By Dylan Yates

Contents

This asset is designed to ease the pain of GUI creation. It features scripts that automatically scale GUI elements based on the screen resolution and scripts that turn simple GUITextures into interactive elements. This asset is also designed to be a learning tool. Included are scripts and example scenes detailing how to make a main menu, pause menu and an in-game HUD.

Prerequisites

Move the “GUI Essentials/Editor” folder to your “Assets” folder.

Pick a resolution and an aspect ratio that you will be designing your GUI for. If you do not know which resolution to use, I recommend a resolution of 1920 x 1080.

GUI Components

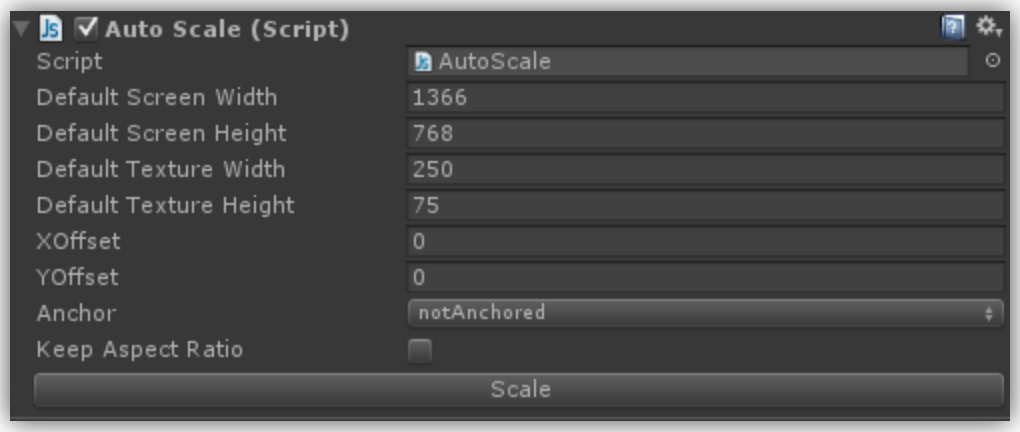
Auto Scale

These Auto Scale scripts are designed to allow you to build your GUI once and have it work on any resolution or aspect ratio. These scripts can be used in combination with any of the other elements in this package.

AutoScale.js – Attach this script to any GameObject with a GUITexture. This script requires a GUITexture. Use this script to automatically scale the GUITexture when the resolution changes. This script executes in edit mode. Click on the game view to see your scaled image without hitting the play button.

Parameters:

- **Default Screen Width** – The standard screen width that you are designing your GUI for.
- **Default Screen Height** – The standard screen height that you are designing your GUI for.
- **Default Texture Width** – The default width of the texture that you are using.
- **Default Texture Height** – The default height of the texture that you are using.
- **X Offset** – If you are anchoring this GUI element this is the offset in the X direction of the anchor in screen space. Must be between -1 and 1.
- **Y Offset** – If you are anchoring this GUI element this is the offset in the Y direction of the anchor in screen space. Must be between -1 and 1.
- **Anchor** – This anchors your GUI element to a specific part of the screen. Use the X offset and Y offset to fine tune the positioning. If you select “Not Anchored”, use the Transform component to position the GUI element. If you decide to anchor the GUI element, you must use the X offset and Y offset to position the GUI element.
- **Keep Aspect Ratio** – Boolean value that determines whether or not this GUITexture should maintain its aspect ratio. Aspect ratio is calculated with the Default Screen Width and Default Screen Height variables*.
- **Scale Button** – Click the scale button to see your changes updated in real time in the game view.



AutoScaleGUIText.js – Attach this script to any GameObject with a GUIText. This script requires a GUIText. Use this script to automatically scale the GUIText when the resolution changes. This script uses two different scaling techniques. The first is for GUIText components that are designed to stretch and shrink when the aspect ratio changes. In this case, uncheck the "Pixel Correct" check box on the GUIText component and use the scale values to resize your text. I also recommend a large font size to make the text look sharper. The second scaling technique is for GUIText components that are going to maintain aspect ratio and position. Check the "Pixel Correct" check box on the GUIText component and resize the text with the default font size parameter.

