
# LECTURE 1

## Introduction to Blueprints

## Goals

The goals of this lecture are to

- Introduce Blueprints
- Explain the Blueprint Editor interface
- Demonstrate how to create Blueprint classes
- Introduce the concept of parent-child inheritance
- Show how to place nodes in the Event Graph
- Explain the Blueprint graph execution

## Outcomes

By the end of this lecture you will be able to

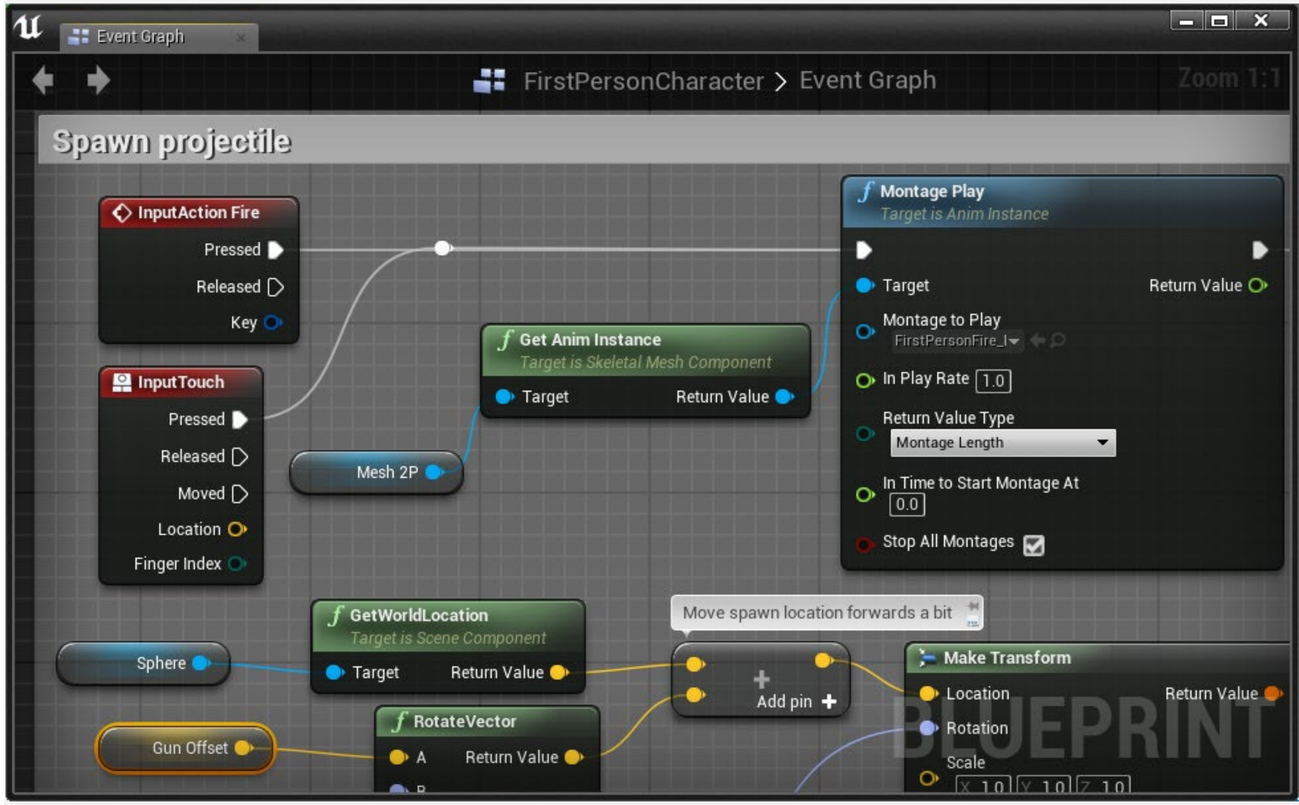
- Create Blueprint classes in the Editor
- Differentiate between Level Blueprints and Blueprint classes
- Place nodes in the Event Graph
- Understand the basic concept of parent-child inheritance
- Understand the execution flow of a Blueprint

# WHAT IS BLUEPRINT?

**Blueprint** is a visual scripting language created by Epic Games for Unreal Engine 4. It is used within the Unreal Editor to create new classes and gameplay functionality.

A script in Blueprint is represented by graphs of nodes connected by wires that define the flow of execution.

The word “Blueprint” is also used to refer to a game object created using Blueprint.



