

09 User Interfaces

Contents

[Overview 3](#_Toc507691688)

[9.1. HUD Types 4](#_Toc507691689)

[Screen Space 4](#_Toc507691690)

[In-World 4](#_Toc507691691)

[VR 5](#_Toc507691692)

[9.2. Unreal Motion Graphics UI Designer 5](#_Toc507691693)

[9.3. Key Widget Types 5](#_Toc507691694)

[9.4. Common Panel Types 6](#_Toc507691695)

[9.5. UI Focus 6](#_Toc507691696)

[9.6. Animation 6](#_Toc507691697)

[Glossary of Terms 8](#_Toc507691698)

[Exercises 9](#_Toc507691699)

[Exercise 9A: Creating a Simple HUD 9](#_Toc507691700)

[Exercise 9B: Extension (UMG and Blueprint Communication) 10](#_Toc507691701)

[Multiple-Choice Quiz 11](#_Toc507691702)

# Overview

**PURPOSE**

The purpose of this tutorial is to introduce you to the toolsets provided for the creation of UI and HUD interfaces within Unreal Engine 4. You will also become familiar with the terminology used to describe the UI and HUD in the context of game design in general as well as within the scope of Unreal Engine development.

**SCOPE**

This section will provide an introductory overview of the following:

* HUD types
* Unreal Motion Graphics UI Designer
* Widgets
* Animation with UMG

**PREREQUISITES**

It is assumed that you have completed the previous tutorials.

## 9.1. HUD Types

The User Interface of a game is the methods and features that allow the player to interact with the game, and the game to provide information to the player. It can include a range of mechanics within the gameplay, input from controllers, menus, and HUDs and utilizes 2D and 3D art, sound, and physical feedback via controllers (rumble, haptic, force feedback).

More often than not, when people discuss User Interface, they are referring to either menus or HUDs. While these are important to most games, remember that other methods exist and in some cases can offer an easier or clearer user experience, or a more immersive experience. A recent trend in game design is to integrate menus and HUDs into the game world and story, a method called Diegetic Interfaces.

### Screen Space

Screen space UIs are HUD Widgets that are drawn directly to the screen and are typically static outside of when they are created/destroyed, during which they are often animated. Screen space UIs have been the standard for a long time; however, in the last ten years, in-world user interface elements have become more and more prevalent. Screen space Widgets are created in Unreal by creating a Widget and adding it to a viewport or screen.

Typical examples of screen space elements include the following:

* Game menus
* On-screen health and stamina counters
* Item-selection dials
* Grayscale effects when the player is beginning to take too much damage (typically performed through a post process effect)

### In-World

The world itself can be used to provide the player with feedback on mechanics and the environment. It can be worked in thematically, such as via a heat counter built into a player’s suit or computer terminals flashing messages within the world, or appear as more conventional UI elements that are simply parented to actors in the world. Due to the wide nature of in-world UI elements, they can be added in a variety of ways: material effects can be driven through dynamic material instances controlled by Blueprint, and parented Widgets are often created by spawning a Widget component on the relevant actor.

Typical examples of in-world UI elements include the following:

* Health bars over the heads of other players
* Terminals inside the world
* Visible damage to a vehicle you’re driving to illustrate its current health. Often a vehicle will exhibit a smoke or fire particle effect as a final warning to players before it explodes.

### VR

Because of the nature of virtual reality, existing UI mechanisms are not always appropriate: the player lacks a mouse, and screen space Widgets have no flat surface to be projected onto by default. In addition, the user no longer has a wide range of inputs at their disposal and, currently, few input standards to follow. For this reason VR interfaces often require development explicitly designed for a VR project.

Common types of VR interface mechanisms include but are not limited to the following:

* Cylindrical projection of Widgets around the player for them to point at and select options from
* Motion Controller parented Widgets for when the player to looks at their hands and interact with accordingly. Typically this style allows for selection from the opposite controller by pointing or selection using a thumb stick on the controller that the panel is being displayed on.