Parameters:

- **Default Screen Width** – The standard screen width that you are designing your GUI for.
- **Default Screen Height** – The standard screen height that you are designing your GUI for.
- **Default Scale X** – The default x-axis scaling of the GUIText. This is only used when you are not maintaining aspect ratio.
- **Default Scale Y** – The default y-axis scaling of the GUIText. This is only used when you are not maintaining aspect ratio.
- **Default Font Size** – The original font size before the GUI is scaled.
- **X Offset** – If you are anchoring this GUI element this is the offset in the X direction of the anchor in screen space. Must be between -1 and 1.
- **Y Offset** – If you are anchoring this GUI element this is the offset in the Y direction of the anchor in screen space. Must be between -1 and 1.
- **Anchor** – This anchors your GUI element to a specific part of the screen. Use the X offset and Y offset to fine tune the positioning. If you select "Not Anchored", use the Transform component to position the GUI element. If you decide to anchor the GUI element, you must use the X offset and Y offset to position the GUI element.
- **Keep Aspect Ratio** – Boolean value determines whether or not this GUIText should maintain aspect ratio. Aspect ratio is calculated with the Default Screen Width and Default Screen Height variables*. Use pixel correct text when maintaining aspect ratio.
- **Is Text Box** – A Boolean value that should be checked when using this GUIText with a text box or text area. This prevents the pixel offset values from changing after modifying the text in the text box.

***Keep Aspect Ratio** - When you tick the "Keep Aspect Ratio" checkbox, the positioning of the GUITexture or GUIText is determined by the aspect ratio screen resolution instead of the actual screen resolution. Let's say your game wants to maintain the 4:3 aspect ratio, but the device that it is running on has a resolution of 1024 x 600. Keeping the aspect ratio will scale the GUI to use a resolution of 800 x 600, which is the largest 4:3 aspect ratio of a screen that size. Since the aspect ratio screen resolution does not fill the whole screen and you get the "black bars" on the sides, the GUITextures and GUIText must be shifted by the offset of the difference in the actual screen resolution and the aspect ratio screen resolution. This allows the GUI elements to stay in the right position inside the bounds of the aspect ratio screen resolution. Here's an image to visualize what I just said:

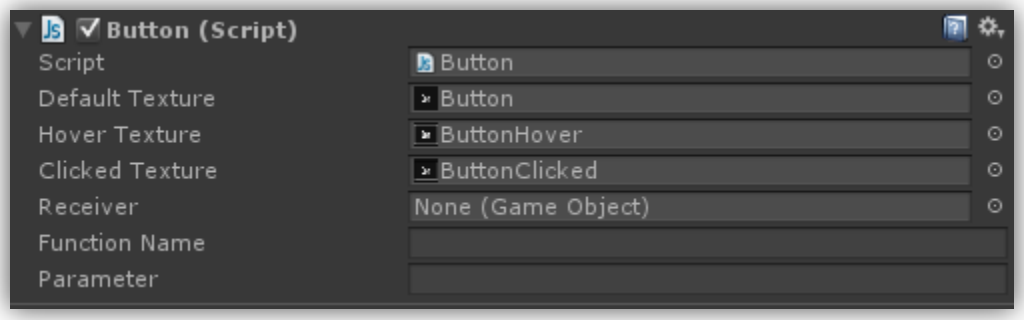
Interactive Elements

All of the interactive elements have the ability to send information to another GameObject. Let's say you press a button and you want it to do something. Create a script (C# or JS) and define a function that you want to be called when the button is pressed. Attach this script to a GameObject in your scene and then set your new GameObject as the "Receiver" variable for any of these interactive elements. Then all you need to do is tell your button what function to call (see "Function Name" variable). Now your button is set up to perform an action. This method of communication uses the GameObject class' SendMessage function to allow the GUI Essentials package to work with C# and JS projects without any headaches.

Button.js – Attach this script to a GameObject with a GUITexture to turn a flat GUITexture into a functioning button.

