LECTURE 5

Simple Game Example

LECTURE GOALS AND OUTCOMES

Goals

The goals of this lecture are to

- Implement a simple game
- Show how to use a Game Mode class to define the rules of a game
- Provide examples of how to use macros, custom events, and functions
- Demonstrate the use of Timers in a game

Outcomes

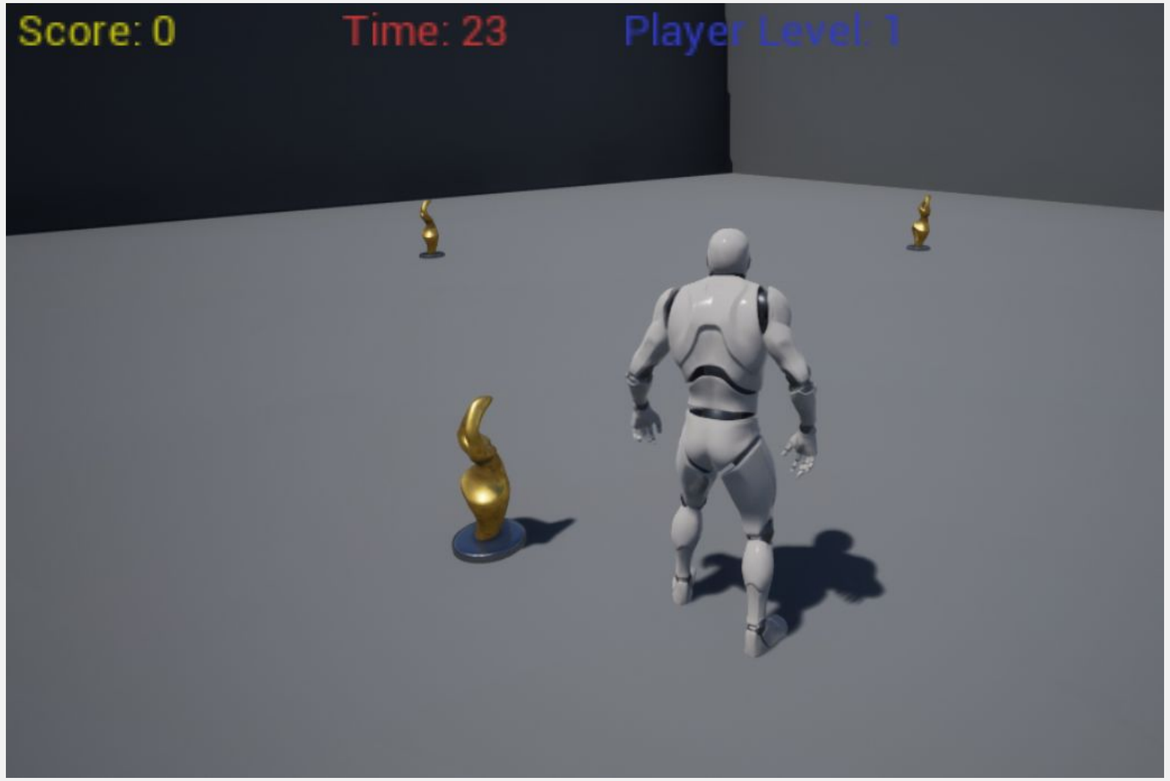
By the end of this lecture you will be able to

- Convert the rules of a game into Blueprint code
- Organize code into macros, custom events, and functions
- Draw some information using the HUD class
- Define Timers using functions or custom events

GAME CLASSES

The game created in this lecture is based on the Third Person template. The game uses four main Blueprint classes:

- **ThirdPersonCharacter**: A Pawn subclass that represents the player and is part of the Third Person template. This class is assigned to the **Default Pawn Class** parameter of the Game Mode.
- **BP_Statue**: An Actor class that represents the statue. It checks for collision with the player and has the logic to change its position periodically.
- **BP_Guide_GameMode**: Controls the state of the game and stores some variables, such as those for time, score, and Level. It also defines the base classes, such as ThirdPersonCharacter.
- **BP_Guide_HUD**: Responsible for drawing the time, score, and Player Level values on the screen.



GAME RULES

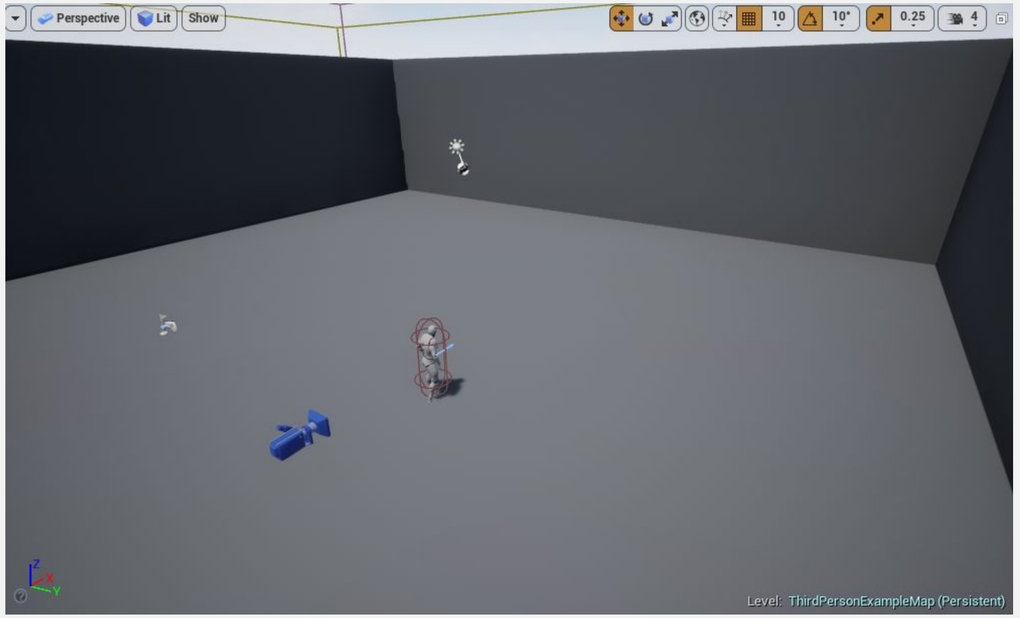
- The player must collect the small statues that appear on screen before time runs out.
- The initial time is set at 30 seconds.
- The game is over when there is no time remaining.
- For every five statues collected, the Player Level increases and 15 seconds are added to the time.
- The player starts at Player Level 1, and the maximum Level a player can reach is Player Level 5.
- There are three statues in the scenario that change position periodically.
- When a statue appears in a position, it will stay there for a period of time that depends on the current Player Level. The number of seconds it takes a statue to change position is the result of the expression “6 – Level”.
- When the player gets a statue, another one is spawned.
- The score for a statue collected is determined by the expression “10 x Level”.
- The time, score, and Player Level values will be drawn on the screen.

CREATING THE PROJECT

Create a new project using the **Third Person** template with **starter content**.

Remove the Static Mesh and Text Render Actors that are in the middle of the scene, leaving only the floor and side walls, as seen in the bottom image on the right.

In the **Content Browser**, create a new folder named “**BP_Guide**” that will store all new Blueprints.



