brush and paint **Clear** (Brush Category) around the edges of our Brush blob in order to shrink it. (See below image)
 - As you can see, we took our Brush (1) and we painted **Clear** (Brush Category) around it in order to shrink it.
 - In the end, we should push the Brush far enough from the **DenseWood** Passability (2) to keep edge hugging tanks from getting stealth.
 - An alternate way is to manually extend the DenseWood Passability a bit outside of the brush (same effect in the long run - tanks unable to reach the "brush" and gain the stealth).



NOTE

- While in the **Movement** Pasability Category, you can use the Debug Displays Tools to change the Movement Class from **Ground Normal** to **Vehicle Normal**.
 - Doing this will allow you to see the Tank Movement(1) while painting Brush (2)



- Once we are happy and tested our Brush in game, we can then proceed to Prop, Color, and Texture accordingly to convey the Dense Forest.



NOTE

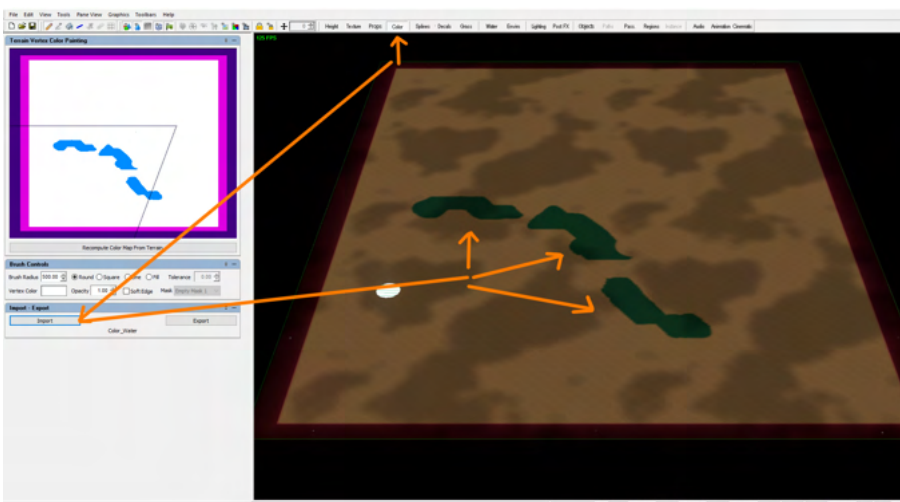
- Although, Dense Forest are intended to be used only by Infantry Companies for custom maps, you can technically choose to erase the **DenseWood** Passability and have forest that stealth ALL Companies (including Vehicles).
- When Dense Forest exist within a players Region, its best to use the Triangle Grid to refine the edges to create forests that allow for trenches to be placed, right at the edges.
 - Click: Menu Bar View Show/Hide Elements Triangle Placement Grid

Setting Up Water Regions

- Now that the **Pit** and **Brush** passability are placed, we can add our final element of shallow water, using the Region Tab.
 - **Region_Water_00 Water** : This region is used to denote the "shallow" water sections of the map that slow down Companies moving through them.
 - The name is tied to an effect generator so make sure you dont change it.
- Since this is a long and involved process, make sure you are constantly saving.

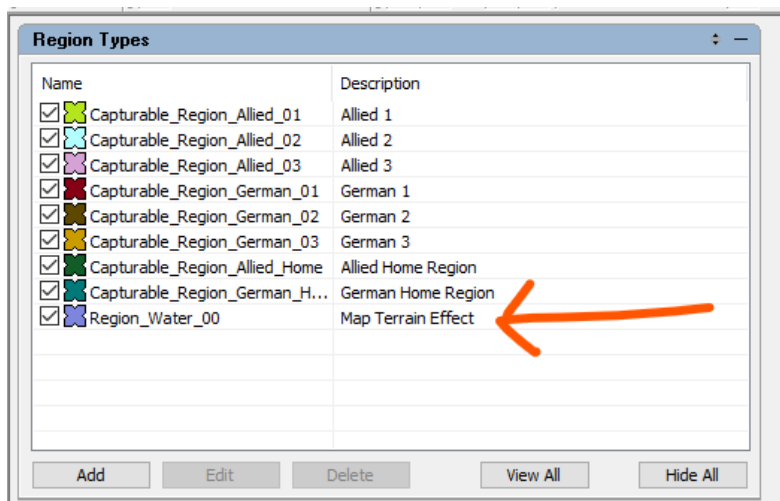
Visualizing Where to paint the regions

- Switch over to the **Color Tab** via Main Tool Bar Editor Tool Bar **Color Tab**
- Use the Import Button to bring up the Open dialogue.
 - Navigate to where Steam has the game installed (See above note for details) and `\TheGreatWar\TerrainEditorData\ModSampleFiles\MyNewMap_01_00\`
 - Select **Color_Water.tga** and click **OPEN**



Painting Water Regions

- With the guide in place, we now have an idea as to where to draw our water regions.
 - Switch over to the **Regions Tab** via Main Tool Bar Editor Tool Bar **Regions Tab**
 - We will be using the Region Table we created in the previous tutorial [02: Using the Map Editor to customize your new skirmish map#SettingUpRegions](#).



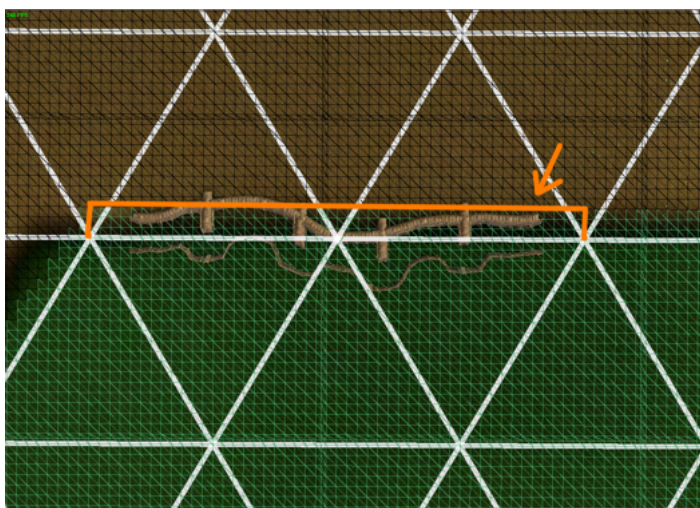
- Select **Region_Water_00** in the Region Types Pane and in the **Tools Pane** ensure that the **Replace Mode** is checked before starting to paint.
 - If can't see the guide, remember to "Toggle Solid Cell" via Menu Bar Graphics View Terrain Debug Toggle Solid Cell

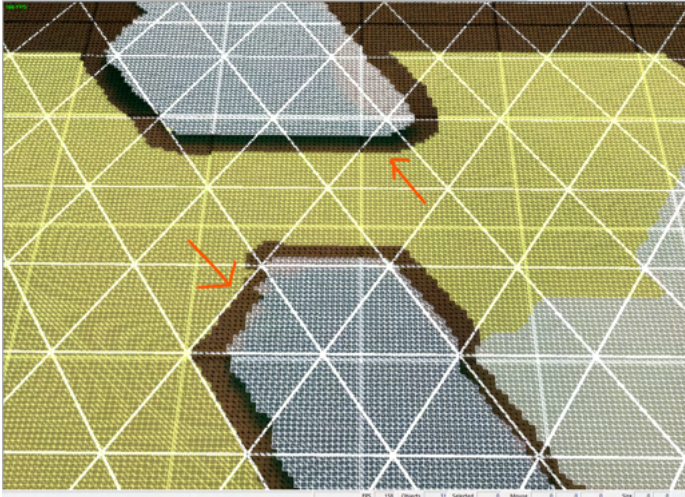


NOTE

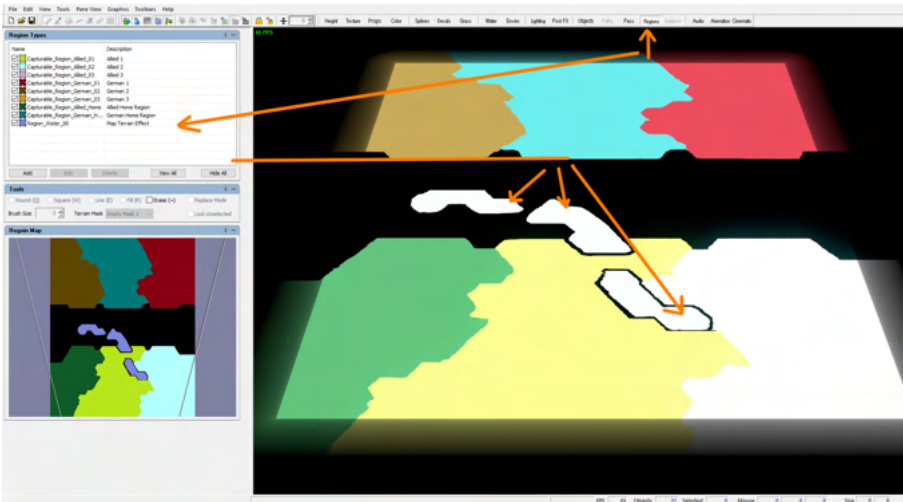
- Although it is possible to have **Region Types** overlap and exist at the same location, for this product we avoid this, since there is no visual element to convey the overlap to the player and it could create some strange confusing behavior with the control points and spawn points tech.
 - ALWAYS use **Replaced Mode** when painting, as it will replace any Region type already there with the one currently being used to paint.
- For more information on how to use the Region Tab See: [14_Regions Tab](#)

- Similar to forest, when painting Shallow Water by buildable regions, it is recommended to use the triangle grid to paint beyond what would be considered the center of the trench (as seen in the image below).
 - Click: Menu Bar View Show/Hide Elements Triangle Placement Grid
- As you can see below, since some of our shallow water sits in the Players Region, we leave a gap between the regions in order to control the placement of Trenches and prevent them from being built into the water.



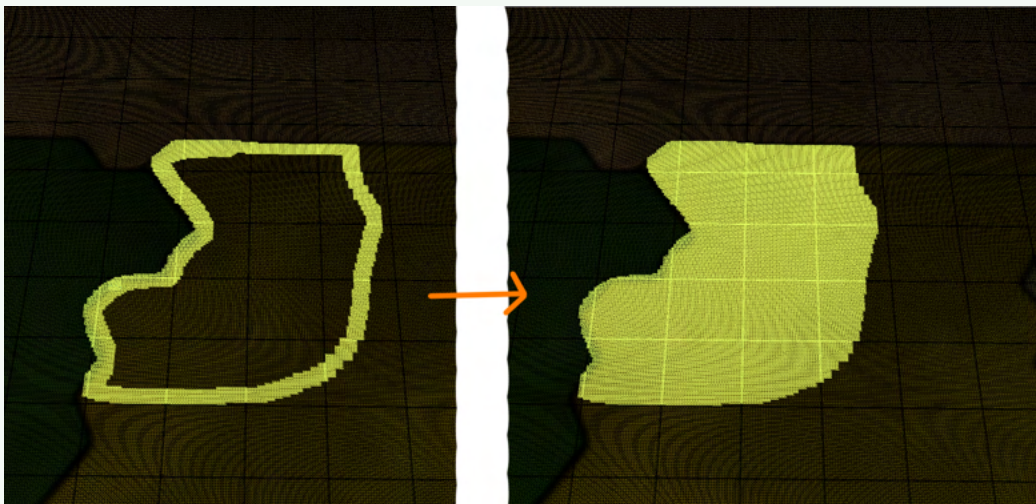


- We continue with this process until all 3 Shallow Water Region Types are painted.
- Once this is complete, we have finished adding Shallow Water functionality to our Main sob.



NOTE

- Remember that painting regions is an extremely iterative process, so don't expect to get it perfect the first time. It's best practice to block it in and through various playthroughs refine it in order to give the player an interesting play space to build their trench layouts.
- The process of Painting Regions can be expedited a bit through careful use of the Fill brush.
 - Essentially, as long as you can create an unbroken line, you can then use the Fill brush to quickly fill the center.

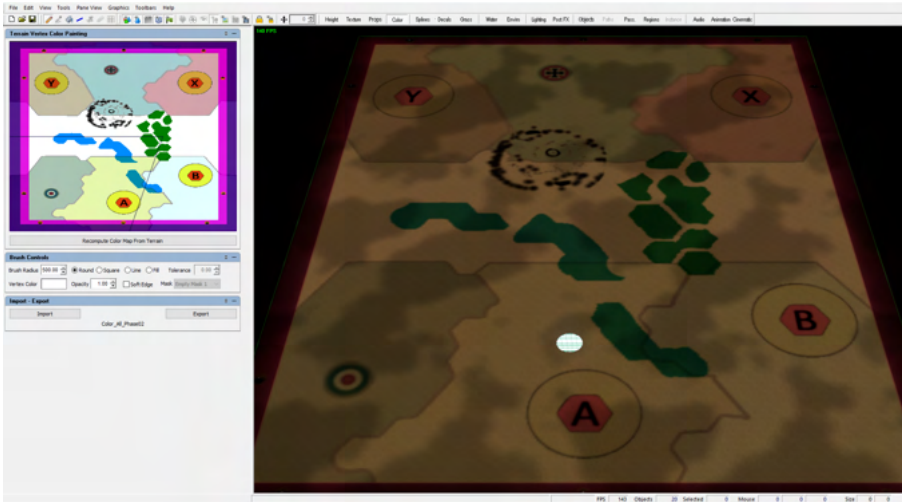


Adjusting Terrain Height to visualize our new map elements

- With **MyNewMap_01_00_SHARED.sob** ready to go, we want to add some visuals to denote our new elements.

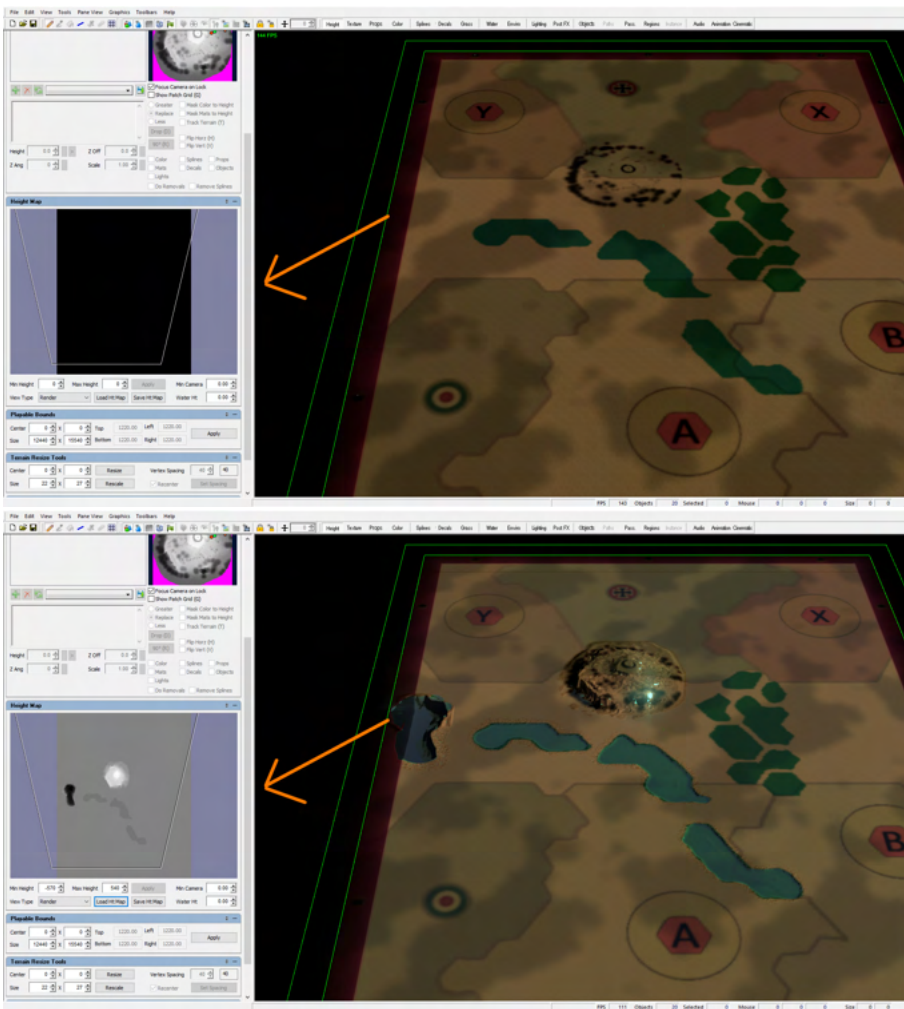
Visualizing Height Map for our new elements

- Switch over to the **Color Tab** via Main Tool Bar Editor Tool Bar **Color Tab**
- Use the Import Button to bring up the Open dialogue.
 - Navigate to where Steam has the game installed (See above note for details) and **\TheGreatWar\TerrainEditorData\ModSampleFiles\MyNewMap_01_00**
 - Select **Color_All_Phase02.tga** and click **OPEN**

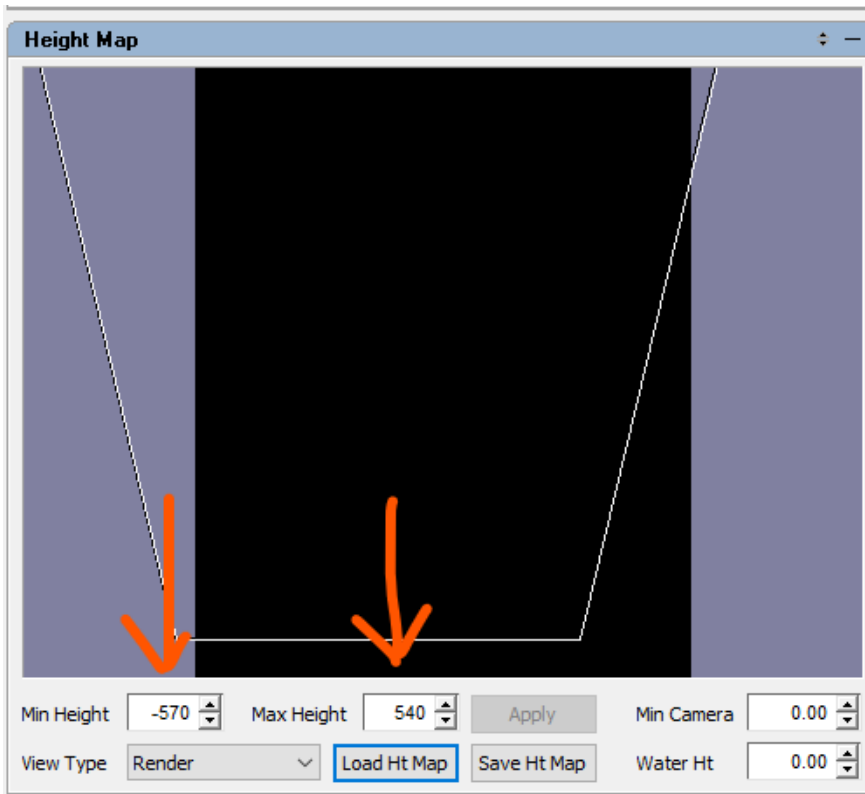


Loading a Height Map

- Switch over to the **Height Tab** via Main Tool Bar Editor Tool Bar **Height Tab**
 - The Height Tab is normally used to alter the height using the mouse to Raise and Lower the terrain.
 - See [01_Height Tab](#) for more information.
 - See [02_Texture Tab](#) for more information.
 - See [03_Props Tab](#) for more information.
 - Now when using the brushes to modifying the terrain, you'll notice that the Height Map, seen in the Height Pane changes colors.
 - This is because any modifications to height are represented as black and white data.



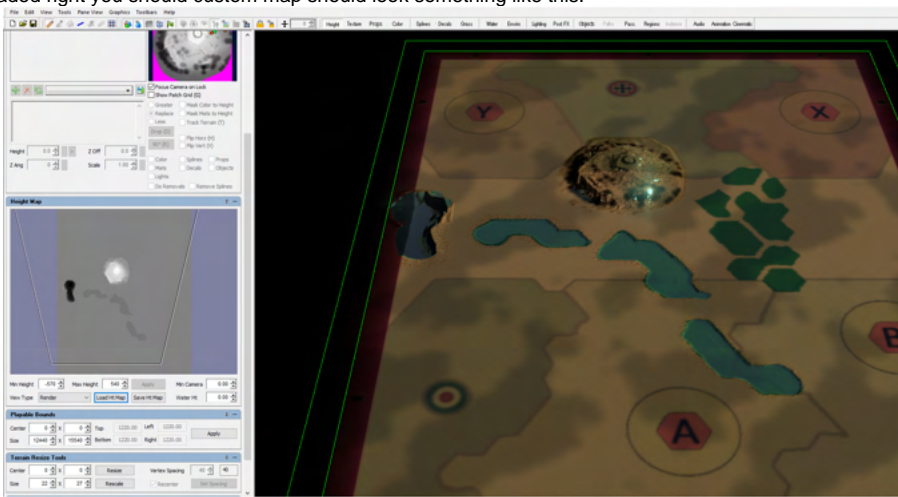
- Since its outside of the scope for this tutorial to go in-depth through the Height and Texture tab, we will learn how to load a height map via the height tab.
- First in our Height Map Pane, we are going to set the following min and max height.
 - Set Min Height: -570
 - Set Max Height: 540



NOTE

- When developing a map, it may be easier to save out the height map and edit it in a program similar to photoshop, as you can copy and paste your way to a quick initial layout.
 - Especially if you paint a "pallet" of different heights before you export out your height map, which would then allow you to quickly block out you map with corresponding heights before loading it back into the Map Editor.
- When sharing heightmaps, it's important to know the min/max setting, since the heightmap will apply those settings to the data.
 - For example, we could make our height map deeper by increasing the min, although we could also entirely negate the height map we are bringing in by making the min/max 0.
 - Black is brought in at the min height, while white is the max height with the grey values filling out the differences

- Click Apply (your map will probably be underwater since changed the min from 0 to -570 and your height map was all black).
- Use the "Load Ht Map" button to bring up the Open dialogue.
 - Navigate to where Steam has the game installed (See above note for details) and `\TheGreatWar\TerrainEditorData\ModSampleFiles\MyNewMap_01_00\`
 - Select **Phase02_Height_Map_00.png** and click **OPEN**
- When loaded right you should look something like this:



- Now with our height map loaded we proceed to our final steps of updating the TGA and moving over our updated files.

- Before we move on to create the Mini Map image, make sure to save your **MyNewMap_01_00.ted** and **MyNewMap_01_00_Shared.sob**

Updating the Mini Map image: MyNewMap_01.tga

- With the enhancements added to both **MyNewMap_01_00_SHARED.sob** and **MyNewMap_01_00.ted** the last thing we need to do is update our Mini Map image, before moving the files over to our Mod Folder for in game testing.
 - These next steps are mostly a repeat of the previous tutorial with the new files.

Visualizing Game Elements for MiniMap Creation

- For this step, we are going to keep the visual guide we loaded in the previous step.

Creating the Mini Map

- With our placeholder visuals in place, we can generate our mini map via the **Generate Map Preview Image**
 - Menu Bar Tools Generate Map Preview Image **1024x1024**
 - Make sure "Use Playable Bounds" is unchecked
- Once its generated, the image can be found here:
 - **C:\Users\USER_DATA\Documents\Petroglyph\TheGreatWar\Custom_Maps\MyNewMap_01.tga**
- And with our data created and functionality added, we can finally move our files over.



NOTE

Optionally customizing the Mini Map

- Once the mini map is created, it tends to be "thinner" than normal when run in game.
 - This means the location of Companies moving around the map will be a bit distorted.
- In order to account for this, it's best to take the original image with the purple borders into an image editing program and stretch the image from left to right so as that the "purple" borders are no longer seen (and then save).



Moving Updated Files from Map Editor directory to our Mod directory

- This final step of the tutorial builds on the work done in the pervious one, since you must have a functional Mod to pass the data over. (See [01: Setting up and running your Skirmish Map Mod](#))
 - Before continuing it is recommended you copy your entire skirmish map mod directory elsewhere to have a backup in case so you don't need to redo the entire first tutorial if something is wrong
 - **C:\Users\USER_NAME\Documents\Petroglyph\TheGreatWar\Mods\MyNewMap_01**
- We now need to move all of our new data from the Map Editor folder to the Mod folder we created that the game uses.:
 - Remember the Map Editor saves **ALL** its files at the following location: **C:\Users\USER_NAME\Documents\Petroglyph\TheGreatWar\Custom_Maps**
 - While the game requires the mod data to be located at: **C:\Users\USER_NAME\Documents\Petroglyph\TheGreatWar\Mods**
- The chart below shows where to place the Updated Map Editor Data and move it over to its respective location in our **MyNewMap_01** mod folder.

Map Editor Data	Mod Data
C: \Users\USER_NAME\Documents\Petroglyph\TheGreatWar\Custom_Maps	C: \Users\USER_NAME\Documents\Petroglyph\TheGreatWar\Mods
MyNewMap_01_00.ted	C: \Users\USER_DATA\Documents\Petroglyph\TheGreatWar\Mods\MyNewMap_01\Data\Art\Maps
MyNewMap_01_00_SHARED.sob	C:

	\Users\USER_DATA\Documents\Petroglyph\TheGreatWar\Mods\MyNewMap_01\Data\Art\Maps\ServerObjects
MyNewMap_01_00_SHARED.cpd	C: \Users\USER_DATA\Documents\Petroglyph\TheGreatWar\Mods\MyNewMap_01\Data\Art\Maps\Passability
MyNewMap_01_00_SHARED.gpd	C: \Users\USER_DATA\Documents\Petroglyph\TheGreatWar\Mods\MyNewMap_01\Data\Art\Maps\Passability
MyNewMap_01_00.tga	C: \Users\USER_DATA\Documents\Petroglyph\TheGreatWar\Mods\MyNewMap_01\Data\Art\Textures\SRGB

- Once everything is moved over, we can run the game and try it out.

Seeing your updated map in game

- Make sure the Mod is loaded via the Mod manager and then you can find your updated map in the Skirmish Map UI.



- Select the Map, add some units, and Click Start
- Once you loaded, test out placing trenches by your new map elements in pre-battle and walking around units in battle.



Congratulations of updating your mod with some of the more advanced Map Editor Features.

TUTORIAL: Creating a Cinematic

- 1 [Cinematic System](#)
 - 1.1 [Quick Start Guide](#)
 - 1.1.1 [Map Editor Steps](#)
 - 1.1.1.1 [Open the Cinematic Editor](#)
 - 1.1.1.2 [Create Your First Camera](#)
 - 1.1.1.3 [Set the Recommended Frustum](#)
 - 1.1.1.4 [Change the Length of the Time Line](#)
 - 1.1.1.5 [Create a Second Camera](#)
 - 1.1.1.6 [Adjust for Desired Shot](#)
 - 1.1.1.7 [Set the Fade in From black](#)
 - 1.1.1.8 [Save Cinematic](#)
 - 1.1.2 [Quick Start FAQ](#)

Cinematic System

- Cinematics are generally used in Historical Missions and called via lua script.
- This system allows the designers to fly the camera along a path to show the player locations, or enemy activity that's relevant to the story or the mission. The camera can either fade to black at the end of a sequence, or transition directly into the game camera. This is a nice effect as the player is put directly into the action at the end of the cinematic.
- Cinematic sequences are created / edited in the Map Editor and then saved out as a .tec file (Located **C:\Users\USER_NAME\Documents\Petroglyph\TheGreatWar\Custom_Maps**).
 - Although it is best to create a Cinematic on the map you wish to record on, Cinematics files can be used on any map, issues only being if you somehow place your camera markers outside of playable bounds (Maps of different size).

Quick Start Guide

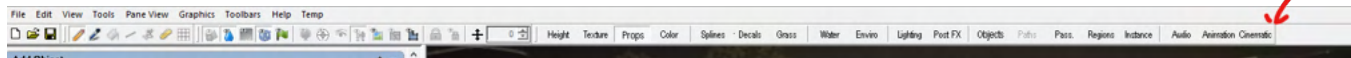
The Purpose of this Section is to take you through the steps to set up your first Camera, then have it Play it in Game. NOTE: Open the Map Editor and Open a .ted file. The work done with Cinematics creates its own .tec file and will not effect .ted or .sob files.

NOTE See the **Quick Start FAQ** section if you run into any issues as they may already be addressed there.

Map Editor Steps

- The Goal of this section is to Create and Save out cinematic file (.tec)
 - The Cinematic will consist of 2 **Cameras** looking at a single **Target** for 200 **Frames**
 - A **Fade-In** from black will also be added to the start of the frame.
- OVERALL PROCESS:
 - Open the Cinematic Editor
 - Create Your First Camera.
 - Set the Recommended Frustum.
 - Change the Length of the Time Line.
 - Create a Second Camera.
 - Adjust for Desired Shot.
 - Set the Fade in From black.
 - Save Cinematic.

Open the Cinematic Editor



- When in the Map Editor, enter the Cinematic editor by clicking on the last button (named Cinematic) in the Editor Tool Bar.
 - Clicking on **Cinematic** will bring up the Cinematics Player (Time Line) in the user window, notice that the time line is set to 600 frames, this is the max amount available to you when creating cinematics.
 - Hold down Mouse Right + Drag in order to move around 3D space in the Terrain Editor.
 - Hold down Control-Right click to rotate around 3D space in the Terrain Editor.
 - Scroll up or down using Middle Mouse button in order to zoom Terrain Editor camera.
 - **NOTE:** While in **Cinematics** Pane (tools window on the left hand side of the editor) you will primarily working in (Camera) (Target) (Frustum) Tabs for this example (Note each tab allows you to Add/Remove/Adjust settings for each frame created).

Create Your First Camera

- Placing your **FIRST** Camera is always done by hovering your Mouse Cursor in 3d Space at the location where you want to record then Shift + Left Clicking to place (**NOTE** this is the **ONLY** camera that will be set this way).
 - Once placed you will see a Camera Dummy Object and Dummy Target Object
 - Dummy Objects denote the location and facing of the camera.

- Dummy Objects can be "grabbed" (Left Click + Drag) and moved along the X,Y axis . HOWEVER, you must have the frame selected and the respective tab selected for that object (Camera/Target), otherwise movement will be prevented.
- Camera will always face the Target Dummy when moved.

Set the Recommended Frustum

- After Setting Down your first Camera, **THE FIRST THING YOU SHOULD DO** is set a Frustum Frame by clicking the Frustum Tab in the Cinematics Pane. Click Add in order to add a Frustum Frame at Frame 0.
 - ADJUST THE FRUSTRUM:
 - While in the Frustum Tab set the following parameter:
 - Near Clip Plane: 2.00
 - Far Clip Plane: 20005
 - Object Fade Distance: 20000
 - Object Rejection Distance: 20005
 - These settings are the most common settings for 90% of your shots.
 - **NOTE:** Its recommended to avoid angling the camera so you can see too much of the horizon

Change the Length of the Time Line

- The **NEXT THING YOU SHOULD DO** is shorten the timeline.
 - CINEMATIC PLAYER:
 - The Cinematic Player allows you to See/Set frames as well Play/Pause cinematic . Furthermore the Cinematic Player allows you to filter vision of each type of frame (Camera, Target, Frustum, Roll, Shake, Light).
 - The Cinematic Player has a slider you can grab to scrub through the timeline as well as input boxes to move directly to specific location.
 - Since our Cinematic will only be 200 Frames long, Set the **Total** input box (which reads 600) to 200. This will shorten the length of the cinematic (NOTE this will change the **End** input box as well).

Create a Second Camera

- The **NEXT THING YOU SHOULD DO** is add a second Camera by using the Cinematic Player and the Cinematic Pane set to the Camera Tab
 - NOTE: When adding another Camera or Target, those objects will be connected via a visible line called a Spline. The power of the system is found here, since designer are able to create smooth movements from Point A to B, while also being able to add other points to the spline if need be.
 - CAMERA:
 - Add a second camera by simply moving the slider on the time line to frame 200 (you can also type in the number in the "Current" input box to move the slider).
 - Once you have done this, in the Cinematic Pane, make sure you are in the Camera Tab, NOTE that the **Add** button should now be available. Press it to set a Camera frame and Spawn in a new camera Dummy in.
 - NOTE: The second camera is spawned within the first camera. You must change its X,Y, or Z position in order to view, which can be done via the Camera Tab OR you can "grab" it and move, although only on the X,Y axis.

Adjust for Desired Shot

- NOTE: This is the more experimental phase of the Quick Start Guide, since there are plenty of settings that can be adjusted to get the perfect shot. HOWEVER for brevity sake, will give a quick run down of what you should do and what you can do.
- **YOU SHOULD**
 - Grab Camera 1 and Camera 2 and move them in such a way, where you can clearly see the spline that connects them.
 - Then using the Cinematic Player Time Line, you can click the Play button to watch the Camera travel from camera 1 to 2 (NOTE how since there is only one target, as it travels the camera will continue to stare at the target the whole run through).
- **YOU COULD**
 - In the Capture Options Pane (if you scroll down to the bottom) there is an option for Cinematic View, click this box on or off as needed to get a view of what the camera sees. You can work in this mode, it's very helpful.
 - Also while in the CAMERA tab you can adjust the X,Y,Z axis. Be weary of the Y axis as it jumps your camera to another portion of the map, and can lead to confusion.
 - You can add a second Frustum Frame at frame 200. Then you can adjust the Camera Field of View from 30 to 78. Since there are 2 different frustum frames on the time line, the system will interpolate between the 2.
 - With the frame and Camera selected, use the Cinematics Pane to adjust the location and height of either of your 2 cameras.

Set the Fade in From black

- With Camera movement set up, you can now add a Fade in from Black to the start of the Cinematic. Look for the option to be located in the same place where you can toggle between the Cinematic View in the **Capture Options Pane** options.
 - Toggle Fade In/Out On (Checked)
 - Frames - Fade In to 150
- **NOTE:** To see the effect you must be in Cinematic View.
- Using the Time line hit the Play button, this should cause the first 150 frames to start black and gradually lighten.
- **NOTE:** It is possible to make it so your screen remains black. This is solvable by making sure you don't stop the cinematic when its entirely black.