Parameters:

- **Default Texture** – The default texture for the button. This is what your button will look like when it is at rest.
- **Hover Texture** – The texture for the button when the mouse is over it.
- **Clicked Texture** – The texture for the button when the mouse clicks it.
- **Receiver** – A GameObject with a script that is designed to receive a message.
- **Function Name** – The name of the function that you want to call that is in a script attached to the receiver GameObject.
- **Parameter** – A string to send as a parameter to the function declared above.



RadioButton.js – Attach this script to a GameObject with a GUITexture to turn a flat GUITexture into a functioning radio button.

Parameters:

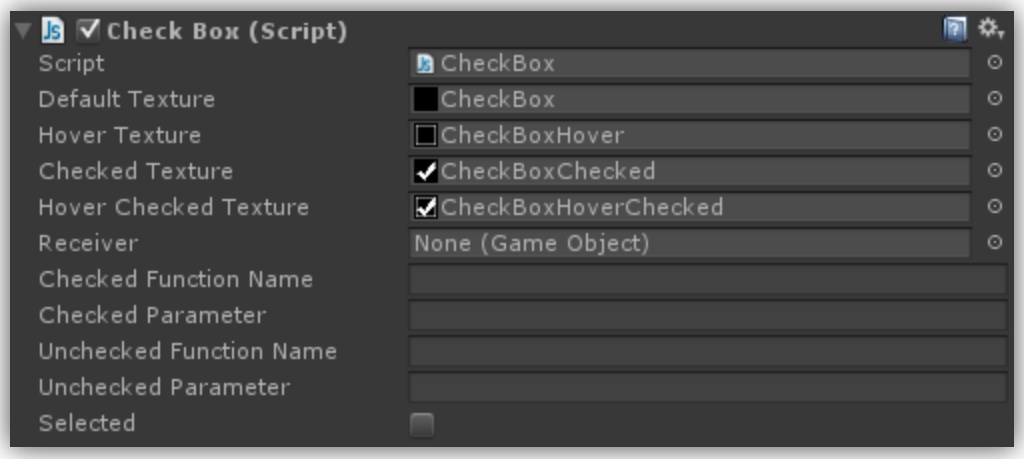
- **Default Texture** – The default texture for the radio button. This is what your radio button will look like when it is at rest.
- **Hover Texture** – The texture for the radio button when the mouse is over it or when the radio button has been selected.
- **Clicked Texture** – The texture for the radio button when the mouse clicks it.
- **Receiver** – A GameObject with a script that is designed to receive a message.
- **Function Name** – The name of the function that you want to call that is in a script attached to the receiver GameObject.
- **Parameter** – A string to send as a parameter to the function declared above.

- **Other Options** – An array of GameObjects containing the RadioButton.js script. These are the other radio buttons in a group of radio buttons. This assures that only one radio button in the group is selected at a time.
- **Selected** – A Boolean value representing whether or not the radio button is selected.

CheckBox.js – Attach this script to a GameObject with a GUITexture to turn a GUITexture into a functioning check box.

Parameters:

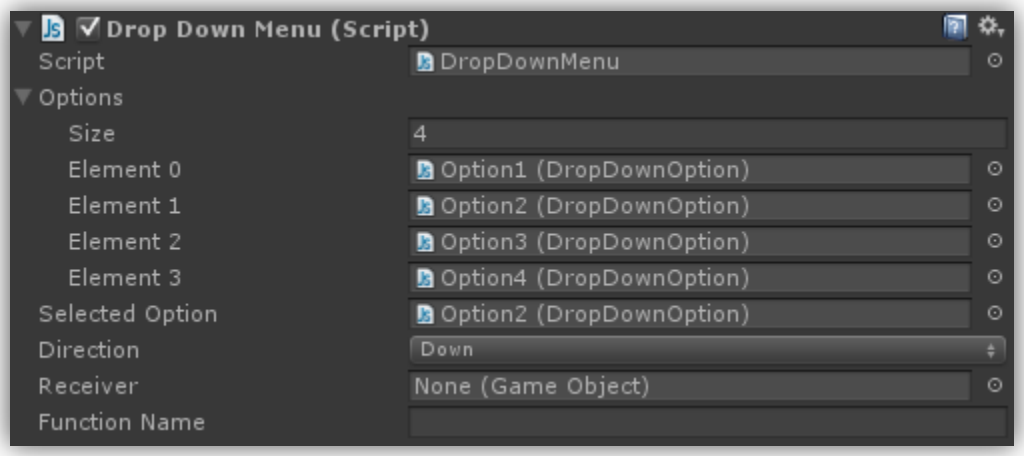
- **Default Texture** – The default texture for the check box. This is what your check box will look like when it is at rest.
- **Hover Texture** – The texture for the check box when the mouse is over it.
- **Checked Texture** – The texture for the check box when it is checked.
- **Hover Checked Texture** – The texture for the check box when it is checked and the mouse is over it.
- **Receiver** – A GameObject with a script that is designed to receive a message.
- **Checked Function Name** – The name of the function that you want to call that is in a script attached to the receiver GameObject. This is called when the check box is checked.
- **Checked Parameter** – A string to send as a parameter to the function declared above.
- **Unchecked Function Name** – The name of the function that you want to call that is in a script attached to the receiver GameObject. This is called when the check box is unchecked.
- **Unchecked Parameter** – A string to send as a parameter to the function declared above.
- **Selected** – A Boolean value showing whether or not the check box is checked.