## 9.2. Unreal Motion Graphics UI Designer

Unreal Motion Graphics UI Designer (UMG) is a visual UI authoring tool that can be used to create 2D UI elements inside Unreal Engine. These elements can be displayed directly on the screen as well as within the game world or as part of 3D meshes.

The main building blocks of UMG are **Widgets**. A Widget can be added to another Widget, which can then be added to a third Widget, creating a hierarchical structure. This method of creation is known as composition. When Widget A is added to Widget B, Widget B is now referred to as the parent of Widget A, which is referred to as the child of Widget B. The parent Widget now can control a range of properties of the child Widget, including where it is rendered, how big it is, and whether or not it is visible.

Not all Widgets can contain other Widgets. To be able to contain another Widget, a Widget must have a **slot**. A Widget can have a set number of slots, or allow any number of slots. A slot may allow the child Widget to specify some properties about how it would like to be displayed. What those properties are and how they are used are up to the parent Widget, however.

## 9.3. Key Widget Types

There are many types of pre-built Widgets within UMG.

Some Widgets are designed to display information or take information from players, while others are used to help format and control the layout and positions of Widgets. Beyond the default types, it’s possible for developers to develop their own Widgets to ensure easier content additions and more streamlined behaviors in their own projects.

***Panels*** *control the layout and placement of other Widgets.*

***Display Widgets****, such as Text and Images, display information to the player. There are a range of pre-built display Widgets to help with more complex ways of displaying information, such as a Progress Bar.*

***Input Widgets****, such as Buttons, Text Boxes, and Combo Boxes, take user input through the keyboard and mouse.*

***User Widgets*** *are Widgets made within UMG, which can be reused many times and placed within other User Widgets.*

## 9.4. Common Panel Types

Panels are easy mechanisms for controlling the placement of elements and sub-Widgets within a Widget. The following are the core types of panels and how they affect placement:

***Canvas Panel:*** *A designer-friendly panel that allows Widgets to be laid out at arbitrary locations, anchored, and* z*-ordered with other children of the canvas.*

***Grid Panel:*** *A panel that evenly divides up available space between all of its child Widgets.*

***Horizontal Box:*** *A layout panel allowing child Widgets to be automatically laid out horizontally. It is great for arranging Widgets side by side.*

***Vertical Box:*** *A layout panel allowing child Widgets to be automatically laid out vertically. It is great for stacking Widgets on top of each other.*

***Scroll Box:*** *An arbitrary scrollable collection of Widgets. It is great for presenting 10–100 Widgets in a list.*

***Size Box:*** *Allows you to place content with a desired size and have it scale to meet the constraints placed on the box’s allotted area.*

***Overlay:*** *Allows Widgets to be stacked on top of each other and uses a simple flow layout for content on each layer.*

## 9.5. UI Focus

Focus indicates what component of the game is allowed to capture the input from controllers, mice, and keyboards. For example, a mouse is often used in a game to control aim; however, it is also often used to control the cursor in a menu. Doing both at the same time would be confusing and problematic for the player, so Focus allows a developer to explicitly set what should be using the input.

Once UMG is given input focus, it is still necessary to be even more specific as to which Widget can accept the input events. For example, you can have multiple text entry boxes; which one should the letters being typed using the keyboard appear in? This sort of focus can be set by the user—that is, clicking on that Widget using a cursor—and it can also be set programmatically.

Once a Widget has focus, it can receive input events and define behavior tied to those events using blueprints in the Graph Editor within the UMG Editor.

## 9.6. Animation

Located along the bottom of the Widget Blueprint Editor are two windows that allow you to implement and control animations for your UI Widgets.

The first, the Animations window, allows you to create animation tracks, which are used to drive the animation of your Widgets.

The second, the Timeline window, allows you to control how an animation is applied to a Widget over time, which is done by placing Keyframes at specified times and then defining how the attached Widget should appear at that Keyframe (this could be size, shape, location, or even coloring options). A Widget can have multiple animations (for example, having a button move across the screen and flash at the same time to control multiple properties).