BP_GUIDE_GAMEMODE

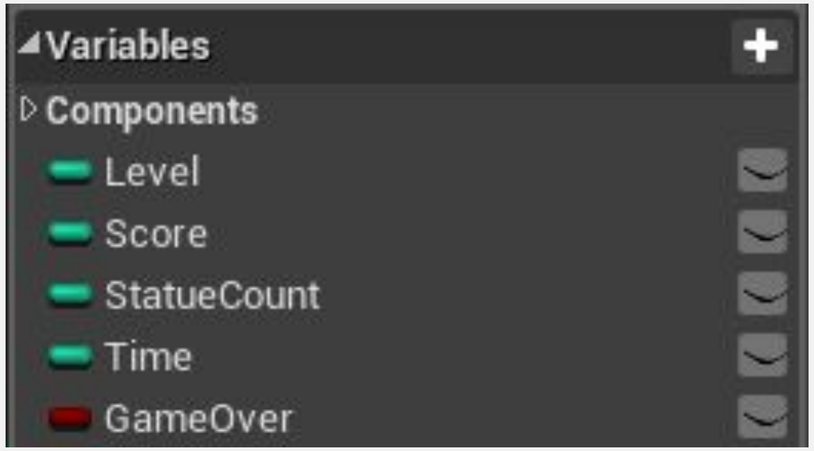
Create a new Blueprint class and choose “**Game Mode Base**” as the parent class. Rename it “**BP_Guide_GameMode**”.

Create the following **Integer** variables:

- **Level**: Stores the current Player Level in the game.
- **Score**: Stores the player’s score.
- **StatueCount**: Keeps track of the number of statues collected.
- **Time**: Stores the time remaining until the end of the game.

Create the following **Boolean** variable:

- **GameOver**: Indicates if the game has ended.

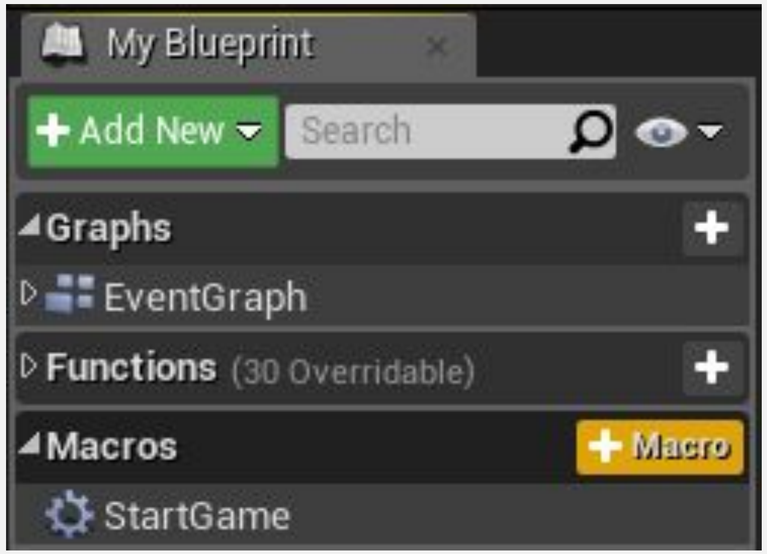


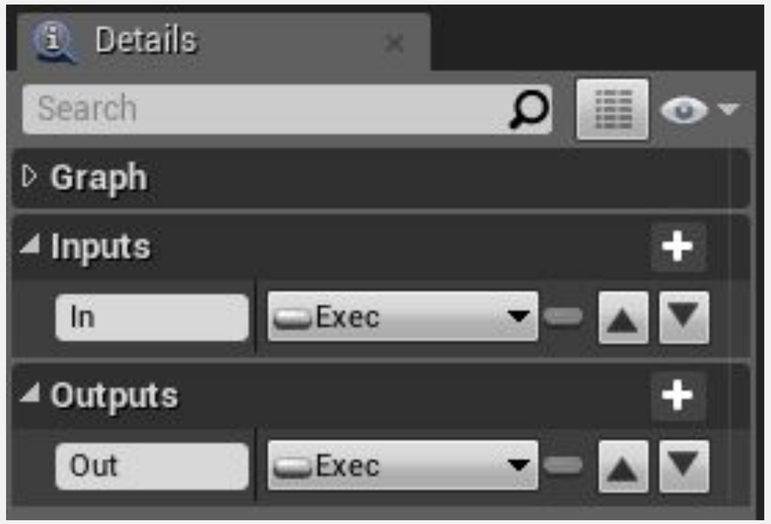
BP_GUIDE_GAMEMODE: START GAME MACRO

In the **My Blueprint** panel, create a macro named “**StartGame**”.

This macro is responsible for initializing the variables that control the state of the game.

In the **Details** panel for the **StartGame** macro, create an input parameter named “**In**” and set the type to “**Exec**”. Create an output parameter named “**Out**” and set the type to “**Exec**”.

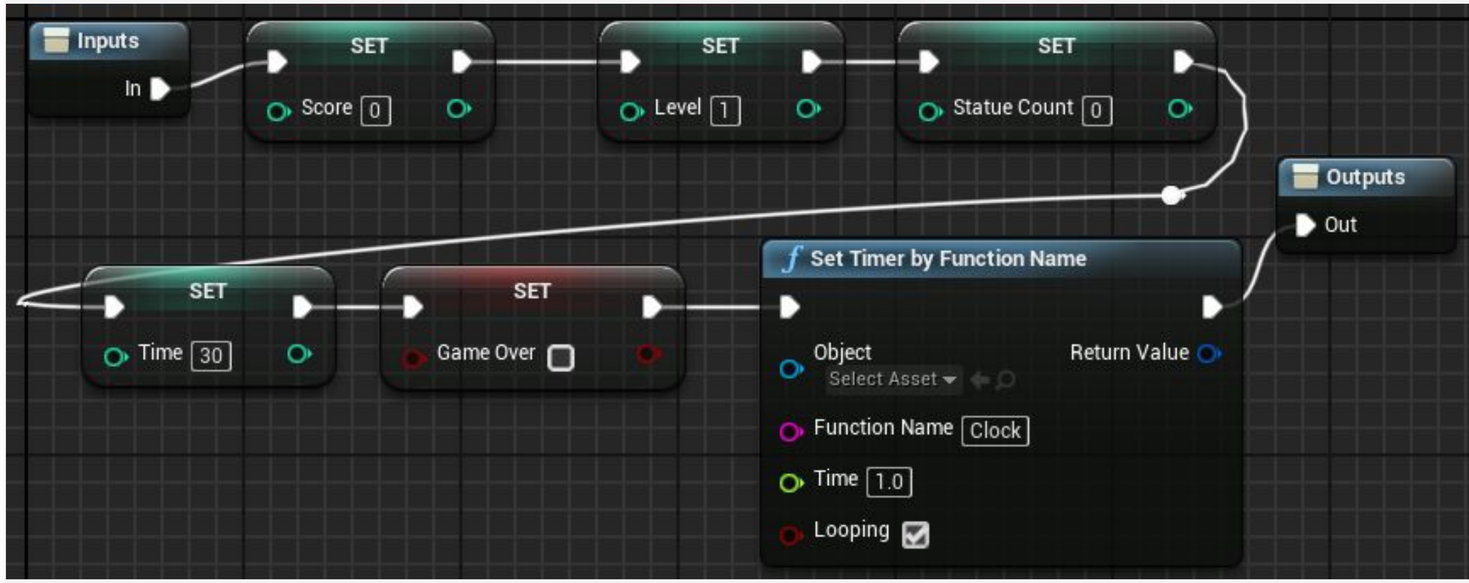




BP_GUIDE_GAMEMODE: START GAME MACRO

The **StartGame** macro will set the initial values of the variables.

The **Set Timer** function creates a Timer object that will call a custom event named “**Clock**” every 1.0 seconds.