Save Cinematic

- To save the sequence go to the SAVE CINEMATIC button located at the top of the Cinematic Pane (tool set on the left hand side of the editor).
- save the file at the following location with the name CinRollTest:
 - C:\Users\USER_NAME\Documents\Petroglyph\TheGreatWar\Custom_Maps

Quick Start FAQ

- Can I chain cinematics together?
 - Can be created separately and chained together in game via LUA Scripts.
- Is the Cinematic affected by the Half Speed/Double Speed Buttons?
 - Yes, the Cinematic will play slower or faster respectively.
- I some how caused the Cinematic Player Time Line to disappear, how do I get it back?
 - You can cause the Player to disappear by having it selected and hitting the Esc key.
 - Save your current cinematic and jump to a different Editor (Height, Objects, Animations, etc), then jump back to the Cinematic editor.
- In the Editor, How can I see what the Camera(s) are seeing?
 - Locate the Capture Options on the Left Hand side of the screen. Within that pane look for **Capture Options**. There you will see a box called **Cinematic View** check to see the view from the cameras perspective (uncheck to go back).
 - NOTE: When in this view, you will not be able to add new Cameras to the time Line.
- I played my cinematic and my 3d space is now permanently black, how do I fix it?
 - It's possible to darken your 3d space when viewing your cinematic.
 - When using the Fade In/Out feature, if you were to stop or pause the cinematic at the start of the fade in or at the end of a fade out the Editor retains the last change making the 3d space pitch black.
 - To Fix it just use the timeline to play your cinematic and pause/stop it past/before the screen is set to all black.
- I keep trying to Set the Cinematic Player Slider to 0, but it keeps moving on its own or "struggles" when I try to move it, how do I fix this?
 - On the Cinematic Player, hit the Stop Button.
 - Essentially, the Player is trying to play (kind of) and that's why it fights you.
- I want to add a new camera to a blank frame, but the Add button in the Cinematic Pane in the Camera Tab is not selectable.
 - This is generally caused by trying to add a camera WHILE in Cinematic View. Exit Cinematic View by unchecking the Cinematic View box in the Capture Options Pane.
- How do I reset the entire process? I've jumped from the Cinematic Editor to another then back, but I can't get rid of my previous work?
 - For the most part as long as you don't exit the Terrain Editor, work you do in the Cinematic will remain. So you could restart the terrain editor.
 - Another way is to Save a .tec file immediately after you place your first camera and set your frustum frame, so you can always return to a clean-is previous state.

INFORMATION: LUA Debug Tools

Getting Started

file:///D:/Projects/RENO_TGW_DES/TGW/Des/Documents/luadocs/modules/ServerTGWUtils.html



The function below, when placed into a Map Script load a suite of Debug Tools which are useful when testing custom content. The following Tools are most commonly used in Map data creation:

LUA SNIPPET

```
--Uncomment in order to Turn on LUA Debug Tools  
Server_Enable_Mod_Debugging_Tools()
```

Tip: The place where you place the function in script matters. It's best to place where it won't trigger constantly, preferably only once. In In Provided sample scripts, we tend to place it in GamePhaseStarted_Handler(), either in GAME_PHASE_TACTICAL_PRE_BATTLE or GAME_PHASE_TACTICAL_BATTLE that way its available from the beginning of the game.

Tip: The top of each window can be used to move it around. To left of the name of each window there is a small arrow. This arrow can be pressed in order to collapse the window, making it easier to see.

SDBG List

- The following list is be toggled on/of by **Toggle List** in the SDBGDIAGNOSTICS window. An element needs to be clicked to turn on (goes from green to red) and the data for it will appear in the SDBGDIAGNOSTICS window. More one element can be on at a time, appearing in that order in the SDBGDIAGNOSTICS window.
 - **Object Name**
 - Provides the user with objects name (Client/Server) on Mouse Hover over object.
 - **Find Objects**
 - Provides the user with a list of all the objects in the instance and on a button click, can snap the camera to its location.
 - **Triangles**
 - Allows the user to see the Triangle grid outside of pre-battle.
 - **Trench Configurator**
 - Allows the user to create Trench Templates to be used with Skirmish/Campaign/Historical mod Content, by allowing the player to create a .lua file of object name and locations.
 - Furthermore, with the Terrain Editor Open while the game is running, it can "Teleport" trenches from game into the Map Editor in their correct x/y positions.
 - **Modding Diag**
 - Opens the SDBG_USER_MODDING_DIAG

SDBGDIAGNOSTICS

- Remains empty, until a command is chosen from the SDBG List. Can hold more than one element chosen from the SDBG List, appearing in that order in the SDBGDIAGNOSTICS window.
 - **Toggle List**
 - Toggles the SDBG List on/off (Hide/Unhide).

- **Hide Diag**
 - Removes the Debug tools entirely.

SDBG_USER_MODDING_DIAG

- A series of Toggles with different functionality.
 - **Obj Diag (Toggle)**
 - This tool provides information on the players as well as any object the mouse cursor is hovered over.
 - Become extremely useful when dealing with AI, Unit/Player States via a GUI Debug window.
 - **Lua Logs (Toggle)**
 - LUA logs are generally something that is not **ON** by default.
 - Dev user generally needs to add the attached file (sdbglogpreferences.txt) to their run folder for the files to be generated in the run folder.
 - It can be found here ".\Projects\RENO_TGW_DES\TGWDes\Project\Run\Log" and is usually used in conjunction with baretail, since it updates during runtime.
 - Logs are generally prefixed by "SDBGLog" and are constantly updated each time LUA triggers, which may be a problem to consider.
 - **Debug Keys (Toggle)**
 - Allows the player to use the following Keyboard debug commands
 - **F3 Keyboard Key**
 - Allows the users to cycle through the different types of Fog of War, useful when monitoring the AI and testing the map as you can remove the fog of war entirely.
 - **Alt+K / Shift+K**
 - Mouse over an object then:
 - Shift+K Damages objects slowly over time.
 - Alt+ K Kill objects immediately.
 - **Structure Friendly Region Override (Toggle)**
 - During Pre battle, allows players to build trenches **anywhere**. This is useful as it makes it easier to build trench layouts and either teleport them over to the terrain editor or create a trench template using the **Trench Configurator**.

LUA Modules

The following documents provide a brief description of the LUA modules available for The Great War: Western Front.

Inside each Module, you will find descriptions and parameters of the functions found within.

- [Misc](#)
- [PG](#)
- [AIRPC](#)
- [ClientCamera](#)
- [ClientCinematic](#)
- [ClientGUIComponent](#)
- [ClientGUIUtils](#)
- [ClientObjectManipulation](#)
- [ClientObjectivesDisplay](#)
- [ClientPGGUI](#)
- [ClientPlayers](#)
- [GUIAdvancedTicker](#)
- [GUIAnimation](#)
- [GUIButtonBase](#)
- [GUICarousel](#)
- [GUIClockProgress](#)
- [GUIComboBox](#)
- [GUIDecoratedIconButton](#)
- [GUIImageBox](#)
- [GUIGeneric](#)
- [GUIIconButton](#)
- [GUILine](#)
- [GUIListBox](#)
- [GUIMPEGMovieQuad](#)
- [GUIMenu](#)
- [GUIModel](#)
- [GUIMovieQuad](#)
- [GUIProgressBar](#)
- [GUIScene](#)
- [GUISceneReference](#)
- [GUISliderBar](#)
- [GUITextBlock](#)
- [GUITextButton](#)
- [GUITexturedQuad](#)
- [GUITicker](#)
- [GUITree](#)
- [FrameUpdate](#)
- [ServerCinematic](#)
- [ServerEffectSystem](#)
- [ServerGUIUtils](#)
- [ServerInstanceManagement](#)
- [ServerObjectManipulation](#)
- [ServerObjectSelection](#)
- [ServerObjectives](#)
- [ServerPlayers](#)
- [ServerTGWUtils](#)
- [ServerUnitOrders](#)
- [ServerVictoryConditions](#)
- [Localization](#)
- [ObjectManipulation](#)
- [Players](#)
- [TGWUtils](#)
- [TerrainRegions](#)
- [UnitOrders](#)
- [PGServer](#)
- [Timers](#)
- [GameSignals](#)
- [StateSignals](#)
- [EXAMPLES](#)

Misc

A module full of Designer created Functions.

Functions

DialogBox_Display_Popup (dialogid, caption, handler, button1text, button2text, button1text, scenename)	Dialog box popup function.
CinLockedGUITextSet_UnHidden (showtextbool)	Functions when called unhides/hides the "CINEMATIC MODE - INPUT LOCKED" (TEXT_NAME_MISSIONS_OBJ_LABEL_07) text
PlayBarkAfterDelay (bark_to_play, delay_seconds)	Funtion allows a bark to be set on a non repeating timer.
FillPlayerInfo (player_obj)	Functions which Generates a useful data table of specific player info.
UpdateLocalizedObjectiveText (update_color, mapdata_objective, final_outcome)	Functions which updates the Mission Objective Tracker in Tactical for Count type Objectives.
ConvertSectoTimeFormatString (textid, timer_count, add_textcontrol)	Function which takes a Tex Tag and returns text with the proper (00:00) added to it.
InitializePhaseTimer_Localized (mapdata_ref, objective_objects_ref, timer_string_name, logictimer_function, add_debugname, add_textcontrol)	Function which initilizes a visual mission timer.
CreateVisualTimerTracker_Localized (data)	Function which updates the GUI Missions Objective tracker.
UpdateVisualTimerText_Localized (update_color, objective_objects_ref)	Function which updates the GUI Missions Objective tracker with color flashes.
OnlyConvertSectoTimeFormatString (timer_count)	Function which takes a Tex Tag and returns text with the proper (00:00) added to it.
InitializePhaseTimer_GUIMTracker (mapdata_ref, objective_objects_ref, timer_string_name, logictimer_function)	Function which initilizes the New GUI Missions Objective tracker.
CreateVisualTimerTracker_GUIMTracker (data)	Function which updates the New GUI Missions Objective tracker.
UpdateVisualTimerText_GUIMTracker (update_color, objective_objects_ref)	Function which updates the New GUI Missions Objective tracker with color flashes.
SetMObjTimer_GUIMTracker (phase_objective_data)	Function used to createa a Timer Objective with the new Mission Tracker GUI elements (Text and Progress boxes).
SetMObjCount_GUIMTracker (phase_objective_data)	Function used to createa a Count Objective with the new Mission Tracker GUI elements (Text and Progress boxes).
BrieflyShowCompletedTimeObjective (completed_objective, objective_list, set_objective_textfunction)	Function allows a completed Timer Objective to be displayed in the GUI Tracker for a short time.
SearchAllPlayerList (player_ref, find_obj_type)	Functions which returns a list of player owned objects, sharing the same state
RaisePlayerBalloons (user_data)	Functions which when given a table of player info, raises or lowers the balloon any balloons owned by that player.
GetAllCPMaxValues (mapdata_cp_list)	Functions which when given a custom table, which we run at Start() in a Map Script, saves all CPs oroginal Max Capture Value, for later manipulation.
FindNewCPMaxValue (cp_data, new_maxvalue)	Functions returns the % (given in decimal form) of the maxcpvalue as a number
SetAdjustCPMaxValue (cp_data, new_maxvalue, mapdta)	Function is similar Set_Capture_Point_Max_Value, EXCEPT it calculates and adds any missing progress , which happens when you go from a small max capture value to a bigger one.
PingObjectSpawnRemove (obj_type, location)	Function intended to act like a fake ping, spawns an object and removes it after a set time.
VideoInProgress_Quad_Set_Hidden (hide_or_unhide)	Function intended to hide/unhide the VideoInProgress_Quad in the lower right hand corner during the cineamtic.

Functions

DialogBox_Display_Popup (dialogid, caption, handler, button1text, button2text, button1text, scenename)

Dialog box popup function.

Parameters:

- dialogid : (string) unique string id assigned to this dialog.
- caption : (string) The text you want displayed inside the popup.
- handler : (function|nil) lua function that receives the result of the popup. Function Params(dialog_id, button_index)
- button1text : (string|nil) Text to show on button 3(right) or nil to hide. Default: nil
- button2text : (string|nil) Text to show on button 2(center) or nil to hide. Default: "TEXT_OK"
- button1text : (string|nil) Text to show on button 3(right) or nil to hide. Default: nil
- scenename : (string|nil) BUI scene to load for the popup. Default: DialogBox

Returns:

nil

CinLockedGUITextSet_UnHidden (showtextbool)

Functions when called unhides/hides the "CINEMATIC MODE - INPUT LOCKED" (TEXT_NAME_MISSIONS_OBJ_LABEL_07) text

Parameters:

- showtextbool : bool value, true = show text in upper right corner false= hide it

Returns:

nil

PlayBarkAfterDelay (bark_to_play, delay_seconds)

Funtion allows a bark to be set on a non repeating timer. If no seconds are provided, then it will wait 0 sec by default.

Parameters:

- bark_to_play : String name of Bark
- delay_seconds : bark delay in seconds. No seconds will mean no delay.

Returns:

nil

FillPlayerInfo (player_obj)

Functions which Generate a useful data table of specific player info.

Parameters:

- player_obj : CObjectRef of player creating the data table of

Returns:

Returns a data table.

Usage:

```
local sideinfo ={ }
```

UpdateLocalizedObjectiveText (update_color, mapdata_objective, final_outcome)

Functions which updates the Mission Objective Tracker in Tactical for Count type Objectives.

Parameters:

- `update_color` : String name of the color you want to flash the Mission Objective Tracker.
- `mapdata_objective` : reference to the ObjectiveObjects data table in the script
- `final_outcome` : Normally left blank but can be used to append another localized text id by providing TEXT_ID as a string.

Returns:

nil

ConvertSectoTimeFormatString (textid, timer_count, add_textcontrol)

Function which takes a Tex Tag and returns text with the proper (00:00) added to it. Used SPECIFICALLY in Conjunction with `InitializePhaseTimer_Localized()`

Parameters:

- `textid` : TextTag ID with Localization Tokens (##0, ##1, ##2, etc.)
- `timer_count` : time in sec to be converted into the (00:00) format.
- `add_textcontrol` : bool - will add the timer to the center GUI if true.

Returns:

string with (00:00) to be displayed.

InitializePhaseTimer_Localized (mapdata_ref, objective_objects_ref, timer_string_name, logictimer_function, add_debugname, add_textcontrol)

Function which initializes a visual mission timer. This version of the function is able to take localized data. Part of a set marked by the suffix "_Localized"

Parameters:

- `mapdata_ref` : Reference to the MapData data table found on ALL Scripted maps.
- `objective_objects_ref` : Reference to the data table and specific Mission Objective Data.
- `timer_string_name` : String which will give a name to the CTimerRef object that will be created.
- `logictimer_function` : Function which will play if the timer Ends or times out.
- `add_debugname` : bool which forces the timer to display the instance name of the instance it's in.
- `add_textcontrol` : bool which if set to "true" will use the timer GUI element.

Returns:

nil Misc.InitializePhaseTimer_Localized(MapData,Phase01_ObjectiveObjects.MainObjective,"P01_MAIN_TIMER",Phase01_MainObjectiveLogicTimer)

CreateVisualTimerTracker_Localized (data)

Function which updates the GUI Missions Objective tracker. This version of the function is able to take localized data. Part of a set marked by the suffix "_Localized"

Parameters:

- `data` : Data Table of references from InitializePhaseTimer_Localized, with information about the timer.

Returns:

nil

UpdateVisualTimerText_Localized (update_color, objective_objects_ref)

Function which updates the GUI Missions Objective tracker with color flashes. This version of the function is able to deal with "paused" timers. Part of a set marked by the suffix "_Localized"

Parameters:

- `update_color` : String name of "color" to use.
- `objective_objects_ref` : Mission Objective Reference

Returns:

nil

OnlyConvertSectoTimeFormatString (timer_count)

Function which takes a Tex Tag and returns text with the proper (00:00) added to it.

Parameters:

- timer_count : time in sec to be converted into the (00:00) format.

Returns:

string with (00:00) to be displayed.

InitializePhaseTimer_GUITracker (mapdata_ref, objective_objects_ref, timer_string_name, logictimer_function)

Function which initializes the New GUI Missions Objective tracker.

Parameters:

- mapdata_ref : Reference to the MapData data table found on ALL Scripted maps.
- objective_objects_ref : Reference to the data table and specific Mission Objective Data.
- timer_string_name : String which will give a name to the CTimerRef object that will be created.
- logictimer_function : Function which will play if the timer Ends or times out.

Returns:

nil

CreateVisualTimerTracker_GUITracker (data)

Function which updates the New GUI Missions Objective tracker.

Parameters:

- data : Data Table of references from InitializePhaseTimer_Localized, with information about the timer.

Returns:

nil

UpdateVisualTimerText_GUITracker (update_color, objective_objects_ref)

Function which updates the New GUI Missions Objective tracker with color flashes. This version of the function is able to deal with "paused" timers.

Parameters:

- update_color : String name of "color" to use.
- objective_objects_ref : Mission Objective Reference

Returns:

nil

SetMObjTimer_GUITracker (phase_objective_data)

Function used to create a a Timer Objective with the new Mission Tracker GUI elements (Text and Progress boxes).

Parameters:

- phase_objective_data : data table of Missions Objective specifics

Returns:

nil

SetMObjCount_GUITracker (phase_objective_data)

Function used to create a Count Objective with the new Mission Tracker GUI elements (Text and Progress boxes).

Parameters:

- phase_objective_data : data table of Missions Objective specifics

Returns:

nil

BrieflyShowCompletedTimeObjective (completed_objective, objective_list, set_objective_textfunction)

Function allows a completed Timer Objective to be displayed in the GUI Tracker for a short time. Shifts the whole list in order to display the completed objective at the top of the tracker, then times out shifts it back removing the completed objective.

Parameters:

- completed_objective : data of the objective you want to display completed for a brief period of time.
- objective_list : list of data with Mission Objective information.
- set_objective_textfunction : mapscript function which normally sets the GUI Mission Tracker.

Returns:

nil

SearchAllPlayerList (player_ref, find_obj_type)

Functions which returns a list of player owned objects, sharing the same state

Parameters:

- player_ref : The CObjectRef of a player
- find_obj_type : unique string, the name of a specific object_type

Returns:

list of player owned objects of that object type, or nil.

RaisePlayerBalloons (user_data)

Functions which when given a table of player info, raises or lowers the balloon any balloons owned by that player.

Parameters:

- user_data : Player Info Table created by the FillPlayerInfo() and a String Bark name.

Returns:

Will only return "nil" if there is a problem otherwise raises balloons.

GetAllCPMaxValues (mapdata_cp_list)

Functions which when given a custom table, which we run at Start() in a Map Script, saves all CPs original Max Capture Value, for later manipulation.

Parameters:

- mapdata_cp_list : a custom data table defined in a Map Script (MUST HAVE), which stores Control Point info.

Returns:

Will only return "nil" if mapdata_cp_list does not have a maxcpvalue value.

FindNewCPMaxValue (cp_data, new_maxvalue)

Functions returns the % (given in decimal form) of the maxcpvalue as a number.

Parameters:

- cp_data : specific location in the mapdata_cp_list
- new_maxvalue : percentage given as decimal when looking to set a new value on a cp, i.e. Lower Max Value of specific CP by 30% (new_maxvalue = .3)

Returns:

Will only return "nil" if mapdata_cp_list does not have a maxcpvalue value

SetAdjustCPMaxValue (cp_data, new_maxvalue, mapdata)

Function is similar Set_Capture_Point_Max_Value, EXCEPT it calculates and adds any missing progress , which happens when you go from a small max capture value to a bigger one.

Parameters:

- cp_data : specific location in the mapdata_cp_list
- new_maxvalue : use the value returned by FindNewCPMaxValue
- mapdata : provided in order to properly change sizes of progress bar

PingObjectSpawnRemove (obj_type, location)

Function intended to act like a fake ping, spawns an object and removes it after a set time.

Parameters:

- obj_type : in order to spawn it, needs a valid obj type.
- location : world location to spawn in the object.

VideolnProgress_Quad_Set_Hidden (hide_or_unhide)

Function intended to hide/unhide the VideolnProgress_Quad in the lower right-hand corner during the cinematic.

Parameters:

- hide_or_unhide : bool, False = Unhide and True = Hide

PG

Sets up global values usable across PG libs.

Fields

InvalidTeamID	A value representing an invalid team ID.
----------------------	--

Fields

InvalidTeamID

A value representing an invalid team ID. May be returned by a function returning a team ID.

Usage:

```
local team_id = Get_Player_Team_ID(player_ref)

if team_id == PG.InvalidTeamID then
    print("The team ID was invalid!")
end
```

AIRPC

Provides functionality to send function calls to AI players.

Functions

Send_AI_Function_Call (player_selector, function_name, ...)	Sends a function call to the given global function in the AI player's Lua state.
--	--

Functions

Send_AI_Function_Call (player_selector, function_name, ...)

Sends a function call to the given global function in the AI player's Lua state.

Parameters:

- `player_selector` : (only when called on the server) RPC player selector (see `rpcplayerselector.lua`)
- `function_name` : string of the name of the global function to call.
- ... parameters to send to the function.

Returns:

nil

Usage:

```
-- There are restrictions on the types of parameters that can be sent.
-- Supported types include: number, boolean, string, CObjectRef, and limited table support
-- Tables can include the above types, and tables can also include other tables.
-- However, tables will not maintain references to tables, which means the same table included in several
places will result in new table instances each time that table is referenced.

local inner_table = { test = "this table will result in several new tables!" }
local t = {
    ["string key"] = 123,
    [5] = "string value",
    ["table1"] = inner_table,
    ["table2"] = inner_table, -- notice: table2 is not table1 in the RPC function!
}

Send_AI_Function_Call(ai_player_var, "A_Global_AI_Function", 123, "some parameter", true, t)
```

ClientCamera

Various functionality that affects the client.

Functions

Client_Set_Camera_Distance (distance)	Sets the camera distance.
Client_Set_Camera_Pitch (distance)	Sets the camera pitch.
Client_Set_Camera_Yaw (distance)	Sets the camera yaw.
Client_Set_Camera_FOV (distance)	Sets the camera field of view.
Client_Set_Camera_Look_At_Position (position_3d)	Sets the camera look at position instantly.
Client_Set_Camera_Look_At_Position_With_Speed (position_3d, speed)	Sets the camera look at position with a camera movement speed specified.
Client_Set_Camera_Look_At_Region (ID)	Sets the camera look at region instantly.
Client_Set_Camera_Look_At_Region_With_Speed (ID, speed)	Sets the camera look at region with a camera movement speed specified.
Client_Set_Unrestricted_Camera ()	Unlocks the camera, allowing for various camera parameters to be set without clamping.
Client_Reset_Camera ()	Resets the camera, setting the various camera clamping parameters back to their defaults.
Client_Set_Camera_Locked (is_locking)	Locks or unlocks the camera.
Client_Set_Camera_Distance (distance, time_seconds)	Sets the camera distance over time with ease in/out.

Functions

Client_Set_Camera_Distance (distance)

Sets the camera distance.

Parameters:

- distance : number value of the new distance

Returns:

nil

Client_Set_Camera_Pitch (distance)

Sets the camera pitch.

Parameters:

- distance : number value of the new pitch

Returns:

nil

Client_Set_Camera_Yaw (distance)

Sets the camera yaw.

Parameters:

- distance : number value of the new yaw

Returns:

nil

Client_Set_Camera_FOV (distance)

Sets the camera field of view.

Parameters:

- distance : number value of the new FOV

Returns:

nil

Client_Set_Camera_Look_At_Position (position_3d)

Sets the camera look at position instantly.

Parameters:

- position_3d : LVector3 of the position to look at.

Returns:

nil

Client_Set_Camera_Look_At_Position_With_Speed (position_3d, speed)

Sets the camera look at position with a camera movement speed specified.

Parameters:

- position_3d : LVector3 of the position to look at.
- speed : number specifying speed of camera movement.

Returns:

nil

Client_Set_Camera_Look_At_Region (ID)

Sets the camera look at region instantly.

Parameters:

- ID : ID of the region to look at

Returns:

nil

Client_Set_Camera_Look_At_Region_With_Speed (ID, speed)

Sets the camera look at region with a camera movement speed specified.

Parameters:

- ID : ID of the region to look at
- speed : number specifying speed of camera movement.

Returns:

nil

Client_Set_Unrestricted_Camera ()

Unlocks the camera, allowing for various camera parameters to be set without clamping.

Returns:

nil

Client_Reset_Camera ()

Resets the camera, setting the various camera clamping parameters back to their defaults.

Returns:

nil

Client_Set_Camera_Locked (is_locking)

Locks or unlocks the camera. When locked, users will be unable to do various camera operations such as translation and rotation.

Parameters:

- `is_locking` : boolean When true, the camera is locked. When false, the camera is unlocked.

Client_Set_Camera_Distance (distance, time_seconds)

Sets the camera distance over time with ease in/out.

Parameters:

- `distance` : number value of the new distance
- `time_seconds` : number of seconds over which to transition to the new distance.

Returns:

nil

ClientCinematic

Module ClientCinematic

Lua client-specific functionality for cinematics.

Functions

Client_Setup_Cinematic_Signals ()	Starts listening to cinematic signals that the client causes during cinematic playback.
--	---

Functions

Client_Setup_Cinematic_Signals ()

Starts listening to cinematic signals that the client causes during cinematic playback. Required to call this once in the client-side in order for the server to receive cinematic events.

Returns:

nil

ClientGUIComponent

Various functionality that affects the client.

Functions

Load_GUI_Scene (scene_name)	Loads a GUI scene by file name.
------------------------------------	---------------------------------

Functions

Load_GUI_Scene (scene_name)

Loads a GUI scene by file name.

Parameters:

- scene_name : string

Returns:

nil

ClientGUIUtils

Various functionality that affects the client.

Functions

Client_Animate_GUI (player_selector, animation_name, scene_name, element_name, forward, looping)	Animates a GUI element.
Client_Radar_Ping (player_selector, vector3, radar_event_type_name)	Pings the radar map.
Client_Block_Event_Notification (player_selector, notification)	Blocks a specific event notification.
Client_Unblock_Event_Notification (player_selector, notification)	Unblocks a specific event notification.
Client_Block_All_Event_Notifications (player_selector)	Blocks all event notifications.
Client_Unblock_All_Event_Notifications (player_selector)	Unblocks all event notifications.
Client_Show_UI_Component (player_selector, scene_name, component_name)	Shows a UI component.
Client_Hide_UI_Component (player_selector, scene_name, component_name)	Hides a UI component.
Client_Start_Texture_Animation (player_selector, scene_name, component_name, looping)	Starts texture animation on a UI component.
Client_Stop_Texture_Animation (player_selector, scene_name, component_name)	Stops texture animation on a UI component.
Client_Set_Component_Text (player_selector, scene_name, component_name, text_id)	Sets text on a UI component.
Client_Hide_UI_Component (player_selector, scene_name, component_name)	Enables a UI component.
Client_Hide_UI_Component (player_selector, scene_name, component_name)	Disables a UI component.
Client_Script_Disable_Or_Hide_UI_Component (scene_name, component_name, true, true)	Disables a UI component and prevents it from being enabled until the disable is removed.
Client_Remove_Script_Disable_Or_Hide_UI_Component (scene_name, component_name)	Removes Disable of a UI component
Client_Show_Hint (player_selector, hint_name)	Shows a hint.
Client_Open_Hint (player_selector, hint_name)	Opens a hint.
Client_Play_Cinematic (player_selector, cinematic_file, show_gui, show_fow)	Plays a cinematic.
Client_Queue_Cinematic (player_selector, cinematic_file, show_gui, show_fow)	Queues a cinematic.
Client_Play_Cinematic_At (player_selector, cinematic_file, show_gui, show_fow, at_object)	Plays a cinematic at an object's location.
Client_Queue_Cinematic_At (player_selector, cinematic_file, show_gui, show_fow, at_object)	Queues a cinematic at an object's location.
Client_Abort_Cinematics (player_selector)	Aborts cinematics.
Client_Strategic_Set_Highlight_Region (region_id)	Turns on/off highlight for a region on the map.
Client_Strategic_Toggle_Region_Name_Display (show, region_id)	Turns on/off name/star display for specified region.
Client_Strategic_Enable_Region_Selection (show, region_id)	Turns on/off selectability for specified region.
Client_Strategic_Enable_Region_Deselection (show, region_id)	Turns on/off deselection for specified region.

LuaAPI_Strategic_Enable_Region_Attack_Target (show, region_id)	Turns on/off ability to attack specified region.
Client_Strategic_Enable_Region_Move_Corps (show, region_id)	Turns on/off ability to move corps for specified region.
Client_Strategic_Enable_Region_Move_Corps_Into (show, region_id)	Turns on/off ability to move corps for specified region.
Client_Strategic_Set_Corps_Move_Selection_Count (region_id, corps_type_name, count)	Turns on/off ability to move corps for specified region.
Client_Strategic_Set_Purchase_Limit (region_id, purchase_index, max_count)	Turns on/off ability to move corps for specified region.
Client_Cancel_Direct_Placement ()	Turns off placement cursor.
Client_Strategic_Play_Movie (movie_name)	Plays a movie in strategic.
Client_Strategic_Quit_To_Main ()	Quit to main menu.
Client_Main_Menu_Jump_To_Last_Screen ()	Query whether the main menu will be jumping directly to a selection screen after initialization.
Client_Enable_FOW (enable)	Enables / Disables FOW on Client.
Client_Start_Bark (player_selector, bark_name)	Starts playing a bark.
Client_End_Bark (player_selector)	Stops the currently playing bark.
Client_Start_SFX_Event_2D (player_selector, sfx_event_name)	Starts playing a 2D SFXEvent.
Client_Set_GUI_Mouse_Input_Locked (player_selector, locked, auto_unlock_seconds)	Lock/Unlock GUI mouse events, for GUI screen transitions and such - to temporarily block button clicks, mouse-overs, tooltip, etc
Client_Set_Unit_Button_Select_Input_Locked (player_selector, locked)	Lock/Unlock unit tray button select click input.
Client_Set_Unit_Button_Move_Camera_Input_Locked (player_selector, locked)	Lock/Unlock unit tray button move camera double-click input.
Client_Cinematic_Register_Frame (player_selector, cinematic_file, frame)	Causes the client to send a frame event on the given frame for the given cinematic file.
Client_Cinematic_Unregister_Frame (player_selector, cinematic_file, frame)	Clears the registered cinematic frame event for the given file and frame.
Get_Profile_String (key_name)	Get a string value associated with the current player/steam profile.
Set_Profile_String (key_name, value)	Set a string value associated with the current player/steam profile.
Set_JSON_Persistence (json_string)	Sets JSON persistence blob that can be used to pass data from one instance to another across a switch and load instance boundary.
Get_JSON_Persistence ()	Gets JSON persistence blob that can be used to pass data from one instance to another across a switch and load instance boundary.
Set_JSON_Campaign_Profile (json_string)	Sets JSON profile blob that can be used to save data associated with the current campaign.
Get_JSON_Campaign_Profile ()	Gets JSON profile blob that can be used to retrieve data associated with the current campaign.
Client_Unblock_All_Event_Notifications (player_selector)	Disables upgrade/delete buttons on existing attached GUI.
Client_Set_Cinematic_Mode_Enabled (player_selector, enable)	Toggles the cinematic mode on or off.
Client_Set_Tactical_Victory_State (value)	Specify the Victory Scale for a tactical campaign battle
Client_Enable_Mod_Debugging_Tools ()	Enables the mod debugging UI and related functionality.

rpcplayerselector.lua

```
local pgm_clientobjectivesdisplay = PG_Import("Libs.Client.ClientObjectiveDisplay")
--[
```

Calling a client rpc function from the server requires a player selector as the initial argument. A player selector tells the server which clients to send the function call to.

Valid values for player selectors include:

- the boolean value true : sends to all players
- a CObjectRef of a player : sends to just that player
- a table of CObjectRef of players : sends to every player included in the table

Examples follow:

```
--]]

-- sends the following RPC calls to all human players
pgm_clientobjectivesdisplay.Add_Objective(true, "destroy_objective", "Destroy 15 structures.")
pgm_clientobjectivesdisplay.Set_Objective_Star_Type(true, "destroy_objective", "Gold")

-- sends the following RPC call to a single human player
local players = Create_ObjectSet()
Get_Players(players)
for player in ObjectSet_Values(players) do
    if Is_Player_Human(player) then
        pgm_clientobjectivesdisplay.Add_Objective(player, "survive_objective", "Survive 10 waves.")
        break
    end
end

-- sends the following RPC call to all human players in the table
local human_players = {}
local potential_players = Create_ObjectSet()
Get_Players(potential_players)
for player in ObjectSet_Values(potential_players) do
    if Is_Player_Human(player) then
        human_players[#human_players + 1] = player
        break
    end
end

pgm_clientobjectivesdisplay.Add_Objective(human_players, "hidden_objective", "Find the secret treasure.")
```

Functions

Client_Animate_GUI (player_selector, animation_name, scene_name, element_name, forward, looping)

Animates a GUI element.

Parameters:

- player_selector : (only when called on the server) RPC player selector (see rpcplayerselector.lua)
- animation_name : string
- scene_name : string
- element_name : string
- forward : boolean
- looping : boolean

Returns:

nil

Client_Radar_Ping (player_selector, vector3, radar_event_type_name)

Pings the radar map.

Parameters:

- player_selector : (only when called on the server) RPC player selector (see rpcplayerselector.lua)

- vector3 : An LVector3 designating the coordinate to ping.
- radar_event_type_name : string of the type of radar event

Returns:

nil

Client_Block_Event_Notification (player_selector, notification)

Blocks a specific event notification.

Parameters:

- player_selector : (only when called on the server) RPC player selector (see rpcplayerselector.lua)
- notification : string of the notification to block

Returns:

nil

Client_Unblock_Event_Notification (player_selector, notification)

Unblocks a specific event notification.

Parameters:

- player_selector : (only when called on the server) RPC player selector (see rpcplayerselector.lua)
- notification : string of the notification to unblock.

Returns:

nil

Client_Block_All_Event_Notifications (player_selector)

Blocks all event notifications.

Parameters:

- player_selector : (only when called on the server) RPC player selector (see rpcplayerselector.lua)

Returns:

nil

Client_Unblock_All_Event_Notifications (player_selector)

Unblocks all event notifications.

Parameters:

- player_selector : (only when called on the server) RPC player selector (see rpcplayerselector.lua)

Returns:

nil

Client_Show_UI_Component (player_selector, scene_name, component_name)

Shows a UI component.

Parameters:

- player_selector : (only when called on the server) RPC player selector (see rpcplayerselector.lua)
- scene_name : string

- component_name : string

Returns:

nil

Client_Hide_UI_Component (player_selector, scene_name, component_name)

Hides a UI component.

Parameters:

- player_selector : (only when called on the server) RPC player selector (see rpcplayerselector.lua)
- scene_name : string
- component_name : string

Returns:

nil

Client_Start_Texture_Animation (player_selector, scene_name, component_name, looping)

Starts texture animation on a UI component.

Parameters:

- player_selector : (only when called on the server) RPC player selector (see rpcplayerselector.lua)
- scene_name string
- component_name string
- looping bool

Returns:

nil

Client_Stop_Texture_Animation (player_selector, scene_name, component_name)

Stops texture animation on a UI component.

Parameters:

- player_selector : (only when called on the server) RPC player selector (see rpcplayerselector.lua)
- scene_name : string
- component_name : string

Returns:

nil

Client_Set_Component_Text (player_selector, scene_name, component_name, text_id)

Sets text on a UI component.

Parameters:

- player_selector : (only when called on the server) RPC player selector (see rpcplayerselector.lua)
- scene_name : string
- component_name : string
- text_id : string

Returns:

nil

Client_Hide_UI_Component (player_selector, scene_name, component_name)

Enables a UI component.

Parameters:

- player_selector : (only when called on the server) RPC player selector (see rpcplayerselector.lua)
- scene_name : string
- component_name : string

Returns:

nil

Client_Hide_UI_Component (player_selector, scene_name, component_name)

Disables a UI component.