When an animation is created, a variable for it is also created. If you drag the newly created variable into the graph, you will be able to issue commands to it such as Play or Stop.

|  |  |
| --- | --- |
| C:\Users\Melanie Nikdel\AppData\Local\Microsoft\Windows\INetCache\Content.Word\tips-png-4.png | **Need More?**   * For tips and guidelines on setting up UMG Widgets, see <https://docs.unrealengine.com/latest/INT/Engine/UMG/UserGuide/BestPractices/>. * For step-by-step guides across multiple UMG features, see <https://docs.unrealengine.com/latest/INT/Engine/UMG/HowTo/index.html>. |

## Glossary of Terms

***UMG (Unreal Motion Graphics UI Designer):*** *A visual UI authoring tool that can be used to create 2D UI elements inside Unreal Engine.*

***Slate:*** *The name for the custom UI programming framework in Unreal. The editor interface is largely built using Slate.*

***Widget:*** *A pre-made function that can be used to construct your interface (things like buttons, checkboxes, sliders, progress bars, and others).*

***User Interface(UI):*** *The methods and features that allow the player to interact with the game, and the game to provide information to the player.*

***Slots:*** *The invisible glue that binds Widgets together. In Slate they are a lot more explicit in that you must first create a Slot and then choose what control to place inside of it. In UMG, however, you have Panel Widgets that automatically use the right kind of Slot when Child Widgets are added to them.*

***Anchor:*** *Used to define a UI Widget’s desired location on a* ***Canvas Panel*** *and maintains that position with varying screen sizes.*

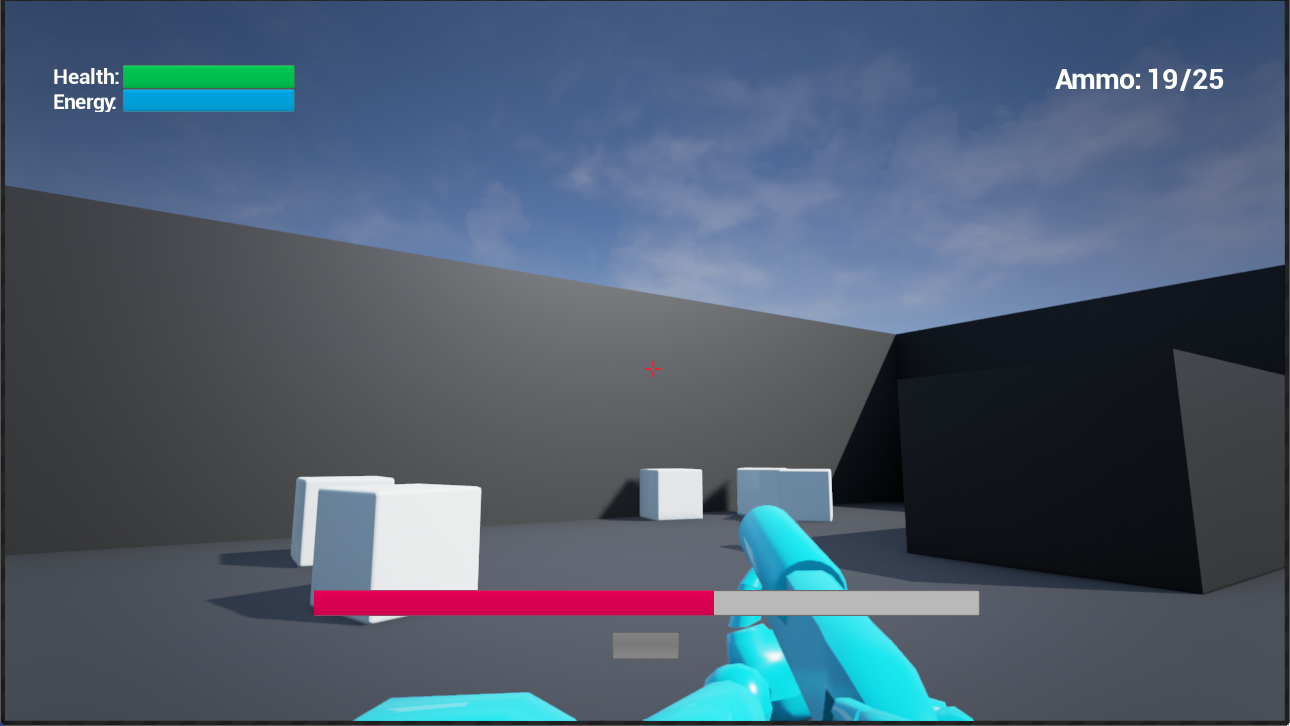
# Exercises

## Exercise 9A: Creating a Simple HUD

**Deliverables:** Project files (map/uassets), In-class observation

Have the students work through the HUD tutorial using the First Person template.