DropDownMenu.js – Attach this script to a GameObject with a GUITexture to create a drop down menu.

Parameters:

- **Options** – An array of DropDownOptions. These are the different options in the drop down menu. See DropDownOptions.js
- **Selected Option** – The option that is currently selected in the drop down menu.
- **Direction** – The direction that the drop down menu will unfold. Either up or down.
- **Receiver** – A GameObject with a script that is designed to receive a message.
- **Function Name** – The name of the function that you want to call that is in a script attached to the receiver GameObject. This script will also send the "Description" variable of the selected DropDownOption as a parameter.

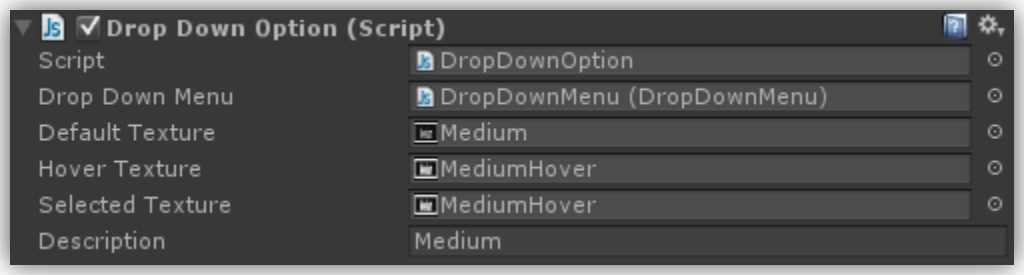


DropDownOption.js – This creates an instance of an option in the drop down menu

Parameters:

- **Drop Down Menu** – The drop down menu that this option is a part of. This GameObject must have the DropDownMenu.js script attached to it.
- **Default Texture** – The default texture for the drop down option. This is what your drop down option will look like when it is at rest.

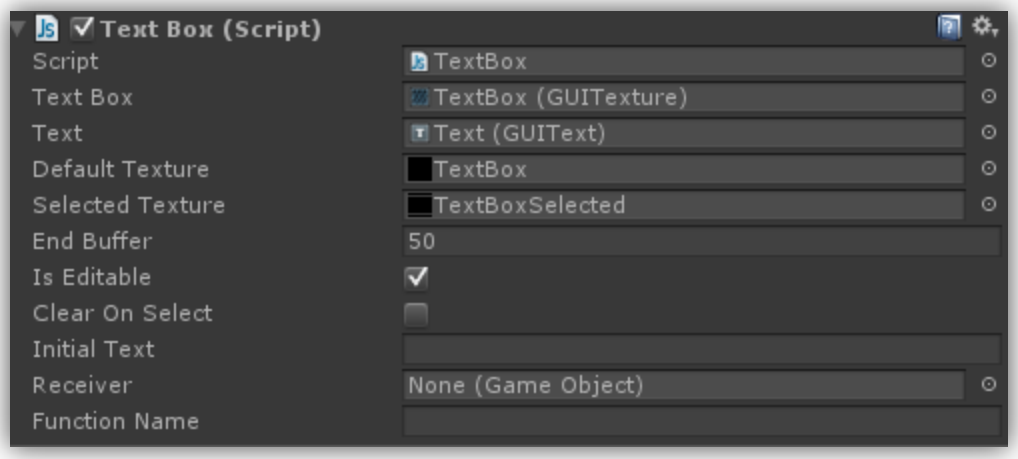
- **Hover Texture** – The texture for the drop down option when the mouse is over it.
- **Selected Texture** – The texture for the drop down option when the mouse clicks it.
- **Description** – A string used to describe this DropDownOption. This string is sent as a parameter to the function specified in the DropDownMenu object when this option is selected.