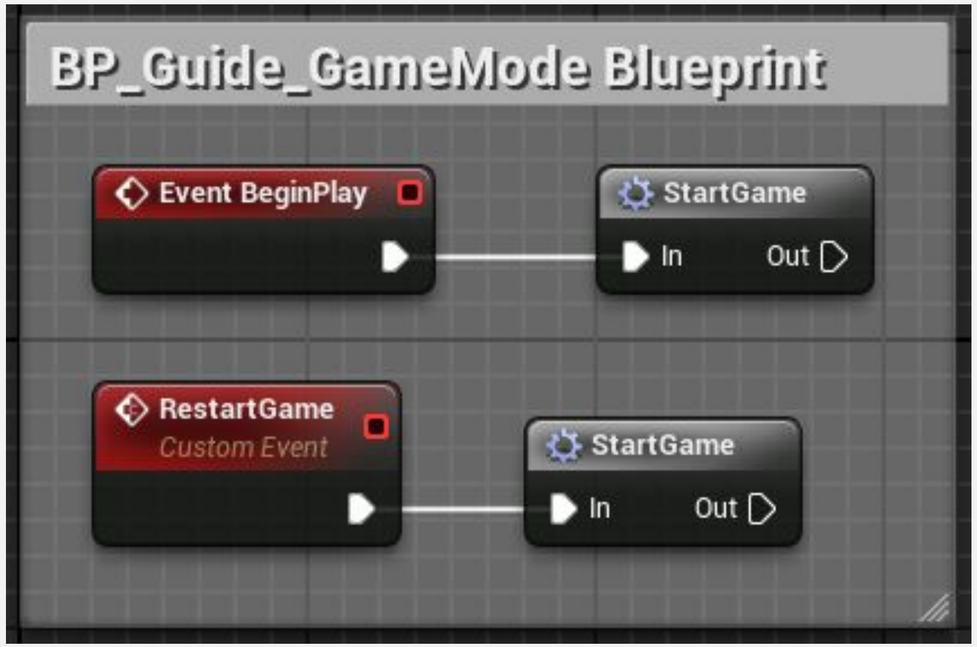


BP_GUIDE_GAMEMODE: USING THE MACRO

The **StartGame** macro is called from the **BeginPlay** event in the **BP_Guide_GameMode** Blueprint.

A custom event named “**RestartGame**” was created that also calls the **StartGame** macro.

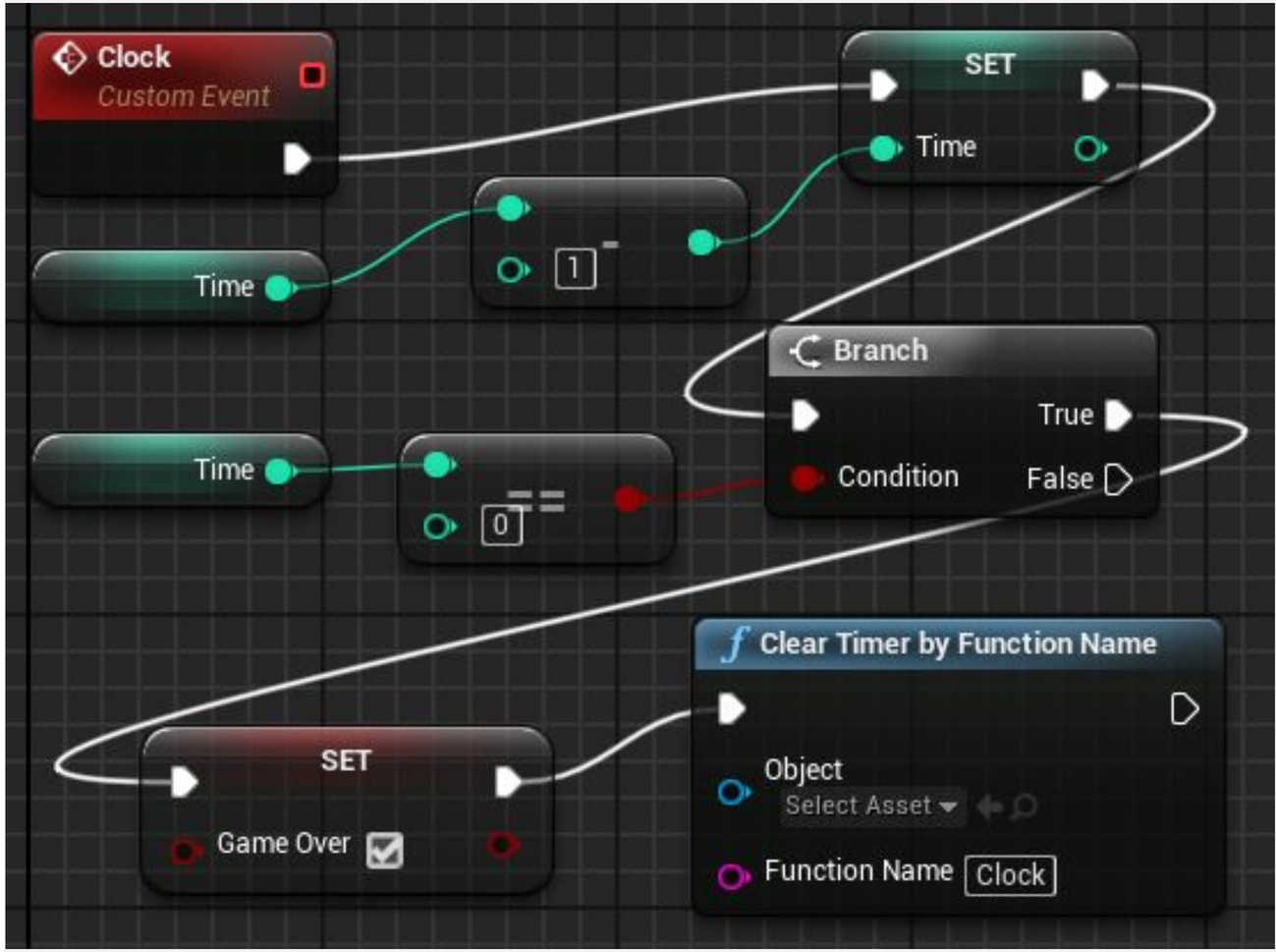
The **RestartGame** custom event is called from the **ThirdPersonCharacter** Blueprint when the player presses the **Enter** key, as seen in the bottom image on the right.



BP_GUIDE_GAMEMODE: CLOCK EVENT

Clock is a custom event that is called every second by the Timer. It has the following responsibilities:

- To decrease by “**1**” the value of the **Time** variable
- To check if the value of the **Time** variable is “**0**”; if “**true**”, to perform the following actions:
  - Set the value of the Boolean **GameOver** variable to “**true**”
  - Clear the **Timer** so that it stops calling the **Clock** event