# MAJOR BLUEPRINT TYPES

## Level Blueprint

A **Level Blueprint** is a special type of Blueprint that belongs to a Level. It is used to define specific events and actions in a Level.

A Level Blueprint can be used to interact with Blueprint Actor classes and to manage some systems, such as cinematics and Level streaming.

## Blueprint Class

A **class** is the definition of data and behavior that will be used by a particular type of object. A Blueprint class can be based on a C++ class or on another Blueprint class.

A Blueprint class is used to create interactive objects for the game and can be reused in any Level.

# OBJECT, ACTOR, AND ACTOR COMPONENT CLASSES

When creating a new Blueprint class, you must define the **parent class**. All variables and actions of the parent class will be part of the new class, which is known as a **child class** or **subclass**. This concept is called **inheritance**.

Below are some parent classes:

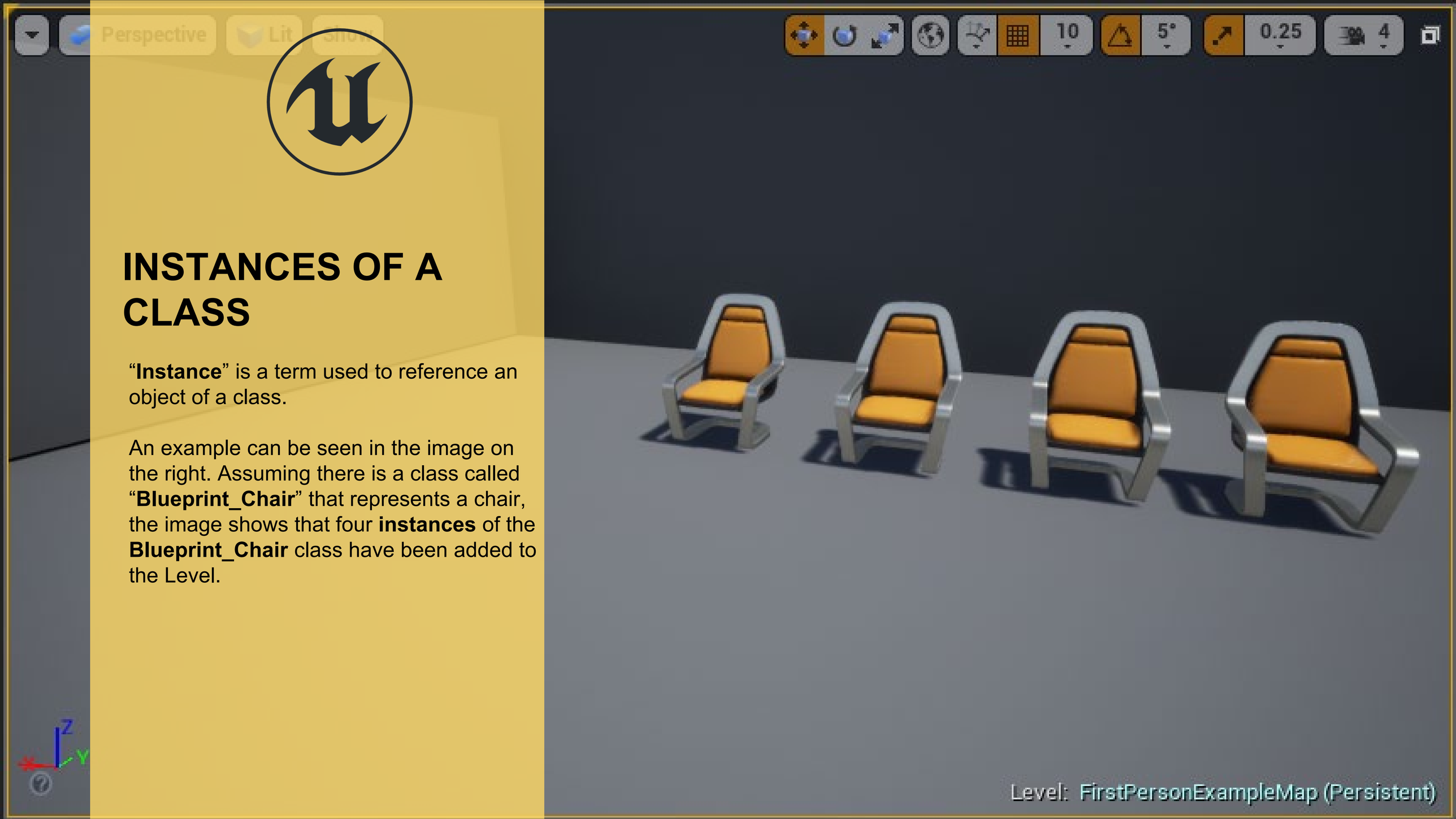
- **Object**: The base class for objects in Unreal Engine. All other classes are subclasses of the Object class.
- **Actor**: The base class used for objects that can be placed or spawned into a Level.
- **Actor Component**: The base class for components that define reusable behavior that can be added to Actors.