TextBox.js – This script uses a GUITexture and a GUIText to create a working text box. This script handles keyboard input when the text box is selected and limits the text to the bounds of the GUITexture. Also contains a function called "EnterText(string)" that allows for text to be entered programmatically.

Parameters:

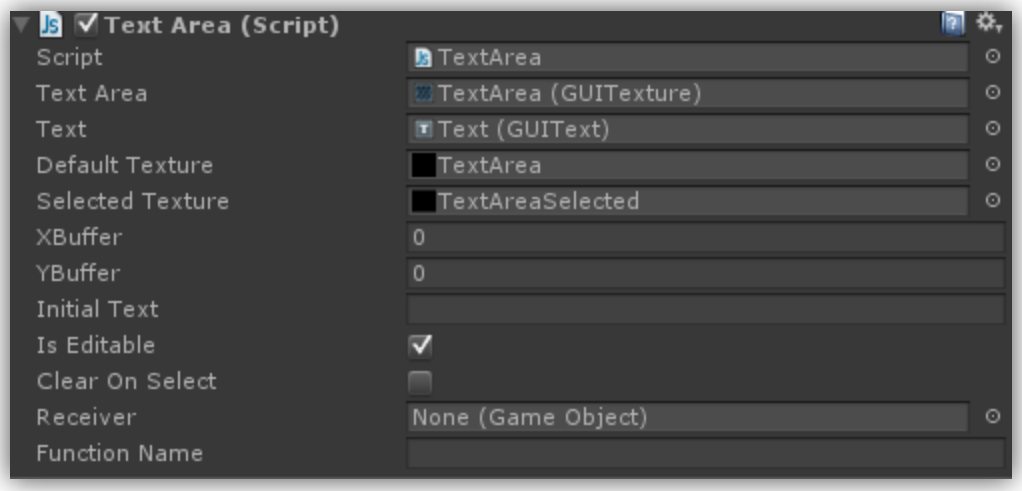
- **Text Box** – The GUITexture for the text box. This script should be attached to that GUITexture.
- **Text** – The GUIText that is displaying the text that the user typed.
- **Default Texture** – The texture used when the text box is inactive.
- **Selected Texture** – The texture used when the user clicks the text box.
- **End Buffer** –This value shifts the text to the right by some amount (in pixels). This is useful when the GUIText and GUITexture are aligned to the left at different points.
- **Is Editable** – Boolean flag for toggling editing the text box.
- **Clear On Select** – Boolean flag for clearing the text box every time it is selected.
- **Initial Text –** Dynamic string that is loaded into the text box at runtime.
- **Receiver** – A GameObject with a script that is designed to receive a message.
- **Function Name** – The name of the function that you want to call that is in a script attached to the receiver GameObject. This script will also send the all of the text in the text box as a parameter. This function is called when the user presses "Enter" after entering text.



TextArea.js – This script uses a GUITexture and a GUIText to create a working text area. This script handles keyboard input when the text area is selected, limits the text to the bounds of the GUITexture and word wraps the text. Also contains a function called "EnterText(string)" that allows for text to be entered programmatically.