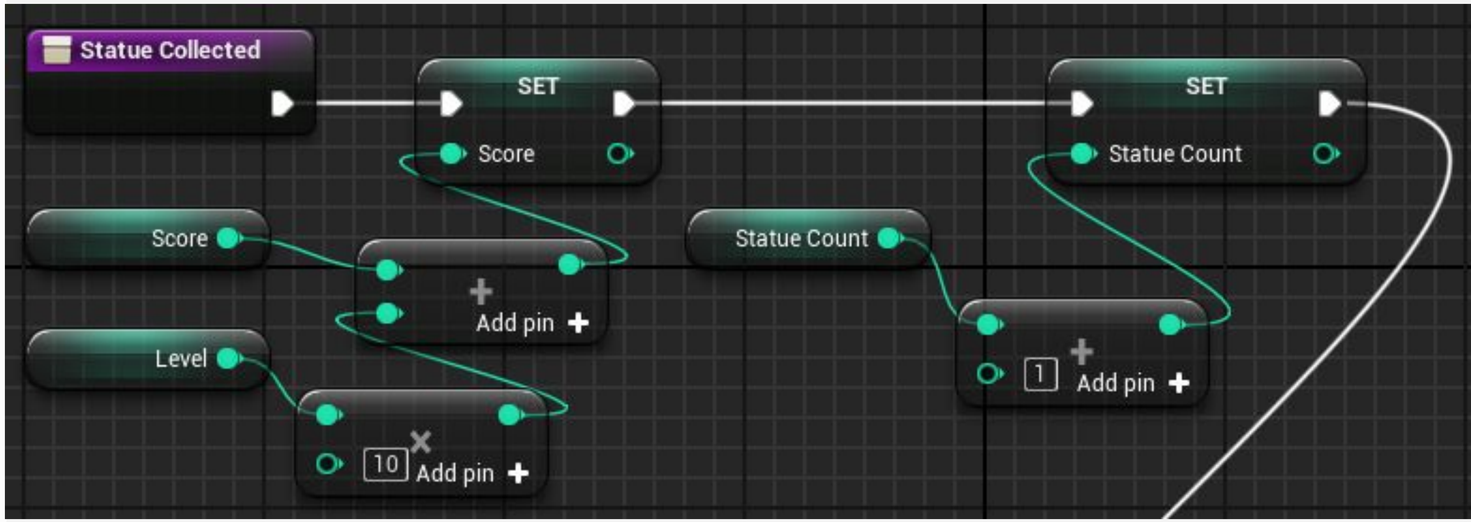


BP_GUIDE_GAMEMODE: STATUE COLLECTED 1/2

“**Statue Collected**” is a function in **BP_Guide_GameMode** that is called by the **BP_Statue** class when the player gets a statue.

The first part of the function performs the following actions:

- Adds to the value of the **Score** variable the points obtained when the player collects a statue, which is calculated with the expression “**10 x Level**”.
- Adds a value of “**1**” to the **StatueCount** variable, which stores the number of statues collected.



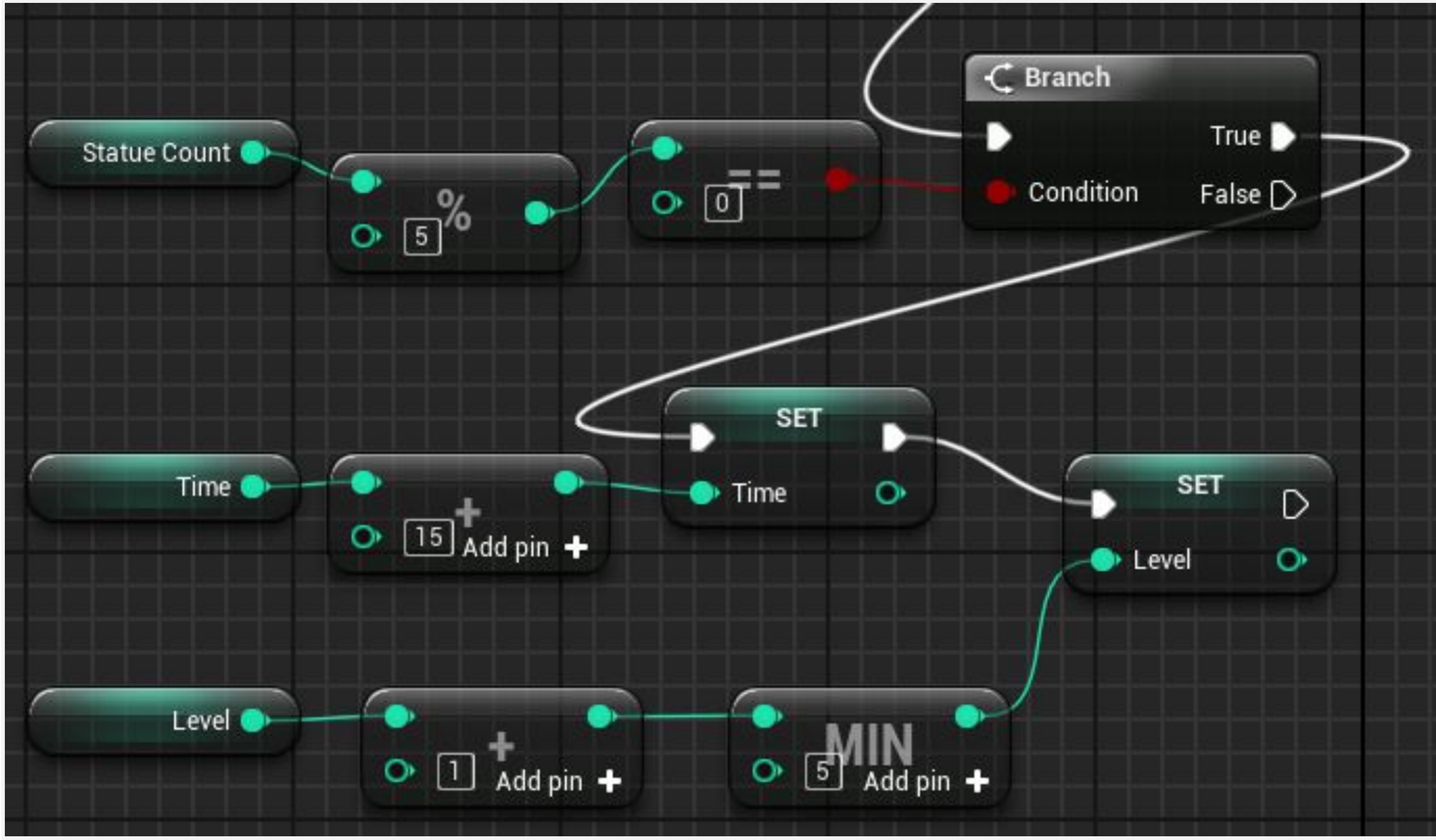
BP_GUIDE_GAMEMODE: STATUE COLLECTED 2/2

The second part of the **Statue Collected** function does the following:

- Tests whether the value of the **StatueCount** variable is a multiple of 5. If “**true**”, it performs the following actions:
    - Adds a value of “**15**” to the **Time** variable.
    - Adds a value of “**1**” to the **Level** variable, with the maximum value of the **Level** variable limited to “**5**”.

This means that for every five statues collected, the player advances a Level and gets 15 additional seconds of time.

The **modulo** (%) operator returns the remainder of the division.



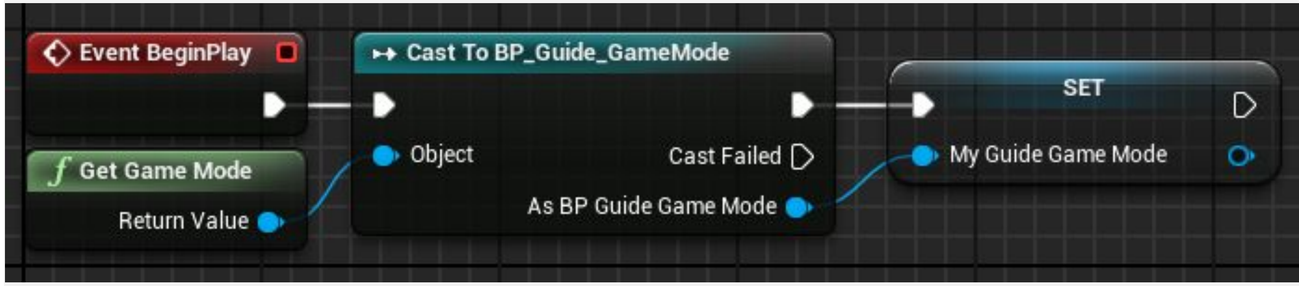
BP_GUIDE_HUD

Create a new Blueprint class and choose “**HUD**” as the parent class. Name it “**BP_Guide_HUD**”.

Create a variable named “**My_Guide_GameMode**” of type “**BP_Guide_GameMode Object Reference**”.

The image on the right shows the **BeginPlay** event getting a reference to the **Game Mode**, casting the reference to **BP_Guide_GameMode**, and saving that reference in the variable.

This reference will be used to access the variables that will be drawn on the screen.