Parameters:

- player_selector : (only when called on the server) RPC player selector (see rpcplayerselector.lua)
- scene_name : string
- component_name : string

Returns:

nil

Client_Script_Disable_Or_Hide_UI_Component (scene_name, component_name, true, true)

Disables a UI component and prevents it from being enabled until the disable is removed.

Parameters:

- scene_name : string
- component_name : string
- true - hidden, false - enabled
- true - hidden, false - enabled

Returns:

nil

Client_Remove_Script_Disable_Or_Hide_UI_Component (scene_name, component_name)

Removes Disable of a UI component

Parameters:

- scene_name : string
- component_name : string

Returns:

nil

Client_Show_Hint (player_selector, hint_name)

Shows a hint.

Parameters:

- player_selector : (only when called on the server) RPC player selector (see rpcplayerselector.lua)
- hint_name : string

Returns:

nil

Client_Open_Hint (player_selector, hint_name)

Opens a hint.

Parameters:

- player_selector : (only when called on the server) RPC player selector (see rpcplayerselector.lua)
- hint_name : string

Returns:

nil

Client_Play_Cinematic (player_selector, cinematic_file, show_gui, show_fow)

Plays a cinematic.

Parameters:

- player_selector : (only when called on the server) RPC player selector (see rpcplayerselector.lua)
- cinematic_file : string of the cinematic file
- show_gui : boolean Whether or not to display the GUI during the cinematic
- show_fow : boolean Whether or not to display fog of war during the cinematic

Returns:

nil

Client_Queue_Cinematic (player_selector, cinematic_file, show_gui, show_fow)

Queues a cinematic.

Parameters:

- player_selector : (only when called on the server) RPC player selector (see rpcplayerselector.lua)
- cinematic_file : string of the cinematic file
- show_gui : boolean Whether or not to display the GUI during the cinematic
- show_fow : boolean Whether or not to display fog of war during the cinematic

Returns:

nil

Client_Play_Cinematic_At (player_selector, cinematic_file, show_gui, show_fow, at_object)

Plays a cinematic at an object's location.

Parameters:

- player_selector : (only when called on the server) RPC player selector (see rpcplayerselector.lua)
- cinematic_file : string of the cinematic file
- show_gui : boolean Whether or not to display the GUI during the cinematic
- show_fow : boolean Whether or not to display fog of war during the cinematic
- at_object : CObjectRef of the object to play the cinematic at

Returns:

nil

Client_Queue_Cinematic_At (player_selector, cinematic_file, show_gui, show_fow, at_object)

Queues a cinematic at an object's location.

Parameters:

- player_selector : (only when called on the server) RPC player selector (see rpcplayerselector.lua)
- cinematic_file : string of the cinematic file
- show_gui : boolean Whether or not to display the GUI during the cinematic
- show_fow : boolean Whether or not to display fog of war during the cinematic
- at_object : CObjectRef of the object to play the cinematic at

Returns:

nil

Client_Abort_Cinematics (player_selector)

Aborts cinematics.

Parameters:

- player_selector : (only when called on the server) RPC player selector (see rpcplayerselector.lua)

Returns:

nil

Client_Strategic_Set_Highlight_Region (region_id)

Turns on/off highlight for a region on the map.

Parameters:

- region_id : int, the region to highlight, -1 to highlight none, int which highlight to use.

Returns:

nil

Client_Strategic_Toggle_Region_Name_Display (show, region_id)

Turns on/off name/star display for specified region.

Parameters:

- show : boolean, whether or not to show the name/stars for this region
- region_id : int, what region to do this on, -1 to specify all regions

Returns:

nil

Client_Strategic_Enable_Region_Selection (show, region_id)

Turns on/off selectability for specified region.

Parameters:

- show : boolean, whether or not to allow the selection of this region.
- region_id : int, what region to do this on, -1 to specify all regions.

Returns:

nil

Client_Strategic_Enable_Region_Deselection (show, region_id)

Turns on/off deselection for specified region.

Parameters:

- show : boolean, whether or not to allow the deselection of this region.
- region_id : int, what region to do this on, -1 to specify all regions.

Returns:

nil

LuaAPI_Strategic_Enable_Region_Attack_Target (show, region_id)

Turns on/off ability to attack specified region.

Parameters:

- show : boolean, whether or not to allow the attack of this region.
- region_id : int, what region to do this on, -1 to specify all regions.

Returns:

nil

Client_Strategic_Enable_Region_Move_Corps (show, region_id)

Turns on/off ability to move corps for specified region.

Parameters:

- show : boolean, whether or not to allow the selection of this region.
- region_id : int, what region to do this on, -1 to specify all regions.

Returns:

nil

Client_Strategic_Enable_Region_Move_Corps_Into (show, region_id)

Turns on/off ability to move corps for specified region.

Parameters:

- show : boolean, whether or not to allow the movement of corps into this region.
- region_id : int, what region to do this on, -1 to specify all regions.

Returns:

nil

Client_Strategic_Set_Corps_Move_Selection_Count (region_id, corps_type_name, count)

Turns on/off ability to move corps for specified region.

Parameters:

- region_id : int, what region to do this on
- corps_type_name : string, what corps to do this on
- count : int, desired selection count

Returns:

nil

Client_Strategic_Set_Purchase_Limit (region_id, purchase_index, max_count)

Turns on/off ability to move corps for specified region.

Parameters:

- region_id : int, what region to do this on
- purchase_index : int, button index
- max_count : int, desired max purchase count

Returns:

nil

Client_Cancel_Direct_Placement ()

Turns off placement cursor.

Returns:

nil

Client_Strategic_Play_Movie (movie_name)

Plays a movie in strategic.

Parameters:

- movie_name : string, what corps to do this on

Returns:

nil

Client_Strategic_Quit_To_Main ()

Quit to main menu.

Returns:

nil

Client_Main_Menu_Jump_To_Last_Screen ()

Query whether the main menu will be jumping directly to a selection screen after initialization.

Returns:

nil

Client_Enable_FOW (enable)

Enables / Disables FOW on Client.

Parameters:

- enable : bool

Returns:

nil

Client_Start_Bark (player_selector, bark_name)

Starts playing a bark.

Parameters:

- player_selector : (only when called on the server) RPC player selector (see rpcplayerselector.lua)
- bark_name : string, name of the bark to play.

Returns:

nil

Client_End_Bark (player_selector)

Stops the currently playing bark.

Parameters:

- player_selector : (only when called on the server) RPC player selector (see rpcplayerselector.lua)

Returns:

nil

Client_Start_SFX_Event_2D (player_selector, sfx_event_name)

Starts playing a 2D SFXEvent.

Parameters:

- player_selector : (only when called on the server) RPC player selector (see rpcplayerselector.lua)
- sfx_event_name : string name of the SFXEvent to play

Returns:

nil

Client_Set_GUI_Mouse_Input_Locked (player_selector, locked, auto_unlock_seconds)

Lock/Unlock GUI mouse events, for GUI screen transitions and such - to temporarily block button clicks, mouse-overs, tooltip, etc

Parameters:

- player_selector : (only when called on the server) RPC player selector (see rpcplayerselector.lua)
- locked : boolean, locked toggle value.
- auto_unlock_seconds : float, automatic unlock seconds after locking.

Returns:

nil

Client_Set_Unit_Button_Select_Input_Locked (player_selector, locked)

Lock/Unlock unit tray button select click input.

Parameters:

- player_selector : (only when called on the server) RPC player selector (see rpcplayerselector.lua)
- locked : boolean, locked toggle value.

Returns:

nil

Client_Set_Unit_Button_Move_Camera_Input_Locked (player_selector, locked)

Lock/Unlock unit tray button move camera double-click input.

Parameters:

- player_selector : (only when called on the server) RPC player selector (see rpcplayerselector.lua)
- locked : boolean, locked toggle value.

Returns:

nil

Client_Cinematic_Register_Frame (player_selector, cinematic_file, frame)

Causes the client to send a frame event on the given frame for the given cinematic file.

Parameters:

- player_selector : (only when called on the server) RPC player selector (see rpcplayerselector.lua)
- cinematic_file : string of the cinematic file
- frame number of the frame for the event

Returns:

nil

Client_Cinematic_Unregister_Frame (player_selector, cinematic_file, frame)

Clears the registered cinematic frame event for the given file and frame.

Parameters:

- player_selector : (only when called on the server) RPC player selector (see rpcplayerselector.lua)
- cinematic_file : string of the cinematic file
- frame number of the frame for the event

Returns:

nil

Get_Profile_String (key_name)

Get a string value associated with the current player/steam profile.

Parameters:

- key_name : string specifying the key name of the player profile item to retrieve.

Returns:

string value associated with the player profile and key name.

Set_Profile_String (key_name, value)

Set a string value associated with the current player/steam profile.

Parameters:

- key_name : string specifying the key name to associate with the value.
- value : string value to save with the player profile and key name.

Returns:

none

Set_JSON_Persistence (json_string)

Sets JSON persistence blob that can be used to pass data from one instance to another across a switch and load instance boundary.

Parameters:

- json_string : json string to be used for persistence.

Returns:

none

Get_JSON_Persistence ()

Gets JSON persistence blob that can be used to pass data from one instance to another across a switch and load instance boundary.

Returns:

json string

Set_JSON_Campaign_Profile (json_string)

Sets JSON profile blob that can be used to save data associated with the current campaign.

Parameters:

- json_string : json string to be used for persistence.

Returns:

none

Get_JSON_Campaign_Profile ()

Gets JSON profile blob that can be used to retrieve data associated with the current campaign.

Returns:

json string

Client_Unblock_All_Event_Notifications (player_selector)

Disables upgrade/delete buttons on existing attached GUI. Objects placed after this call will still have the upgrade/delete buttons enabled.

Parameters:

- player_selector : (only when called on the server) RPC player selector (see rpcplayerselector.lua)

Returns:

nil

Client_Set_Cinematic_Mode_Enabled (player_selector, enable)

Toggles the cinematic mode on or off. Various UI elements are hidden when the cinematic mode is turned on.

Parameters:

- player_selector: (only when called on the server) RPC player selector (see rpcplayerselector.lua)
- enable : boolean, cinematic mode is turned on when true and is turned off when false

Returns:

nil

Client_Set_Tactical_Victory_State (value)

Specify the Victory Scale for a tactical campaign battle.

Parameters:

- value

Returns:

nil

Client_Enable_Mod_Debugging_Tools ()

Enables the mod debugging UI and related functionality.

Returns:

nil

ClientObjectManipulation

Lua client-specific functionality for dealing with objects.

Functions

Client_Set_Unit_Colorization (<i>player_selector</i> , <i>object_ref</i> , <i>faction_type_name</i> , <i>team_id</i>)	Sets the unit's colorization settings based on the given faction and team id.
--	---

```
rpcplayerselector.lua

local pgm_clientobjectivesdisplay = PG_Import("Libs.Client.ClientObjectiveDisplay")

--[[
Calling a client rpc function from the server requires a player selector as the initial argument.
A player selector tells the server which clients to send the function call to.
Valid values for player selectors include:
    - the boolean value true : sends to all players
    - a CObjectRef of a player : sends to just that player
    - a table of CObjectRef of players : sends to every player included in the table

Examples follow:
--]]

-- sends the following RPC calls to all human players
pgm_clientobjectivesdisplay.Add_Objective(true, "destroy_objective", "Destroy 15 structures.")
pgm_clientobjectivesdisplay.Set_Objective_Star_Type(true, "destroy_objective", "Gold")

-- sends the following RPC call to a single human player
local players = Create_ObjectSet()
Get_Players(players)
for player in ObjectSet_Values(players) do
    if Is_Player_Human(player) then
        pgm_clientobjectivesdisplay.Add_Objective(player, "survive_objective", "Survive 10 waves.")
        break
    end
end

-- sends the following RPC call to all human players in the table
local human_players = {}
local potential_players = Create_ObjectSet()
Get_Players(potential_players)
for player in ObjectSet_Values(potential_players) do
    if Is_Player_Human(player) then
        human_players[#human_players + 1] = player
        break
    end
end

pgm_clientobjectivesdisplay.Add_Objective(human_players, "hidden_objective", "Find the secret treasure.")
```

Functions

Client_Set_Unit_Colorization (*player_selector*, *object_ref*, *faction_type_name*, *team_id*)

Sets the unit's colorization settings based on the given faction and team id.

Parameters:

- *player_selector* : (only when called on the server) RPC player selector (see `rpcplayerselector.lua`)
- *object_ref* : CObjectRef of unit

- `faction_type_name` : string of the faction type name
- `team_id` : number of the team id to use.

Returns:

`nil`

ClientObjectivesDisplay

Allows manipulating objectives in the built in objective UI.

This is a module that directly interacts with the client. For a more user-friendly module, see ServerObjectives.

Functions

Add_Objective (player_selector, id, objective_text)	Add a new objective with a given ID and display text.
Remove_Objective (player_selector, id)	Removes the objective with the given ID if it exists.
Set_Objective_Completion_State (player_selector, id, completion_state)	Sets the completion state of the objective specified by the given ID.
Set_Objective_Text (player_selector, id, objective_text)	Sets the displayed text to the given localized text value for the given objective specified by the given ID.
Set_Objective_Text_Non_Localized (player_selector, id, objective_text)	Sets the displayed text to the passed in value (and not a localized text value) for the given objective specified by the given ID.
Set_Objective_Star_Type (player_selector, id, star_type)	Sets the shown star type of the objective with the given ID.
Set_Objective_Timer_Display (player_selector, id, seconds)	Adds a counting down timer display next to the objective text for the given objective.
Set_Objective_Counter_Data (player_selector, id, counter_value, counter_max)	Adds a counter display of the form [X/Y] to the objective text.
Remove_Objective_Timer (player_selector, id)	Removes the visible timer from the objective.

Tables

ObjectiveStarType	The types of stars objectives may display.
ObjectiveCompletionState	The states objectives can be in.

rpcplayerselector.lua

```
local pgm_clientobjectivesdisplay = PG_Import("Libs.Client.ClientObjectiveDisplay")

--[
Calling a client rpc function from the server requires a player selector as the initial argument.
A player selector tells the server which clients to send the function call to.
Valid values for player selectors include:
    - the boolean value true : sends to all players
    - a CObjectRef of a player : sends to just that player
    - a table of CObjectRef of players : sends to every player included in the table

Examples follow:
--]]

-- sends the following RPC calls to all human players
pgm_clientobjectivesdisplay.Add_Objective(true, "destroy_objective", "Destroy 15 structures.")
pgm_clientobjectivesdisplay.Set_Objective_Star_Type(true, "destroy_objective", "Gold")

-- sends the following RPC call to a single human player
local players = Create_ObjectSet()
Get_Players(players)
for player in ObjectSet_Values(players) do
    if Is_Player_Human(player) then
        pgm_clientobjectivesdisplay.Add_Objective(player, "survive_objective", "Survive 10 waves.")
        break
    end
end
end
```

```

-- sends the following RPC call to all human players in the table
local human_players = {}
local potential_players = Create_ObjectSet()
Get_Players(potential_players)
for player in ObjectSet_Values(potential_players) do
    if Is_Player_Human(player) then
        human_players[#human_players + 1] = player
        break
    end
end
end

pgm_clientobjectivesdisplay.Add_Objective(human_players, "hidden_objective", "Find the secret treasure.")

```

Functions

Add_Objective (player_selector, id, objective_text)

Add a new objective with a given ID and display text. IDs are unique. If an objective already has the given ID, nothing occurs.

Parameters:

- player_selector : (only when called on the server) RPC player selector (see rpcplayerselector.lua)
- id : string of unique ID
- objective_text : string of localized text ID containing text to display in the UI.

Returns:

nil

Remove_Objective (player_selector, id)

Removes the objective with the given ID if it exists.

Parameters:

- player_selector : (only when called on the server) RPC player selector (see rpcplayerselector.lua)
- id : string of objective's ID

Returns:

nil

Set_Objective_Completion_State (player_selector, id, completion_state)

Sets the completion state of the objective specified by the given ID.

Parameters:

- player_selector : (only when called on the server) RPC player selector (see rpcplayerselector.lua)
- id : string of objective's ID
- completion_state : ObjectiveCompletionState

Returns:

nil

Set_Objective_Text (player_selector, id, objective_text)

Sets the displayed text to the given localized text value for the given objective specified by the given ID.

Parameters:

- `player_selector` : (only when called on the server) RPC player selector (see `rpcplayerselector.lua`)
- `id` : string of objective's ID
- `objective_text` : string of localized text ID containing text to display in the UI.

Returns:

nil

Set_Objective_Text_Non_Localized (player_selector, id, objective_text)

Sets the displayed text to the passed in value (and not a localized text value) for the given objective specified by the given ID.

Parameters:

- `player_selector` : (only when called on the server) RPC player selector (see `rpcplayerselector.lua`)
- `id` : string of objective's ID
- `objective_text` : Exact text to display in the objective.

Returns:

nil

Set_Objective_Star_Type (player_selector, id, star_type)

Sets the shown star type of the objective with the given ID.

Parameters:

- `player_selector` : (only when called on the server) RPC player selector (see `rpcplayerselector.lua`)
- `id` : string of objective's ID
- `star_type` : ObjectiveStarType

Returns:

nil

Set_Objective_Timer_Display (player_selector, id, seconds)

Adds a counting down timer display next to the objective text for the given objective. Note that no functionality is attached to when the timer completes the countdown. Users should have their own timer to, for example, fail the objective.

Parameters:

- `player_selector` : (only when called on the server) RPC player selector (see `rpcplayerselector.lua`)
- `id` : string of objective's ID
- `seconds` : How many seconds the timer starts with

Returns:

nil

Set_Objective_Counter_Data (player_selector, id, counter_value, counter_max)

Adds a counter display of the form [X/Y] to the objective text. Note that no functionality is attached to when the counter is full. Users would have to have their own logic to, for example, complete the objective when the counter is full.

Parameters:

- `player_selector` : (only when called on the server) RPC player selector (see `rpcplayerselector.lua`)
- `id` : string of objective's ID
- `counter_value` : number for the current counter value
- `counter_max` : number for the maximum counter value

Returns:

nil

Remove_Objective_Timer (player_selector, id)

Removes the visible timer from the objective.

Parameters:

- player_selector : (only when called on the server) RPC player selector (see rpcplayerselector.lua)
- id : string of objective's ID

Returns:

nil

Tables

ObjectiveStarType

The types of stars objectives may display.

Fields:

- None
- Bronze
- Silver
- Gold

ObjectiveCompletionState

The states objectives can be in.

Fields:

- InProgress
- Complete
- Failed

ClientPGGUI

Various functionality that affects the client.

Functions

Delete_GUI_Scene_Ref (CGUIScene)	Deletes a GUI scene reference.
---	--------------------------------

Functions

Delete_GUI_Scene_Ref (CGUIScene)

Deletes a GUI scene reference.

Parameters:

- CGUIScene ref to be deleted.

Returns:

nil

ClientPlayers

Lua client-specific functionality for dealing with players.

Functions

Client_Get_Local_Player ()	Returns a CObjectRef for the local player.
-----------------------------------	--

Functions

Client_Get_Local_Player ()

Returns a CObjectRef for the local player.

Returns:

CObjectRef of player

GUIAdvancedTicker

Module GUIAdvancedTicker

Functions to manipulate an AdvancedTicker GUI component.

Class GUIAdvancedTicker

GUIAdvancedTicker:Move_To_Section (section_id)	Moves to the given section.
GUIAdvancedTicker:Move_To_Card_Offset (section_id, offset_pct)	Moves to an offset of the given card.
GUIAdvancedTicker:Center_On_Card_Offset (section_id, offset_pct)	Moves to the offset from the center of the given card.
GUIAdvancedTicker:Get_Current_Center_Card_ID ()	Gets the ID of the current center card.
GUIAdvancedTicker:Clear_All ()	Clears all entries.
GUIAdvancedTicker:Set_Ticker_Follower (item_ref)	Sets a GUI component to follow the ticker.
GUIAdvancedTicker:Get_Ticker_Follower ()	Gets the current follower.
GUIAdvancedTicker:Add_Section_To_Ticker (desired_id)	Adds a new section to the ticker with the given desired ID.
GUIAdvancedTicker:Remove_Section_By_ID (remove_id)	Removes a section via ID.
GUIAdvancedTicker:Add_Header_To_Section (component_type, section_id)	Creates a new GUI component of the given type and makes it the header of the section specified by the ID.
GUIAdvancedTicker:Clone_Header_To_Section (CGUIComponent, section_id)	Clones the given GUI component and makes the clone the header of the section specified by the ID.
GUIAdvancedTicker:Insert_Header_Into_Section (CGUIComponent, section_id)	Inserts the given GUI component as the header of the section specified by the ID.
GUIAdvancedTicker:Remove_Header_From_Section (section_id)	Removes the header from the given section.
GUIAdvancedTicker:Add_Item_To_Section (component_type, section_id, section_id)	Creates a new GUI component of the given type and adds it to the given section with a desired ID value.
GUIAdvancedTicker:Clone_Item_To_Section (component_type, section_id, section_id)	Clones a given GUI component and adds the clone to the given section with a desired ID value.
GUIAdvancedTicker:Insert_Item_Into_Section (component_type, section_id, section_id)	Adds the given GUI component to the given section with a desired ID value.
GUIAdvancedTicker:Remove_Item_By_ID (section_id)	Removes an item by ID.
GUIAdvancedTicker:Get_Number_Of_Sections ()	Returns the number of sections.
GUIAdvancedTicker:Get_Number_Of_Items_In_Section (section_id)	Returns the number of items in the section.
GUIAdvancedTicker:Get_Section_ID_From_Index (section_index)	Returns the section ID of the section at the given index.
GUIAdvancedTicker:Get_Item_ID_From_Index_In_Section (index, section_id)	Returns the item ID of the item at the given index in the given section.
GUIAdvancedTicker:Get_Header_Item_By_Section_ID (section_id)	Returns the header GUI component of the given section.
GUIAdvancedTicker:Get_Item_By_ID (item_id)	Returns the item GUI component with the given ID.
GUIAdvancedTicker:Get_ID_Of_Item (item_ref)	Returns the ID of the given CGUIComponent

Class GUIAdvancedTicker

Contains functions to manipulate a GUIAdvancedTicker.

GUIAdvancedTicker:Move_To_Section (section_id)

Moves to the given section.

Parameters:

- section_id: number or CLuaCValue of the section ID

Returns:

nil

GUIAdvancedTicker:Move_To_Card_Offset (section_id, offset_pct)

Moves to an offset of the given card.

Parameters:

- section_id: number or CLuaCValue of the card ID
- offset_pct: number for the offset percentage

Returns:

nil

GUIAdvancedTicker:Center_On_Card_Offset (section_id, offset_pct)

Moves to the offset from the center of the given card.

Parameters:

- section_id: number or CLuaCValue of the card ID
- offset_pct: number for the offset percentage

Returns:

nil

GUIAdvancedTicker:Get_Current_Center_Card_ID ()

Gets the ID of the current center card.

Returns:

CLuaCValue

GUIAdvancedTicker:Clear_All ()

Clears all entries.

Returns:

nil

GUIAdvancedTicker:Set_Ticker_Follower (item_ref)

Sets a GUI component to follow the ticker. (?)

Parameters:

- item_ref: CGUIComponent of the follower

Returns:

nil

GUIAdvancedTicker:Get_Ticker_Follower ()

Gets the current follower.

Returns:

CGUIComponent or nil

GUIAdvancedTicker:Add_Section_To_Ticker (desired_id)

Adds a new section to the ticker with the given desired ID.

Parameters:

- desired_id: (optional) number or CLuaCValue for the desired ID

Returns:

CLuaCValue of the assigned ID

GUIAdvancedTicker:Remove_Section_By_ID (remove_id)

Removes a section via ID.

Parameters:

- remove_id: (optional) number or CLuaCValue for the ID to remove

Returns:

nil

GUIAdvancedTicker:Add_Header_To_Section (component_type, section_id)

Creates a new GUI component of the given type and makes it the header of the section specified by the ID.

Parameters:

- component_type: string of the type of GUI component to create
- section_id: CLuaCValue holding the section ID

Returns:

boolean for success or failure, or nil

GUIAdvancedTicker:Clone_Header_To_Section (CGUIComponent, section_id)

Clones the given GUI component and makes the clone the header of the section specified by the ID.

Parameters:

- CGUIComponent: of the item to clone
- section_id: CLuaCValue holding the section ID

Returns:

boolean for success or failure, or nil

GUIAdvancedTicker:Insert_Header_Into_Section (CGUIComponent, section_id)

Inserts the given GUI component as the header of the section specified by the ID.

Parameters:

- CGUIComponent: of the item to insert
- section_id: CLuaCValue holding the section ID

Returns:

boolean for success or failure, or nil

GUIAdvancedTicker:Remove_Header_From_Section (section_id)

Removes the header from the given section.

Parameters:

- section_id: CLuaCValue holding the section ID

Returns:

boolean for success or failure, or nil

GUIAdvancedTicker:Add_Item_To_Section (component_type, section_id, section_id)

Creates a new GUI component of the given type and adds it to the given section with a desired ID value.

Parameters:

- component_type: string of the type of GUI component to create
- section_id: (optional) number or CLuaCValue holding the desired ID
- section_id: (optional) number or CLuaCValue holding the desired ID

Returns:

CLuaCValue of the ID of the added item, or nil

GUIAdvancedTicker:Clone_Item_To_Section (component_type, section_id, section_id)

Clones a given GUI component and adds the clone to the given section with a desired ID value.

Parameters:

- component_type: string of the type of GUI component to create
- section_id: (optional) number or CLuaCValue holding the desired ID
- section_id: (optional) number or CLuaCValue holding the desired ID

Returns:

CLuaCValue of the ID of the added item, or nil

GUIAdvancedTicker:Insert_Item_Into_Section (component_type, section_id, section_id)

Adds the given GUI component to the given section with a desired ID value.

Parameters:

- component_type: string of the type of GUI component to create
- section_id: (optional) number or CLuaCValue holding the desired ID
- section_id: (optional) number or CLuaCValue holding the desired ID

Returns:

CLuaCValue of the ID of the added item, or nil

GUIAdvancedTicker:Remove_Item_By_ID (section_id)

Removes an item by ID.

Parameters:

- section_id: CLuaCValue holding the item ID

Returns:

boolean for success or failure, or nil

GUIAdvancedTicker:Get_Number_Of_Sections ()

Returns the number of sections.

Returns:

number of sections

GUIAdvancedTicker:Get_Number_Of_Items_In_Section (section_id)

Returns the number of items in the section.

Parameters:

- section_id: number or CLuaCValue holding the section ID

Returns:

number of items

GUIAdvancedTicker:Get_Section_ID_From_Index (section_index)

Returns the section ID of the section at the given index.

Parameters:

- section_index: number for section index

Returns:

CLuaCValue of section ID, or nil

GUIAdvancedTicker:Get_Item_ID_From_Index_In_Section (index, section_id)

Returns the item ID of the item at the given index in the given section.

Parameters:

- index: number for item index
- section_id: CLuaCValue for section ID

Returns:

CLuaCValue of item ID, or nil

GUIAdvancedTicker:Get_Header_Item_By_Section_ID (section_id)

Returns the header GUI component of the given section.

Parameters:

- section_id: CLuaCValue for section ID

Returns:

CGUIComponent, or nil

GUIAdvancedTicker:Get_Item_By_ID (item_id)

Returns the item GUI component with the given ID.

Parameters:

- item_id: CLuaCValue for item ID

Returns:

CGUIComponent, or nil

GUIAdvancedTicker:Get_ID_Of_Item (item_ref)

Returns the ID of the given CGUIComponent

Parameters:

- item_ref: CGUIComponent of the item

Returns:

CLuaCValue of item ID, or nil

GUIAnimation

Functions to manipulate an Animation GUI item.

Class GUIAnimation

GUIAnimation:Is_Valid ()	Returns a boolean specifying if the animation is valid.
GUIAnimation:Add_Key_Frame (application_name, time, data)	Returns a boolean specifying if the animation is valid.

Class GUIAnimation

Contains functions to manipulate a GUIAnimation.

GUIAnimation:Is_Valid ()

Returns a boolean specifying if the animation is valid.

Returns:

boolean

GUIAnimation:Add_Key_Frame (application_name, time, data)

Returns a boolean specifying if the animation is valid.

Parameters:

- application_name: string of the type of data; valid values are "PosX", "PosY", "SizeX", "SizeY", "Tint", and "Texture"
- time: number for the time the data will apply to
- data: table (array) containing the required values to apply to the desired application_name

Returns:

nil

Usage:

```
-- for specifying PosX, PosY, SizeX, and SizeY:  
data = { 123.456 } -- any number inside a table like so  
-- for specifying Tint  
data = { 1.0, 0.0, 1.0, 1.0 } -- four numbers in a table like so, specifying R G B A values  
-- for specifying Texture  
data = { "Blue.tga" } -- a string for the texture name inside a table like so
```

UIButtonBase

Functions to manipulate a UIButtonBase GUI component.

Class UIButtonBase

UIButtonBase:Set_Check_State (check_state)	Sets the check state of the button.
UIButtonBase:Get_Check_State ()	Returns the check state of the button.
UIButtonBase:Set_Checked (checked, optional, optional)	Sets the check state of the button.
UIButtonBase:Get_Is_Checked ()	Returns whether the button is checked or not.
UIButtonBase:Set_Is_Mouse_Over_Hilight_Enabled (enabled)	Sets whether the mouse over highlight is enabled or not.
UIButtonBase:Get_Is_Mouse_Over_Hilight_Enabled ()	Returns whether the mouse over highlight is enabled or not.
UIButtonBase:Highlight (highlight)	Sets whether the highlight is active or not.
UIButtonBase:Set_Is_Automated (is_automated)	Turns on/off the automated behavior.

Class UIButtonBase

Contains functions to manipulate a UIButtonBase.

UIButtonBase:Set_Check_State (check_state)

Sets the check state of the button.

Parameters:

- check_state: number of the check state (0 = unchecked, 1 = checked, 2 = partially checked)

Returns:

nil

UIButtonBase:Get_Check_State ()

Returns the check state of the button.

Returns:

number for the check state (see UIButtonBase:Set_Check_State)

UIButtonBase:Set_Checked (checked, optional, optional)

Sets the check state of the button.

Parameters:

- checked: number of the check state (0 = unchecked, 1 = checked, 2 = partially checked)
- optional:) boolean for whether to forcefully set the state (default = false)
- optional:) boolean for whether to forcefully set the state (default = false)

Returns:

nil

UIButtonBase:Get_Is_Checked ()

Returns whether the button is checked or not.

Returns:

boolean of check state

UIButtonBase:Set_Is_Mouse_Over_Hilight_Enabled (enabled)

Sets whether the mouse over highlight is enabled or not.

Parameters:

- enabled: boolean

Returns:

nil

UIButtonBase:Get_Is_Mouse_Over_Hilight_Enabled ()

Returns whether the mouse over highlight is enabled or not.

Returns:

boolean

UIButtonBase:Highlight (highlight)

Sets whether the highlight is active or not.

Parameters:

- highlight: boolean

Returns:

nil

UIButtonBase:Set_Is_Automated (is_automated)

Turns on/off the automated behavior. Automated behavior while active will allow the button to be checked/unchecked via mouse clicks.

Parameters:

- is_automated: boolean

Returns:

nil

GUICarousel

Functions to manipulate a Carousel GUI component.

Class GUICarousel

GUICarousel:Set_Front_Scale (scale)	Sets the front scale.
GUICarousel:Set_Back_Scale (scale)	Sets the back scale.
GUICarousel:Set_Active_Scale (scale)	Sets the active scale.
GUICarousel:Get_Front_Scale ()	Gets the front scale.
GUICarousel:Get_Back_Scale ()	Gets the back scale.
GUICarousel:Get_Active_Scale ()	Gets the active scale.
GUICarousel:Set_Front_Color (r, g, b, a)	Sets the front color.
GUICarousel:Set_Back_Color (r, g, b, a)	Sets the back color.
GUICarousel:Set_Active_Color (r, g, b, a)	Sets the active color.
GUICarousel:Get_Front_Color ()	Gets the front color.
GUICarousel:Get_Back_Color ()	Gets the back color.
GUICarousel:Get_Active_Color ()	Gets the active color.
GUICarousel:Set_Scroll_Speed (speed)	Sets the scroll speed.
GUICarousel:Get_Scroll_Speed ()	Gets the scroll speed.
GUICarousel:Get_Active_Component_ID ()	Gets the ID of the active component.
GUICarousel:Set_Active_Component (component_ref)	Sets the active component to the given GUI component.
GUICarousel:Get_Active_Component ()	Gets the active GUI component.
GUICarousel:Get_Active_Child_Count ()	Gets the number of active children.

Class GUICarousel

Contains functions to manipulate a GUICarousel.

GUICarousel:Set_Front_Scale (scale)

Sets the front scale.

Parameters:

- scale: number for the scale

Returns:

nil

GUICarousel:Set_Back_Scale (scale)

Sets the back scale.

Parameters:

- scale: number for the scale

Returns:

nil

GUICarousel:Set_Active_Scale (scale)

Sets the active scale.

Parameters:

- scale: number for the scale

Returns:

nil

GUICarousel:Get_Front_Scale ()

Gets the front scale.

Returns:

number

GUICarousel:Get_Back_Scale ()

Gets the back scale.

Returns:

number

GUICarousel:Get_Active_Scale ()

Gets the active scale.

Returns:

number

GUICarousel:Set_Front_Color (r, g, b, a)

Sets the front color.

Parameters:

- r: number color R value
- g: number color G value
- b: number color B value
- a: number color A value

Returns:

nil

GUICarousel:Set_Back_Color (r, g, b, a)

Sets the back color.

Parameters:

- r: number color R value
- g: number color G value
- b: number color B value
- a: number color A value

Returns:

nil

GUICarousel:Set_Active_Color (r, g, b, a)

Sets the active color.

Parameters:

- r: number color R value
- g: number color G value
- b: number color B value
- a: number color A value

Returns:

nil

GUICarousel:Get_Front_Color ()

Gets the front color.

Returns:

table containing "r", "g", "b", "a" number values

GUICarousel:Get_Back_Color ()

Gets the back color.

Returns:

table containing "r", "g", "b", "a" number values

GUICarousel:Get_Active_Color ()

Gets the active color.

Returns:

table containing "r", "g", "b", "a" number values

GUICarousel:Set_Scroll_Speed (speed)

Sets the scroll speed.

Parameters:

- speed: number

Returns:

nil

GUICarousel:Get_Scroll_Speed ()

Gets the scroll speed.

Returns:

number

GUICarousel:Get_Active_Component_ID ()

Gets the ID of the active component.

Returns:

number of active component's ID

GUICarousel:Set_Active_Component (component_ref)

Sets the active component to the given GUI component.

Parameters:

- component_ref: CGUIComponent item to set active

Returns:

nil

GUICarousel:Get_Active_Component ()

Gets the active GUI component.

Returns:

CGUIComponent of the active component

GUICarousel:Get_Active_Child_Count ()

Gets the number of active children.

Returns:

number

GUIClockProgress

Functions to manipulate a ClockProgress GUI component.