Parameters:

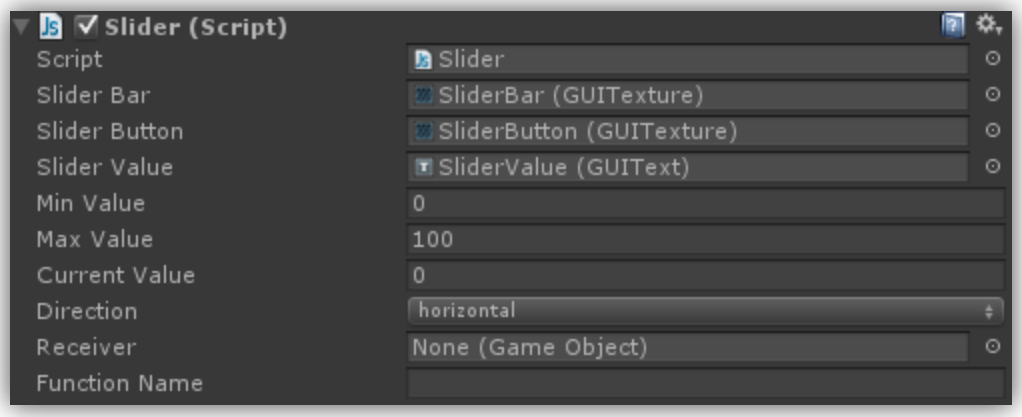
- **Text Area** – The GUITexture for the text area. This script should be attached to that GUITexture.
- **Text** – The GUIText that is displaying the text that the user typed.
- **Default Texture** – The texture used when the text area is inactive.
- **Selected Texture** – The texture used when the user clicks the text area.
- **X Buffer** – Shifts the text by this many pixels inside the text area on the x-axis.
- **Y Buffer** – Shifts the text by this many pixels inside the text area on the y-axis.
- **Initial Text –** Dynamic string that is loaded into the text area at runtime.
- **Is Editable** – Boolean flag for toggling editing the text are.
- **Clear On Select** – Boolean flag for clearing the text area every time it is selected.
- **Receiver** – A GameObject with a script that is designed to receive a message.
- **Function Name** – The name of the function that you want to call that is in a script attached to the receiver GameObject. This script will also send the all of the text in the text area as a parameter. This function is called when the user presses "Enter" after entering text.



Slider.js – The slider component is more complicated to set up. Please use the provided Slider prefab in the Prefabs folder. The prefab contains three child GameObjects: SliderBar, SliderButton, and SliderValue. The SliderBar contains a GUITexture with a texture that makes up the bar for the slider. SliderButton contains a GUITexture for the drag-able button on the slider and the Slider.js script. The SliderValue GameObject contains a GUIText component that displays the current value of the slider.

Parameters:

- **Slider Bar** – A GameObject with a GUITexture containing the bar for the slider. See the SliderBar GameObject.
- **Slider Button** – A GameObject with a GUITexture containing the drag-able button for the slider. This GameObject should have the Slider.js script attached.
- **Slider Value** – A GameObject with a GUIText attached. Displays the current value of the slider.
- **Min Value** – The minimum value of the slider.
- **Max Value** – The maximum value of the slider.
- **Current Value** – The current value of the slider based on where the SliderButton is relative to the SliderBar. (Read Only)
- **Direction** – Select horizontal or vertical based on the direction of the slider.
- **Receiver** – A GameObject with a script that is designed to receive a message.
- **Function Name** – The name of the function that you want to call that is in a script attached to the receiver GameObject. This script will also send the value of the slider (as a float) as a parameter. This function is called every time that the slider is modified.

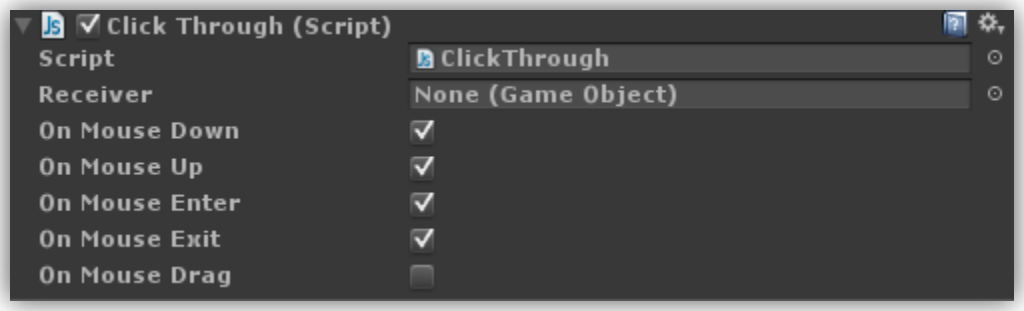


ClickAndDrag.js – Attach this script to a GameObject with a GUIText or GUITexture on it. This script allows the user to drag the GUI element around the screen. Also works with other interactive elements like buttons and radio buttons.