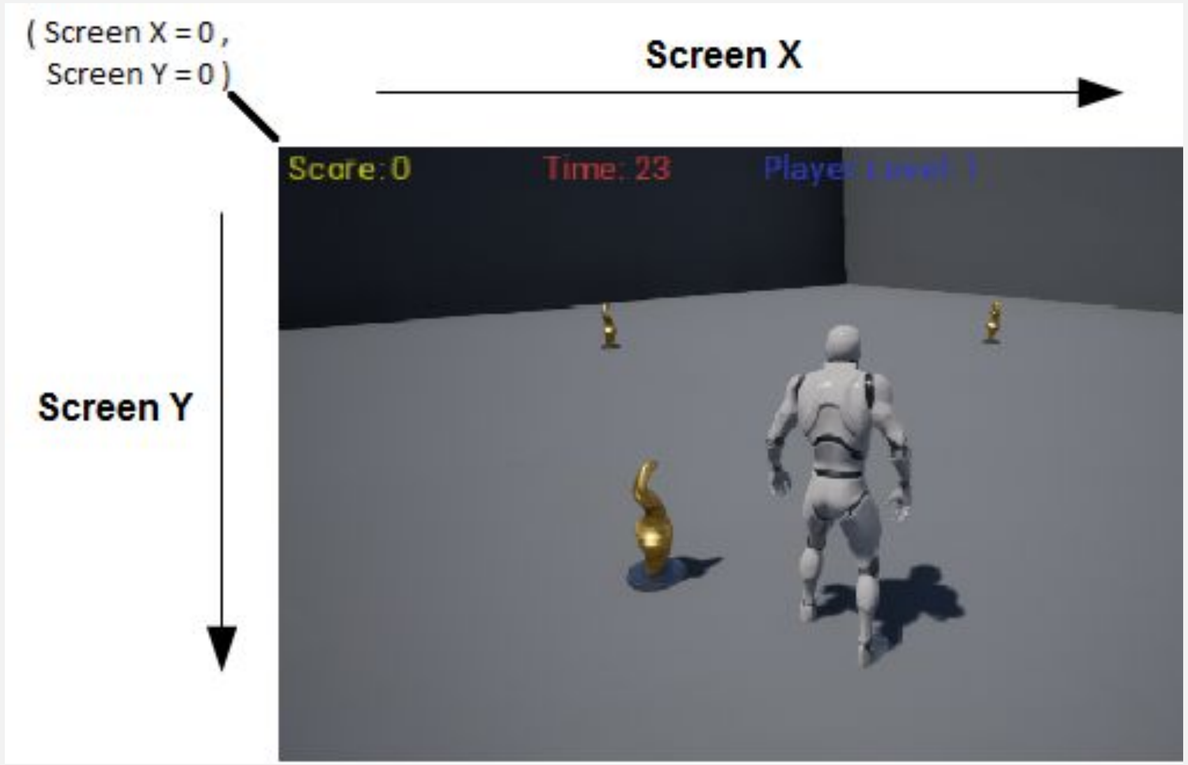
BP_GUIDE_HUD: SCREEN COORDINATES

The only action that will be used to draw in this game is the **Draw Text** function. This function receives some input values, among them those for two parameters known as “**Screen X**” and “**Screen Y**”, which depend on the resolution of the screen. For example, 1280 x 720 is a common resolution.

These values represent the coordinates of the screen on which the text will be drawn. The top left position of the screen is the origin, where the values of **Screen X** and **Screen Y** are “**0**”. The image on the right shows how the values of these two parameters determine where text will be drawn.

The **Screen Y** value used by the **Score**, **Time**, and **Level** variables is “**10**”, so the associated text appears at the top of the screen.

The **Screen X** values used are “**10**” by the **Score** variable, “**300**” by the **Time** variable, and “**550**” by the **Level** variable.

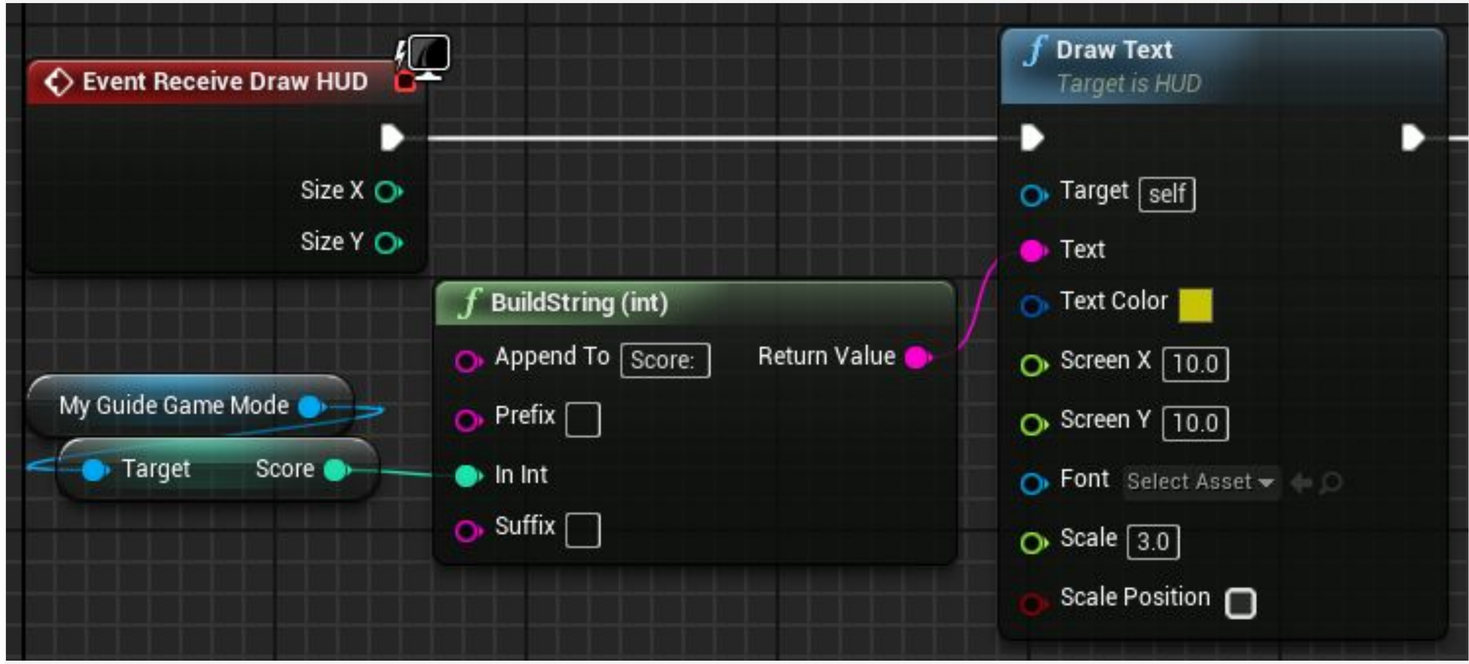


BP_GUIDE_HUD: RECEIVE DRAW HUD 1/3

To draw on the screen, the **Receive Draw HUD** event must be added to the Event Graph. This event is available only in Blueprints based on the HUD class.

The **Draw Text** function of the HUD class is used to print the score on the screen, as shown in the image on the right.