Class GUIClockProgress

GUIClockProgress:Set_Filled (fill_percent)	Sets the progress bar's filled percentage.
GUIClockProgress:Get_Filled ()	Gets the progress bar's filled percentage.
GUIClockProgress:Set_Texture (texture_name)	Sets the texture.
GUIClockProgress:Set_Autofill_Time (seconds_to_fill, seconds_elapsed)	Sets the progress bar to fill over time.

Class GUIClockProgress

Contains functions to manipulate a GUIClockProgress.

GUIClockProgress:Set_Filled (fill_percent)

Sets the progress bar's filled percentage.

Parameters:

- fill_percent: number for the fill amount (0 to 1)

Returns:

nil

GUIClockProgress:Get_Filled ()

Gets the progress bar's filled percentage.

Returns:

number

GUIClockProgress:Set_Texture (texture_name)

Sets the texture.

Parameters:

- texture_name: string of the texture name

Returns:

nil

GUIClockProgress:Set_Autofill_Time (seconds_to_fill, seconds_elapsed)

Sets the progress bar to fill over time.

Parameters:

- seconds_to_fill: number of seconds to take
- seconds_elapsed: (optional) number of seconds already taken (default = 0)

Returns:

nil

GUIComboBox

Functions to manipulate a ComboBox GUI component.

Class GUIComboBox

GUIComboBox:Clear ()	Clears the combo box.
GUIComboBox:Add_Item (text)	Adds an item with the given text to the combo box.
GUIComboBox:Add_Texture (texture_name)	Adds an item with the given texture to the combo box.
GUIComboBox:Set_Item_Color (index, r, g, b, a)	Sets the color of the item with the given index.
GUIComboBox:Get_Item_Count ()	Returns the number of items.
GUIComboBox:Refresh ()	Refreshes the combo box.
GUIComboBox:Set_Selected_Index (index, raise_event)	Sets the selected index.
GUIComboBox:Set_Selected_Text_Data (new_text, raise_event)	Sets the selected item by finding the item with matching text.
GUIComboBox:Get_Selected_Index ()	Gets the selected index.
GUIComboBox:Get_Selected_Text_Data ()	Gets the text of the item for the selected item.
GUIComboBox:Contains_Text_Data (text)	Returns whether or not the given text is in the combo box.
GUIComboBox:Set_Interactive (interactive)	Sets whether the combo box is interactive or not.
GUIComboBox:Get_Is_List_Box_Hidden ()	Returns whether the list box is hidden or not.

Class GUIComboBox

Contains functions to manipulate a GUIComboBox.

GUIComboBox:Clear ()

Clears the combo box.

Returns:

nil

GUIComboBox:Add_Item (text)

Adds an item with the given text to the combo box.

Parameters:

- text: string

Returns:

nil

GUIComboBox:Add_Texture (texture_name)

Adds an item with the given texture to the combo box.

Parameters:

- texture_name: string

Returns:

nil

GUIComboBox:Set_Item_Color (index, r, g, b, a)

Sets the color of the item with the given index.

Parameters:

- index: number
- r: number color R value
- g: number color G value
- b: number color B value
- a: number color A value

Returns:

nil

GUIComboBox:Get_Item_Count ()

Returns the number of items.

Returns:

number

GUIComboBox:Refresh ()

Refreshes the combo box.

Returns:

nil

GUIComboBox:Set_Selected_Index (index, raise_event)

Sets the selected index.

Parameters:

- index: number
- raise_event: (optional) boolean for whether to raise an event (default = true)

Returns:

nil

GUIComboBox:Set_Selected_Text_Data (new_text, raise_event)

Sets the selected item by finding the item with matching text.

Parameters:

- new_text: string of text to find
- raise_event: (optional) boolean for whether to raise an event (default = true)

Returns:

nil

GUIComboBox:Get_Selected_Index ()

Gets the selected index.

Returns:

number

GUIComboBox:Get_Selected_Text_Data ()

Gets the text of the item for the selected item.

Returns:

string

GUIComboBox:Contains_Text_Data (text)

Returns whether or not the given text is in the combo box.

Parameters:

- text: string of the text to find

Returns:

boolean

GUIComboBox:Set_Interactive (interactive)

Sets whether the combo box is interactive or not.

Parameters:

- interactive: boolean

Returns:

boolean

GUIComboBox:Get_Is_List_Box_Hidden ()

Returns whether the list box is hidden or not.

Returns:

boolean

GUIDecoratedIconButton

Functions to manipulate a DecoratedIconButton GUI component.

Class GUIDecoratedIconButton

GUIDecoratedIconButton:Set_Text (text_index, new_text)	Sets the text of the given text index.
GUIDecoratedIconButton:Set_Formatted_Text (text_index, text_id, ...)	Sets the formatted text of the given text index.
GUIDecoratedIconButton:Set_Quad_Texture (quad_index, texture_name)	Sets the texture of the given quad index.
GUIDecoratedIconButton:Get_Quad_Texture (quad_index)	Gets the texture name of the quad at the given index.
GUIDecoratedIconButton:Set_Quad_Hidden (quad_index, hidden)	Sets the quad visibility of the given quad index.
GUIDecoratedIconButton:Set_Quad_Tint (quad_index, r, g, b, a)	Sets the quad tint of the given quad index.
GUIDecoratedIconButton:Set_Text_Hidden (text_index, hidden)	Sets the text visibility of the given text index.
GUIDecoratedIconButton:Get_Component_Index (component_name)	Searches for the component name and returns its index.

Class GUIDecoratedIconButton

Contains functions to manipulate a GUIDecoratedIconButton.

GUIDecoratedIconButton:Set_Text (text_index, new_text)

Sets the text of the given text index.

Parameters:

- text_index: number
- new_text: string of the text to display

Returns:

nil

GUIDecoratedIconButton:Set_Formatted_Text (text_index, text_id, ...)

Sets the formatted text of the given text index.

Parameters:

- text_index: number
- text_id: string of the text to format
- ...: values to add to the formatted string

Returns:

nil

GUIDecoratedIconButton:Set_Quad_Texture (quad_index, texture_name)

Sets the texture of the given quad index.

Parameters:

- quad_index: number
- texture_name: string of the texture

Returns:

nil

GUIDecoratedIconButton:Get_Quad_Texture (quad_index)

Gets the texture name of the quad at the given index.

Parameters:

- quad_index: number

Returns:

string, or nil

GUIDecoratedIconButton:Set_Quad_Hidden (quad_index, hidden)

Sets the quad visibility of the given quad index.

Parameters:

- quad_index: number
- hidden: boolean (when true, hides the quad)

Returns:

string, or nil

GUIDecoratedIconButton:Set_Quad_Tint (quad_index, r, g, b, a)

Sets the quad tint of the given quad index.

Parameters:

- quad_index: number
- r: number color R value
- g: number color G value
- b: number color B value
- a: number color A value

Returns:

nil

GUIDecoratedIconButton:Set_Text_Hidden (text_index, hidden)

Sets the text visibility of the given text index.

Parameters:

- text_index: number
- hidden: boolean (when true, hides the text)

Returns:

string, or nil

GUIDecoratedIconButton:Get_Component_Index (component_name)

Searches for the component name and returns its index.

Parameters:

- component_name: string

Returns:

number

GUIEditBox

Functions to manipulate an EditBox GUI component.

Class GUIEditBox

GUIEditBox:Set_Text (new_text, raise_event)	Sets the text in the edit box.
GUIEditBox:Get_Text ()	Gets the text in the edit box.
GUIEditBox:Set_Texture (texture_name)	Sets the edit box texture.
GUIEditBox:Set_Font (font_name)	Sets the edit box font.
GUIEditBox:Set_Color (r, g, b, a)	Sets the edit box color.
GUIEditBox:Set_Text_Limit (limit)	Sets the edit box text limit.
GUIEditBox:Set_Editable (editable)	Sets whether one can modify the contents of the edit box or not.
GUIEditBox:Set_Cursor_Index (index)	Sets the location of the text cursor in the edit box.
GUIEditBox:Get_Cursor_Index ()	Gets the location of the text cursor in the edit box.

Class GUIEditBox

Contains functions to manipulate a GUIEditBox.

GUIEditBox:Set_Text (new_text, raise_event)

Sets the text in the edit box.

Parameters:

- new_text: string
- raise_event: (optional) whether to raise the event or not (default = true)

Returns:

nil

GUIEditBox:Get_Text ()

Gets the text in the edit box.

Returns:

string, or nil

GUIEditBox:Set_Texture (texture_name)

Sets the edit box texture.

Parameters:

- texture_name: string

Returns:

nil

GUIEditBox:Set_Font (font_name)

Sets the edit box font.

Parameters:

- font_name: string

Returns:

nil

GUIEditBox:Set_Color (r, g, b, a)

Sets the edit box color.

Parameters:

- r: number color R value
- g: number color G value
- b: number color B value
- a: number color A value

Returns:

nil

GUIEditBox:Set_Text_Limit (limit)

Sets the edit box text limit.

Parameters:

- limit number of characters

Returns:

nil

GUIEditBox:Set_Editable (editable)

Sets whether one can modify the contents of the edit box or not.

Parameters:

- editable: boolean

Returns:

nil

GUIEditBox:Set_Cursor_Index (index)

Sets the location of the text cursor in the edit box.

Parameters:

- index: number

Returns:

nil

GUIEditBox:Get_Cursor_Index ()

Gets the location of the text cursor in the edit box.

Returns:

number

GUIGeneric

Functions to manipulate a Generic GUI component.

Class GUIGeneric

GUIGeneric:Is_Pool_Safe ()	?
GUIGeneric:Is_Valid ()	Returns whether the component is valid.
GUIGeneric:Get_Name ()	Gets the name of the component.
GUIGeneric:Set_Name (name)	Sets the name of the component.
GUIGeneric:Get_Fully_Qualified_Name ()	Gets the fully qualified name of the component.
GUIGeneric:Set_Bounds (x, y, width, height)	Sets the bounds of the component.
GUIGeneric:Get_Bounds ()	Gets the bounds of the component.
GUIGeneric:Set_Tint (r, g, b, a)	Sets the tint of the component.
GUIGeneric:Get_Tint ()	Gets the tint of the component.
GUIGeneric:Begin_Flash (r, g, b, a, seconds, speed, flash_type)	Flash this component using a tint color interpolation over time.
GUIGeneric:End_Flash (restore_color)	Stop any flash animation and set the tint back to it's starting tint.
GUIGeneric:Translate (dx, dy)	Moves the component relative to its current position.
GUIGeneric:Set_Hidden (hidden)	Sets the component's visibility.
GUIGeneric:Get_Hidden ()	Gets the component's visibility.
GUIGeneric:Scale (scale_factor)	Scales the component relative to its current size.
GUIGeneric:Clone (new_parent_ref, new_name)	Clones the component and returns the cloned component.
GUIGeneric:Is_Animation_Playing (animation_name)	Returns whether or not the given animation is playing, or if any animation is playing if nil.
GUIGeneric:Stop_Animation (pass_to_children)	Stops animating.
GUIGeneric:Play_Animation (animation_name, loop, pass_to_children)	Plays animation.
GUIGeneric:Add_Animation (animation_ref)	Adds the given animation.
GUIGeneric:Destroy ()	Destroys the component.
GUIGeneric:Begin_Drag ()	Begins dragging the component (currently seems to hang the game, do not call).
GUIGeneric:Is_Dragged ()	Returns boolean for being dragged.
GUIGeneric:Schedule_Drag (drag_grab_x, drag_grab_y)	Schedules a drag (may not be working, avoid).
GUIGeneric:Get_Screen_Position ()	Gets the component's screen position.
GUIGeneric:Set_Screen_Position (screen_x, screen_y)	Sets the component's screen position.
GUIGeneric:Set_Position_Using_Justification (screen_margin_x, screen_margin_y)	Sets position.
GUIGeneric:Play_Animation_Backwards (animation_name, loop)	Plays animation backwards.
GUIGeneric:Pause_Animation ()	Pauses animation.
GUIGeneric:Resume_Animation ()	Resumes animation.
GUIGeneric:Get_Animation_Frame ()	Gets the frame the active animation is on.
GUIGeneric:Set_Animation_Frame (frame)	Sets the frame the active animation is on.
GUIGeneric:Get_Animation_Length ()	Gets animation length.

GUIGeneric:Set_Animate_Backwards (yesno)	Sets whether to animate backwards.
GUIGeneric:Set_Animate_Looping (loop)	Sets whether to loop.
GUIGeneric:Set_Animation_Loop_Count (loop_count)	Sets how many loops to do.
GUIGeneric:Bring_To_Front ()	Brings the component to the front.
GUIGeneric:Send_To_Back ()	Sends the component to the back.
GUIGeneric:Set_Tab_Order (raw_tab_traversal)	Sets the component's tab order value.
GUIGeneric:Get_World_Bounds ()	Gets the component's world bounds.
GUIGeneric:Set_World_Bounds (x, y, width, height)	Sets the component's world bounds.
GUIGeneric:Get_Pixel_Bounds ()	Gets the component's pixel bounds.
GUIGeneric:Set_Key_Focus ()	Sets the component as the key focus.
GUIGeneric:Clear_Key_Focus ()	Clears the component from being the key focus.
GUIGeneric:Get_Mouse_Pointer ()	Gets the mouse pointer type of the component.
GUIGeneric:Enable (enable)	Sets whether or not the component is enabled.
GUIGeneric:Is_Enabled ()	Returns whether or not the component is enabled.
GUIGeneric:Set_Screen_Justification (justification)	Sets the screen justification value.
GUIGeneric:Get_Screen_Justification ()	Gets the screen justification value.
GUIGeneric:Set_Screen_Vertical_Justification (justification)	Sets the screen vertical justification value.
GUIGeneric:Get_Screen_Vertical_Justification ()	Gets the screen vertical justification value.
GUIGeneric:Is_Visible ()	Returns a boolean for the visibility of the component.
GUIGeneric:Set_Can_Be_Disabled (yesno)	Sets if the component can be disabled.
GUIGeneric:Can_Be_Disabled ()	Returns if the component can be disabled.
GUIGeneric:Debug_Get_Pointer_String ()	Gets the debug pointer string for this component.
GUIGeneric:Set_Grayscale (grayscale)	Sets the grayscale setting to on or off.
GUIGeneric:Add_Extended_Tooltip ()	Adds an extended tooltip to the component.
GUIGeneric:Remove_Tooltip ()	Removes tooltip from the component.
GUIGeneric:Attach_To_Mouse (offset_x, offset_y)	Attaches the component to the mouse.
GUIGeneric:Set_Tooltip_Text_ID (text_id)	Sets the tooltip's text via text ID.
GUIGeneric:Set_Tooltip_Preformatted_Text (text, is_dynamic)	Sets the tooltip's text via raw text (and not text ID).
GUIGeneric:Enable_Mouse_Collision (yesno)	Enables or disables mouse collision.
GUIGeneric:Set_Is_Event_Source (yesno)	Enables or disables whether the component is an event source.
GUIGeneric:Remove_All_Children ()	Removes all children.
GUIGeneric:Get_Parent ()	Gets the component's parent component.
GUIGeneric:Get_Scene ()	Gets the component's parent scene.
GUIGeneric:Is_Mouse_Over_Component ()	Returns boolean value for if the component is a mouse over component.
GUIGeneric:Set_Requests_Next_Close_Event (yesno)	Sets if this component will request the next close event.
GUIGeneric:Get_Requests_Next_Close_Event ()	Gets if this component will request the next close event.
GUIGeneric:Set_Hides_On_Close_Event (yesno)	Sets if this component will hide on close events.
GUIGeneric:Set_Hides_On_Close_Event ()	Gets if this component will hide on close events.
GUIGeneric:Set_Hides_Scene_On_Close_Event (yesno)	Sets if this component's scene will hide on close events.
GUIGeneric:Get_Hides_Scene_On_Close_Event ()	Gets if this component's scene will hide on close events.
GUIGeneric:Set_Requests_Unhandled_Close_Events (yesno)	Sets if this component receives unhandled close events.

GUIGeneric:Set_Requests_Unhandled_Close_Events ()	Gets if this component receives unhandled close events.
GUIGeneric:Set_User_ID (id)	Sets this component's user ID.
GUIGeneric:Get_User_ID ()	Gets this component's user ID.
GUIGeneric:Has_User_ID ()	Returns true if the component has a user ID, false otherwise.
GUIGeneric:Clear_User_ID ()	Clears the component's user ID.
GUIGeneric:Set_Texture_Set (texture_set)	Sets the component's texture set.
GUIGeneric:Get_Notes ()	Gets the component's notes.
GUIGeneric:Set_Ignores_Parent_Tint (yesno)	Sets whether or not this component will ignore the parent tint.
GUIGeneric:Get_Ignores_Parent_Tint ()	Gets whether or not this component will ignore the parent tint.

Class GUIGeneric

Contains functions to manipulate a GUIGeneric.

GUIGeneric:Is_Pool_Safe ()

?

Returns:

boolean

GUIGeneric:Is_Valid ()

Returns whether the component is valid.

Returns:

boolean

GUIGeneric:Get_Name ()

Gets the name of the component.

Returns:

string

GUIGeneric:Set_Name (name)

Sets the name of the component.

Parameters:

- name: string

Returns:

nil

GUIGeneric:Get_Fully_Qualified_Name ()

Gets the fully qualified name of the component.

Returns:

string

GUIGeneric:Set_Bounds (x, y, width, height)

Sets the bounds of the component.

Parameters:

- x: number
- y: number
- width: number
- height: number

Returns:

nil

GUIGeneric:Get_Bounds ()

Gets the bounds of the component.

Returns:

table containing values for "x", "y", "width", and "height"

GUIGeneric:Set_Tint (r, g, b, a)

Sets the tint of the component.

Parameters:

- r: number color R value
- g: number color G value
- b: number color B value
- a: number color A value

Returns:

nil

GUIGeneric:Get_Tint ()

Gets the tint of the component.

Returns:

table containing values for "r", "g", "b", "a"

GUIGeneric:Begin_Flash (r, g, b, a, seconds, speed, flash_type)

Flash this component using a tint color interpolation over time.

Parameters:

- r: number color R value
- g: number color G value
- b: number color B value
- a: number color A value
- seconds: number of seconds for how long the flashing should run for. default=-1, forever
- speed: number of how fast/slow the speed of the flash is. default=1.0
- flash_type: string specifying the kind of flash("log", "linear", "flip_log"). default="log"

Returns:

nil

GUIGeneric:End_Flash (restore_color)

Stop any flash animation and set the tint back to it's starting tint.

Parameters:

- restore_color: boolean restore component color to it's original value (default: true)

Returns:

nil

GUIGeneric:Translate (dx, dy)

Moves the component relative to its current position.

Parameters:

- dx: number offset by x
- dy: number offset by y

Returns:

nil

GUIGeneric:Set_Hidden (hidden)

Sets the component's visibility.

Parameters:

- hidden: boolean when true, the component is hidden

Returns:

nil

GUIGeneric:Get_Hidden ()

Gets the component's visibility.

Returns:

boolean true for hidden

GUIGeneric:Scale (scale_factor)

Scales the component relative to its current size.

Parameters:

- scale_factor: number scale factor

Returns:

nil

GUIGeneric:Clone (new_parent_ref, new_name)

Clones the component and returns the cloned component.

Parameters:

- new_parent_ref: CGUIComponent of the component to clone

- `new_name`: string of the name of the newly cloned component

Returns:

CGUIComponent, or nil

GUIGeneric:Is_Animation_Playing (animation_name)

Returns whether or not the given animation is playing, or if any animation is playing if nil.

Parameters:

- `animation_name`: (optional) string of animation

Returns:

boolean

GUIGeneric:Stop_Animation (pass_to_children)

Stops animating.

Parameters:

- `pass_to_children`: boolean pass this call to children as well when true

Returns:

nil

GUIGeneric:Play_Animation (animation_name, loop, pass_to_children)

Plays animation.

Parameters:

- `animation_name`: string of animation
- `loop`: boolean
- `pass_to_children`: (optional) boolean pass this call to children as well when true (default = true)

Returns:

nil

GUIGeneric:Add_Animation (animation_ref)

Adds the given animation.

Parameters:

- `animation_ref`: ???

Returns:

nil

GUIGeneric:Destroy ()

Destroys the component.

Returns:

nil

GUIGeneric:Begin_Drag ()

Begins dragging the component (currently seems to hang the game, do not call).

Returns:

nil

GUIGeneric:Is_Dragged ()

Returns boolean for being dragged.

Returns:

boolean

GUIGeneric:Schedule_Drag (drag_grab_x, drag_grab_y)

Schedules a drag (may not be working, avoid).

Parameters:

- drag_grab_x: number for grab point x
- drag_grab_y: number for grab point y

Returns:

nil

GUIGeneric:Get_Screen_Position ()

Gets the component's screen position.

Returns:

table containing values for "x" and "y"

GUIGeneric:Set_Screen_Position (screen_x, screen_y)

Sets the component's screen position.

Parameters:

- screen_x: number
- screen_y: number

Returns:

nil

GUIGeneric:Set_Position_Using_Justification (screen_margin_x, screen_margin_y)

Sets position.

Parameters:

- screen_margin_x: number
- screen_margin_y: number

Returns:

nil

GUIGeneric:Play_Animation_Backwards (animation_name, loop)

Plays animation backwards.

Parameters:

- animation_name: string
- loop: boolean

Returns:

nil

GUIGeneric:Pause_Animation ()

Pauses animation.

Returns:

nil

GUIGeneric:Resume_Animation ()

Resumes animation.

Returns:

nil

GUIGeneric:Get_Animation_Frame ()

Gets the frame the active animation is on.

Returns:

number

GUIGeneric:Set_Animation_Frame (frame)

Sets the frame the active animation is on.

Parameters:

- frame: number

Returns:

nil

GUIGeneric:Get_Animation_Length ()

Gets animation length.

Returns:

number

GUIGeneric:Set_Animate_Backwards (yesno)

Sets whether to animate backwards.

Parameters:

- yesno: boolean

Returns:

nil

GUIGeneric:Set_Animate_Looping (loop)

Sets whether to loop.

Parameters:

- loop: boolean

Returns:

nil

GUIGeneric:Set_Animation_Loop_Count (loop_count)

Sets how many loops to do.

Parameters:

- loop_count: number

Returns:

nil

GUIGeneric:Bring_To_Front ()

Brings the component to the front.

Returns:

nil

GUIGeneric:Send_To_Back ()

Sends the component to the back.

Returns:

nil

GUIGeneric:Set_Tab_Order (raw_tab_traversal)

Sets the component's tab order value.

Parameters:

- raw_tab_traversal: number for tab order value of this component

Returns:

nil

GUIGeneric:Get_World_Bounds ()

Gets the component's world bounds.

Returns:

table containing values for "x", "y", "width", and "height"

GUIGeneric:Set_World_Bounds (x, y, width, height)

Sets the component's world bounds.

Parameters:

- x: number
- y: number
- width: number
- height: number

Returns:

nil

GUIGeneric:Get_Pixel_Bounds ()

Gets the component's pixel bounds.

Returns:

table containing values for "x", "y", "width", and "height"

GUIGeneric:Set_Key_Focus ()

Sets the component as the key focus.

Returns:

nil

GUIGeneric:Clear_Key_Focus ()

Clears the component from being the key focus.

Returns:

nil

GUIGeneric:Get_Mouse_Pointer ()

Gets the mouse pointer type of the component.

Returns:

number

GUIGeneric:Enable (enable)

Sets whether or not the component is enabled.

Parameters:

- enable: boolean

Returns:

nil

GUIGeneric:Is_Enabled ()

Returns whether or not the component is enabled.

Returns:

boolean

GUIGeneric:Set_Screen_Justification (justification)

Sets the screen justification value.

Parameters:

- justification number (values starting from 0 = left, center, right, true position, use parent, left and right, stretch, aspect stretch)

Returns:

nil

GUIGeneric:Get_Screen_Justification ()

Gets the screen justification value.

Returns:

number (see **GUIGeneric:Set_Screen_Justification**)

GUIGeneric:Set_Screen_Vertical_Justification (justification)

Sets the screen vertical justification value.

Parameters:

- justification: number (values starting from 0 = top, center, bottom, use parent, top and bottom, stretch, true position, aspect stretch)

Returns:

nil

GUIGeneric:Get_Screen_Vertical_Justification ()

Gets the screen vertical justification value.

Returns:

number (see **GUIGeneric:Set_Screen_Vertical_Justification**)

GUIGeneric:Is_Visible ()

Returns a boolean for the visibility of the component.

Returns:

boolean

GUIGeneric:Set_Can_Be_Disabled (yesno)

Sets if the component can be disabled.

Parameters:

- yesno: boolean

Returns:

nil

GUIGeneric:Can_Be_Disabled ()

Returns if the component can be disabled.

Returns:

boolean

GUIGeneric:Debug_Get_Pointer_String ()

Gets the debug pointer string for this component.

Returns:

string

GUIGeneric:Set_Grayscale (grayscale)

Sets the grayscale setting to on or off.

Parameters:

- grayscale: boolean

Returns:

nil

GUIGeneric:Add_Extended_Tooltip ()

Adds an extended tooltip to the component.

Returns:

nil

GUIGeneric:Remove_Tooltip ()

Removes tooltip from the component.

Returns:

nil

GUIGeneric:Attach_To_Mouse (offset_x, offset_y)

Attaches the component to the mouse.

Parameters:

- offset_x: number
- offset_y: number

Returns:

nil

GUIGeneric:Set_Tooltip_Text_ID (text_id)

Sets the tooltip's text via text ID.

Parameters:

- text_id: string

Returns:

nil

GUIGeneric:Set_Tooltip_Preformatted_Text (text, is_dynamic)

Sets the tooltip's text via raw text (and not text ID).

Parameters:

- text: string
- is_dynamic: boolean

Returns:

nil

GUIGeneric:Enable_Mouse_Collision (yesno)

Enables or disables mouse collision.

Parameters:

- yesno: boolean

Returns:

nil

GUIGeneric:Set_Is_Event_Source (yesno)

Enables or disables whether the component is an event source.

Parameters:

- yesno: boolean

Returns:

nil

GUIGeneric:Remove_All_Children ()

Removes all children.

Returns:

nil

GUIGeneric:Get_Parent ()

Gets the component's parent component.

Returns:

CGetComponent, or nil

GUIGeneric:Get_Scene ()

Gets the component's parent scene.

Returns:

CGUIComponent, or nil

GUIGeneric:Is_Mouse_Over_Component ()

Returns boolean value for if the component is a mouse over component.

Returns:

boolean

GUIGeneric:Set_Requests_Next_Close_Event (yesno)

Sets if this component will request the next close event.

Parameters:

- yesno: boolean

Returns:

nil

GUIGeneric:Get_Requests_Next_Close_Event ()

Gets if this component will request the next close event.

Returns:

boolean

GUIGeneric:Set_Hides_On_Close_Event (yesno)

Sets if this component will hide on close events.

Parameters:

- yesno: boolean

Returns:

nil

GUIGeneric:Set_Hides_On_Close_Event ()

Gets if this component will hide on close events.

Returns:

boolean

GUIGeneric:Set_Hides_Scene_On_Close_Event (yesno)

Sets if this component's scene will hide on close events.

Parameters:

- yesno: boolean

Returns:

nil

GUIGeneric:Get_Hides_Scene_On_Close_Event ()

Gets if this component's scene will hide on close events.

Returns:

boolean

GUIGeneric:Set_Requests_Unhandled_Close_Events (yesno)

Sets if this component receives unhandled close events.

Parameters:

- yesno: boolean

Returns:

nil

GUIGeneric:Set_Requests_Unhandled_Close_Events ()

Gets if this component receives unhandled close events.

Returns:

boolean

GUIGeneric:Set_User_ID (id)

Sets this component's user ID.

Parameters:

- id: number or CLuaCValue

Returns:

CLuaCValue of the assigned ID

GUIGeneric:Get_User_ID ()

Gets this component's user ID.

Returns:

CLuaCValue of the assigned ID

GUIGeneric:Has_User_ID ()

Returns true if the component has a user ID, false otherwise.

Returns:

boolean

GUIGeneric:Clear_User_ID ()

Clears the component's user ID.

Returns:

nil

GUIGeneric:Set_Texture_Set (texture_set)

Sets the component's texture set.

Parameters:

- texture_set: string

Returns:

nil

GUIGeneric:Get_Notes ()

Gets the component's notes.

Returns:

string

GUIGeneric:Set_Ignores_Parent_Tint (yesno)

Sets whether or not this component will ignore the parent tint.

Parameters:

- yesno: boolean

Returns:

nil

GUIGeneric:Get_Ignores_Parent_Tint ()

Gets whether or not this component will ignore the parent tint.

Returns:

boolean

GUIIconButton

Functions to manipulate an IconButton GUI component.

Class GUIIconButton

GUIIconButton:Get_Texture (quad_number)	Gets the texture for the given quad number.
GUIIconButton:Set_Texture (quad_number, texture_name)	Sets the texture for the given quad number.
GUIIconButton:Set_Clock_Filled (filled_percent)	Sets the clock filled amount.
GUIIconButton:Get_Clock_Filled ()	Gets the clock filled amount.
GUIIconButton:Set_Clock_Tint (r, g, b, a)	Sets the clock tint.
GUIIconButton:Set_Clock_Autofill_Time (seconds_to_fill, seconds_elapsed, reverse_count)	Sets clock autofill settings.

Class GUIIconButton

Contains functions to manipulate a GUIIconButton.

GUIIconButton:Get_Texture (quad_number)

Gets the texture for the given quad number.

Parameters:

- quad_number: number

Returns:

string

GUIIconButton:Set_Texture (quad_number, texture_name)

Sets the texture for the given quad number.

Parameters:

- quad_number: number
- texture_name: string

Returns:

nil

GUIIconButton:Set_Clock_Filled (filled_percent)

Sets the clock filled amount.

Parameters:

- filled_percent: number (0 to 1)

Returns:

nil

GUIconButton:Get_Clock_Filled ()

Gets the clock filled amount.

Returns:

number (0 to 1)

GUIconButton:Set_Clock_Tint (r, g, b, a)

Sets the clock tint.

Parameters:

- r: number color R value
- g: number color G value
- b: number color B value
- a: number color A value

Returns:

nil

GUIconButton:Set_Clock_Autofill_Time (seconds_to_fill, seconds_elapsed, reverse_count)

Sets clock autofill settings.

Parameters:

- seconds_to_fill: number time in seconds to fill
- seconds_elapsed: (optional) number time in seconds already elapsed (default = 0)
- reverse_count: (optional) boolean counts backwards if true (default = false)

Returns:

number (0 to 1)

GUILine

Functions to manipulate a Line GUI component.

Class GUILine

GUILine:Set_Start_Point (x, y)	Sets line start point.
GUILine:Set_End_Point (x, y)	Sets line end point.
GUILine:Set_World_Start_Point (x, y)	Sets line world start point.
GUILine:Set_World_End_Point (x, y)	Sets line world end point.
GUILine:Set_Width (width)	Sets line width.

Class GUILine

Contains functions to manipulate a GUILine.

GUILine:Set_Start_Point (x, y)

Sets line start point.

Parameters:

- x: number
- y: number

Returns:

nil

GUILine:Set_End_Point (x, y)

Sets line end point.

Parameters:

- x: number
- y: number

Returns:

nil

GUILine:Set_World_Start_Point (x, y)

Sets line world start point.

Parameters:

- x: number
- y: number

Returns:

nil

GUILine:Set_World_End_Point (x, y)

Sets line world end point.

Parameters:

- x: number
- y: number

Returns:

nil

GUILine:Set_Width (width)

Sets line width.

Parameters:

- width: number

Returns:

nil

GUIListBox

Functions to manipulate a ListBox GUI component.

Class GUIListBox

GUIListBox:Add_Column (text_id, justification, word_wrap)	Adds a column.
GUIListBox:Set_Column_Width (column_id, width)	Sets the column's width.
GUIListBox:Set_Column_Header_Style (column_id, header_style)	Sets the column's header style.
GUIListBox:Set_Column_Is_Numeric (column_id, is_numeric)	Sets the column to be numeric or not.
GUIListBox:Set_Sort_On_Column (column_id)	Sets the given column to be the sorted column.
GUIListBox:Get_Sort_On_Column ()	Gets the column that is sorted.
GUIListBox:Get_Selected_Col_Index ()	Gets the selected column index.
GUIListBox:Set_Selected_Col_Index (index, raise_event)	Sets the selected column index.
GUIListBox:Add_Row ()	Adds a new row and returns its ID.
GUIListBox:Remove_Row (row_id)	Removes a row by ID.
GUIListBox:Swap_Rows (first_row_id, second_row_id)	Swaps two rows by ID.
GUIListBox:Get_Visible_Row_Count ()	Returns the number of visible rows.
GUIListBox:Get_Max_Visible_Row_Count ()	Returns the maximum number of visible rows.
GUIListBox:Set_Maximum_Rows (num_rows)	Sets the maximum number of rows for the list box.
GUIListBox:Clear ()	Clears the list box.
GUIListBox:Get_Selected_Row_ID ()	Gets the selected row's ID.
GUIListBox:Set_Selected_Row_ID (row_id, raise_event)	Sets the selected row by ID.
GUIListBox:Get_Previous_From_Selected_Row_ID (wraparound)	Gets the row before the selected row, wrapping around if specified.
GUIListBox:Get_Next_From_Selected_Row_ID (wraparound)	Gets the row after the selected row, wrapping around if specified.
GUIListBox:Get_Selected_Row_Index ()	Gets the selected row's index.
GUIListBox:Get_Row_ID_At_Index (index)	Gets the ID of the row at the given index.
GUIListBox:Get_Minimum_World_Height_For_Content ()	Gets the height required for the content.
GUIListBox:Set_Row_Hidden_By_Name (row_name, hidden)	Sets the given row to be hidden or not.
GUIListBox:Set_Selected_Text_Data (column_id, text_data, raise_event)	Sets a row to be selected via its text contents.
GUIListBox:Get_Selected_Text_Data (column_id)	Gets the text data of the selected row in the given column.
GUIListBox:Set_Allow_Selected_Row_Click (allow)	Enable/disable clicking rows.
GUIListBox:Refresh ()	Refreshes the list box.
GUIListBox:Set_Text_Data_Via_Text_ID (column_id, row_id, text_data)	Sets the text data for the given column and row using the given text ID.
GUIListBox:Set_Text_Data_Via_Raw_String (column_id, row_id, text_data)	Sets the text data for the given column and row using the given raw text.
GUIListBox:Get_Text_Data (column_id, row_id)	Gets the text data for the given column and row.
GUIListBox:Set_Texture (column_id, row_id, texture_name)	Sets the texture for the given row and column.
GUIListBox:Set_Item_From_Example (column_id, row_id, example_ref)	Sets the given row and column to use a clone of the given GUI component.
GUIListBox:Set_Item_From_Example (column_id, row_identifier)	Gets the CGUIComponent at the given column and row.
GUIListBox:Set_Expands_Non_Text_Data (expands)	Turns on/off expanding non text data.

