|  |
| --- |
| IMAT 2103 – Progressive Game Engines |
| Lab 3: Gameplay Framework |

***Part 1:***

In this exercise, you will create a Level containing three characters that can be possessed by the Player Controller using the “**1**”, “**2**”, and “**3**” keys.

## Directions

1. Create a new project using the **Third Person** template.
2. Add to the Level two more instances of the **ThirdPersonCharacter** Blueprint, which is in the folder “**Content > ThirdPersonBP > Blueprints**”.
3. In the **Details** panel, rename the **ThirdPersonCharacter** instances in the Level “**Character1**”, “**Character2**”, and “**Character3**”.
4. Click the **Blueprints** button on the **Toolbar** and select “**Open Level Blueprint**”.
5. Select the **Character1** Actor in the Level, and in the Level Blueprint right-click in the **Event Graph** and choose “**Create a Reference to Character1**” (see Figure 1).

Graphical user interface, text, application

Description automatically generated

*Figure 1: Creating a Reference*

1. Repeat the previous step to create a reference to **Character2** and another to **Character3**.
2. Right-click in the **Event Graph** and create **keyboard events** for the “**1**”, “**2**”, and “**3**” keys.
3. Add the **Get Player Controller** function to the **Event Graph**.
4. Drag a wire from the **Get Player Controller** output execution pin and release the mouse button to open the **context menu**. Select the **Possess** function.
5. Complete the script in the Level Blueprint as shown in Figure 2.
6. Compile and play the Level. Use the “**1**”, “**2**”, and “**3**” keys to change the character being controlled.

Graphical user interface, application

Description automatically generated

*Figure 2: Level Blueprint*

## Outcome

After pressing **Play**, you can control each of the third-person characters in the Level by using the “**1**”, “**2**”, and “**3**” keys.

# Submission Requirements

Time for completion: 20–30 minutes

* Teacher observation and posting of the discussion board

***Part 2:***

In this exercise, you will create a simple HUD and a new Game Mode that will use the new HUD as the default HUD class.

## Directions

1. Create a new project using the **Third Person** template.
2. In the **Content Browser**, create a new folder named “**BPGuide**” to use for storing the new Blueprints that will be created.
3. Next, click the green **Add New** button in the **Content Browser** and select “**Blueprint Class**”.
4. In the **Pick Parent Class** window, expand the **All Classes** section and search for “**hud**”. Choose “**HUD**” as the parent class (see Figure 1).

Background pattern

Description automatically generated

*Figure 1: Parent Class: HUD*

1. Rename the new HUD Blueprint “**BP\_HUD\_Timer**”.
2. Double-click **BP\_HUD\_Timer** to open the **Blueprint Editor**.
3. Create an **Integer** variable named “**Time**”.
4. Right-click in the **EventGraph** and search for “**draw hud**”. Add the **Receive Draw HUD** event.
5. Add a **BuildString (int)** function and a **Draw Text** function. Create the script seen in Figure 2.

Graphical user interface, text, application, chat or text message

Description automatically generated

*Figure 2: Drawing the Time Variable*

1. Right-click in the **Event Graph** and search for “**custom event**”. Select “**Add Custom Event**” and rename it “**Count**”.
2. Add the **Set Timer by Event** function. Set the **Time** parameter to “**1.0**”, check the **Looping** parameter, and connect the red square of the **Count** custom event to the **Event** input parameter of the **Set Timer by Event** function. (See Figure 3.)
3. Add the **BeginPlay** event and complete the script seen in Figure 3.

Graphical user interface, application

Description automatically generated

*Figure 3: Drawing the Time Variable*

1. Create a new Blueprint class and choose “**Game Mode Base**” as the parent class. Rename it “**BP\_GameMode\_HUD**”.
2. Open **BP\_GameMode\_HUD** in the **Blueprint Editor** and click the **Class Defaults** button on the **Toolbar**. Set the **HUD Class** property to “**BP\_HUD\_Timer**” and the **Default Pawn Class** property to “**ThirdPersonCharacter**” (see Figure 4).

Graphical user interface, application

Description automatically generated

*Figure 4: Game Mode Classes*

1. To change the **Game Mode** of the current Level, click the **Settings** button in the **Level Editor** and choose “**World Settings**”.
2. In the **World Settings** panel, select “**BP\_GameMode\_HUD**” in the **GameMode Override** property.
3. Play the Level to see the **BP\_HUD\_Timer** Blueprint in action.

## Outcome

After pressing **Play**, a text like “Time: 1” will be drawn on the screen, and the number will increase and be updated on the screen every second.

# Submission Requirements

Time for completion: 20–30 minutes

* Teacher observation and posting of the discussion board

***Part 3:***

# Overview

In this exercise, you will practice working with the World Outliner by creating folders and placing Actors in them. You will also explore attaching Actors in the World Outliner to create parent-child relationships and gain a better understanding of world and relative transforms.

# Outcome

Students will demonstrate their ability to use the World Outliner by attaching Actors to one another and creating folders.

# Process

1. Create a new Blueprint project with the following settings:
   * Project name: Outliner
   * Template: Third Person
   * Quality: Maximum Quality
   * Starter content: Yes
2. From the Place tab in the Modes panel, drag a Sphere Static Mesh Actor into the default Level.
3. With the sphere selected in the Details panel, navigate to the Physics section and activate *Simulate Physics*.
4. Playtest the Level and push the sphere around.
5. With the sphere selected, in the World Outliner to the right of the search bar, click on the icon of a plus symbol on a folder to create a new folder.
6. Give the folder a name.
7. From the Place tab in the Modes panel, drag a Cube Static Mesh Actor into the Level and place it next to the Sphere Static Mesh Actor.
8. With the cube selected, in the Details panel set the *Scale* property under “Transform” to “0.75, 0.75, 0.75” and the *Mobility* setting to “Movable”.
9. In the World Outliner, click on the cube and drag it onto the sphere in the folder.
10. Playtest the Level and push the sphere around. The cube should move and rotate with the sphere. Since the sphere is the parent and the cube’s default transform is relative, the sphere is the center of the cube’s world, so all of the cube’s transforms are relative to those of the sphere.
11. Now select the cube, and in the Details panel under “Transform” click on the triangle next to “Rotation” and select “World” from the drop-down menu.
12. Playtest the Level again and push the sphere around. The cube moves with the sphere but ignores the sphere’s rotation, even though the sphere is the cube’s parent.
13. With the cube selected, in the Details panel set the *Location* property under “Transform” to “0.0, 0.0, 0.0”. The Cube Static Mesh Actor should move to the Sphere Static Mesh Actor. Because the sphere is the cube’s parent, it is the center of the cube’s world.
14. If you playtest the Level now, you will need to turn off collision for the cube so it does not affect the movement of the sphere.

# Submission Requirements

Time for completion: 20–30 minutes

* Teacher observation and posting of the discussion board