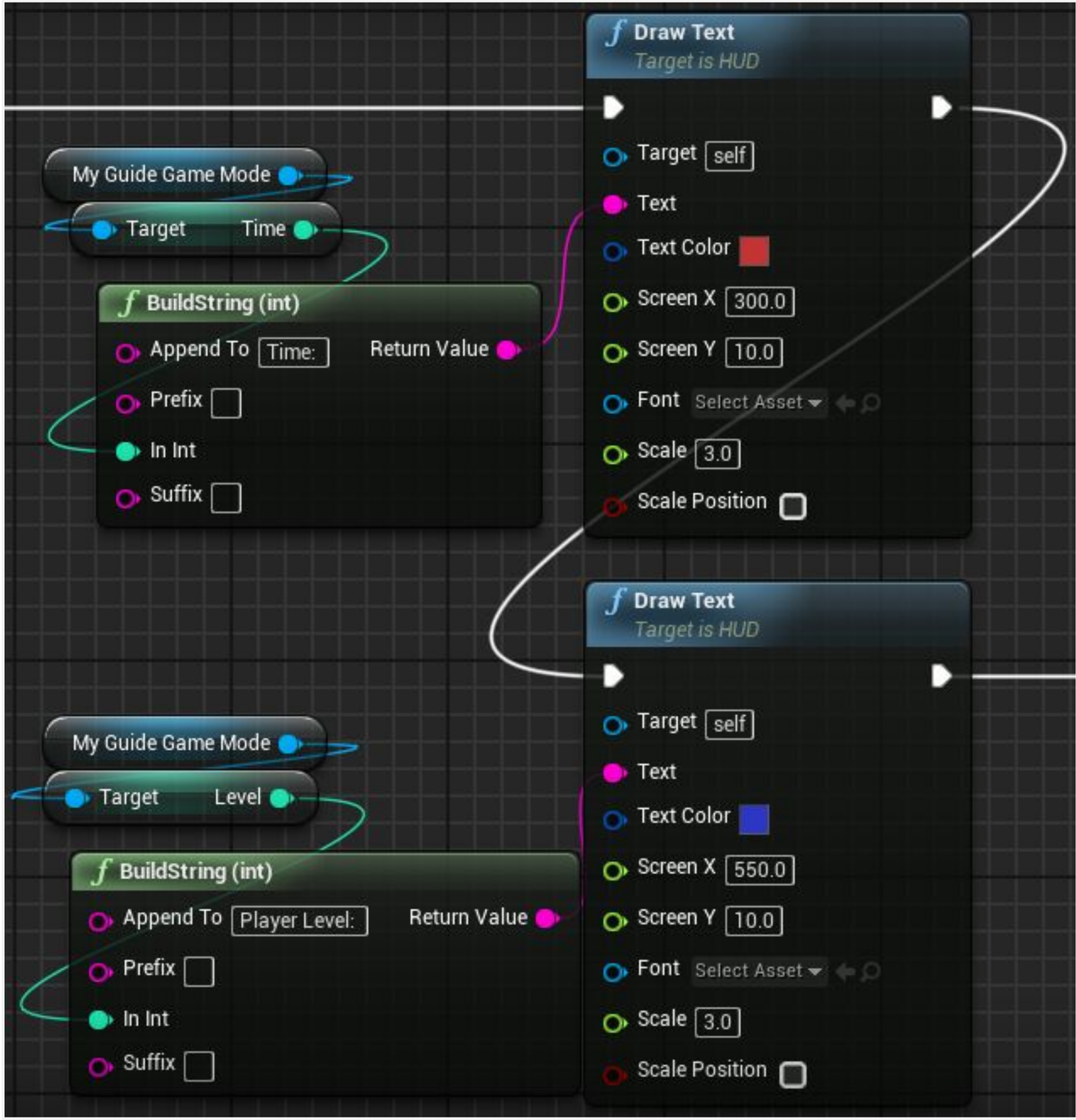
The **BuildString (int)** function is used to produce a string containing the text “**Score:**” plus the current value of the **Score** variable. The resulting string is passed to the **Text** parameter of the **Draw Text** function.



BP_GUIDE_HUD: RECEIVE DRAW HUD 2/3

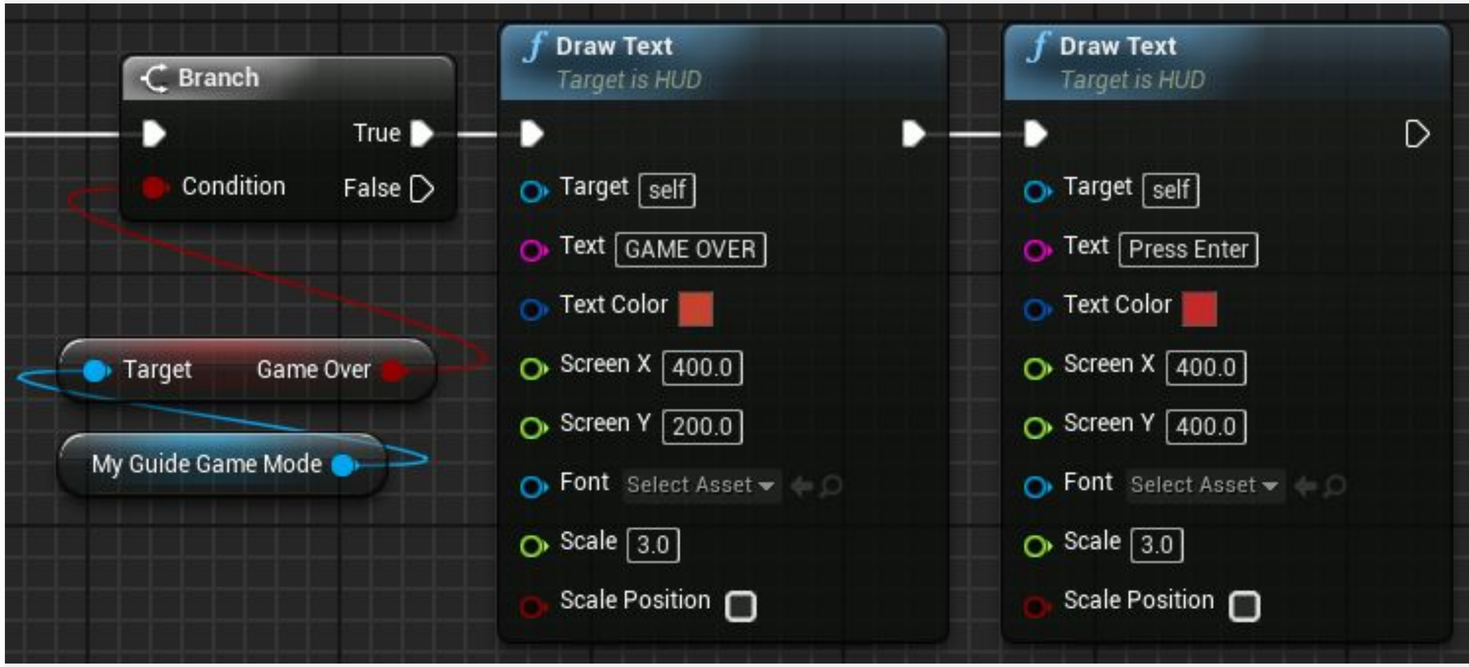
The setup for drawing text for the **Time** and **Level** variables is very similar to that for drawing text for the **Score** variable.

The only changes in the **Draw Text** function involve the parameters **Text**, **Text Color**, and **Screen X.**



BP_GUIDE_HUD: RECEIVE DRAW HUD 3/3

The last part of the **Receive Draw HUD** event tests if the value of the **GameOver** variable is “**true**”. If it is, then it draws on the screen the strings “**GAME OVER**” and “**Press Enter**”.



BP_STATUE

Create a new Blueprint class and choose “**Actor**” as the parent class. Rename it “**BP_Statue**”.