GUIListBox:Set_Account_For_Scrollbar_Size (yesno)	Turns on/off accounting for the scrollbar size.
GUIListBox:Set_All_Entries_Use_Fore_Color (yesno)	Turns on/off all entries using the fore color.
GUIListBox:Clear_All ()	Clears all entries from the list box.
GUIListBox:Set_Header_Font (font_name)	Sets the header font.
GUIListBox:Set_Item_Sort_Value (column_id, row_id, sort_value)	Sets the sort value of the given item.
GUIListBox:Get_Item_Sort_Value (column_id, row_id)	Gets the sort value of the given item.
GUIListBox:Scroll_To_Bottom ()	Scrolls the list box to the bottom.
GUIListBox:Set_List_Entry_Font (font_name)	Sets the font of list entries.
GUIListBox:Set_Row_Color (row_identifier, r, g, b, a)	Sets the row color.
GUIListBox:Set_Row_Background (row_id, texture_name, r, g, b, a)	Sets the row background.
GUIListBox:Get_Slider_Bar_Position ()	Gets the slider bar's position.
GUIListBox:Set_Slider_Bar_Position (position)	Sets the slider bar's position.
GUIListBox:Get_Inverse_Slider_Bar_Position ()	Gets the slider bar's inverse position.
GUIListBox:Set_Inverse_Slider_Bar_Position (position)	Sets the slider bar's inverse position.
GUIListBox:Enable_Selection_Highlight (enable)	Enables/disables selection highlighting.
GUIListBox:Set_Interactive (interactive)	Enables/disables the interactive setting.
GUIListBox:Set_Allow_Row_Deselect (yesno)	Enables/disables deselecting a row.
GUIListBox:Get_Allows_Row_Deselect ()	Returns if rows are allowed to be deselected.
GUIListBox:Set_Allow_Row_Select (allow)	Enables/disables selecting rows.
GUIListBox:Get_Allows_Row_Select ()	Returns if rows are allowed to be selected.
GUIListBox:Get_Scrollbar_Is_Hidden ()	Returns if the scrollbar is hidden.
GUIListBox:Get_Row_Count ()	Returns the row count.

Class GUIListBox

Contains functions to manipulate a GUIListBox.

GUIListBox:Add_Column (text_id, justification, word_wrap)

Adds a column.

Parameters:

- text_id: string of the text ID
- justification: (optional) number (starting from 0: left, center, right, none) (default = 0)
- word_wrap: (optional) boolean (default = false)

Returns:

nil

GUIListBox:Set_Column_Width (column_id, width)

Sets the column's width.

Parameters:

- column_id: string

- width: number

Returns:

nil

GUIListBox:Set_Column_Header_Style (column_id, header_style)

Sets the column's header style.

Parameters:

- column_id: string
- header_style: number (starting from 0: none, test, button, iconbutton)

Returns:

nil

GUIListBox:Set_Column_Is_Numeric (column_id, is_numeric)

Sets the column to be numeric or not.

Parameters:

- column_id: string
- is_numeric: boolean

Returns:

nil

GUIListBox:Set_Sort_On_Column (column_id)

Sets the given column to be the sorted column.

Parameters:

- column_id: string

Returns:

nil

GUIListBox:Get_Sort_On_Column ()

Gets the column that is sorted.

Returns:

string

GUIListBox:Get_Selected_Col_Index ()

Gets the selected column index.

Returns:

number

GUIListBox:Set_Selected_Col_Index (index, raise_event)

Sets the selected column index.

Parameters:

- index: number
- raise_event: (optional) boolean (default = false)

Returns:

nil

GUIListBox:Add_Row ()

Adds a new row and returns its ID.

Returns:

number

GUIListBox:Remove_Row (row_id)

Removes a row by ID.

Parameters:

- row_id: number

Returns:

nil

GUIListBox:Swap_Rows (first_row_id, second_row_id)

Swaps two rows by ID.

Parameters:

- first_row_id: number
- second_row_id: number

Returns:

nil

GUIListBox:Get_Visible_Row_Count ()

Returns the number of visible rows.

Returns:

number

GUIListBox:Get_Max_Visible_Row_Count ()

Returns the maximum number of visible rows.

Returns:

number

GUIListBox:Set_Maximum_Rows (num_rows)

Sets the maximum number of rows for the list box.

Parameters:

- num_rows: number

Returns:

nil

GUIListBox:Clear ()

Clears the list box.

Returns:

nil

GUIListBox:Get_Selected_Row_ID ()

Gets the selected row's ID.

Returns:

number

GUIListBox:Set_Selected_Row_ID (row_id, raise_event)

Sets the selected row by ID.

Parameters:

- row_id: number
- raise_event: (optional) boolean (default = false)

Returns:

nil

GUIListBox:Get_Previous_From_Selected_Row_ID (wraparound)

Gets the row before the selected row, wrapping around if specified. May return -1 to indicate an invalid ID.

Parameters:

- wraparound: boolean

Returns:

number

GUIListBox:Get_Next_From_Selected_Row_ID (wraparound)

Gets the row after the selected row, wrapping around if specified. May return -1 to indicate an invalid ID.

Parameters:

- wraparound: boolean

Returns:

number

GUIListBox:Get_Selected_Row_Index ()

Gets the selected row's index.

Returns:

number

GUIListBox:Get_Row_ID_At_Index (index)

Gets the ID of the row at the given index.

Parameters:

- index: number

Returns:

number

GUIListBox:Get_Minimum_World_Height_For_Content ()

Gets the height required for the content.

Returns:

number

GUIListBox:Set_Row_Hidden_By_Name (row_name, hidden)

Sets the given row to be hidden or not.

Parameters:

- row_name: string
- hidden: boolean

Returns:

nil

GUIListBox:Set_Selected_Text_Data (column_id, text_data, raise_event)

Sets a row to be selected via its text contents.

Parameters:

- column_id: string
- text_data: string
- raise_event: (optional) boolean (default = false)

Returns:

nil

GUIListBox:Get_Selected_Text_Data (column_id)

Gets the text data of the selected row in the given column.

Parameters:

- column_id: string

Returns:

string

GUIListBox:Set_Allow_Selected_Row_Click (allow)

Enable/disable clicking rows.

Parameters:

- allow: boolean

Returns:

nil

GUIListBox:Refresh ()

Refreshes the list box.

Returns:

nil

GUIListBox:Set_Text_Data_Via_Text_ID (column_id, row_id, text_data)

Sets the text data for the given column and row using the given text ID.

Parameters:

- column_id: string
- row_id: number
- text_data: string of the text ID

Returns:

nil

GUIListBox:Set_Text_Data_Via_Raw_String (column_id, row_id, text_data)

Sets the text data for the given column and row using the given raw text.

Parameters:

- column_id: string
- row_id: number
- text_data: string of the raw text to use

Returns:

nil

GUIListBox:Get_Text_Data (column_id, row_id)

Gets the text data for the given column and row.

Parameters:

- column_id: string
- row_id: number

Returns:

string

GUIListBox:Set_Texture (column_id, row_id, texture_name)

Sets the texture for the given row and column.

Parameters:

- column_id: string
- row_id: number
- texture_name: string

Returns:

nil

GUIListBox:Set_Item_From_Example (column_id, row_id, example_ref)

Sets the given row and column to use a clone of the given GUI component.

Parameters:

- column_id: string
- row_id: number
- example_ref: CGUIComponent item to clone

Returns:

nil

GUIListBox:Set_Item_From_Example (column_id, row_identifier)

Gets the CGUIComponent at the given column and row.

Parameters:

- column_id: string
- row_identifier: number or string

Returns:

CGUIComponent

GUIListBox:Set_Expands_Non_Text_Data (expands)

Turns on/off expanding non text data.

Parameters:

- expands: boolean

Returns:

nil

GUIListBox:Set_Account_For_Scrollbar_Size (yesno)

Turns on/off accounting for the scrollbar size.

Parameters:

- yesno: boolean

Returns:

nil

GUIListBox:Set_All_Entries_Use_Fore_Color (yesno)

Turns on/off all entries using the fore color.

Parameters:

- yesno: boolean

Returns:

nil

GUIListBox:Clear_All ()

Clears all entries from the list box.

Returns:

nil

GUIListBox:Set_Header_Font (font_name)

Sets the header font.

Parameters:

- font_name: string

Returns:

nil

GUIListBox:Set_Item_Sort_Value (column_id, row_id, sort_value)

Sets the sort value of the given item.

Parameters:

- column_id: string
- row_id: number
- sort_value: number

Returns:

nil

GUIListBox:Get_Item_Sort_Value (column_id, row_id)

Gets the sort value of the given item.

Parameters:

- column_id: string
- row_id: number

Returns:

number

GUIListBox:Scroll_To_Bottom ()

Scrolls the list box to the bottom.

Returns:

nil

GUIListBox:Set_List_Entry_Font (font_name)

Sets the font of list entries.

Parameters:

- font_name: string

Returns:

nil

GUIListBox:Set_Row_Color (row_identifier, r, g, b, a)

Sets the row color.

Parameters:

- row_identifier: number or string
- r: number color R value
- g: number color G value
- b: number color B value
- a: number color A value

Returns:

nil

GUIListBox:Set_Row_Background (row_id, texture_name, r, g, b, a)

Sets the row background.

Parameters:

- row_id: number
- texture_name: string
- r: number color R value
- g: number color G value
- b: number color B value
- a: number color A value

Returns:

nil

GUIListBox:Get_Slider_Bar_Position ()

Gets the slider bar's position.

Returns:

number

GUIListBox:Set_Slider_Bar_Position (position)

Sets the slider bar's position.

Parameters:

- position: number

Returns:

nil

GUIListBox:Get_Inverse_Slider_Bar_Position ()

Gets the slider bar's inverse position.

Returns:

number

GUIListBox:Set_Inverse_Slider_Bar_Position (position)

Sets the slider bar's inverse position.

Parameters:

- position: number

Returns:

nil

GUIListBox:Enable_Selection_Highlight (enable)

Enables/disables selection highlighting.

Parameters:

- enable: boolean

Returns:

nil

GUIListBox:Set_Interactive (interactive)

Enables/disables the interactive setting.

Parameters:

- interactive: boolean

Returns:

nil

GUIListBox:Set_Allow_Row_Deselect (yesno)

Enables/disables deselecting a row.

Parameters:

- yesno: boolean

Returns:

nil

GUIListBox:Get_Allows_Row_Deselect ()

Returns if rows are allowed to be deselected.

Returns:

boolean

GUIListBox:Set_Allow_Row_Select (allow)

Enables/disables selecting rows.

Parameters:

- allow: boolean

Returns:

nil

GUIListBox:Get_Allows_Row_Select ()

Returns if rows are allowed to be selected.

Returns:

boolean

GUIListBox:Get_Scrollbar_Is_Hidden ()

Returns if the scrollbar is hidden.

Returns:

boolean

GUIListBox:Get_Row_Count ()

Returns the row count.

Returns:

number

GUIMPEGMovieQuad

Functions to manipulate an MPEGMovieQuad GUI component.

Class GUIMPEGMovieQuad

GUIMPEGMovieQuad:Set_Movie (movie_name)	Sets the movie file.
GUIMPEGMovieQuad:Get_Movie_Name ()	Gets the name of the movie.
GUIMPEGMovieQuad:Get_Movie_Path ()	Gets the path of the movie.
GUIMPEGMovieQuad:Is_Movie_Active ()	Returns boolean depending on whether a movie is loaded or not.
GUIMPEGMovieQuad:Is_Looping ()	Returns boolean for the looping setting.
GUIMPEGMovieQuad:Set_Looping ()	Sets the looping setting.
GUIMPEGMovieQuad:Set_Render_Mode (render_mode)	Sets the render mode.
GUIMPEGMovieQuad:Get_Render_Mode ()	Gets the render mode.
GUIMPEGMovieQuad:Get_Width ()	Gets the width.
GUIMPEGMovieQuad:Get_Height ()	Gets the height.
GUIMPEGMovieQuad:Play ()	Starts playing movie.
GUIMPEGMovieQuad:Pause (pause)	Pauses or unpauses the movie.
GUIMPEGMovieQuad:Step ()	Steps the movie one time.
GUIMPEGMovieQuad:Is_Paused ()	Returns boolean for pause status.
GUIMPEGMovieQuad:Stop_And_Clear_Movie ()	Stops and clears the movie.
GUIMPEGMovieQuad:Send_Movie_Finished_Event ()	Sends the movie finished event.

Class GUIMPEGMovieQuad

Contains functions to manipulate a GUIMovieQuad.

GUIMPEGMovieQuad:Set_Movie (movie_name)

Sets the movie file.

Parameters:

- movie_name: string

Returns:

nil

GUIMPEGMovieQuad:Get_Movie_Name ()

Gets the name of the movie.

Returns:

string of movie name

GUIMPEGMovieQuad:Get_Movie_Path ()

Gets the path of the movie.

Returns:

string of movie path

GUIMPEGMovieQuad:Is_Movie_Active ()

Returns boolean depending on whether a movie is loaded or not.

Returns:

boolean

GUIMPEGMovieQuad:Is_Looping ()

Returns boolean for the looping setting.

Returns:

boolean

GUIMPEGMovieQuad:Set_Looping ()

Sets the looping setting.

Returns:

nil

GUIMPEGMovieQuad:Set_Render_Mode (render_mode)

Sets the render mode.

Parameters:

- render_mode: string

Returns:

nil

GUIMPEGMovieQuad:Get_Render_Mode ()

Gets the render mode.

Returns:

string of render mode

GUIMPEGMovieQuad:Get_Width ()

Gets the width.

Returns:

number

GUIMPEGMovieQuad:Get_Height ()

Gets the height.

Returns:

number

GUIMPEGMovieQuad:Play ()

Starts playing movie.

Returns:

nil

GUIMPEGMovieQuad:Pause (pause)

Pauses or unpauses the movie.

Parameters:

- pause: boolean for whether to pause or unpauses

Returns:

nil

GUIMPEGMovieQuad:Step ()

Steps the movie one time.

Returns:

nil

GUIMPEGMovieQuad:Is_Paused ()

Returns boolean for pause status.

Returns:

boolean

GUIMPEGMovieQuad:Stop_And_Clear_Movie ()

Stops and clears the movie.

Returns:

nil

GUIMPEGMovieQuad:Send_Movie_Finished_Event ()

Sends the movie finished event.

Returns:

nil

GUIMenu

Functions to manipulate a Menu GUI component.

Class GUIMenu

<i>GUIMenu:Toggle_Opened ()</i>	<i>Toggles on/off if the menu is opened.</i>
<i>GUIMenu:Set_Opening_Animation_Seconds (seconds)</i>	<i>Sets the duration of the opening animation.</i>

Class GUIMenu

Contains functions to manipulate a GUIMenu.

GUIMenu:Toggle_Opened ()

Toggles on/off if the menu is opened.

Returns:

nil

GUIMenu:Set_Opening_Animation_Seconds (seconds)

Sets the duration of the opening animation.

Parameters:

- *seconds number*

Returns:

nil

GUIModel

Functions to manipulate a Model GUI component.

Class GUIModel

<i>GUIModel:Set_Model (model_name)</i>	<i>Sets the model.</i>
<i>GUIModel:Play_Facial_Animation (lip_sync_name)</i>	<i>Plays facial animation.</i>
<i>GUIModel:Stop_Facial_Animation ()</i>	<i>Stops facial animation.</i>
<i>GUIModel:Play_Model_Animation (animation_name)</i>	<i>Plays animation.</i>
<i>GUIModel:Play_Randomized_Animation (anim_suffix)</i>	<i>Plays randomized animation.</i>
<i>GUIModel:Set_Blend_Duration (duration)</i>	<i>Sets animation blend duration.</i>
<i>GUIModel:Is_Model_Ready ()</i>	<i>Returns if the model is ready or not.</i>
<i>GUIModel:Force_Dump_Assets ()</i>	<i>?</i>

Class GUIModel

Contains functions to manipulate a GUIModel.

GUIModel:Set_Model (model_name)

Sets the model.

Parameters:

- *model_name string*

Returns:

nil

GUIModel:Play_Facial_Animation (lip_sync_name)

Plays facial animation.

Parameters:

- *lip_sync_name string*

Returns:

nil

GUIModel:Stop_Facial_Animation ()

Stops facial animation.

Returns:

nil

GUIModel:Play_Model_Animation (animation_name)

Plays animation.

Parameters:

- *animation_name string*

Returns:

nil

GUIModel:Play_Randomized_Animation (anim_suffix)

Plays randomized animation.

Parameters:

- *anim_suffix string*

Returns:

nil

GUIModel:Set_Blend_Duration (duration)

Sets animation blend duration.

Parameters:

- *duration number*

Returns:

nil

GUIModel:Is_Model_Ready ()

Returns if the model is ready or not.

Returns:

boolean

GUIModel:Force_Dump_Assets ()

?

Returns:

nil

GUIMovieQuad

Functions to manipulate a MovieQuad GUI component.

Class GUIMovieQuad

<i>GUIMovieQuad:Set_Movie (movie_name)</i>	<i>Sets the movie file.</i>
<i>GUIMovieQuad:Set_Movie_With_Audio_Events (movie_name, audio_events_name)</i>	<i>Sets the movie file with the given audio events.</i>
<i>GUIMovieQuad:Play ()</i>	<i>Starts playing movie.</i>
<i>GUIMovieQuad:Play ()</i>	<i>Pauses movie.</i>
<i>GUIMovieQuad:Stop ()</i>	<i>Stops playing movie.</i>
<i>GUIMovieQuad:Has_Movie ()</i>	<i>Returns if there is a movie loaded or not.</i>
<i>GUIMovieQuad:Restart ()</i>	<i>Restarts movie.</i>
<i>GUIMovieQuad:Force_Frame_Rate (rate)</i>	<i>Forces a given frame rate (currently this function actually does nothing).</i>
<i>GUIMovieQuad:Reset_Frame_Rate ()</i>	<i>Resets frame rate (currently this function actually does nothing).</i>
<i>GUIMovieQuad:Set_Loop (loops)</i>	<i>Enables/disables looping.</i>
<i>GUIMovieQuad:Set_Surround (loops)</i>	<i>Enables/disables surround.</i>
<i>GUIMovieQuad:Set_Plays_While_Hidden (plays)</i>	<i>Enables/disables playing while hidden.</i>
<i>GUIMovieQuad:Get_Plays_While_Hidden ()</i>	<i>Returns if the movie plays while hidden.</i>
<i>GUIMovieQuad:Set_Holds_On_Last_Frame (holds)</i>	<i>Enables/disables holding on last frame.</i>
<i>GUIMovieQuad:Get_Holds_On_Last_Frame ()</i>	<i>Returns if the movie holds on the last frame.</i>

Class GUIMovieQuad

Contains functions to manipulate a GUIMovieQuad.

GUIMovieQuad:Set_Movie (movie_name)

Sets the movie file.

Parameters:

- *movie_name string*

Returns:

nil

GUIMovieQuad:Set_Movie_With_Audio_Events (movie_name, audio_events_name)

Sets the movie file with the given audio events.

Parameters:

- *movie_name string*
- *audio_events_name string*

Returns:

nil

GUIMovieQuad:Play ()

Starts playing movie.

Returns:

nil

GUIMovieQuad:Play ()

Pauses movie.

Returns:

nil

GUIMovieQuad:Stop ()

Stops playing movie.

Returns:

nil

GUIMovieQuad:Has_Movie ()

Returns if there is a movie loaded or not.

Returns:

boolean

GUIMovieQuad:Restart ()

Restarts movie.

Returns:

nil

GUIMovieQuad:Force_Frame_Rate (rate)

Forces a given frame rate (currently this function actually does nothing).

Parameters:

- *rate number*

Returns:

nil

GUIMovieQuad:Reset_Frame_Rate ()

Resets frame rate (currently this function actually does nothing).

Returns:

nil

GUIMovieQuad:Set_Loop (loops)

Enables/disables looping.

Parameters:

- *loops boolean*

Returns:

nil

GUIMovieQuad:Set_Surround (loops)

Enables/disables surround.

Parameters:

- *loops boolean*

Returns:

nil

GUIMovieQuad:Set_Plays_While_Hidden (plays)

Enables/disables playing while hidden.

Parameters:

- *plays boolean*

Returns:

nil

GUIMovieQuad:Get_Plays_While_Hidden ()

Returns if the movie plays while hidden.

Returns:

boolean

GUIMovieQuad:Set_Holds_On_Last_Frame (holds)

Enables/disables holding on last frame.

Parameters:

- *holds boolean*

Returns:

nil

GUIMovieQuad:Get_Holds_On_Last_Frame ()

Returns if the movie holds on the last frame.

Returns:

boolean

GUIProgressBar

Functions to manipulate a ProgressBar GUI component.

Class GUIProgressBar

<code>GUIProgressBar:Set_Bar_Tint (r, g, b, a)</code>	Sets the bar tint.
<code>GUIProgressBar:Set_Filled (fill_percent)</code>	Sets how much the bar is filled.
<code>GUIProgressBar:Get_Filled ()</code>	Gets how much the bar is filled.
<code>GUIProgressBar:Set_Autofill_Time (seconds_to_fill, from_current, end_value, elapsed_seconds)</code>	Sets the bar to fill up over a certain amount of time.

Class GUIProgressBar

Contains functions to manipulate a GUIProgressBar.

`GUIProgressBar:Set_Bar_Tint (r, g, b, a)`

Sets the bar tint.

Parameters:

- *r* number color R value
- *g* number color G value
- *b* number color B value
- *a* number color A value

Returns:

nil

`GUIProgressBar:Set_Filled (fill_percent)`

Sets how much the bar is filled.

Parameters:

- *fill_percent* number (0 to 1)

Returns:

nil

`GUIProgressBar:Get_Filled ()`

Gets how much the bar is filled.

Returns:

number

`GUIProgressBar:Set_Autofill_Time (seconds_to_fill, from_current, end_value, elapsed_seconds)`

Sets the bar to fill up over a certain amount of time.

Parameters:

- *seconds_to_fill* number
- *from_current* (optional) number (default = false)

- *end_value (optional) number (default = 1.0)*
- *elapsed_seconds (optional) number (default = 0)*

Returns:

nil

GUIScene

Functions to manipulate a Scene GUI component.

Class GUIScene

<i>GUIScene:Get_Local_Component (component_name)</i>	<i>Finds and returns the given component by local name in the scene.</i>
<i>GUIScene:Get_Component (component_name)</i>	<i>Finds and returns the given component by full name in the scene.</i>
<i>GUIScene:Get_Component_By_Partial_Name (component_name)</i>	<i>Finds and returns the given component by partial name match in the scene.</i>

Class GUIScene

Contains functions to manipulate a GUIScene.

GUIScene:Get_Local_Component (component_name)

Finds and returns the given component by local name in the scene. If components have the same name, only one of them will be returned.

Parameters:

- *component_name string*

Returns:

CGUIComponent, or nil

GUIScene:Get_Component (component_name)

Finds and returns the given component by full name in the scene.

Parameters:

- *component_name string*

Returns:

CGUIComponent, or nil

GUIScene:Get_Component_By_Partial_Name (component_name)

Finds and returns the given component by partial name match in the scene. If there is more than one match, only one match is returned.

Parameters:

- *component_name string*

Returns:

CGUIComponent, or nil

GUISceneReference

Functions to manipulate a SceneReference GUI component.

Class GUISceneReference

<i>GUISceneReference:Set_Screen_Position (x, y)</i>	<i>Sets screen position.</i>
<i>GUISceneReference:Start_Modal ()</i>	<i>Starts modal mode.</i>
<i>GUISceneReference:Start_Modal ()</i>	<i>Ends modal mode.</i>
<i>GUISceneReference:Is_Modal_Scene ()</i>	<i>Returns if the scene is modal.</i>
<i>GUISceneReference:Set_Rotation (rotation_degrees)</i>	<i>Sets the rotation.</i>
<i>GUISceneReference:Rebuild_Graphics ()</i>	<i>Rebuilds graphics.</i>
<i>GUISceneReference:Set_Mouse_Pointer (mouse_pointer)</i>	<i>Sets the mouse pointer.</i>
<i>GUISceneReference:Get_Scene_Link ()</i>	<i>Gets the scene the scene reference is referencing.</i>

Class GUISceneReference

Contains functions to manipulate a GUISceneReference.

GUISceneReference:Set_Screen_Position (x, y)

Sets screen position.

Parameters:

- *x number*
- *y number*

Returns:

nil

GUISceneReference:Start_Modal ()

Starts modal mode.

Returns:

nil

GUISceneReference:Start_Modal ()

Ends modal mode.

Returns:

nil

GUISceneReference:Is_Modal_Scene ()

Returns if the scene is modal.

Returns:

boolean

GUISceneReference:Set_Rotation (rotation_degrees)

Sets the rotation.

Parameters:

- *rotation_degrees* number angle in degrees

Returns:

boolean

GUISceneReference:Rebuild_Graphics ()

Rebuilds graphics.

Returns:

nil

GUISceneReference:Set_Mouse_Pointer (mouse_pointer)

Sets the mouse pointer.

Parameters:

- *mouse_pointer* number

Returns:

nil

GUISceneReference:Get_Scene_Link ()

Gets the scene the scene reference is referencing.

Returns:

CGUIComponent, or nil

GUISliderBar

Functions to manipulate a SliderBar GUI component.

Class GUISliderBar

<i>GUIAdvancedTicker:Get_Steps ()</i>	<i>Returns the number of steps in the slider bar.</i>
<i>GUIAdvancedTicker:Set_Steps (steps)</i>	<i>Sets the number of steps in the slider bar.</i>
<i>GUIAdvancedTicker:Get_Current_Step_Index ()</i>	<i>Returns the current step.</i>
<i>GUIAdvancedTicker:Set_Current_Step_Index (step_index, raise_event)</i>	<i>Sets the current step.</i>
<i>GUIAdvancedTicker:Set_Current_Fraction (fraction, raise_event)</i>	<i>Sets step via a specified fraction/percentage.</i>
<i>GUIAdvancedTicker:Get_Current_Fraction ()</i>	<i>Returns step as a percentage.</i>
<i>GUIAdvancedTicker:Get_Stops ()</i>	<i>Returns number of stops.</i>
<i>GUIAdvancedTicker:Set_Stops (stops)</i>	<i>Sets number of stops.</i>
<i>GUIAdvancedTicker:Get_Current_Stop_Index ()</i>	<i>Gets current stop.</i>
<i>GUIAdvancedTicker:Set_Current_Stop_Index (index, raise_event)</i>	<i>Sets current stop.</i>
<i>GUIAdvancedTicker:Set_Value (min, max, step_size, value, raise_event)</i>	<i>Sets all parameters simultaneously.</i>
<i>GUIAdvancedTicker:Get_Value ()</i>	<i>Gets the current value specified by the slider bar.</i>

Class GUISliderBar

Contains functions to manipulate a GUISliderBar.

GUIAdvancedTicker:Get_Steps ()

Returns the number of steps in the slider bar.

Returns:

number

GUIAdvancedTicker:Set_Steps (steps)

Sets the number of steps in the slider bar.

Parameters:

- *steps number*

Returns:

nil

GUIAdvancedTicker:Get_Current_Step_Index ()

Returns the current step.

Returns:

number

GUIAdvancedTicker:Set_Current_Step_Index (step_index, raise_event)

Sets the current step.

Parameters:

- *step_index* number
- *raise_event* (optional) boolean (default = true)

Returns:

nil

GUIAdvancedTicker:Set_Current_Fraction (*fraction*, *raise_event*)

Sets step via a specified fraction/percentage.

Parameters:

- *fraction* number
- *raise_event* (optional) boolean (default = true)

Returns:

nil

GUIAdvancedTicker:Get_Current_Fraction ()

Returns step as a percentage.

Returns:

number

GUIAdvancedTicker:Get_Stops ()

Returns number of stops.

Returns:

number

GUIAdvancedTicker:Set_Stops (*stops*)

Sets number of stops.

Parameters:

- *stops* number

Returns:

nil

GUIAdvancedTicker:Get_Current_Stop_Index ()

Gets current stop.

Returns:

number

GUIAdvancedTicker:Set_Current_Stop_Index (*index*, *raise_event*)

Sets current stop.

Parameters:

- *index* number
- *raise_event* (optional) boolean (default = true)

Returns:

nil

GUIAdvancedTicker:Set_Value (min, max, step_size, value, raise_event)

Sets all parameters simultaneously.

Parameters:

- *min number min value*
- *max number max value*
- *step_size number size of step*
- *value number current value after call*
- *raise_event (optional) boolean (default = true)*

Returns:

nil

GUIAdvancedTicker:Get_Value ()

Gets the current value specified by the slider bar.

Returns:

number

GUITextBlock

Functions to manipulate a TextBlock GUI component.

Class GUITextBlock

<code>GUITextBlock:Set_Text (new_text)</code>	Sets the text for the text block via text ID.
<code>GUITextBlock:Set_Formatted_Text (text_id, ...)</code>	Formats text using the given text ID and parameters, then sets the text of the text block to the result.
<code>GUITextBlock:Get_Text ()</code>	Gets the text block's text.
<code>GUITextBlock:Get_Text_Width ()</code>	Gets the text block's width.
<code>GUITextBlock:Get_Text_Height ()</code>	Gets the text block's height.
<code>GUITextBlock:Set_Font (font_name)</code>	Sets the font.
<code>GUITextBlock:Set_Word_Wrap (use_word_wrap)</code>	Enables/disables word wrap.
<code>GUITextBlock:Set_Highlight (use_highlight)</code>	Turns on/off highlighting.

Class GUITextBlock

Contains functions to manipulate a GUITextBlock.

`GUITextBlock:Set_Text (new_text)`

Sets the text for the text block via text ID.

Parameters:

- `new_text` string text ID

Returns:

`nil`

`GUITextBlock:Set_Formatted_Text (text_id, ...)`

Formats text using the given text ID and parameters, then sets the text of the text block to the result.

Parameters:

- `text_id` string text ID
- ... values to fill in the format string

Returns:

`nil`

`GUITextBlock:Get_Text ()`

Gets the text block's text.

Returns:

string, or `nil`

`GUITextBlock:Get_Text_Width ()`

Gets the text block's width.

Returns:

number

GUITextBlock:Get_Text_Height ()

Gets the text block's height.

Returns:

number

GUITextBlock:Set_Font (font_name)

Sets the font.

Parameters:

- *font_name string*

Returns:

nil

GUITextBlock:Set_Word_Wrap (use_word_wrap)

Enables/disables word wrap.

Parameters:

- *use_word_wrap boolean*

Returns:

nil

GUITextBlock:Set_Highlight (use_highlight)

Turns on/off highlighting.

Parameters:

- *use_highlight boolean*

Returns:

nil

UIButton

Functions to manipulate a UIButton GUI component.

Class UIButton

<code>UIButton:Set_Text (new_text)</code>	Sets the text for the button via text ID.
<code>UIButton:Set_Enabled (is_enabled)</code>	Enables/disables the button.
<code>UIButton:Set_Tint (r, g, b, a)</code>	Sets the tint.

Class UIButton

Contains functions to manipulate a UIButton.

`UIButton:Set_Text (new_text)`

Sets the text for the button via text ID.

Parameters:

- `new_text` string of text ID

Returns:

`nil`

`UIButton:Set_Enabled (is_enabled)`

Enables/disables the button.

Parameters:

- `is_enabled` boolean

Returns:

`nil`

`UIButton:Set_Tint (r, g, b, a)`

Sets the tint.

Parameters:

- `r` number color R value
- `g` number color G value
- `b` number color B value
- `a` number color A value

Returns:

`nil`

GUITexturedQuad

Functions to manipulate a TexturedQuad GUI component.

Class GUITexturedQuad

<code>GUITexturedQuad:Set_Texture (texture_name)</code>	<i>Sets the texture.</i>
<code>GUITexturedQuad:Get_Texture_Name ()</code>	<i>Gets the texture.</i>
<code>GUITexturedQuad:Get_Full_Texture_Name ()</code>	<i>Gets the full texture name.</i>
<code>GUITexturedQuad:Start_Texture_Animation (loop)</code>	<i>Starts the texture animation.</i>
<code>GUITexturedQuad:Stop_Texture_Animation ()</code>	<i>Stops the texture animation.</i>
<code>GUITexturedQuad:Pause_Texture_Animation ()</code>	<i>Pauses the texture animation.</i>
<code>GUITexturedQuad:Resume_Texture_Animation (request_animation_traverse)</code>	<i>Resumes the texture animation (from a pause).</i>
<code>GUITexturedQuad:Reverse_Texture_Animation ()</code>	<i>Toggles the reversed animation setting on and off.</i>
<code>GUITexturedQuad:Is_Texture_Animating ()</code>	<i>Returns if the texture is already animating.</i>
<code>GUITexturedQuad:Set_Texture_Animation_Frame (frame)</code>	<i>Sets the current frame of the animation.</i>
<code>GUITexturedQuad:Get_Texture_Animation_Frame ()</code>	<i>Gets the current frame of the animation.</i>
<code>GUITexturedQuad:Get_Texture_Animation_Backwards ()</code>	<i>Returns if the animation is reversed or not.</i>

Class GUITexturedQuad

Contains functions to manipulate a GUITexturedQuad.

`GUITexturedQuad:Set_Texture (texture_name)`

Sets the texture.

Parameters:

- `texture_name` string

Returns:

nil

`GUITexturedQuad:Get_Texture_Name ()`

Gets the texture.

Returns:

string, or nil

`GUITexturedQuad:Get_Full_Texture_Name ()`

Gets the full texture name.

Returns:

string, or nil

`GUITexturedQuad:Start_Texture_Animation (loop)`

Starts the texture animation.

Parameters:

- *loop* boolean determines if the animation is looping

Returns:

nil

GUITexturedQuad:Stop_Texture_Animation ()

Stops the texture animation.

Returns:

nil

GUITexturedQuad:Pause_Texture_Animation ()

Pauses the texture animation.

Returns:

nil

GUITexturedQuad:Resume_Texture_Animation (request_animation_traverse)

Resumes the texture animation (from a pause).

Parameters:

- *request_animation_traverse* (optional) boolean (default = false)

Returns:

nil

GUITexturedQuad:Reverse_Texture_Animation ()

Toggles the reversed animation setting on and off.

Returns:

string, or nil

GUITexturedQuad:Is_Texture_Animating ()

Returns if the texture is already animating.

Returns:

boolean

GUITexturedQuad:Set_Texture_Animation_Frame (frame)

Sets the current frame of the animation.

Parameters:

- *frame number*

Returns:

nil

GUITexturedQuad:Get_Texture_Animation_Frame ()

Gets the current frame of the animation.

Returns:

number

GUITexturedQuad:Get_Texture_Animation_Backwards ()

Returns if the animation is reversed or not.

Returns:

boolean

GUITicker

Functions to manipulate a TextButton GUI component.