ClickThrough.js – This script sends mouse events to another GameObject. This is useful when you have a text box or text area that you want to receive mouse events but there is a GUIText on top of it that is taking the mouse events instead. When deciding which mouse events to send, make sure the script on the receiver GameObject has a defined function for that mouse event.

Parameters:

- **Receiver** – A GameObject with a script that is designed to receive a message.
- **On Mouse Down** – Boolean flag for relaying the OnMouseDown function.
- **On Mouse Up**– Boolean flag for relaying the OnMouseUp function.
- **On Mouse Enter** – Boolean flag for relaying the OnMouseEnter function.
- **On Mouse Exit** – Boolean flag for relaying the OnMouseExit function.
- **On Mouse Drag** – Boolean flag for relaying the OnMouseDrag function.

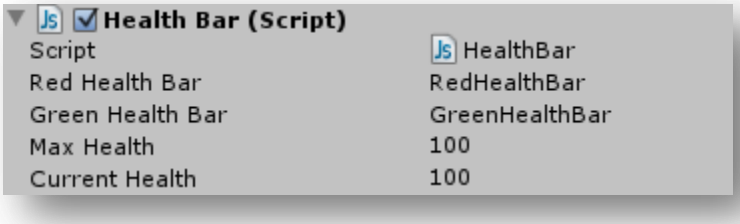


HUD Elements

HealthBar.js – Uses two overlapping GUITextures to create a health bar. Scales and repositions the top GUITexture based on the current health.

Parameters:

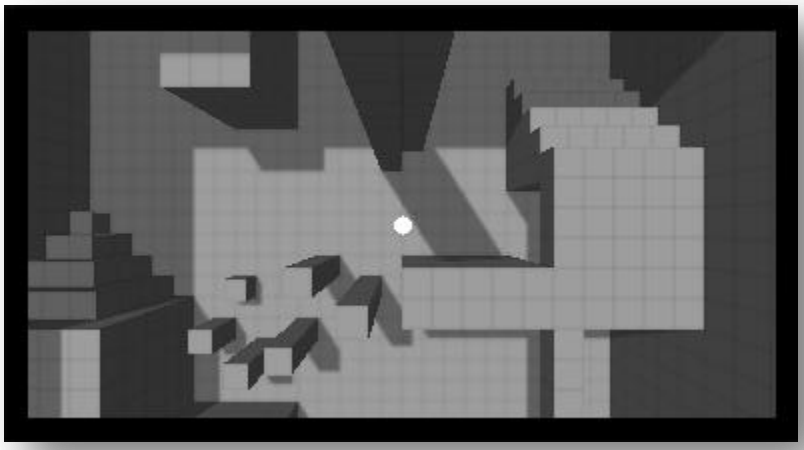
- **Green Health Bar** – A GameObject with a GUITexture attached that graphically represents the player's health. Must be on top of the Red Health Bar.
- **Red Health Bar** – A GameObject with a GUITexture attached that represents the total health that the player can have. Must be underneath the Green Health Bar.
- **Max Health** – The maximum health of the player.
- **Current Health** – The current health of the player.



MiniMap.js – Use this script in conjunction with a camera. Uses a top down camera that follows the player to create a mini map. Check the Prefabs folder for a ready-made prefab. Just assign the target parameter.

Parameters:

- **Target** – The Transform of an object to follow.
- **Height** – How high the camera is above the target.



Timer.js – Attach this script to a GameObject with a GUIText component. Displays the number of seconds that have passed since the scene was loaded.