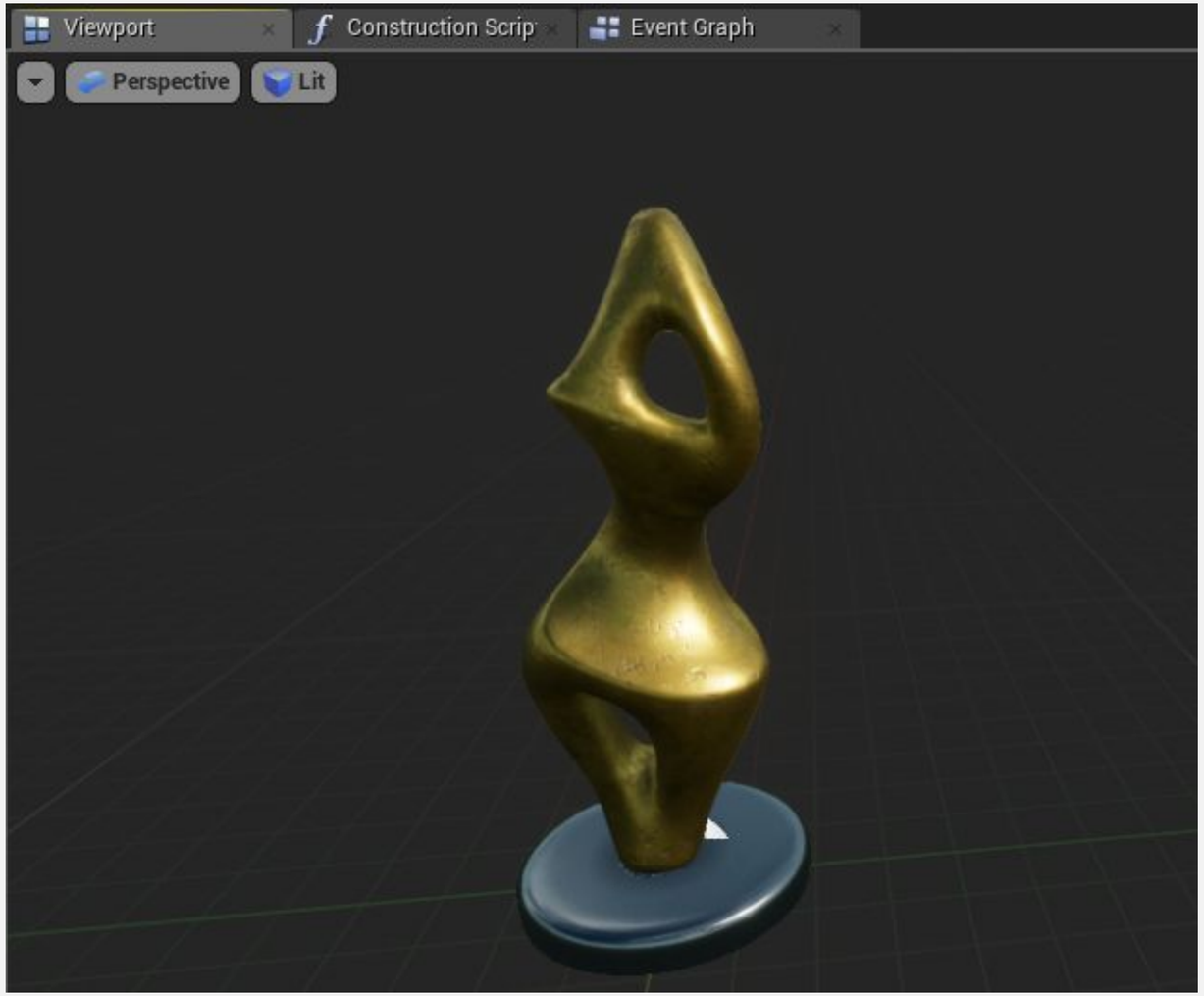
Add a **Static Mesh** component. In the **Details** panel for the **Static Mesh** component, set the **Static Mesh** property to “**SM_Statue**”.

Set the **Element 0** property in the **Materials** category to “**M_Metal_Gold**”.

Set the **Collision Presets** property to “**OverlapAllDynamic**”.

Compile and save your Blueprint class.

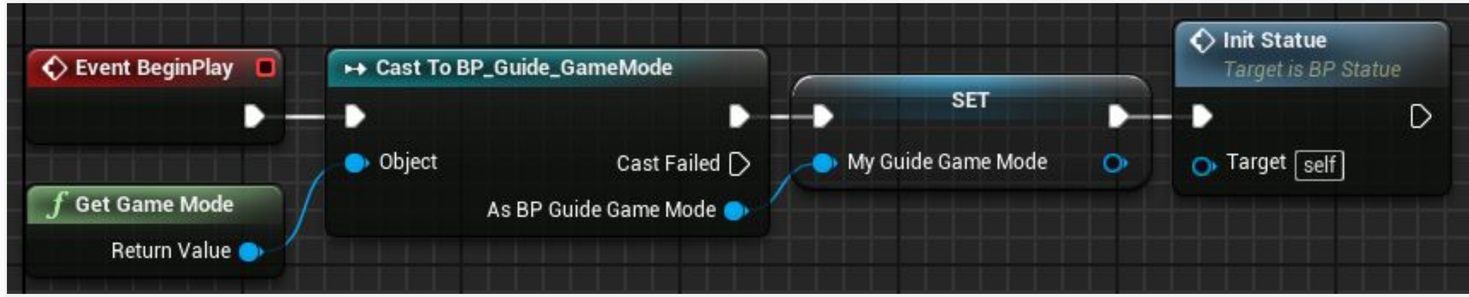
Add three instances of the **BP_Statue** Blueprint to the Level at any location.



BP_STATUE:
EVENT BEGIN PLAY

The image on the right shows the **BeginPlay** event getting a reference to the **Game Mode**, casting the reference to **BP_Guide_GameMode**, and saving that reference in the variable.

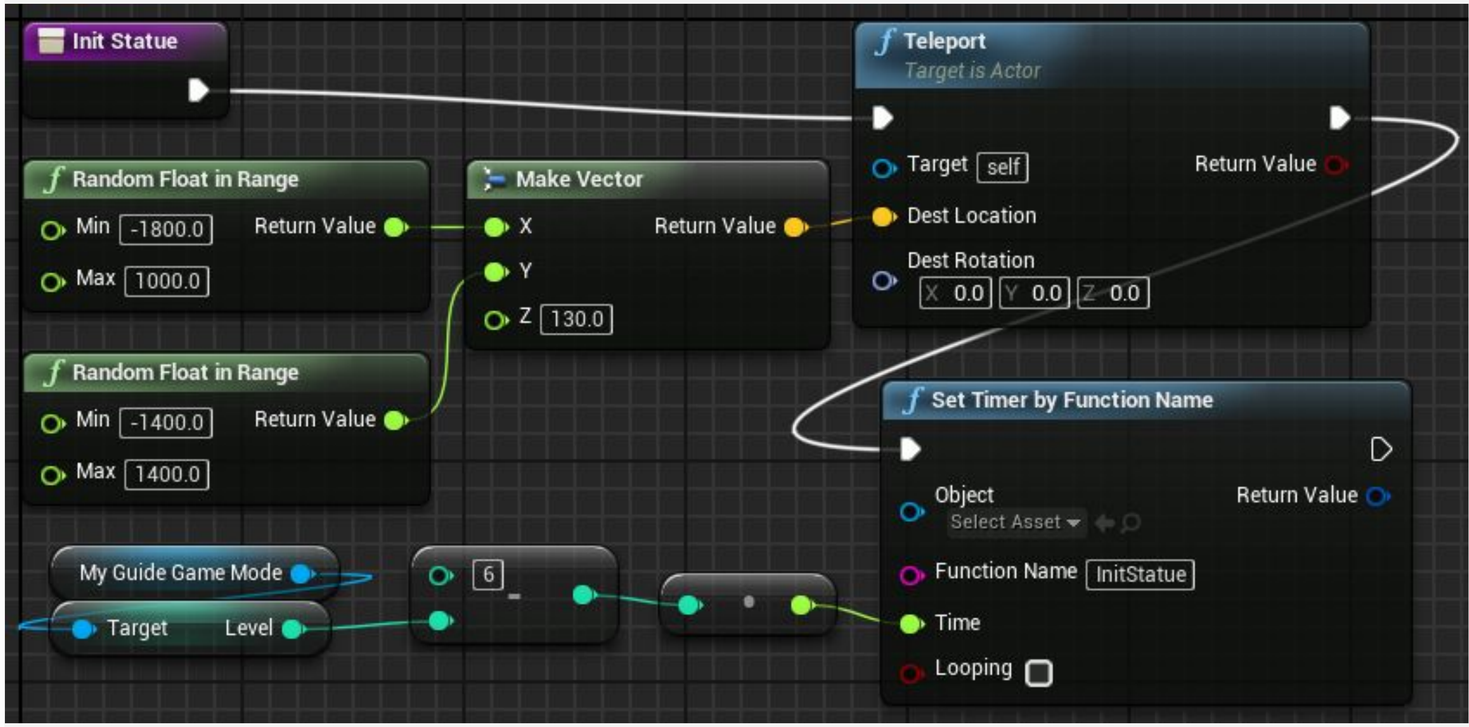
After that, the function **InitStatue** is called.