Class GUITicker

<i>GUITicker:Scroll_To_Top ()</i>	<i>Scrolls to the top.</i>
<i>GUITicker:Scroll_To_Bottom ()</i>	<i>Scrolls to the bottom.</i>
<i>GUITicker:Ticker_Page_Up ()</i>	<i>Goes up a page.</i>
<i>GUITicker:Ticker_Page_Down ()</i>	<i>Goes down a page.</i>
<i>GUITicker:Ticker_Move_Up ()</i>	<i>Moves ticker up once.</i>
<i>GUITicker:Ticker_Move_Down ()</i>	<i>Moves ticker down once.</i>
<i>GUITicker:Set_Base_Velocity (velocity)</i>	<i>Sets base velocity.</i>
<i>GUITicker:Get_Base_Velocity ()</i>	<i>Gets base velocity.</i>
<i>GUITicker:Set_Looping_Display (looping)</i>	<i>Enables/disables looping setting.</i>
<i>GUITicker:Get_Looping_Display ()</i>	<i>Gets if the display is looping.</i>
<i>GUITicker:Set_Cache_Displayed_Elements (yesno)</i>	<i>Enables/disables caching setting.</i>
<i>GUITicker:Get_Cache_Displayed_Elements ()</i>	<i>Gets if the display uses caching.</i>
<i>GUITicker:Clone_Item_Into_Ticker (item_ref, desired_id, insert_before_id)</i>	<i>Clones the given GUI component and adds it to the ticker.</i>
<i>GUITicker:Insert_Item_Into_Ticker (item_ref, desired_id, insert_before_id)</i>	<i>Inserts given GUI component to the ticker.</i>
<i>GUITicker:Remove_Item_At_Index (index)</i>	<i>Removes item at given index.</i>
<i>GUITicker:Remove_Item_By_ID (item_id)</i>	<i>Removes item with the given ID.</i>
<i>GUITicker:Clear ()</i>	<i>Clears the ticker.</i>
<i>GUITicker:Get_Number_Of_Items ()</i>	<i>Returns number of items in the ticker.</i>
<i>GUITicker:Get_Index_From_ID (item_id)</i>	<i>Returns index of the given item.</i>
<i>GUITicker:Get_ID_From_Index (index)</i>	<i>Returns ID of the given item.</i>
<i>GUITicker:Get_Item_At_Index (index)</i>	<i>Finds and returns the GUI component at the given index.</i>
<i>GUITicker:Get_Item_By_ID (item_id)</i>	<i>Finds and returns the GUI component with the given ID.</i>
<i>GUITicker:Get_ID_Of_Item (item_ref)</i>	<i>Gets the ID of the given GUI component.</i>

Class GUITicker

Contains functions to manipulate a GUITicker.

GUITicker:Scroll_To_Top ()

Scrolls to the top.

Returns:

nil

GUITicker:Scroll_To_Bottom ()

Scrolls to the bottom.

Returns:

nil

GUITicker:Ticker_Page_Up ()

Goes up a page.

Returns:

nil

GUITicker:Ticker_Page_Down ()

Goes down a page.

Returns:

nil

GUITicker:Ticker_Move_Up ()

Moves ticker up once.

Returns:

nil

GUITicker:Ticker_Move_Down ()

Moves ticker down once.

Returns:

nil

GUITicker:Set_Base_Velocity (velocity)

Sets base velocity.

Parameters:

- *velocity number*

Returns:

nil

GUITicker:Get_Base_Velocity ()

Gets base velocity.

Returns:

number

GUITicker:Set_Looping_Display (looping)

Enables/disables looping setting.

Parameters:

- *looping boolean*

Returns:

nil

GUITicker:Get_Looping_Display ()

Gets if the display is looping.

Returns:

boolean

GUITicker:Set_Cache_Displayed_Elements (yesno)

Enables/disables caching setting.

Parameters:

- *yesno boolean*

Returns:

nil

GUITicker:Get_Cache_Displayed_Elements ()

Gets if the display uses caching.

Returns:

boolean

GUITicker:Clone_Item_Into_Ticker (item_ref, desired_id, insert_before_id)

Clones the given GUI component and adds it to the ticker.

Parameters:

- *item_ref CGUIComponent*
- *desired_id number or CLuaCValue for the desired ID*
- *insert_before_id number or CLuaCValue for the ID to insert before*

Returns:

CLuaCValue of result ID, or nil

GUITicker:Insert_Item_Into_Ticker (item_ref, desired_id, insert_before_id)

Inserts given GUI component to the ticker.

Parameters:

- *item_ref CGUIComponent*
- *desired_id number or CLuaCValue for the desired ID*
- *insert_before_id number or CLuaCValue for the ID to insert before*

Returns:

CLuaCValue of result ID, or nil

GUITicker:Remove_Item_At_Index (index)

Removes item at given index.

Parameters:

- *index number*

Returns:

boolean

GUITicker:Remove_Item_By_ID (item_id)

Removes item with the given ID.

Parameters:

- *item_id* number or *CLuaCValue* for the ID to remove

Returns:

boolean

GUITicker:Clear ()

Clears the ticker.

Returns:

nil

GUITicker:Get_Number_Of_Items ()

Returns number of items in the ticker.

Returns:

number

GUITicker:Get_Index_From_ID (item_id)

Returns index of the given item.

Parameters:

- *item_id* number or *CLuaCValue* for the ID to find

Returns:

number, or nil

GUITicker:Get_ID_From_Index (index)

Returns ID of the given item.

Parameters:

- *index* number

Returns:

number, or nil

GUITicker:Get_Item_At_Index (index)

Finds and returns the GUI component at the given index.

Parameters:

- *index* number

Returns:

CGUIComponent, or nil

GUITicker:Get_Item_By_ID (item_id)

Finds and returns the GUI component with the given ID.

Parameters:

- *item_id* number or *CLuaCValue* for the ID to find

Returns:

CGUIComponent, or nil

GUITicker:Get_ID_Of_Item (item_ref)

Gets the ID of the given GUI component.

Parameters:

- *item_ref CGUIComponent*

Returns:

CLuaCValue of ID, or nil

GUITree

Functions to manipulate a Tree GUI component.

Class GUITree

<code>GUITree:Set_Node_Text_ID (node_id, text_id)</code>	Sets the text ID for the node.
<code>GUITree:Set_Node_Raw_Text (node_id, text)</code>	Sets the raw text used in the node.
<code>GUITree:Get_Node_Text (node_id)</code>	Gets the text in the node.
<code>GUITree:Set_Node_Text_Color (node_id, r, g, b, a)</code>	Sets the node's text color.
<code>GUITree:Remove_Node (node_id)</code>	Removes the given node.
<code>GUITree:Get_Child_Count (node_id)</code>	Gets the number of children for the node.
<code>GUITree:Get_Child_At_Index (node_id, index)</code>	Gets the child GUI component at the given index of the given node.
<code>GUITree:Set_Expanded (node_id, expanded, immediate)</code>	Sets the given node to be expanded or collapsed.
<code>GUITree:Toggle_Expanded (node_id)</code>	Toggles between collapsed and expanded for the given node.
<code>GUITree:Get_Is_Expanded (node_id)</code>	Returns if the node is expanded or not.
<code>GUITree:Get_Selected_Node_At_Index (index)</code>	Returns the selected node's ID at the given index.
<code>GUITree:Get_Selected_Nodes_Count ()</code>	Returns the selected nodes count.
<code>GUITree:Add_New_Node (parent_node_id)</code>	Adds a new node and returns its ID.
<code>GUITree:Get_Root_Node_At_Index (index)</code>	Gets the root node at the given index.
<code>GUITree:Get_Root_Node_Count ()</code>	Gets the number of root nodes.
<code>GUITree:Clear ()</code>	Clears the tree.

Class GUITree

Contains functions to manipulate a GUITree.

`GUITree:Set_Node_Text_ID (node_id, text_id)`

Sets the text ID for the node.

Parameters:

- `node_id` number or `CLuaCValue` of the node ID
- `text_id` string

Returns:

`nil`

`GUITree:Set_Node_Raw_Text (node_id, text)`

Sets the raw text used in the node.

Parameters:

- `node_id` number or `CLuaCValue` of the node ID
- `text` string

Returns:

nil

GUITree:Get_Node_Text (*node_id*)

Gets the text in the node.

Parameters:

- *node_id* number or CLuaCValue of the node ID

Returns:

string, or nil

GUITree:Set_Node_Text_Color (*node_id*, *r*, *g*, *b*, *a*)

Sets the node's text color.

Parameters:

- *node_id* number or CLuaCValue of the node ID
- *r* number color R value
- *g* number color G value
- *b* number color B value
- *a* number color A value

Returns:

nil

GUITree:Remove_Node (*node_id*)

Removes the given node.

Parameters:

- *node_id* number or CLuaCValue of the node ID

Returns:

nil

GUITree:Get_Child_Count (*node_id*)

Gets the number of children for the node.

Parameters:

- *node_id* number or CLuaCValue of the node ID

Returns:

number, or nil

GUITree:Get_Child_At_Index (*node_id*, *index*)

Gets the child GUI component at the given index of the given node.

Parameters:

- *node_id* number or CLuaCValue of the node ID
- *index* number

Returns:

CGUIComponent, or nil

GUITree:Set_Expanded (node_id, expanded, immediate)

Sets the given node to be expanded or collapsed.

Parameters:

- *node_id* number or CLuaCValue of the node ID
- *expanded* boolean
- *immediate* boolean

Returns:

nil

GUITree:Toggle_Expanded (node_id)

Toggles between collapsed and expanded for the given node.

Parameters:

- *node_id* number or CLuaCValue of the node ID

Returns:

nil

GUITree:Get_Is_Expanded (node_id)

Returns if the node is expanded or not.

Parameters:

- *node_id* number or CLuaCValue of the node ID

Returns:

boolean, or nil

GUITree:Get_Selected_Node_At_Index (index)

Returns the selected node's ID at the given index.

Parameters:

- *index* number

Returns:

CLuaCValue of ID, or nil

GUITree:Get_Selected_Nodes_Count ()

Returns the selected nodes count.

Returns:

number

GUITree:Add_New_Node (parent_node_id)

Adds a new node and returns its ID.

Parameters:

- *parent_node_id* number or CLuaCValue of the parent node ID

Returns:

CLuaCValue of ID, or nil

GUITree:Get_Root_Node_At_Index (index)

Gets the root node at the given index.

Parameters:

- *index number*

Returns:

CLuaCValue of ID, or nil

GUITree:Get_Root_Node_Count ()

Gets the number of root nodes.

Returns:

number

GUITree:Clear ()

Clears the tree.

Returns:

nil

FrameUpdate

Allows registering functions to be run every Lua frame.

Functions

<i>Register_Function (func)</i>	<i>Registers a function to run on each update.</i>
<i>Unregister_Function (func)</i>	<i>Stops a function from running each update.</i>
<i>Get_Registered_Funcs ()</i>	<i>Gets the table of registered functions.</i>

Functions

Register_Function (func)

Registers a function to run on each update.

Parameters:

- *func* A Lua function to execute each frame

Returns:

nil

Unregister_Function (func)

Stops a function from running each update.

Parameters:

- *func* The Lua function to not run anymore

Returns:

nil

Get_Registered_Funcs ()

Gets the table of registered functions. May be removed from public availability later.

Returns:

table of functions

ServerCinematic

Lua client-specific functionality for cinematics.

Functions

<code>Server_Register_Cinematic_Start_Event (cinematic_file, user_func)</code>	Registers the given function to run on the given cinematic when the cinematic starts.
<code>Server_Register_Cinematic_Finished_Event (cinematic_file, user_func)</code>	Registers the given function to run on the given cinematic when the cinematic completes.
<code>Server_Register_Cinematic_Frame_Event (cinematic_file, frame, user_func)</code>	Registers the given function to run on the given cinematic when the cinematic reaches the given frame.
<code>Server_Register_Cinematic_Skipped_Event (cinematic_file, user_func)</code>	Registers the given function to run on the given cinematic when the cinematic is skipped.
<code>Server_Register_Bark_Finished_Event (bark_name, user_func)</code>	Registers the given function to run when the given bark type finishes.
<code>Server_Register_Bark_Started_Event (bark_name, user_func)</code>	Registers the given function to run when the given bark type finishes.
<code>Server_Register_Strategic_Event_Popped_Up (user_func)</code>	Registers the given function to run when a strategic event dialog pops up.
<code>Server_Register_Strategic_Event_Closed (user_func)</code>	Registers the given function to run when a strategic event dialog is closed.
<code>Server_Unregister_Cinematic_Event (registration_ref)</code>	Removes the registered cinematic event.
<code>Set_Forwards_Barks (value)</code>	Sets whether the lua state this script is running in will handle forwarding barks to server AI players.

Functions

`Server_Register_Cinematic_Start_Event (cinematic_file, user_func)`

Registers the given function to run on the given cinematic when the cinematic starts.

Parameters:

- `cinematic_file` string of the cinematic file to apply to
- `user_func` function to call; `user_func` receives (`LCinematicRegistration`) as a parameter

Returns:

`LCinematicRegistration` ref, which can be used to unregister the event

`Server_Register_Cinematic_Finished_Event (cinematic_file, user_func)`

Registers the given function to run on the given cinematic when the cinematic completes.

Parameters:

- `cinematic_file` string of the cinematic file to apply to
- `user_func` function to call; `user_func` receives (`LCinematicRegistration`) as a parameter

Returns:

`LCinematicRegistration` ref, which can be used to unregister the event

`Server_Register_Cinematic_Frame_Event (cinematic_file, frame, user_func)`

Registers the given function to run on the given cinematic when the cinematic reaches the given frame.

Parameters:

- *cinematic_file* string of the cinematic file to apply to
- *frame number* of the frame
- *user_func* function to call; *user_func* receives (*LCinematicRegistration*) as a parameter

Returns:

LCinematicRegistration ref, which can be used to unregister the event

Server_Register_Cinematic_Skipped_Event (*cinematic_file*, *user_func*)

Registers the given function to run on the given cinematic when the cinematic is skipped.

Parameters:

- *cinematic_file* string of the cinematic file to apply to
- *user_func* function to call; *user_func* receives (*LCinematicRegistration*) as a parameter

Returns:

LCinematicRegistration ref, which can be used to unregister the event

Server_Register_Bark_Finished_Event (*bark_name*, *user_func*)

Registers the given function to run when the given bark type finishes.

Parameters:

- *bark_name* string of the bark name to apply to
- *user_func* function to call; *user_func* receives (string of bark name, *LCinematicRegistration*) as parameters

Returns:

LCinematicRegistration ref, which can be used to unregister the event

Server_Register_Bark_Started_Event (*bark_name*, *user_func*)

Registers the given function to run when the given bark type finishes.

Parameters:

- *bark_name* string of the bark name to apply to
- *user_func* function to call; *user_func* receives (string of bark name, *LCinematicRegistration*) as parameters

Returns:

LCinematicRegistration ref, which can be used to unregister the event

Server_Register_Strategic_Event_Popped_Up (*user_func*)

Registers the given function to run when a strategic event dialog pops up. The supplied function is called with a string parameter indicating the event type.

Parameters:

- *user_func* function to call; *user_func* receives (string of event name, *LCinematicRegistration*) as parameters

Returns:

LCinematicRegistration ref, which can be used to unregister the event

Server_Register_Strategic_Event_Closed (*user_func*)

Registers the given function to run when a strategic event dialog is closed. The supplied function is called with a string parameter indicating the event type.

Parameters:

- *user_func* function to call; *user_func* receives (string of event name, *LCinematicRegistration*) as parameters

Returns:

LCinematicRegistration ref, which can be used to unregister the event

Server_Unregister_Cinematic_Event (*registration_ref*)

Removes the registered cinematic event.

Parameters:

- *registration_ref* *LCinematicRegistration* of the registration

Returns:

nil

Set_Forwards_Barks (*value*)

Sets whether the lua state this script is running in will handle forwarding barks to server AI players.

Parameters:

- *value* boolean

Returns:

nil

ServerEffectSystem

Effect system functionality.

Class EffectApplicationStruct

EffectApplicationStruct:Set_Source_Object (object_ref)	Sets the source object for the CEffectApplicationStruct.
EffectApplicationStruct:Set_Target_Object (object_ref)	Sets the target object for the CEffectApplicationStruct.
EffectApplicationStruct:Set_Target_Position (x, y)	Sets the target position for the CEffectApplicationStruct.
EffectApplicationStruct:Set_Original_Source_Object (object_ref)	Sets the original source object for the CEffectApplicationStruct.
EffectApplicationStruct:Set_Original_Target_Object (object_ref)	Sets the original target object for the CEffectApplicationStruct.
EffectApplicationStruct:Set_Context_Value (value)	Sets the context value for the CEffectApplicationStruct.
EffectApplicationStruct:Set_Original_Player (object_ref)	Sets the original player for the CEffectApplicationStruct.

Functions

Create_EffectApplicationData ()	Creates a new CEffectApplicationStruct and returns a reference to it.
Activate_Effect_Generator (effect_generator_name, application_data_ref)	Creates a new effect generator via type name and applies it using the given application data.
Deactivate_Effect_Generator (generator_id)	Deactivates an effect generator via ID.

Class EffectApplicationStruct

An EffectApplicationStruct contains information for applying effect generators to objects.

EffectApplicationStruct:Set_Source_Object (object_ref)

Sets the source object for the CEffectApplicationStruct.

Parameters:

- object_ref CObjectRef of object

Returns:

nil

EffectApplicationStruct:Set_Target_Object (object_ref)

Sets the target object for the CEffectApplicationStruct.

Parameters:

- object_ref CObjectRef of object

Returns:

nil

EffectApplicationStruct:Set_Target_Position (x, y)

Sets the target position for the CEffectApplicationStruct.

Parameters:

- x number of x coordinate
- y number of y coordinate

Returns:

nil

EffectApplicationStruct:Set_Original_Source_Object (object_ref)

Sets the original source object for the CEffectApplicationStruct.

Parameters:

- object_ref CObjectRef of object

Returns:

nil

EffectApplicationStruct:Set_Original_Target_Object (object_ref)

Sets the original target object for the CEffectApplicationStruct.

Parameters:

- object_ref CObjectRef of object

Returns:

nil

EffectApplicationStruct:Set_Context_Value (value)

Sets the context value for the CEffectApplicationStruct.

Parameters:

- value number of context value

Returns:

nil

EffectApplicationStruct:Set_Original_Player (object_ref)

Sets the original player for the CEffectApplicationStruct.

Parameters:

- object_ref CObjectRef of player

Returns:

nil

Functions

Create_EffectApplicationData ()

Creates a new CEffectApplicationStruct and returns a reference to it.

Returns:

CEffectApplicationStruct

Activate_Effect_Generator (effect_generator_name, application_data_ref)

Creates a new effect generator via type name and applies it using the given application data.

Parameters:

- effect_generator_name string of effect generator type name

- application_data_ref CEffectApplicationStruct ref

Returns:

generator ID - which can be used to deactivate the generator (see **Deactivate_Effect_Generator**)

Deactivate_Effect_Generator (generator_id)

Deactivates an effect generator via ID.

Parameters:

- generator_id number of the ID

Returns:

boolean - true if it was deactivated, false if it failed to deactivate for any reason (such as the generator not existing anymore)

ServerGUIUtils

Various functionality that affects the client (via the server).

Functions

<code>Server_Start_Structure_Highlight (object_ref)</code>	Turns on the structure highlight effect on the object.
<code>Server_Stop_Structure_Highlight (object_ref)</code>	Turns off the structure highlight effect on the object.
<code>Server_Start_Unit_Highlight (object_ref)</code>	Turns on the unit highlight effect on the object.
<code>Server_Stop_Unit_Highlight (object_ref)</code>	Turns off the unit highlight effect on the object.

Functions

`Server_Start_Structure_Highlight (object_ref)`

Turns on the structure highlight effect on the object.

Parameters:

- `object_ref` `CObjectRef` of the object to highlight

Returns:

`nil`

`Server_Stop_Structure_Highlight (object_ref)`

Turns off the structure highlight effect on the object.

Parameters:

- `object_ref` `CObjectRef` of the object to remove the highlight of

Returns:

`nil`

`Server_Start_Unit_Highlight (object_ref)`

Turns on the unit highlight effect on the object.

Parameters:

- `object_ref` `CObjectRef` of the object to highlight

Returns:

`nil`

`Server_Stop_Unit_Highlight (object_ref)`

Turns off the unit highlight effect on the object.

Parameters:

- `object_ref` `CObjectRef` of the object to remove the highlight of

Returns:

`nil`

ServerInstanceManagement

Instance management functions that run on the server.

Functions

Switch_Instance (instance_name)	Advances into another instance.
Load_Instance (save_name)	Reloads a previously saved instance.
Save_Instance (save_name)	Save the current instance.
Set_JSON_Persistence (json_string)	Sets JSON persistence blob that can be used to pass data from one instance to another across a switch and load instance boundary.
Get_JSON_Persistence ()	Gets JSON persistence blob that can be used to pass data from one instance to another across a switch and load instance boundary.
Get_Instance_Extra_Settings ()	Gets JSON instance extra settings blob.
Tracked_Stats_Rebuild ()	Generate a new Stats snapshot to be used by the Tracked_Stats functions.
Tracked_Stats_Get_Total_For_Player (player_ref, stat_type)	Returns the value of the given stat for the given player.
Tracked_Stats_Get_Total_For_Team (team_id, stat_type)	Returns the value of the given stat for the given team.
Tracked_Stats_Get_Objects_Built_For_Player (player_ref, type_name)	Returns the total number of objects types built for the given player.

Functions

Switch_Instance (instance_name)

Advances into another instance.

Parameters:

- instance_name Name of the new instance to switch to.

Returns:

bool true on success

Load_Instance (save_name)

Reloads a previously saved instance.

Parameters:

- save_name Name of instance save (no path or extension).

Returns:

bool true on success

Save_Instance (save_name)

Save the current instance.

Parameters:

- save_name Name of instance save (no path or extension)(Can be null).

Returns:

Name of the save or null on failure.

Set_JSON_Persistence (json_string)

Sets JSON persistence blob that can be used to pass data from one instance to another across a switch and load instance boundary.

Parameters:

- json_string json string to be used for persistence.

Returns:

none

Get_JSON_Persistence ()

Gets JSON persistence blob that can be used to pass data from one instance to another across a switch and load instance boundary.

Returns:

json string

Get_Instance_Extra_Settings ()

Gets JSON instance extra settings blob.

Returns:

json string

Tracked_Stats_Rebuild ()

Generate a new Stats snapshot to be used by the Tracked_Stats functions.

Returns:

boolean true if successful

Tracked_Stats_Get_Total_For_Player (player_ref, stat_type)

Returns the value of the given stat for the given player.

Parameters:

- player_ref CObjectRef of player
- stat_type TrackedStatsEnum type (TSE_) of the stat to be queried.

Returns:

number, or nil

Tracked_Stats_Get_Total_For_Team (team_id, stat_type)

Returns the value of the given stat for the given team.

Parameters:

- team_id id of the team
- stat_type TeamTotalStatsEnum type (TTSE_) of the stat to be queried.

Returns:

number, or nil

Tracked_Stats_Get_Objects_Built_For_Player (player_ref, type_name)

Returns the total number of objects types built for the given player.

Parameters:

- player_ref CObjectRef of player
- type_name Object Type name to be queried for total built.

Returns:

number, or nil

ServerObjectManipulation

A server-only module that aids in manipulating objects.

Functions

<i>Add_Object_State (object_ref, state_string)</i>	<i>Adds a state to an object.</i>
<i>Remove_Object_State (object_ref, state_string)</i>	<i>Removes a state from an object.</i>
<i>Finish_Structure_Construction (object_ref)</i>	<i>Finishes construction of the given structure if it is constructing.</i>
<i>Set_Owner_Player (object_ref, player_ref)</i>	<i>Changes the unit's owner to the specified player.</i>
<i>Set_Object_X (object_ref, x)</i>	<i>Sets an object's x coordinate</i>
<i>Set_Object_Y (object_ref, y)</i>	<i>Sets an object's y coordinate</i>
<i>Kill_Object (object_ref)</i>	<i>Kills an object.</i>
<i>Create_Object (player_ref, object_type, x, y, facing, add_population)</i>	<i>Creates an object owned by the passed in player.</i>
<i>Request_Create_Structure (player_ref, object_type, x, y, facing, add_population)</i>	<i>Creates a structure owned by the given player and returns a reference to the created object.</i>
<i>Create_Reinforcement (player_ref, object_type, x, y, target_ref, fail_reason_table)</i>	<i>Creates reinforcements owned by the passed in player.</i>
<i>Get_Company_Reinforcement_Limit (player_ref, object_type)</i>	<i>Queries the number of reinforcements available to be spawned.</i>
<i>Set_Object_Attribute (object_ref, attribute_name, value)</i>	<i>Sets the attribute of an object.</i>
<i>Set_Capture_Point_Progress (object_ref, player_ref, progress)</i>	<i>Sets the capture point's capture progress.</i>
<i>Set_Capture_Point_Max_Value (object_ref, max_value)</i>	<i>Sets the capture point's max capture progress value.</i>
<i>Set_Capture_Point_Owning_Team (object_ref, player_ref)</i>	<i>Sets the capture point's owning team via the given player.</i>

Functions

Add_Object_State (object_ref, state_string)

Adds a state to an object.

Parameters:

- *object_ref* *CObjectRef* of object
- *state_string* The string representation of the state to add

Returns:

bool for whether or not the state was added, or *nil*

Remove_Object_State (object_ref, state_string)

Removes a state from an object.

Parameters:

- *object_ref* *CObjectRef* of object
- *state_string* The string representation of the state to remove

Returns:

bool for whether or not the state was removed, or *nil*

Finish_Structure_Construction (object_ref)

Finishes construction of the given structure if it is constructing. Finishes unit production of the currently producing unit if the structure is producing a unit.

Parameters:

- *object_ref* CObjectRef of structure

Returns:

nil

Set_Owner_Player (object_ref, player_ref)

Changes the unit's owner to the specified player.

Parameters:

- *object_ref* CObjectRef of object
- *player_ref* CObjectRef of player

Returns:

nil

Set_Object_X (object_ref, x)

Sets an object's x coordinate

Parameters:

- *object_ref* CObjectRef of object
- *x* The new x coordinate

Returns:

nil

Set_Object_Y (object_ref, y)

Sets an object's y coordinate

Parameters:

- *object_ref* CObjectRef of object
- *y* The new y coordinate

Returns:

nil

Kill_Object (object_ref)

Kills an object.

Parameters:

- *object_ref* CObjectRef of object to kill

Returns:

nil

Create_Object (player_ref, object_type, x, y, facing, add_population)

Creates an object owned by the passed in player.

Parameters:

- *player_ref* CObjectRef of player that will own the object
- *object_type* (string) the object type name
- *x* (number) the spawn x coordinate
- *y* (number) the spawn y coordinate
- *facing* (number) the facing the object should have
- *add_population* (boolean) should this add to popcap?

Returns:

CObjectRef of created unit

Request_Create_Structure (*player_ref*, *object_type*, *x*, *y*, *facing*, *add_population*)

Creates a structure owned by the given player and returns a reference to the created object. The created structure's position and facing will be adjusted according to placement rules.

Parameters:

- *player_ref* CObjectRef of player that will own the object
- *object_type* (string) the object type name
- *x* (number) the spawn x coordinate
- *y* (number) the spawn y coordinate
- *facing* (number) the facing the object should have
- *add_population* (boolean) should this add to popcap?

Returns:

- case #1 CObjectRef of created unit (or nil) and string of fail reason (may be empty)
- case #2 nil

Create_Reinforcement (*player_ref*, *object_type*, *x*, *y*, *target_ref*, *fail_reason_table*)

Creates reinforcements owned by the passed in player. Will obey pop-cap, structure voters, resource rules etc. Will fail if requirements not met

Parameters:

- *player_ref* CObjectRef of player that will own the object
- *object_type* (string) the object type name (must be a company type)
- *x* (number) the target move x coordinate
- *y* (number) the target move y coordinate
- *target_ref* the CObjectRef of a trench to place the reinforcement in, or nil
- *fail_reason_table* (table) table to contain the error result (if any)

Returns:

CObjectRef of created unit

Get_Company_Reinforcement_Limit (*player_ref*, *object_type*)

Queries the number of reinforcements available to be spawned.

Parameters:

- *player_ref* CObjectRef of player to be queried.
- *object_type* (string) the object type name (must be a company type)

Returns:

number of companies that can still be created, and fail_reason string

Set_Object_Attribute (object_ref, attribute_name, value)

Sets the attribute of an object.

Parameters:

- *object_ref CObjectRef of object to kill*
- *attribute_name string of attribute to modify*
- *value number of value to set attribute to*

Returns:

nil

Set_Capture_Point_Progress (object_ref, player_ref, progress)

Sets the capture point's capture progress.

Parameters:

- *object_ref CObjectRef of capture point*
- *player_ref CObjectRef of player capturing the point*
- *progress number of the new progress value*

Returns:

nil

Set_Capture_Point_Max_Value (object_ref, max_value)

Sets the capture point's max capture progress value.

Parameters:

- *object_ref CObjectRef of capture point*
- *max_value number of the new max progress value*

Returns:

nil

Set_Capture_Point_Owning_Team (object_ref, player_ref)

Sets the capture point's owning team via the given player.

Parameters:

- *object_ref CObjectRef of capture point*
- *player_ref CObjectRef of player whose team will own the capture point*

Returns:

nil

ServerObjectSelection

Server functions for controlling players' object selection.

Functions

Server_Is_Object_Selected (player_ref, object_ref)	Determines if the player has the object selected.
Server_Is_Object_Selectable (player_ref, object_ref)	Determines if the player can select the object.
Server_Set_Object_Selectable_For_Player (player_ref, object_ref, selectable)	Sets whether or not the player can select the object.
Server_Set_Object_Selectable (object_ref, selectable)	Sets whether or not any player can select the object.
Server_Select_Object (player_ref, object_ref)	Unselects all objects and then selects the given object for the given player.
Server_Unselect_Object (player_ref, object_ref)	Unselects the given object for the given player.
Server_Add_Object_To_Selection (player_ref, object_ref)	Adds the given object to the player's currently selected objects.
Server_Clear_Object_Selection (player_ref)	Unselects all objects for the given player.

Functions

Server_Is_Object_Selected (player_ref, object_ref)

Determines if the player has the object selected.

Parameters:

- player_ref CObjectRef of player
- object_ref CObjectRef of object to check

Returns:

nil

Server_Is_Object_Selectable (player_ref, object_ref)

Determines if the player can select the object.

Parameters:

- player_ref CObjectRef of player
- object_ref CObjectRef of object to check

Returns:

nil

Server_Set_Object_Selectable_For_Player (player_ref, object_ref, selectable)

Sets whether or not the player can select the object.

Parameters:

- player_ref CObjectRef of player
- object_ref CObjectRef of object
- selectable boolean; true to enable selection, false to disable selection

Returns:

nil

Server_Set_Object_Selectable (object_ref, selectable)

Sets whether or not any player can select the object.

Parameters:

- object_ref CObjectRef of object
- selectable boolean; true to enable selection, false to disable selection

Returns:

nil

Server_Select_Object (player_ref, object_ref)

Unselects all objects and then selects the given object for the given player.

Parameters:

- player_ref CObjectRef of player
- object_ref CObjectRef of object to select

Returns:

nil

Server_Unselect_Object (player_ref, object_ref)

Unselects the given object for the given player. Other selected objects are unaffected.

Parameters:

- player_ref CObjectRef of player
- object_ref CObjectRef of object to unselect

Returns:

nil

Server_Add_Object_To_Selection (player_ref, object_ref)

Adds the given object to the player's currently selected objects.

Parameters:

- player_ref CObjectRef of player
- object_ref CObjectRef of object to select

Returns:

nil

Server_Clear_Object_Selection (player_ref)

Unselects all objects for the given player.

Parameters:

- player_ref CObjectRef of player

Returns:

nil

ServerObjectives

Allows creation and manipulation of objectives that are saved and loaded.

Functions

<code>Create_Objective (player_selector, text_id)</code>	Creates a new objective, adds it to the UI, and returns the objective.
--	--

Class Objective

<code>Objective:Set_Objective_Text (text)</code>	Sets the displayed text for the objective.
<code>Objective:Set_Objective_Text_Non_Localized (text)</code>	Sets the displayed text for the objective.
<code>Objective:Set_Objective_Star_Type (star_type)</code>	Sets the type of star to display for the objective.
<code>Objective:Set_Objective_Completion_State (state)</code>	Sets the completion status for the objective.
<code>Objective:Set_Objective_Timer_Display (seconds, func, user_data)</code>	Sets a timer to be displayed along with the objective.
<code>Objective:Set_Objective_Fail_Timer (seconds)</code>	Sets a timer to be displayed along with the objective.
<code>Objective:Set_Objective_Complete_Timer (seconds)</code>	Sets a timer to be displayed along with the objective.
<code>Objective:Set_Objective_Counter_Data (counter, counter_max)</code>	Adds a counter to the objective display.
<code>Objective:Add_Objective_Counter (add_value)</code>	Adds to the current counter value of the objective.
<code>Objective:Get_Counter ()</code>	Returns the current counter value.
<code>Objective:Get_Counter_Max ()</code>	Returns the current counter maximum.
<code>Objective:Remove_Objective_Timer ()</code>	Removes the timer associated with the objective.
<code>Objective:Remove_Objective ()</code>	Removes the objective from the display.

Functions

`Create_Objective (player_selector, text_id)`

Creates a new objective, adds it to the UI, and returns the objective. The text value is localized, for non localized call `Objective:Set_Objective_Text_Non_Localized` afterward.

Parameters:

- `player_selector` (only when called on the server) RPC player selector (see `rpcplayerselector.lua`)
- `text_id` The text id of the localized text to use

Returns:

Objective of the created objective

Class Objective

An objective is able to display to a user their progress and other information in a built-in UI.

`Objective:Set_Objective_Text (text)`

Sets the displayed text for the objective. Uses localized text if available.

Parameters:

- text string of the text id to get the localized text of; falls back to just displaying the text directly if the text id is not found

Returns:

nil

Objective:Set_Objective_Text_Non_Localized (text)

Sets the displayed text for the objective. Uses the text directly as is, and does not attempt to localize.

Parameters:

- *text string of the text to display*

Returns:

nil

Objective:Set_Objective_Star_Type (star_type)

*Sets the type of star to display for the objective. See *ClientObjectivesDisplay.ObjectiveStarType**

Parameters:

- *star_type string for the type of star to display*

Returns:

nil

Objective:Set_Objective_Completion_State (state)

*Sets the completion status for the objective. See *ClientObjectivesDisplay.ObjectiveCompletionState**

Parameters:

- *state string for the completion state*

Returns:

nil

Objective:Set_Objective_Timer_Display (seconds, func, user_data)

Sets a timer to be displayed along with the objective. The timer may be set to execute a passed in function when it elapses.

Parameters:

- *seconds number for seconds*
- *func (optional) function to execute when the timer elapses*
- *user_data (optional) user_data to pass in to the passed in function*

Returns:

nil

Objective:Set_Objective_Fail_Timer (seconds)

Sets a timer to be displayed along with the objective. The timer will fail the objective if it elapses.

Parameters:

- *seconds number for seconds*

Returns:

nil

Objective:Set_Objective_Complete_Timer (seconds)

Sets a timer to be displayed along with the objective. The timer will complete the objective if it elapses.