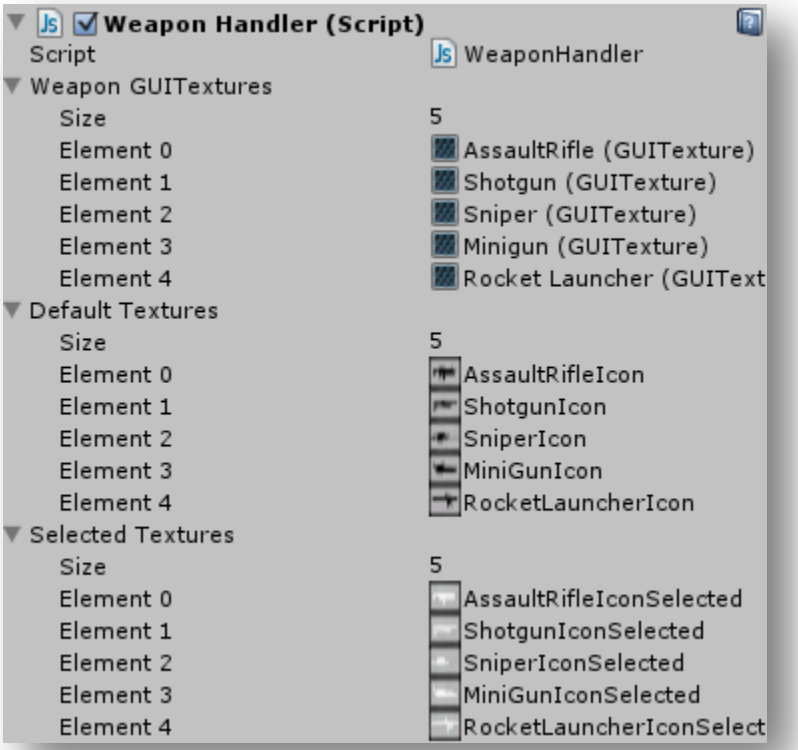
Parameters:

- **Time** – the amount of time since the scene was loaded.
- **Gui** – The GUIText component that is displaying the time.

WeaponHandler.js – Switches the selected weapon when the mouse scroll wheel is used.

Parameters:

- **Weapon GUITextures** – An array of all of the GUITextures for the different weapon icons. Make sure order in this array is the same as the order of the arrays in the following parameters.
- **Default Textures** – An array of textures representing the unselected weapons.
- **Selected Textures** – An array of textures representing the selected weapons.



GUI Systems

Main Menu – The MainMenu scene contains a fully functioning main menu. It combines some of the GUI components listed above with some simple logic to hide and show groups of GUI elements.

MainMenu.js – Handles the GUI elements used to make a functioning main menu. The parameters represent the groups of GUI elements that make up the submenus. The Message function receives messages from buttons and radio buttons, parses the message, and executes the appropriate command. To add additional functionality to the main menu, edit the Message function to support new messages. This script is attached to the Main Camera in the MainMenu scene.

Options.js – Handles all of the interactions in the options submenu. The Message function in this script handles the buttons for changing resolution, quality level, anti aliasing and vertical sync. To add additional functionality to the options menu, edit the Message function to support new messages.

Singleplayer.js – Handles all of the interactions in the singleplayer submenu. The Message function in this script handles the buttons for starting a new game or loading a game. These functions are intentionally left blank and are to be filled in by you! To add additional functionality to the singleplayer menu, edit the Message function to support new messages.

Multiplayer.js - Handles all of the interactions in the multiplayer submenu. The Message function in this script handles the buttons for hosting a game or joining a game. These functions are intentionally left blank and are to be filled in by you! To add additional functionality to the multiplayer menu, edit the Message function to support new messages.

Pause Menu – The HUD_Pause scene contains a HUD and a pause menu. Pressing escape will hide the HUD elements, pause the game and bring up the pause menu.

PauseMenu.js – Handles the GUI elements that make up the pause menu. When the escape key is pressed, this script hides the HUD elements, pauses the game and displays the pause menu elements. The Message function in this script parses messages from the various buttons in the pause menu.

PauseMenuOptions.js – When the Options button is selected in the pause menu this script displays interactive GUI elements to modify various graphics options. Each button and radio button sends a different message to the Message function. The Message function parses these messages and executes the appropriate command.

Contact

If you have any questions or comments for me please contact me at dylanyates92@gmail.com