BP_STATUE: FUNCTION INIT STATUE

In the **InitStatue** function, there are two main nodes:

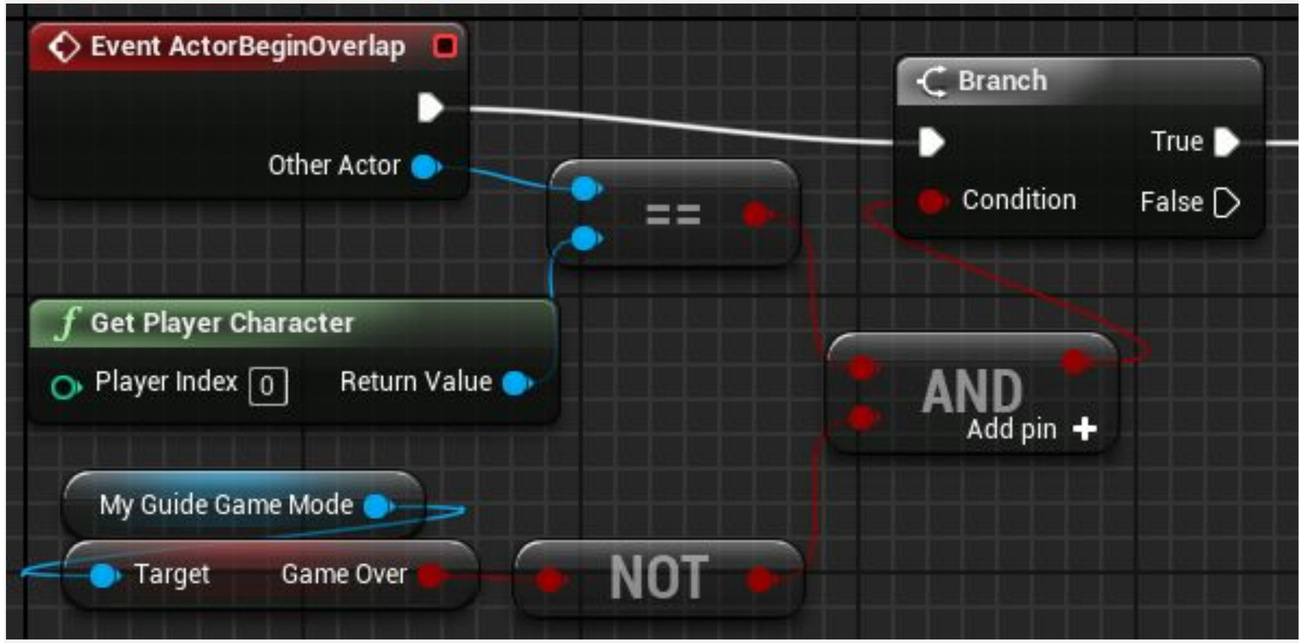
- **Teleport**: This node sets a new location for the statue. The location is represented by a vector (X, Y, Z) where the values of the **X** and **Y** parameters are random. The minimum and maximum values of the **X** and **Y** parameters represent the area of the game.
- **Set Timer by Function Name**: This Timer represents the time in seconds that the statue will take to change position. When this time expires, the **InitStatue** function is called to set a new position. The value of the **Time** parameter is the result of the expression "**6 – Level**". "**Level**" is a variable from the **My_Guide_GameMode** class that represents the Player Level.



BP_STATUE:
ACTOR BEGIN OVERLAP 1/2

The **ActorBeginOverlap** collision event is triggered when an Actor in the game overlaps the statue.

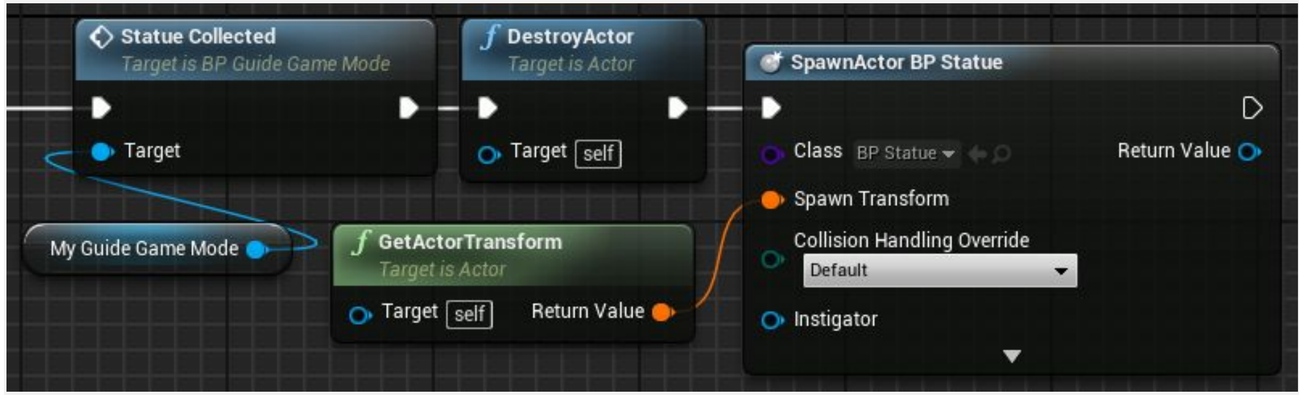
The value passed to the **Condition** input parameter of the **Branch** node is the result of the expression that checks to see if the Actor who is overlapping the statue is the Player Character and if the **GameOver** variable in the **Game Mode** is not set to “**true**”. The statue will be collected only if these two conditions are both true.



BP_STATUE:
ACTOR BEGIN OVERLAP 2/2

If the statue is collected, the **Statue Collected** function of the **My_Guide_GameMode** class is called to manage the player's score and the game Level.

After that, the current statue is destroyed and a new one is created.



SETTING THE GAME MODE

Open the **BP_Guide_GameMode** Blueprint and click the **Class Defaults** button. Set the **Default Pawn Class** property to “**ThirdPersonCharacter**” and the **HUD Class** property to “**BP_Guide_HUD**”**.**

In the **Level Editor**, click the **Settings** button and choose “**World Settings**”**.** In the **World Settings** panel, set the **GameMode Override** property to “**BP_Guide_GameMode**”**.**

The game is now ready to play.

PLAYING THE GAME

If everything is correct, when playing the game, the player will control a character and must get the statues.

There are three statues in the scenario that change position periodically.

When the player overlaps a statue, they will receive a score and the statue will reappear in another location.

There is a countdown, and the game is over when there is no time remaining.

For every five statues collected, the Player Level increases and 15 seconds are added to the time.

SUMMARY

A simple game was implemented in this lecture to show how Blueprint classes interact with each other in a game.

This game uses macros, custom events, functions, and Timers. The HUD class was used to draw some information on screen.

Also, the Game Mode was used to define the game rules and uses logic to enforce those rules.