Parameters:

- *seconds* number for seconds

Returns:

nil

Objective:Set_Objective_Counter_Data (counter, counter_max)

Adds a counter to the objective display. The counter display takes the form [x/y] where x is the current value and y is the maximum value.

Parameters:

- *counter* Number for the current value
- *counter_max* Number for the maximum value

Returns:

nil

Objective:Add_Objective_Counter (add_value)

Adds to the current counter value of the objective.

Parameters:

- *add_value* number for the amount to add

Returns:

nil

Objective:Get_Counter ()

Returns the current counter value.

Returns:

number for the current counter value, or nil

Objective:Get_Counter_Max ()

Returns the current counter maximum.

Returns:

number for the current counter maximum, or nil

Objective:Remove_Objective_Timer ()

Removes the timer associated with the objective. The timer does not execute any functionality as a result.

Returns:

nil

Objective:Remove_Objective ()

Removes the objective from the display.

Returns:

nil

ServerPlayers

Lua server-specific functionality for dealing with players.

Functions

Set_Player_Currency (player_ref, value)	Sets the player's currency to the given value.
Modify_Player_Currency (player_ref, value)	Modifies the player's currency by adding the given value to the current currency value.
Set_AI_Player_Enabled (player_ref, enable)	Enables (if true) or disables (if false) the specified AI player.
Modify_Player_National_Will (player_ref, value)	Modifies the player's national will by the given value.
Get_Player_Object_Price_Modifier_Invalid_ID ()	Returns the value that signifies an invalid modifier ID.
Add_Player_Object_Price_Modifier (player_ref, object_type_string, modifier_value, is_additive)	Adds a price modifier to the player for the given object type.
Remove_Player_Object_Price_Modifier (player_ref, object_type_string, modifier_id)	Removes a price modifier from a player specified by the given object type string and modifier ID.

Functions

Set_Player_Currency (player_ref, value)

Sets the player's currency to the given value.

Parameters:

- player_ref : CObjectRef of player
- value : number of the new currency value

Returns:

nil

Modify_Player_Currency (player_ref, value)

Modifies the player's currency by adding the given value to the current currency value. Positive and negative values are accepted.

Parameters:

- player_ref : CObjectRef of player
- value : number to modify the currency by

Returns:

nil

Set_AI_Player_Enabled (player_ref, enable)

Enables (if true) or disables (if false) the specified AI player. Only works if the player is defined as an AI player originally.

Parameters:

- player_ref : CObjectRef of AI player
- enable : boolean true to enable, false to disable

Returns:

nil

Modify_Player_National_Will (player_ref, value)

Modifies the player's national will by the given value.

Parameters:

- player_ref : CObjectRef of player
- value : number of the national will delta

Returns:

nil

Get_Player_Object_Price_Modifier_Invalid_ID ()

Returns the value that signifies an invalid modifier ID.

Returns:

number

Add_Player_Object_Price_Modifier (player_ref, object_type_string, modifier_value, is_additive)

Adds a price modifier to the player for the given object type.

Parameters:

- player_ref : CObjectRef of player
- object_type_string : string of object type
- modifier_value : number of the modification value
- is_additive : boolean of whether this modifier is additive (true) or multiplicative (false)

Returns:

number of modifier id, or nil

Remove_Player_Object_Price_Modifier (player_ref, object_type_string, modifier_id)

Removes a price modifier from a player specified by the given object type string and modifier ID.

Parameters:

- player_ref : CObjectRef of player
- object_type_string : string of object type
- modifier_id : number of the modifier ID to remove

Returns:

boolean of whether the modifier was successfully removed, or nil

ServerTGWUtils

Project TGW utility functions that run on the server.

Functions

Order_Company_Garrison_Trench (company_ref, trench_ref)	Orders the given company to garrison inside the given trench.
Order_Company_Ungarrison (company_ref)	Orders the given company to leave its current garrisoned trench.
Find_Target_Company_Lead_Point (object_ref, lead_time_seconds)	Finds the position to target such that the given target will be hit x seconds later.
Use_Ability_Item_Leading_Target (order_object_ref, item_name, order_object_ref, lead_time_seconds, facing_offset)	Fires the given ability item leading the target object.
Order_Company_With_Target_Object (company_ref, order_type, target_ref)	Gives an order with a target object to the specified company.
Order_Company_With_Target_Position (company_ref, order_type, x, y)	Gives an order with a target position to the specified company.
Order_Company_With_No_Target (company_ref, order_type)	Gives an order with no target to the specified company.
Set_Company_Formation (company_ref, formation_type)	Changes the formation of the specified company.
Retreat_Company (company_ref)	Sets the company to retreat.
Is_Trench_Connected_To_Trench (start_trench, end_trench, filter_full_trenches)	Performs a pathfind and determines if a move order between the trenches would follow the trench network.
Use_Ability_Item_With_Target_Object (object_ref, item_name, target_ref)	Uses an ability item of an object with the specified target object.
Use_Ability_Item_With_Target_Position (object_ref, item_name, x, y, facing_degrees)	Uses an ability item of an object with the specified target position.
Use_Ability_Item_With_No_Target (object_ref, item_name)	Uses an ability item of an object with no target.
Start_Strategic_Event (event_name)	Starts a strategic event of the given object type name.
Add_Strategic_Region_Effect (region_name, effect_name)	Adds a region effect to a specific strategic region.
Get_Appropriate_Spawn_Point_For_Reinforcement (object_type_name, player_ref, x, y)	Finds a spawn location object nearest to the destination point usable by the given object type and player.
Get_Company_Known_Combat_Targets (company_ref)	Returns a table containing the company's tracked enemies when in combat.
Is_Target_Within_Company_Firing_Range (company_ref, target_ref)	Determines if the company is within range to fire at the given object.
Request_Instance_Settings (object_ref, lead_time_seconds)	Returns the current match settings for skirmish/multiplayer.
Set_Map_Environment (environment, season)	Changes to the specified map visual environment
Queue_Pre_Battle_Artillery_Event (player_ref, days_of_bombardment)	Queues up days of pre-battle bombardment
Get_Current_Strategic_Region ()	Gets the defender's region ID.
Get_Current_Front_Age ()	Gets the age of the front.
Get_Current_Front_Battle_Count ()	Gets the number of battles at the front.
Get_Instance_Random_Template_Value ()	Gets the random value used to pick template instance for trenches
Can_See_Position ()	Checks to see if provided position is visible by the specified player
Can_Target_Position ()	Checks to see if provided position is targetable by the specified object
Can_Ability_Target_Position (object_ref, item_name, x, y)	Checks to see if an ability item of an object can be used with the specified target position.
Server_Enable_Mod_Debugging_Tools ()	Enables mod debugging tools.

Functions

Order_Company_Garrison_Trench (company_ref, trench_ref)

Orders the given company to garrison inside the given trench.

Parameters:

- company_ref: CObjectRef of company object
- trench_ref: CObjectRef of trench object

Returns:

nil

Order_Company_Ungarrison (company_ref)

Orders the given company to leave its current garrisoned trench.

Parameters:

- company_ref: CObjectRef of company object

Returns:

nil

Find_Target_Company_Lead_Point (object_ref, lead_time_seconds)

Finds the position to target such that the given target will be hit x seconds later.

Parameters:

- object_ref: CObjectRef of target object
- lead_time_seconds: number of seconds ahead to look

Returns:

(number) x of position, (number) y of position

Use_Ability_Item_Leading_Target (order_object_ref, item_name, order_object_ref, lead_time_seconds, facing_offset)

Fires the given ability item leading the target object. If the ability can be given a facing, it will face toward the lead point from the target.

Parameters:

- order_object_ref: CObjectRef of object to target the ability towards
- item_name: string of the item ability to use
- order_object_ref: CObjectRef of object to target the ability towards
- lead_time_seconds: number of seconds to lead the target by
- facing_offset: (optional) number of degrees to offset the calculated ability facing by

Returns:

boolean true if the ability was activated, false otherwise

Order_Company_With_Target_Object (company_ref, order_type, target_ref)

Gives an order with a target object to the specified company.

Parameters:

- company_ref: CObjectRef of company object
- order_type: string of order type
- target_ref: CObjectRef of target object

Returns:

nil

Usage:
Available orders are: "MoveToLocation", "Attack"

Order_Company_With_Target_Position (company_ref, order_type, x, y)

Gives an order with a target position to the specified company.

Parameters:

- company_ref: CObjectRef of company object
- order_type: string of order type
- x: number of x position
- y: number of y position

Returns:

nil

Usage:
Available orders are: "MoveToLocation", "Attack"

Order_Company_With_No_Target (company_ref, order_type)

Gives an order with no target to the specified company.

Parameters:

- company_ref: CObjectRef of company object
- order_type: string of order type

Returns:

Available orders are: "MoveToLocation", "Attack"

Usage:
Available orders are: "Stop"

Set_Company_Formation (company_ref, formation_type)

Changes the formation of the specified company.

Parameters:

- company_ref: CObjectRef of company object
- formation_type: string of formation type (Skirmish, Column, Square, TrenchTraversal)

Returns:

nil

Usage:

Available formation types are: "Skirmish", "Column", "Square", "TrenchTraversal"

Retreat_Company (company_ref)

Sets the company to retreat.

Parameters:

- company_ref: CObjectRef of company to retreat

Returns:

nil

Is_Trench_Connected_To_Trench (start_trench, end_trench, filter_full_trenches)

Performs a pathfind and determines if a move order between the trenches would follow the trench network. Notice: does not necessarily return whether there is a trench path between the two trenches, as there are situations during normal movement that would cause companies to leave the trench network even with an available trench path.

Parameters:

- start_trench: CObjectRef of starting trench
- end_trench: CObjectRef of destination trench
- filter_full_trenches: boolean to determine if the pathfind should path around full trenches; the normal behavior sets this to true

Use_Ability_Item_With_Target_Object (object_ref, item_name, target_ref)

Uses an ability item of an object with the specified target object.

Parameters:

- object_ref: CObjectRef of item owner object
- item_name: string of item name
- target_ref: CObjectRef of target object

Returns:

true if the ability was activated, false otherwise

Usage:

Item must be in an ability item inventory

Use_Ability_Item_With_Target_Position (object_ref, item_name, x, y, facing_degrees)

Uses an ability item of an object with the specified target position.

Parameters:

- object_ref: CObjectRef of item owner object
- item_name: string of item name
- x: number of x position
- y: number of y position
- facing_degrees: (optional) number for the facing degrees to aim the ability at the position

Returns:

true if the ability was activated, false otherwise

Usage:

Item must be in an ability item inventory

Use_Ability_Item_With_No_Target (object_ref, item_name)

Uses an ability item of an object with no target.

Parameters:

- object_ref: CObjectRef of item owner object
- item_name: string of item name

Returns:

true if the ability was activated, false otherwise

Usage:

Item must be in an ability item inventory

Start_Strategic_Event (event_name)

Starts a strategic event of the given object type name.

Parameters:

- event_name: string of strategic event object name.

Returns:

true if the event was started, false otherwise

Add_Strategic_Region_Effect (region_name, effect_name)

Adds a region effect to a specific strategic region.

Parameters:

- region_name: string of name of the strategic region to be affected.
- effect_name: string of name of the strategic effect to be added to the region.

Returns:

true if the effect was added, false otherwise

Get_Appropriate_Spawn_Point_For_Reinforcement (object_type_name, player_ref, x, y)

Finds a spawn location object nearest to the destination point usable by the given object type and player.

Parameters:

- object_type_name: string of object type
- player_ref CObjectRef: of player that will spawn the object
- x: (number) x-coordinate of the destination
- y: (number) y-coordinate of the destination

Returns:

CObjectRef of spawn location object, or nil

Get_Company_Known_Combat_Targets (company_ref)

Returns a table containing the company's tracked enemies when in combat.

Parameters:

- company_ref CObjectRef: of the company

Returns:

table of CObjectRefs (may be empty), or nil

Is_Target_Within_Company_Firing_Range (company_ref, target_ref)

Determines if the company is within range to fire at the given object.

Parameters:

- company_ref: CObjectRef of company checking the target
- target_ref: CObjectRef of target being fired upon

Returns:

boolean, or nil

Request_Instance_Settings (object_ref, lead_time_seconds)

Returns the current match settings for skirmish/multiplayer.

Parameters:

- object_ref: CObjectRef of target object
- lead_time_seconds: number of seconds ahead to look

Returns:

table of settings

Set_Map_Environment (environment, season)

Changes to the specified map visual environment

Parameters:

- environment: string of environment weather name ("Standard", "Snow", "Rain")
- season string: of season name ("Summer", "Winter", "Fall")

Returns:

true

Queue_Pre_Battle_Artillery_Event (player_ref, days_of_bombardment)

Queues up days of pre-battle bombardment

Parameters:

- player_ref: CObjectRef of player to queue the bombardment for. Player must have enough currency to pay for each day of bombardment
- days_of_bombardment: the number of days of bombardment to queue up

Returns:

nil

Get_Current_Strategic_Region ()

Gets the defender's region ID. Only useful during a tactical match played as part of the campaign

Returns:

Region id, or -1 if not found

Get_Current_Front_Age ()

Gets the age of the front. Only useful during a tactical match played as part of the campaign

Returns:

Region id, or -1 if not found

Get_Current_Front_Battle_Count ()

Gets the number of battles at the front. Only useful during a tactical match played as part of the campaign

Returns:

0 if not found

Get_Instance_Random_Template_Value ()

Gets the random value used to pick template instance for trenches

Returns:

0 if not found

Can_See_Position ()

Checks to see if provided position is visible by the specified player

Returns:

true if visible

Can_Target_Position ()

Checks to see if provided position is targetable by the specified object

Returns:

true if visible

Can_Ability_Target_Position (object_ref, item_name, x, y)

Checks to see if an ability item of an object can be used with the specified target position.

Parameters:

- object_ref: CObjectRef of item owner object
- item_name: string of item name
- x: number of x position
- y: number of y position

Returns:

true if the ability can activate, false otherwise

Usage:

Item must be in an ability item inventory

Server_Enable_Mod_Debugging_Tools ()

Enables mod debugging tools. Called on the server. Activates mod debugging tools functionality on the client.

Returns:

nil

ServerUnitOrders

Defines server-only methods for dealing with unit orders.

Functions

Raw_Order_Object (ordered_object_ref, order_ref)	Orders an object using a CUnitOrder.
Order_Object (ordered_object_ref, order)	Orders an object using a Lua table containing values controlling the order.

Tables

Order	Fields of a Lua order table
--------------	-----------------------------

Functions

Raw_Order_Object (ordered_object_ref, order_ref)

Orders an object using a CUnitOrder.

Parameters:

- ordered_object_ref : CObjectRef of object to order
- order_ref : CUnitOrder input

Returns:

nil

Order_Object (ordered_object_ref, order)

Orders an object using a Lua table containing values controlling the order.

Parameters:

- ordered_object_ref : CObjectRef of object to order
- order : A Lua table containing data to create the order

Returns:

nil

Usage:

```
local order = {}
order.OrderType = "MoveToLocation"
order.MaxSpeed = 500
order.TargetX = -1234
order.TargetY = 4321
order.ContextValue = 500
order.MoveRange = 0
order.Protected = false

pgserver.Order_Object(object, order) -- assumes you already have a CObjectRef stored in variable object
```

Tables

Order

Fields of a Lua order table

Fields:

- OrderType (number) : Check **UnitOrders.Get_Order_Type_From_String** to get this value
- Autonomous (boolean)
- Protected (boolean) : A protected order cannot be canceled by the user
- UseSimpleOverridePath (boolean)
- FromRally (boolean)
- OverrideMovementPriority (number) : A specific number value (see UnitOrders.h)
- TargetX (number) : The target x coordinate
- TargetY (number) : The target y coordinate
- TargetObject (CObjectRef) : The target object
- MoveRange (number) : How close the object must get to the desired location to fulfill the order
- AttachTarget (CObjectRef) : The attach target object
- CachedContextTargetX (number) : Cached context target x coordinate
- CachedContextTargetY (number) : Cached context target y coordinate
- CachedContextTargetObject (CObjectRef) : The cached context target object
- ContextValue (number) : A context value whose usage depends on the type of order
- MaxSpeed (number) : The speed the object can move at while fulfilling a movement order
- LastKnownTargetPositionX (number) : Last known target position x coordinate
- LastKnownTargetPositionY (number) : Last known target position y coordinate

ServerVictoryConditions

Functions that modify the victory condition state of the game.

Functions

Set_Game_Victory_Mode_Standard ()	Sets the victory mode to standard.
Set_Game_Victory_Mode_DestroyHQ ()	Sets the victory mode to destroy headquarters.
Set_Game_Victory_Mode_Annihilation ()	Sets the victory mode to annihilation.
Set_Game_Victory_Mode_Regicide ()	Sets the victory mode to regicide.
Set_Game_Victory_Mode_None ()	Sets the victory mode to nothing.

Functions

Set_Game_Victory_Mode_Standard ()

Sets the victory mode to standard.

Returns:

nil

Set_Game_Victory_Mode_DestroyHQ ()

Sets the victory mode to destroy headquarters.

Returns:

nil

Set_Game_Victory_Mode_Annihilation ()

Sets the victory mode to annihilation.

Returns:

nil

Set_Game_Victory_Mode_Regicide ()

Sets the victory mode to regicide.

Returns:

nil

Set_Game_Victory_Mode_None ()

Sets the victory mode to nothing. The game will not end automatically - and should be controlled by map script.

Returns:

nil

Localization

Localization string manipulation functions.

Functions

Localize (text_id)	Returns the localized text string for a given text id.
Replace_Tokens (text, tokens)	Replaces numbered tokens within a string with the indexed elements in the table.

Functions

Localize (text_id)

Returns the localized text string for a given text id.

Parameters:

- text_id : text id of the string to be localized.

Returns:

utf8 localized string based on the current game language.

Replace_Tokens (text, tokens)

Replaces numbered tokens within a string with the indexed elements in the table. Starts replacing tokens at ##0, continuing until no more entries in the token table.

Parameters:

- text : text containing numbered(##0, ##1, ##2, ...) tokens to be replaced.
- tokens : indexed table containing the tokens to substitute. 1-Based table, tostring() called on each element.

Returns:

utf8 localized string based on the current game language.

Usage:

```
Replace_Tokens("Send $##1,$##0 resources to ally.", {34, 21}), Output: Send $21,$34 resources to ally.
```

ObjectManipulation

A module that aids in manipulating game objects.

Functions

Create_ObjectSet ()	Creates a new wrapped CObjectSet type.
ObjectSet_Values ()	Iterates over a CObjectSet type's values.
ObjectSet_As_Table ()	Returns a Lua table containing the CObjectRef objects in the given CObjectSet.
Collect_Objects_In_Range_Of_Point (objectset_ref, point_x, point_y, radius)	Collects objects near a point into a CObjectSet.
Collect_Player_Objects (objectset_ref, player_ref)	Collects a player's objects into a CObjectSet.
Collect_Trench_Objects_Connected_To_Point (objectset_ref, trench_obj, point_index)	Collects the objects(trenches) attached to the connector point associated with trench_obj into a CObjectSet.
Collect_Objects_By_Type (objectset_ref, type_string)	Collects a player's objects into a CObjectSet.
Collect_Objects_By_Classification (objectset_ref, classification_string)	Collects a player's objects into a CObjectSet.
Get_Object_By_Name (name)	Find and return an object with the given name.
Get_Object_By_ID (id)	Find and return an object with the given ID.
Get_Object_ID_In_Use (id)	Checks if the object ID is in use.
Get_Object_X (object_ref)	Gets an object's x coordinate
Get_Object_Y (object_ref)	Gets an object's y coordinate
Get_Object_Name (object_ref)	Get the given object's name.
Get_Object_ID (object_ref)	Get the given object's ID.
Is_Object_Valid (object_ref)	Returns true if the object is valid.
Get_Object_Classification (object_ref)	Get the given object's classification.
Object_Has_Locomotor (object_ref)	Returns whether an object has a locomotor or not.
Get_Object_Owner_Player (object_ref)	Returns the owning player of an object.
Get_Object_Type_Name (object_ref)	Returns the object type name of an object.
Is_Object_In_State (object_ref, state_string)	Returns whether or not the object is in the passed in state.
Get_Object_Facing (CObjectRef)	Returns the object's logical Z facing.
Is_Object_State_String_Valid (state_string)	With a given string, returns true if the string matches an existing object state (returns false otherwise).
Get_Object_Attribute_Value (object_ref, attribute_name)	Returns the value associated with the given attribute name for the object.
Get_Default_Attribute_Value (object_type, attribute_name, level)	Returns the default value associated with the given attribute name for the object type.
Get_Capture_Point_Progress (object_ref)	Gets the capture point's capture progress.
Get_Capture_Point_Max_Value (object_ref)	Gets the capture point's max value for capture progress.
Get_Capture_Point_Capturing_Team (object_ref)	Gets the capture point's capturing team ID.
Get_Capture_Point_Owning_Team (object_ref)	Gets the capture point's owning team ID.

Functions

Create_ObjectSet ()

Creates a new wrapped CObjectSet type. This type is used in various library functions that deal with multiple objects at a time. See the referenced ObjectSet_Values function for iterating over its values.

Returns:

CObjectSet

See also:

ObjectSet_Values

ObjectSet_Values ()

Iterates over a CObjectSet type's values.

Returns:

CObjectSet iterator

Usage:

```
local objectset = Create_ObjectSet()
-- add objects to the objectset via some other call
for object in ObjectSet_Values(objectset) do
  -- do something with object
end
```

ObjectSet_As_Table ()

Returns a Lua table containing the CObjectRef objects in the given CObjectSet.

Returns:

table of CObjectRefs

Collect_Objects_In_Range_Of_Point (objectset_ref, point_x, point_y, radius)

Collects objects near a point into a CObjectSet.

Parameters:

- objectset_ref: CObjectSet to write into
- point_x: The x coordinate
- point_y: The y coordinate
- radius: The radius to collect within

Returns:

nil

Collect_Player_Objects (objectset_ref, player_ref)

Collects a player's objects into a CObjectSet.

Parameters:

- objectset_ref: CObjectSet to write into
- player_ref: The CObjectRef of a player

Returns:

nil

Collect_Trench_Objects_Connected_To_Point (objectset_ref, trench_obj, point_index)

Collects the objects(trenches) attached to the connector point associated with trench_obj into a CObjectSet.

Parameters:

- objectset_ref: CObjectSet to write into
- trench_obj: The CObjectRef of a trench
- point_index: 0 based index of the connector point to collect objects from, nil for all connector points.

Returns:

nil

Collect_Objects_By_Type (objectset_ref, type_string)

Collects a player's objects into a CObjectSet.

Parameters:

- objectset_ref: CObjectSet to write into
- type_string: The object type's name

Returns:

nil

Collect_Objects_By_Classification (objectset_ref, classification_string)

Collects a player's objects into a CObjectSet.

Parameters:

- objectset_ref: CObjectSet to write into
- classification_string: The object's classification

Returns:

nil

Get_Object_By_Name (name)

Find and return an object with the given name. Returns nil if the object is not found.

Parameters:

- name: The name of the object to find.

Returns:

CObjectRef of object, or nil

Get_Object_By_ID (id)

Find and return an object with the given ID. Returns nil if the object is not found.

Parameters:

- id number: of the object ID

Returns:

CObjectRef of object, or nil

Get_Object_ID_In_Use (id)

Checks if the object ID is in use.

Parameters:

- id: number of object ID

Returns:

boolean

Get_Object_X (object_ref)

Gets an object's x coordinate

Parameters:

- object_ref: CObjectRef of object

Returns:

number, or nil

Get_Object_Y (object_ref)

Gets an object's y coordinate

Parameters:

- object_ref: CObjectRef of object

Returns:

number, or nil

Get_Object_Name (object_ref)

Get the given object's name.

Parameters:

- object_ref: CObjectRef of object

Returns:

string, or nil

Get_Object_ID (object_ref)

Get the given object's ID.

Parameters:

- object_ref: CObjectRef of object

Returns:

number, or nil

Is_Object_Valid (object_ref)

Returns true if the object is valid. If false, the object has probably been deleted already.

Parameters:

- object_ref: CObjectRef of object

Returns:

boolean

Get_Object_Classification (object_ref)

Get the given object's classification.

Parameters:

- object_ref: CObjectRef of object

Returns:

string, or nil

Object_Has_Locomotor (object_ref)

Returns whether an object has a locomotor or not.

Parameters:

- object_ref: CObjectRef of object

Returns:

boolean, or nil

Get_Object_Owner_Player (object_ref)

Returns the owning player of an object.

Parameters:

- object_ref: CObjectRef of object

Returns:

CObjectRef of player, or nil

Get_Object_Type_Name (object_ref)

Returns the object type name of an object.

Parameters:

- object_ref: CObjectRef of object

Returns:

string, or nil

Is_Object_In_State (object_ref, state_string)

Returns whether or not the object is in the passed in state.

Parameters:

- object_ref: CObjectRef of object
- state_string: State name as string

Returns:

bool, or nil

Get_Object_Facing (CObjectRef)

Returns the object's logical Z facing.

Parameters:

- CObjectRef: of object

Returns:

number, or nil

Is_Object_State_String_Valid (state_string)

With a given string, returns true if the string matches an existing object state (returns false otherwise).

Parameters:

- state_string: The string to validate

Returns:

bool

Get_Object_Attribute_Value (object_ref, attribute_name)

Returns the value associated with the given attribute name for the object.

Parameters:

- object_ref: CObjectRef of object
- attribute_name: Attribute name as string

Returns:

number, or nil

Get_Default_Attribute_Value (object_type, attribute_name, level)

Returns the default value associated with the given attribute name for the object type.

Parameters:

- object_type: Name of the object type
- attribute_name: Attribute name as string
- level: the level of the object (defaults to 0)

Returns:

number, or nil

Get_Capture_Point_Progress (object_ref)

Gets the capture point's capture progress.

Parameters:

- object_ref: CObjectRef of capture point object

Returns:

number, or nil

Get_Capture_Point_Max_Value (object_ref)

Gets the capture point's max value for capture progress.

Parameters:

- object_ref: CObjectRef of capture point object

Returns:

number, or nil

Get_Capture_Point_Capturing_Team (object_ref)

Gets the capture point's capturing team ID.

Parameters:

- object_ref: CObjectRef of capture point object

Returns:

number of team ID, or nil

Get_Capture_Point_Owning_Team (object_ref)

Gets the capture point's owning team ID.

Parameters:

- object_ref: CObjectRef of capture point object

Returns:

number of team ID, or nil

Players

Lua functionality for dealing with players.

Functions

Get_Players (objectset_ref)	Adds the players in the game into the CObjectSet.
Get_Environment_Player ()	Gets the environment player.
Get_Player_Name (object_ref)	Gets the given player's name.
Get_Player_Currency (CObjectRef)	Gets the given player's current currency value.
Get_Player_Strategic_Currency (CObjectRef)	Gets the given player's current strategic currency value.
Is_Player_Human (object_ref)	Returns whether or not a player is human.
Is_Player_Ally (checking_player_ref, target_player_ref)	Returns whether or not the first player is an ally to the second player
Is_Player_Enemy (checking_player_ref, target_player_ref)	Returns whether or not the first player is an enemy to the second player
Is_Player_Neutral (checking_player_ref, target_player_ref)	Returns whether or not the first player is neutral to the second player
Get_Player_Team_ID (player_ref)	Returns the team ID of the given player.
Get_Player_Faction (player_ref)	Returns the player's faction as a string.
Get_Player_Faction_Index (player_ref)	Returns the index of the player's faction.
Get_First_Player_Of_Faction ()	Gets the first player of specified faction.
Can_Player_Build_Structure (player_ref, string)	Checks whether the player can build the given structure type.
Can_Player_Build_Unit (player_ref, factory_ref, unit_type_name)	Checks whether the player can build the given unit type.
Get_Player_Factories_For_Unit (player_ref, unit_type_name, objectset_ref)	Gets the given player's factories that can build the given unit type.
Get_Player_Max_Population (player_ref)	Gets the player's maximum population.
Get_Player_Population (player_ref)	Gets the player's population.
Get_Player_Can_End_Turn (player_ref)	Gets the player can click end turn button.
Set_Player_Can_End_Turn (player_ref)	Sets the player can click end turn button.

Functions

Get_Players (objectset_ref)

Adds the players in the game into the CObjectSet.

Parameters:

- objectset_ref : The CObjectSet to write into

Returns:

nil

Get_Environment_Player ()

Gets the environment player.

Returns:

CObjectRef of player, or nil

Get_Player_Name (object_ref)

Gets the given player's name.

Parameters:

- object_ref : CObjectRef of player

Returns:

string, or nil

Get_Player_Currency (CObjectRef)

Gets the given player's current currency value.

Parameters:

- CObjectRef of player

Returns:

number, or nil

Get_Player_Strategic_Currency (CObjectRef)

Gets the given player's current strategic currency value.

Parameters:

- CObjectRef of player

Returns:

number, or nil

Is_Player_Human (object_ref)

Returns whether or not a player is human.

Parameters:

- object_ref : CObjectRef of player

Returns:

boolean, or nil

Is_Player_Ally (checking_player_ref, target_player_ref)

Returns whether or not the first player is an ally to the second player.

Parameters:

- checking_player_ref : CObjectRef of player doing the ally check
- target_player_ref : CObjectRef of player being checked against

Returns:

boolean, or nil

Is_Player_Enemy (checking_player_ref, target_player_ref)

Returns whether or not the first player is an enemy to the second player.

Parameters:

- `checking_player_ref` : CObjectRef of player doing the enemy check
- `target_player_ref` : CObjectRef of player being checked against

Returns:

boolean, or nil

Is_Player_Neutral (checking_player_ref, target_player_ref)

Returns whether or not the first player is neutral to the second player.

Parameters:

- `checking_player_ref` : CObjectRef of player doing the ally check
- `target_player_ref` : CObjectRef of player being checked against

Returns:

boolean, or nil

Get_Player_Team_ID (player_ref)

Returns the team ID of the given player.

Parameters:

- `player_ref` : CObjectRef of player doing the ally check

Returns:

number, or nil

See also:

`PG.InvalidTeamID`

Get_Player_Faction (player_ref)

Returns the player's faction as a string.

Parameters:

- `player_ref` : CObjectRef of player

Returns:

string of faction, or nil

Get_Player_Faction_Index (player_ref)

Returns the index of the player's faction.

Parameters:

- `player_ref` : CObjectRef of player

Returns:

number, or nil

Get_First_Player_Of_Faction ()

Gets the first player of specified faction.

Returns:

CObjectRef of player, or nil

Can_Player_Build_Structure (player_ref, string)

Checks whether the player can build the given structure type.

Parameters:

- player_ref : CObjectRef of player
- string of the structure type

Returns:

boolean, or nil

Can_Player_Build_Unit (player_ref, factory_ref, unit_type_name)

Checks whether the player can build the given unit type. Checks whether a specific factory can build it if factory_ref is supplied, otherwise checks every factory.

Parameters:

- player_ref : CObjectRef of player
- factory_ref : CObjectRef of factory to check; can be nil to check every factory
- unit_type_name : string of unit type to check

Returns:

boolean, or nil

Get_Player_Factories_For_Unit (player_ref, unit_type_name, objectset_ref)

Gets the given player's factories that can build the given unit type.

Parameters:

- player_ref : CObjectRef of player
- unit_type_name : string of unit type
- objectset_ref : CObjectSet to write to; note that the set is cleared before any additions are made

Returns:

nil

Get_Player_Max_Population (player_ref)

Gets the player's maximum population.

Parameters:

- player_ref : CObjectRef of player

Returns:

number, or nil

Get_Player_Population (player_ref)

Gets the player's population.

Parameters:

- player_ref : CObjectRef of player

Returns:

number, or nil

Get_Player_Can_End_Turn (player_ref)

Gets the player can click end turn button.

Parameters:

- player_ref : CObjectRef of player

Returns:

number, or nil

Set_Player_Can_End_Turn (player_ref)

Sets the player can click end turn button.

Parameters:

- player_ref : CObjectRef of player

Returns:

nil

TGWUtils

Project TGW utility functions that run on the client and server.

Functions

Get_Company_From_Object (object_ref)	Attempts to get the company object associated with the passed in object.
Get_Company_Health (object_ref)	Returns the current health value of the company.
Get_Company_Max_Health (object_ref)	Returns the max health value of the company.
Get_Company_Morale (object_ref)	Returns the current morale value of the company.
Get_Company_Max_Morale (object_ref)	Returns the max morale value of the company.
Get_Player_Supply (player_ref)	Returns the current supply of the player.
Get_Trench_Occupation (object_ref, player_ref)	Returns the amount of companies manning a trench for the given player.
Get_Trench_Max_Occupation (object_ref)	Returns the maximum number of companies that can occupy the trench for one player.
Get_Trench_Occupied_By_Company (CObjectRef)	Returns the trench being occupied by the given company.
Get_Companies_Occupying_Trench (CObjectRef, CObjectSet)	Gets the companies occupying the trench and adds them to the given object set.
Get_Strategic_Path_Size (player_ref)	Returns if there is a valid path between 2 regions.
Get_Map_Playable_Boundary_Coordinates ()	Returns the coordinates for the map's boundary in a table.
Get_Nearest_Passable_Point (movement_class, x, y)	Finds the nearest passable point that the given object type can occupy with the given target location.
Get_Bit_Value_Of_Passability_Type ()	Returns the integer value in code for the specified passability type.
Location_Has_Any_Matching_Passability_Type_Bit ()	Determines whether or not any of the passed in passability bits are present at the given x/y coordinates.

Functions

Get_Company_From_Object (object_ref)

Attempts to get the company object associated with the passed in object. Passing in either the company itself or a company member is valid.

Parameters:

- object_ref : CObjectRef of object to grab company from

Returns:

CObjectRef of company, or nil (if finding the company fails)

Get_Company_Health (object_ref)

Returns the current health value of the company.

Parameters:

- object_ref : CObjectRef of company

Returns:

number of health, or nil

Get_Company_Max_Health (object_ref)

Returns the max health value of the company.

Parameters:

- object_ref : CObjectRef of company

Returns:

number of max health, or nil

Get_Company_Morale (object_ref)

Returns the current morale value of the company.

Parameters:

- object_ref : CObjectRef of company

Returns:

number of morale, or nil

Get_Company_Max_Morale (object_ref)

Returns the max morale value of the company.

Parameters:

- object_ref : CObjectRef of company

Returns:

number of max morale, or nil

Get_Player_Supply (player_ref)

Returns the current supply of the player.

Parameters:

- player_ref : CObjectRef of player

Returns:

number of supply, or nil

Get_Trench_Occupation (object_ref, player_ref)

Returns the amount of companies manning a trench for the given player.

Parameters:

- object_ref : CObjectRef of trench object
- player_ref : CObjectRef of player

Returns:

number, or nil

Get_Trench_Max_Occupation (object_ref)

Returns the maximum number of companies that can occupy the trench for one player.

Parameters:

- object_ref : CObjectRef of trench object

Returns:

number, or nil

Get_Trench_Occupied_By_Company (CObjectRef)

Returns the trench being occupied by the given company.