The purpose of this exercise is to teach the students to use the toolset and understand the pipeline for creating a simple UI in Unreal Engine 4.



**Directions**

1. Start the Epic Games Launcher and log in to your account.
2. Launch the latest version of Unreal Engine (4.16 or above).
3. Load up and read the information in the following section of the Unreal Engine Documentation:

<https://docs.unrealengine.com/latest/INT/Engine/UMG/QuickStart/index.html>.

1. Create a new project following the instructions in the above link. Make sure to give your project an appropriate name. For this task, you can name it as follows: “*yourname*\_EX9A”.
2. Once you have completed the tutorial, demonstrate your work to the teacher (where applicable) and save a copy of your project as a zipped file with the following naming convention: *“yourname*\_EX9A.zip”.

## Exercise 9B: Extension (UMG and Blueprint Communication)

**Deliverables:** Project files (map/uassets), In-class observation

**Student Task:** Using your project and map from the previous exercise, create a new pickup object that includes its own UI components.

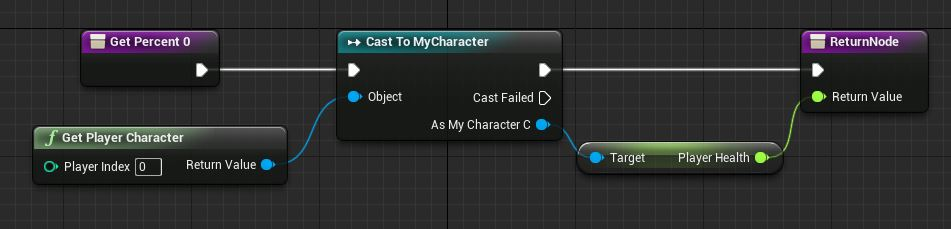


|  |  |
| --- | --- |
| C:\Users\Melanie Nikdel\AppData\Local\Microsoft\Windows\INetCache\Content.Word\tips-png-4.png | Check out the Unreal Documentation for some tips on how to get started:  <https://docs.unrealengine.com/latest/INT/Resources/ContentExamples/Blueprints_HUD/1_1/index.html>. |

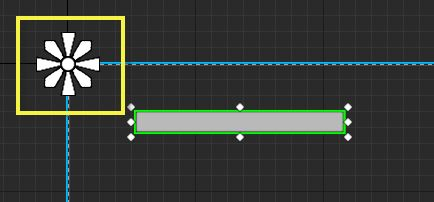
# Multiple-Choice Quiz

|  |  |  |  |
| --- | --- | --- | --- |
| Name: |  | Date: |  |

1. Which answer best describes a UMG Widget?
   1. A visual element that can be connected with other elements in a UI
   2. A pre-made function that can be used to construct an interface
   3. A specialized interactive UI component
   4. All of the above
   5. None of the above



1. Which of the following best describes the above image?
   1. Property Binding
   2. DPI Binding
   3. Function Binding
   4. Event Binding
   5. None of the above
2. Users are limited to only using Widgets that are available by default.
   1. True
   2. False
3. Which of the following is *not* a UMG Widget?
   1. Button
   2. Switch
   3. Checkbox
   4. Throbber



1. What is the name of the object contained within the yellow box?
   1. Anchor Cross
   2. Anchor Pin
   3. Anchor Medallion
   4. Anchor Star
2. VR interfaces are
   1. Unique in design but use the same tools
   2. The same as regular user interfaces and use the same tools
   3. Unique in design and use unique tools
   4. The same as regular user interfaces but use unique tools
3. A health meter above teammates would be an example of a screen space element.
   1. True
   2. False