Parameters:

- CObjectRef of company

Returns:

CObjectRef of occupied trench, or nil

Get_Companies_Occupying_Trench (CObjectRef, CObjectSet)

Gets the companies occupying the trench and adds them to the given object set.

Parameters:

- CObjectRef of trench
- CObjectSet

Returns:

nil

Get_Strategic_Path_Size (player_ref)

Returns if there is a valid path between 2 regions.

Parameters:

- player_ref : CObjectRef of player

Returns:

number of supply, or nil

Get_Map_Playable_Boundary_Coordinates ()

Returns the coordinates for the map's boundary in a table.

Returns:

- table with entries: (number) ["MinX"], (number) ["MaxX"], (number) ["MinY"], (number) ["MaxY"]
- returns nil if not in map

Get_Nearest_Passable_Point (movement_class, x, y)

Finds the nearest passable point that the given object type can occupy with the given target location.

Parameters:

- movement_class : string of the movement class to check
- x : number for the x-coordinate

- y : number for the y-coordinate

Returns:

- boolean (for whether input is passable), number for nearest x point, number for nearest y point
- returns nil if not in map

Get_Bit_Value_Of_Passability_Type ()

Returns the integer value in code for the specified passability type.

Returns:

number

Location_Has_Any_Matching_Passability_Type_Bit ()

Determines whether or not any of the passed in passability bits are present at the given x/y coordinates.

Returns:

boolean

TerrainRegions

Terrain regions functionality. (Limited to the server)

Functions

Get_Terrain_Region_Bits_At_Location (x, y)	Gets the terrain region bits at the location.
Get_Terrain_Region_Bits_At_Location_Table (x, y)	Gets the terrain region bits at the location and returns the bit indices as entries in a table.
Get_Terrain_Region_Bits_In_Box (center_x, center_y, extent_x, extent_y)	Gets the terrain region bits found in the given box.
Get_Num_Terrain_Regions ()	Gets the number of terrain regions.
Get_Terrain_Region_Bit_Index (region_index)	Gets the bit index of the terrain region with the given region index.
Get_Terrain_Region_Name (number)	Gets the terrain region name of the given terrain region specified by bit index.

Functions

Get_Terrain_Region_Bits_At_Location (x, y)

Gets the terrain region bits at the location.

Parameters:

- x : number of x coordinate
- y : number of y coordinate

Returns:

number : this will be a bit mask with each set bit corresponding to a terrain region being present.

Get_Terrain_Region_Bits_At_Location_Table (x, y)

Gets the terrain region bits at the location and returns the bit indices as entries in a table.

Parameters:

- x : number of x coordinate
- y : number of y coordinate

Returns:

table of numbers, each entry is a terrain region bit index.

Get_Terrain_Region_Bits_In_Box (center_x, center_y, extent_x, extent_y)

Gets the terrain region bits found in the given box.

Parameters:

- center_x : number box center's x coordinate
- center_y : number box center's y coordinate
- extent_x : number length of box along x axis
- extent_y : number length of box along y axis

Returns:

number : this will be a bit mask with each set bit corresponding to a terrain region being present.

Get_Num_Terrain_Regions ()

Gets the number of terrain regions. One can iterate from 0 to the value returned here and pass in to other functions taking a region index.

Returns:

number

Get_Terrain_Region_Bit_Index (region_index)

Gets the bit index of the terrain region with the given region index.

Parameters:

- region_index : number

Returns:

number of bit index of the given region

Get_Terrain_Region_Name (number)

Gets the terrain region name of the given terrain region specified by bit index.

Parameters:

- number : bit index

Returns:

string of terrain region name, or nil

UnitOrders

Defines some client/server accessible methods for dealing with unit orders.

Functions

Create_Unit_Order ()	Creates a new CUnitOrder.
Reset_Unit_Order ()	Resets a CUnitOrder to have its default values.
Get_Order_Type_From_String (order_string)	Returns an order value matching the given order string.

Order Strings

Orders	Order string values
---------------	---------------------

Functions

Create_Unit_Order ()

Creates a new CUnitOrder.

Returns:

CUnitOrder

Reset_Unit_Order ()

Resets a CUnitOrder to have its default values.

Returns:

nil

Get_Order_Type_From_String (order_string)

Returns an order value matching the given order string.

Parameters:

- order_string : The string value of the order to find the number value of (check UnitOrders.h for the latest values)

Returns:

number (may return -1 to represent an invalid order)

Order Strings

Orders

Order string values

Fields:

- Invalid
- MoveToLocation
- Attack
- AttackMove

- ExpiredAttackCloseDistance
- AircraftRefuel
- Rollout
- MovingToAttach
- Patrol
- Guard
- MovingToBoardTransport
- AggressiveIdle
- HoldPosition
- Stop
- GuardIdle
- MovingToGetConsumed
- MovingToDeploy
- TransportMovingToLoadObject
- UnloadTransport
- DeployingExtractor
- CollectResources
- DeliverResources
- MoveToRendezvous
- TransportMovingToUnloadObjects
- TeleportToLocation
- JumpTerrain
- AttackAndClearObstacles
- DetachObjects
- WaveUnitMovingUnseen
- WaveUnitGuard
- WaveUnitGuardIdle
- WaveUnitAttack
- WaveUnitPreAttackWait
- WaveUnitAttackMove
- CastAbility

PGServer

Various server-only functions that may be moved to other modules eventually.

Functions

Force_Team_Win (team_id, victory_scale)	Causes the passed in team ID to win.
Force_Team_Loss (team_id, defeat_scale)	Causes the passed in team ID to lose.
On_Strategic_Victory ()	Initiates end of campaign victory
Request_Cease_Fire (player)	Initiates a cease fire on behalf of the player passed into the function.

Functions

Force_Team_Win (team_id, victory_scale)

Causes the passed in team ID to win.

Parameters:

- team_id : number of team ID
- victory_scale : number representing the scale of victory (default: 0)

Returns:

nil

Force_Team_Loss (team_id, defeat_scale)

Causes the passed in team ID to lose.

Parameters:

- team_id : number of team ID
- defeat_scale : number representing the scale of defeat (default: 0)

Returns:

nil

On_Strategic_Victory ()

Initiates end of campaign victory

Returns:

nil

Request_Cease_Fire (player)

Initiates a cease fire on behalf of the player passed into the function.

Parameters:

- player : Object Ref of the player that initiates the cease fire.

Returns:

bool true if successfully started a cease fire.

Timers

Timer functionality: execute functions after a delay.

Functions

Stop_Timer (timer_ref)	Stops a timer from completing.
Start_Timer (timer_ref, recurring, timeout, func, data)	Starts a timer.
Create_Timer ()	Creates a new CTimerRef.
Destroy_Timer (timer_ref)	Destroys a CTimerRef.
Get_Timer_Remaining_Time (timer_ref)	Gets the remaining seconds on the input timer.
Get_Timer_Elapsed_Time (timer_ref)	Gets the time the timer has been actively running.
Pause_Timer (timer_ref)	Pauses the timer, preventing it from timing out and executing its registered function.
Resume_Timer (timer_ref)	Resumes a timer that was paused via Pause_Timer .
Is_Timer_Paused (timer_ref)	Returns whether or not the timer is paused.
Is_Timer_Active (timer_ref)	Returns whether the timer is actively counting down (has not already timed out and is not paused).
Timeout_Timer_Now (timer_ref)	Times out the timer and causes its registered function to execute upon the next timer update.

Functions

Stop_Timer (timer_ref)

Stops a timer from completing. Does not execute the function registered to the timer.

Parameters:

- timer_ref : CTimerRef

Returns:

nil

Start_Timer (timer_ref, recurring, timeout, func, data)

Starts a timer. Optionally registers additional data that the timer can use on timeout.

Parameters:

- timer_ref : CTimerRef
- recurring : If true, the timer will repeat every "timeout" seconds.
- timeout : The time, in seconds, to wait before executing the registered function.
- func : The function to run upon timeout.
- data : optional Lua data that will be passed to the registered function.

Returns:

nil

Usage:

```
function A_Timeout_Function(data, timer_ref) -- the passed in timer_ref can be used to stop the timer if desired
    print("Another 3 seconds have passed! Also, the passed in data was ", data.some_data)
end
```

```
local timer = Create_Timer()
Start_Timer(timer, true, 3, A_Timeout_Function, { some_data = 43 })
-- prints "Another 3 seconds have passed! Also, the passed in data was 43" every three seconds
```

Create_Timer ()

Creates a new CTimerRef.

Returns:

CTimerRef

Destroy_Timer (timer_ref)

Destroys a CTimerRef. Stops it from executing upon timeout. It is not necessary to explicitly call this function, as it will be cleaned up by the garbage collector when the variable becomes inaccessible.

Parameters:

- timer_ref : CTimerRef

Returns:

nil

Get_Timer_Remaining_Time (timer_ref)

Gets the remaining seconds on the input timer.

Parameters:

- timer_ref : CTimerRef

Returns:

number, or nil

Get_Timer_Elapsed_Time (timer_ref)

Gets the time the timer has been actively running.

Parameters:

- timer_ref : CTimerRef

Returns:

nil

Pause_Timer (timer_ref)

Pauses the timer, preventing it from timing out and executing its registered function. Can be resumed via **Resume_Timer**.

Parameters:

- timer_ref : CTimerRef

Returns:

nil

Resume_Timer (timer_ref)

Resumes a timer that was paused via **Pause_Timer**.

Parameters:

- timer_ref : CTimerRef

Returns:

nil

Is_Timer_Paused (timer_ref)

Returns whether or not the timer is paused.

Parameters:

- timer_ref : CTimerRef

Returns:

nil

Is_Timer_Active (timer_ref)

Returns whether the timer is actively counting down (has not already timed out and is not paused).

Parameters:

- timer_ref : CTimerRef

Returns:

nil

Timeout_Timer_Now (timer_ref)

Times out the timer and causes its registered function to execute upon the next timer update.

Parameters:

- timer_ref : CTimerRef

Returns:

nil

GameSignals

A module allowing functions to be registered to various game events.

Functions

Register_Global_Signal (signal_type, handler_func, user_data, unregister_func, existing_handler)	Registers a function to a global signal.
Unregister_Global_Signal (signal_registration_data)	Removes a registered signal.
Register_Object_Signal (object_ref, signal_type, handler_func, user_data, existing_handler)	Registers a function to a signal of an object.
Unregister_Object_Signal (signal_registration_data)	Removes a registered object signal.

Types of signals

ObjectKilled	Occurs when an object is killed.
ObjectMoraleBroken	Occurs when an object has their morale broken.
ObjectHealthChanged	Occurs when an object's health is changed.
ObjectMoraleChanged	Occurs when an object's morale is changed.
TriggerEntered	Occurs when an object enters a trigger box.
TriggerExited	Occurs when an object exits a trigger box.
ObjectStateChanged	Occurs when an object's state changes.
ObjectAttributeChanged	Occurs when an object's attribute changes.
ObjectSelected	Occurs when an object is selected by a player.
ObjectUnselected	Occurs when an object is unselected by a player.
ObjectDeleted	Occurs when an object is deleted.
ObjectEnteredTerrainRegion	Occurs when an object enters a terrain region.
ObjectExitedTerrainRegion	Occurs when an object exits a terrain region.
CinematicStarted	Occurs when cinematic starts (restricted to the client)
CinematicFinished	Occurs when cinematic finishes (client only)
CinematicFrame	Occurs when cinematic reaches a registered frame (client only)
CinematicSkipped	Occurs when cinematic is requested to be skipped by the user (client only)
BarkFinished	Occurs when a bark finishes playing (client only)
HarvesterArrivedAtRefinery	Occurs when a harvester returns to a refinery and dumps its resources
StructureConstructionComplete	Occurs when a structure completes construction
StructureConstructionStart	Occurs when a structure starts constructing
StructureReplaced	Occurs when a structure is replaced (this occurs when a structure is upgraded, for example)
StructureSold	Occurs when a structure is sold
UnitProductionStarted	Occurs when a structure begins producing a unit
UnitProductionCompleted	Occurs when a structure finishes producing a unit
UnitProductionCanceled	Occurs when a structure cancels producing a unit
UnitProductionQueued	Occurs when a structure queues a unit for production
UnitProductionRequeueToggled	Occurs when a structure has its requeue status toggled for a producing or queued unit
UnitSpawned	Occurs when an object is spawned by a spawner (server only)

GUIEvent	Occurs when a GUI event that is being listened for fires (client only)
GamePhaseStarted	Occurs when a game phase is entered.
CompanyEnteredCombat	Occurs when a company enters combat.
CompanyExitedCombat	Occurs when a company exits combat.
ObjectCaptureEvent	Occurs when a capture point changes state.
ServerObjectCreationEvent	Occurs when a server object creation event is received on the server(re-inforcements)
ServerAITurnStart	Occurs when a server AI player's turn is starting
ServerAIClearedToStart	Occurs when a server AI player can run its script
ServerAIHeatMapFinished	Occurs when AI's heat map is finished being created
ServerAIStrategicAttackFinished	Occurs when AI returns to strategic from tactical
ServerAIStrategicEspionagePopupFinished	Occurs when espionage popup is closed
ServerStrategicGameRoundAdvanced	Occurs when a server strategic game round has advanced.
ServerStrategicRegionCaptured	Occurs when a server strategic region has been captured.
ServerStrategicRegionDefended	Occurs when a server strategic region has been defended.
ServerStrategicTechResearched	Occurs when a server strategic tech has been researched.
ServerStrategicObjectsMoved	Occurs when a server strategic object moves from one region to another.
ServerStrategicObjectsBuilt	Occurs when a server strategic object is built at a given region.
StartingMatch	Occurs when a match start event is triggered.
ServerTacticalStrategicScriptEvent	Occurs when a server tactical strategic script event has been triggered.
ServerStrategicTacticalObjectsBuilt	Occurs after a tactical battle in strategic, listing the tactical objects that were built at a given region.
ServerStrategicTacticalObjectsDestroyed	Occurs after a tactical battle in strategic, listing the tactical objects that were destroyed at a given region.
ServerStrategicTacticalObjectsCaptured	Occurs after a tactical battle in strategic, listing the tactical objects that were captured at a given region.
ServerStrategicNationalWillUpdated	Occurs when a faction's national will has changed.
ServerStrategicEventCompleted	Occurs after the completion of a strategic event.
ServerStrategicEventCompletedSuccess	Occurs after the completion of a strategic event (success case).
ServerStrategicEventCompletedExpired	Occurs after the completion of a strategic event (expired case).
ServerStrategicEventCompletedFailed	Occurs after the completion of a strategic event (failed case).
ServerStrategicEventAdded	Occurs after the addition of a strategic event.
CompanyReceivedOrder	Occurs when a company is issued a new order.
CompanyObjectCreated	Occurs when a company object is created.
TrenchObjectCreated	Occurs when a trench object is created.
ObjectItemAbilityActivated	Occurs when an object uses an item ability.
StrategicRegionSelected	Occurs when a region is selected in strategic.
StrategicPawnSelected	Occurs when a pawn is selected in strategic.
ClientStrategicEventPoppedUp	Occurs when a strategic event dialog pops up (client only)
ClientStrategicEventClosed	Occurs when a strategic event dialog is closed (client only)
BarkStarted	Occurs when a bark starts playing (client only)
ServerStrategicNationalWillVictoryDeclined	Occurs when player declines a National Will victory.
ServerStrategicCorpsDestroyed	Occurs when a corps has been removed on the strategic map
ServerStrategicItemPurchased	Occurs when player makes a purchase in strategic mode, currently only triggered by luxury goods

ClientNationalWillEvent	Occurs when national will UI in client strategic changes.
StrategicTransferCorpsChanged	Occurs when the contents of the Corps transfer window changes.
ServerStrategicVictoryDetected	Occurs when a strategic victory has been detected
StrategicMovieEndDetected	Occurs when a strategic movie ends

Functions

Register_Global_Signal (signal_type, handler_func, user_data, unregister_func, existing_handler)

Registers a function to a global signal.

Parameters:

- signal_type : the string identifier for the signal type.
- handler_func : the function to execute.
- user_data : optional user data to pass into the handler_func when it executes.
- unregister_func : optional function to call when the signal is unregistered.
- existing_handler : library reserved parameter. Do not use!

Returns:

LGameSignalRegistration

Usage:

```
function Object_Killed_Handler(data, user_data, signal_ref)
    local name = Get_Object_Name(data.KilledObject)

    if name then
        print(string.format("object %s was killed!", name))
    else
        print("an object was killed!")
    end

    -- user_data contains the optional passed in data
    -- user_data.cool_data == 777
    -- you can unregister the signal using the signal_ref
    Unregister_Global_Signal(signal_ref)
end

Register_Global_Signal("ObjectKilled", Object_Killed_Handler)
```

Unregister_Global_Signal (signal_registration_data)

Removes a registered signal. The associated function will no longer execute on the given signal.

Parameters:

- signal_registration_data : LGameSignalRegistration of the signal

Returns:

nil

Register_Object_Signal (object_ref, signal_type, handler_func, user_data, existing_handler)

Registers a function to a signal of an object.

Parameters:

- `object_ref` : CObjectRef of object to register signal to
- `signal_type` : the string identifier for the signal type.
- `handler_func` : the function to execute.
- `user_data` : optional user data to pass into the `handler_func` when it executes.
- `existing_handler` : library reserved parameter. Do not use!

Returns:

LGameSignalRegistration

Unregister_Object_Signal (signal_registration_data)

Removes a registered object signal. The associated function will no longer execute on the given signal.

Parameters:

- `signal_registration_data` : LGameSignalRegistration of the signal

Returns:

nil

Types of signals

ObjectKilled

Occurs when an object is killed.

Fields:

- `KilledObject` : The killed object
- `Attacker` : The attacking object, if applicable (may be nil)
- `Damage` : The amount of damage that was done

ObjectMoraleBroken

Occurs when an object has their morale broken.

Fields:

- `MoraleBrokenObject` : The broken object
- `Attacker` : The attacking object, if applicable (may be nil)
- `Damage` : The amount of damage that was done

ObjectHealthChanged

Occurs when an object's health is changed.

Fields:

- `Object` : The object whose health was changed
- `OldHealth` : The health before the change
- `NewHealth` : The health after the change
- `Attacker` : The attacking object, if applicable (may be nil)

ObjectMoraleChanged

Occurs when an object's morale is changed.

Fields:

- `Object` : The object whose morale was changed
- `OldMorale` : The morale before the change

- NewMorale : The morale after the change
- Attacker : The attacking object, if applicable (may be nil)

TriggerEntered

Occurs when an object enters a trigger box.

Fields:

- Trigger : The trigger that was entered
- Object : The object that entered the trigger

TriggerExited

Occurs when an object exits a trigger box.

Fields:

- Trigger : The trigger that was exited
- Object : The object that left the trigger

ObjectStateChanged

Occurs when an object's state changes.

Fields:

- Object : The object whose state changed
- State : The state (as a string) that changed
- ChangeType : The change type (as a string); possible values are "Added", "Removed", "Incremented", and "Decrement"

ObjectAttributeChanged

Occurs when an object's attribute changes.

Fields:

- Object : The object whose attribute changed
- Attribute : The attribute (as a string) that changed
- Value : The new value of the attribute(number)

ObjectSelected

Occurs when an object is selected by a player.

Fields:

- Object : The object that became selected
- Player : The player that selected the object
- IsSelected : A boolean with the value true

ObjectUnselected

Occurs when an object is unselected by a player.

Fields:

- Object : The object that became unselected
- Player : The player that unselected the object
- IsSelected : A boolean with the value false

ObjectDeleted

Occurs when an object is deleted. At this point, API calls requiring a CObjectRef will no longer work.

Fields:

- ObjectID : The object ID of the object that was deleted
- Object : CObjectRef of the deleted object; note that this value can no longer be used in API calls requiring a CObjectRef

ObjectEnteredTerrainRegion

Occurs when an object enters a terrain region.

Fields:

- Object : The object that entered the terrain region
- BitIndex : The bit index of the terrain region

ObjectExitedTerrainRegion

Occurs when an object exits a terrain region.

Fields:

- Object : The object that exited the terrain region
- BitIndex : The bit index of the terrain region

CinematicStarted

Occurs when cinematic starts (restricted to the client)

Fields:

- CinematicFile : string of the cinematic file path

CinematicFinished

Occurs when cinematic finishes (client only)

Fields:

- CinematicFile : string of the cinematic file path

CinematicFrame

Occurs when a cinematic reaches a registered frame (client only)

Fields:

- CinematicFile : string of the cinematic file path
- Frame : number of the frame reached

CinematicSkipped

Occurs when a cinematic is requested to be skipped by the user (client only)

Fields:

- CinematicFile : string of the cinematic file path

BarkFinished

Occurs when a bark finishes playing (client only)

Fields:

- BarkName : string of the bark name

HarvesterArrivedAtRefinery

Occurs when a harvester returns to a refinery and dumps its resources.

Fields:

- ResourceNodeObjectID : number for the object ID of the resource node
- ResourceFieldID : number for the (not an object) field's ID
- Resources : number for amount of resources the harvester is dropping off

StructureConstructionComplete

Occurs when a structure completes construction.

Fields:

- Object : CObjectRef of the structure that completed construction

StructureConstructionStart

Occurs when a structure starts constructing.

Fields:

- Object : CObjectRef of the structure that started construction

StructureReplaced

Occurs when a structure is replaced (this occurs when a structure is upgraded, for example)

Fields:

- OldStructureID : number of the replaced structure's ID
- NewStructureID : number of the new structure's ID

StructureSold

Occurs when a structure is sold.

Fields:

- ObjectID : number of the sold structure's ID

UnitProductionStarted

Occurs when a structure begins producing a unit.

Fields:

- StructureObject : CObjectRef of the structure producing the unit
- ObjectTypeString : string of the unit type
- Object : CObjectRef of the producing unit

UnitProductionCompleted

Occurs when a structure finishes producing a unit.

Fields:

- StructureObject : CObjectRef of the structure producing the unit
- ObjectTypeString : string of the unit type
- Object : CObjectRef of the producing unit

UnitProductionCanceled

Occurs when a structure cancels producing a unit.

Fields:

- StructureObject : CObjectRef of the structure producing the unit
- ObjectTypeString : string of the unit type

UnitProductionQueued

Occurs when a structure queues a unit for production.

Fields:

- StructureObject : CObjectRef of the structure producing the unit
- ObjectTypeString : string of the unit type
- WillRequeue : boolean of whether the queued item will requeue itself upon completion

UnitProductionRequeueToggled

Occurs when a structure has its requeue status toggled for a producing or queued unit.

Fields:

- StructureObject : CObjectRef of the structure producing the unit
- ObjectTypeString : string of the unit type
- WillRequeue : boolean of whether the queued item will requeue itself upon completion

UnitSpawned

Occurs when an object is spawned by a spawner (server only)

Fields:

- SpawnerObject : CObjectRef of the spawner
- SpawnedObject : CObjectRef of the spawned object

GUIEvent

Occurs when a GUI event that is being listened for fires (client only)

Fields:

- EventType : number of the event type's numeric enum value
- EventSource : string of the identifying key of the component that fired the event

GamePhaseStarted

Occurs when a game phase is entered.

Fields:

- PhaseName : string of the name of the game phase

CompanyEnteredCombat

Occurs when a company enters combat.

Fields:

- EnemyCompany : The company we entered combat with

CompanyExitedCombat

Occurs when a company exits combat.

Fields:

- EnemyCompany : The company we exited combat from

ObjectCaptureEvent

Occurs when a capture point changes state.

Fields:

- CaptureEventType : The type of state change event. (Attacking, Reinforcing, Captured)
- ObjectID : Object ID of the Capture point object.
- Object : Object of the Capture point object.
- CaptureProgress : Current progress value for the state of capture.
- CaptureValue : Maximum value when the point is captured.
- TeamID : The teamid that's currently attacking or reinforcing the capture point.
- OwningTeamID : The teamid that currently owns the control point.

ServerObjectCreationEvent

Occurs when a server object creation event is received on the server(re-inforcements)

Fields:

- ObjectID : Object ID of the newly created object.
- Object : The newly created object.

ServerAITurnStart

Occurs when a server AI player's turn is starting.

ServerAIClearedToStart

Occurs when a server AI player can run its script.

ServerAIHeatMapFinished

Occurs when AI's heat map is finished being created.

ServerAIStrategicAttackFinished

Occurs when AI returns to strategic from tactical.

Fields:

- ObjectID : Object ID of the AI object.
- Object : The AI player object.

ServerAIStrategicEspionagePopupFinished

Occurs when espionage popup is closed.

Fields:

- ObjectID : Object ID of the AI object.
- Object : The AI player object.

ServerStrategicGameRoundAdvanced

Occurs when a server strategic game round has advanced.

Fields:

- Round : New game round.

ServerStrategicRegionCaptured

Occurs when a server strategic region has been captured.

Fields:

- Region : Name of the region captured.

ServerStrategicRegionDefended

Occurs when a server strategic region has been defended.

Fields:

- Region : Name of the region defended.

ServerStrategicTechResearched

Occurs when a server strategic tech has been researched.

Fields:

- Tech : Name of the tech that was researched.

ServerStrategicObjectsMoved

Occurs when a server strategic objects move from one region to another.

Fields:

- ObjectType : Type Name of the object that was moved.
- Source : Source region name the objects were moved from.
- Dest : Destination region name the objects were moved to.
- Count : The number of ObjectType objects that were moved.

ServerStrategicObjectsBuilt

Occurs when a server strategic object is built at a given region.

Fields:

- ObjectType : Type Name of the object that was built.
- Source : Source region name the objects were built at.
- Dest : Destination region name the objects were built at.
- Count : The number of ObjectType objects that were built, currently always 1.

StartingMatch

Occurs when a match start event is triggered.

ServerTacticalStrategicScriptEvent

Occurs when a server tactical strategic script event has been triggered.

Fields:

- StrategicEvent : Name of the Strategic Event type.
- ScriptString : String associated with the button on the UI for the event.

ServerStrategicTacticalObjectsBuilt

Occurs after a tactical battle in strategic, listing the tactical objects that were built at a given region.

Fields:

- ObjectType : Type Name of the object that was built.
- Source : Source region name the objects were built at.
- Dest : Destination region name the objects were built at.
- Count : The number of ObjectType objects that were built

ServerStrategicTacticalObjectsDestroyed

Occurs after a tactical battle in strategic, listing the tactical objects that were destroyed at a given region.

Fields:

- ObjectType : Type Name of the object that was destroyed.
- Source : Source region name the objects were destroyed at.
- Dest : Destination region name the objects were destroyed at.
- Count : The number of ObjectType objects that were destroyed

ServerStrategicTacticalObjectsCaptured

Occurs after a tactical battle in strategic, listing the tactical objects that were captured at a given region.

Fields:

- ObjectType : Type Name of the object that was captured.
- Source : Source region name the objects were captured at.
- Dest : Destination region name the objects were captured at.
- Count : The number of ObjectType objects that were captured

ServerStrategicNationalWillUpdated

Occurs when a faction's national will has changed.

Fields:

- FactionType
- National Will

ServerStrategicEventCompleted

Occurs after the completion of a strategic event.

Fields:

- EventObjectType : Event Object Type Name.
- RegionObjectType : Region Object Type Name.
- CompletionContext : Completion Context Name (Invalid, None, Success, Expired, Failed).
- StartingTurn : Turn number the event started on.
- PlayerChoiceIndex : Index of the player choice.

ServerStrategicEventCompletedSuccess

Occurs after the completion of a strategic event (success case).

Fields:

- EventObjectType : Event Object Type Name.
- RegionObjectType : Region Object Type Name.
- CompletionContext : Completion Context Name (Invalid, None, Success, Expired, Failed).
- StartingTurn : Turn number the event started on.
- PlayerChoiceIndex : Index of the player choice.

ServerStrategicEventCompletedExpired

Occurs after the completion of a strategic event (expired case).

Fields:

- EventObjectType : Event Object Type Name.
- RegionObjectType : Region Object Type Name.
- CompletionContext : Completion Context Name (Invalid, None, Success, Expired, Failed).
- StartingTurn : Turn number the event started on.
- PlayerChoiceIndex : Index of the player choice.

ServerStrategicEventCompletedFailed

Occurs after the completion of a strategic event (failed case).

Fields:

- EventObjectType : Event Object Type Name.
- RegionObjectType : Region Object Type Name.
- CompletionContext : Completion Context Name (Invalid, None, Success, Expired, Failed).
- StartingTurn : Turn number the event started on.
- PlayerChoiceIndex : Index of the player choice.

ServerStrategicEventAdded

Occurs after the addition of a strategic event.

Fields:

- EventObjectType : Event Object Type Name.
- RegionObjectType : Region Object Type Name.
- CompletionContext : Completion Context Name (Invalid, None, Success, Expired, Failed).
- StartingTurn : Turn number the event started on.
- PlayerChoiceIndex : Index of the player choice.

CompanyReceivedOrder

Occurs when a company is issued a new order.

Fields:

- Company : CObjectRef of company receiving order
- Order : string of the order type received
- TargetObject : CObjectRef of the target object; may be nil if there is no target object
- TargetX : number of X coordinate of target
- TargetY : number of Y coordinate of target

CompanyObjectCreated

Occurs when a company object is created.

Fields:

- Object : CObjectRef of company created

TrenchObjectCreated

Occurs when a trench object is created.

Fields:

- Object : CObjectRef of trench created

ObjectItemAbilityActivated

Occurs when an object uses an item ability.

Fields:

- AbilityName : string of the ability used

StrategicRegionSelected

Occurs when a region is selected in strategic.

Fields:

- Object : CObjectRef of region object selected

StrategicPawnSelected

Occurs when a pawn is selected in strategic.

Fields:

- PawnType : string of the pawn type that was selected.

ClientStrategicEventPoppedUp

Occurs when a strategic event dialog pops up (client only)

Fields:

- EventObjectType : string of the event object type name

ClientStrategicEventClosed

Occurs when a strategic event dialog is closed (client only)

Fields:

- EventObjectType : string of the event object type name

BarkStarted

Occurs when a bark starts playing (client only)

Fields:

- BarkName : string of the bark name

ServerStrategicNationalWillVictoryDeclined

Occurs when player declines a National Will victory.

ServerStrategicCorpsDestroyed

Occurs when a corps has been removed on the strategic map

Fields:

- FactionType
- CorpsType
- CorpsCount

ServerStrategicItemPurchased

Occurs when player makes a purchase in strategic mode, currently only triggered by luxury goods.

Fields:

- ItemType.

ClientNationalWillEvent

Occurs when national will UI in client strategic changes.

Fields:

- UpOrDown : boolean, true if ticking up, false if down.
- LeftOrRight : boolean, true if left UI is changing, false if right.
- EndOrBegin : boolean, true if this is a begin event, false if end event.

StrategicTransferCorpsChanged

Occurs when the contents of the Corps transfer window changes.

Fields:

- Corps : table containing an array of tables, each table contains CorpsType (string), TotalCount(number), CooldownCount(number)

ServerStrategicVictoryDetected

Occurs when a strategic victory has been detected.

StrategicMovieEndDetected

Occurs when a strategic movie ends.

Fields:

- MovieName

StateSignals

State signal listening logic built upon the lower-level game signals module.

Functions

Register_On_State_Changed_Signal (<i>object_ref</i> , <i>state_string</i> , <i>handler_func</i> , <i>user_data</i>)	Registers a function to run when the given state is added or removed from the given object.
Unregister_On_State_Changed_Signal (<i>signal_registration_data_ref</i>)	Removes an existing state signal handler via the passed in LStateSignalRegistration reference.

Functions

Register_On_State_Changed_Signal (*object_ref*, *state_string*, *handler_func*, *user_data*)

Registers a function to run when the given state is added or removed from the given object.

Parameters:

- *object_ref* : CObjectRef of object
- *state_string* : The string representation of the state
- *handler_func* : The function to run when the state is added/removed
- *user_data* : Optional user data that will be passed into the *handler_func* when called

Returns:

LStateSignalRegistration reference, or nil

Usage:

```
function Object_Combat_State_Changed(state, change_type, user_data, signal_ref)
    local object_name = Get_Object_Name(signal_ref.ObjectRef)

    -- change_type could also be either "INCREMENTED" or "DECREMENTED" for reference counted states
    if change_type == "ADDED" then
        print(string.format("Object %s entered combat", object_name))
    elseif change_type == "REMOVED" then
        print(string.format("Object %s left combat", object_name))
    end

    -- prints "userdata == 55"
    print(string.format("userdata == %d", user_data.thedata))
end

-- assuming you already have an object stored in variable "object_ref" ...
-- the state string passed in is in all uppercase
Register_On_State_Changed_Signal(object_ref, "INCOMBAT", Object_Combat_State_Changed, { thedata = 55 })
```

Unregister_On_State_Changed_Signal (*signal_registration_data_ref*)

Removes an existing state signal handler via the passed in LStateSignalRegistration reference.

Parameters:

- *signal_registration_data_ref* : LStateSignalRegistration of signal to unregister

Returns:

nil

EXAMPLES

emptyscript.lua

emptyscript.lua

```
-- emptyscript.lua
-- A basic script one can use as a starting point for implementing new map scripts.

Lazy = PG_Import("Utils.LazyAPI")
Lazy.Copy_API_Into_Table(_G, Lazy)

function SaveLoad_Init()

end

-- This function automatically executes when a map is started and is not a loaded game
function Main_NoLoad()

end

-- Like Main_NoLoad(), only this function runs when it is a loaded game
function Main_Load()

end

-- Runs at map start in both the load and non-load cases
function Main_Common()

end

-- Main() is always called at map start by the script manager. Has a basic implementation to call the separate
-- "Main" functions above.
function Main()
    if not PG.IsLoad then
        Main_NoLoad()
    else
        Main_Load()
    end

    Main_Common()
end
```

rpcplayerselector.lua

rpcplayerselector.lua

```
local pgm_clientobjectivesdisplay = PG_Import("Libs.Client.ClientObjectiveDisplay")

--[[
Calling a client rpc function from the server requires a player selector as the initial argument.
A player selector tells the server which clients to send the function call to.
Valid values for player selectors include:
    - the boolean value true : sends to all players
    - a CObjectRef of a player : sends to just that player
    - a table of CObjectRef of players : sends to every player included in the table

Examples follow:
--]]

-- sends the following RPC calls to all human players
pgm_clientobjectivesdisplay.Add_Objective(true, "destroy_objective", "Destroy 15 structures.")
pgm_clientobjectivesdisplay.Set_Objective_Star_Type(true, "destroy_objective", "Gold")

-- sends the following RPC call to a single human player
local players = Create_ObjectSet()
Get_Players(players)
for player in ObjectSet_Values(players) do
    if Is_Player_Human(player) then
        pgm_clientobjectivesdisplay.Add_Objective(player, "survive_objective", "Survive 10 waves.")
        break
    end
end

-- sends the following RPC call to all human players in the table
local human_players = {}
local potential_players = Create_ObjectSet()
Get_Players(potential_players)
for player in ObjectSet_Values(potential_players) do
    if Is_Player_Human(player) then
        human_players[#human_players + 1] = player
        break
    end
end

pgm_clientobjectivesdisplay.Add_Objective(human_players, "hidden_objective", "Find the secret treasure.")
```