Therefore, every Actor is an Object, but not all Objects are Actors. For example, an Actor Component is an Object, but it is not an Actor.

# INSTANCES OF A CLASS

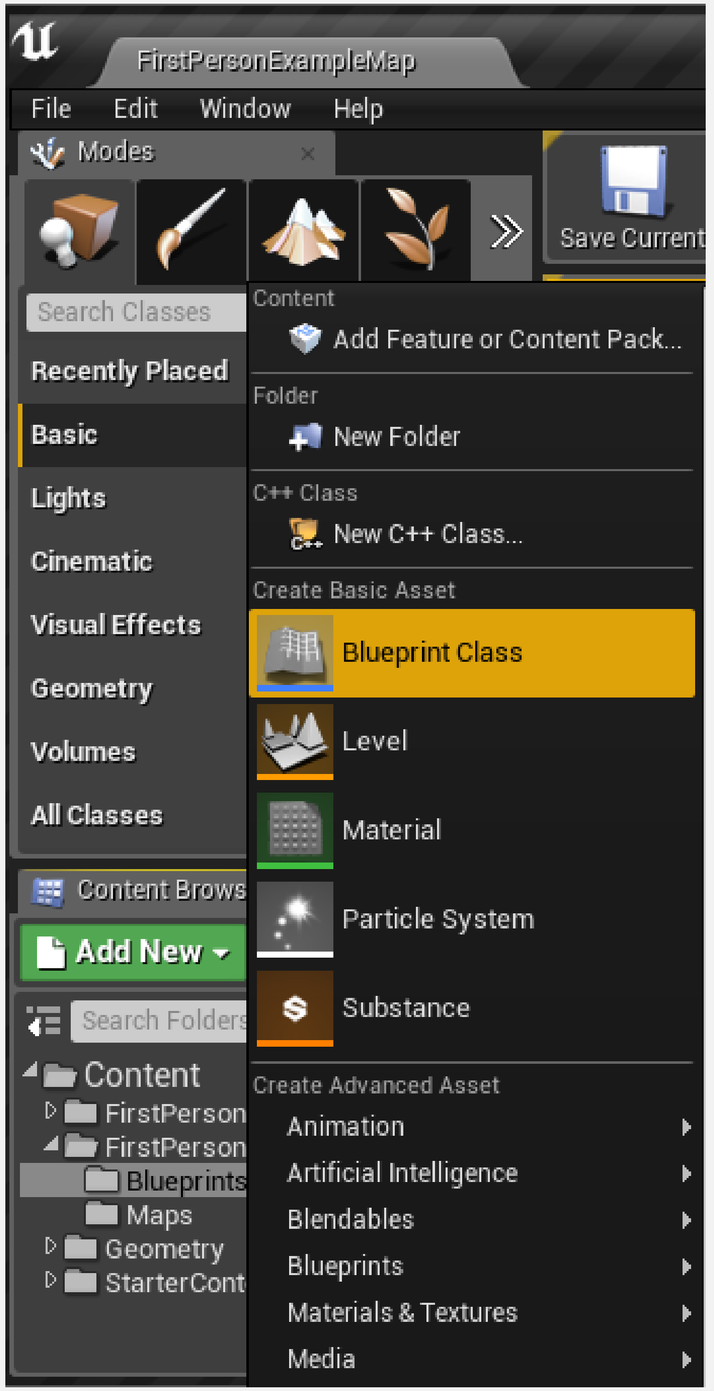
"**Instance**" is a term used to reference an object of a class.

An example can be seen in the image on the right. Assuming there is a class called "**Blueprint_Chair**" that represents a chair, the image shows that four **instances** of the **Blueprint_Chair** class have been added to the Level.



# CREATING BLUEPRINT CLASSES: THE CONTENT BROWSER

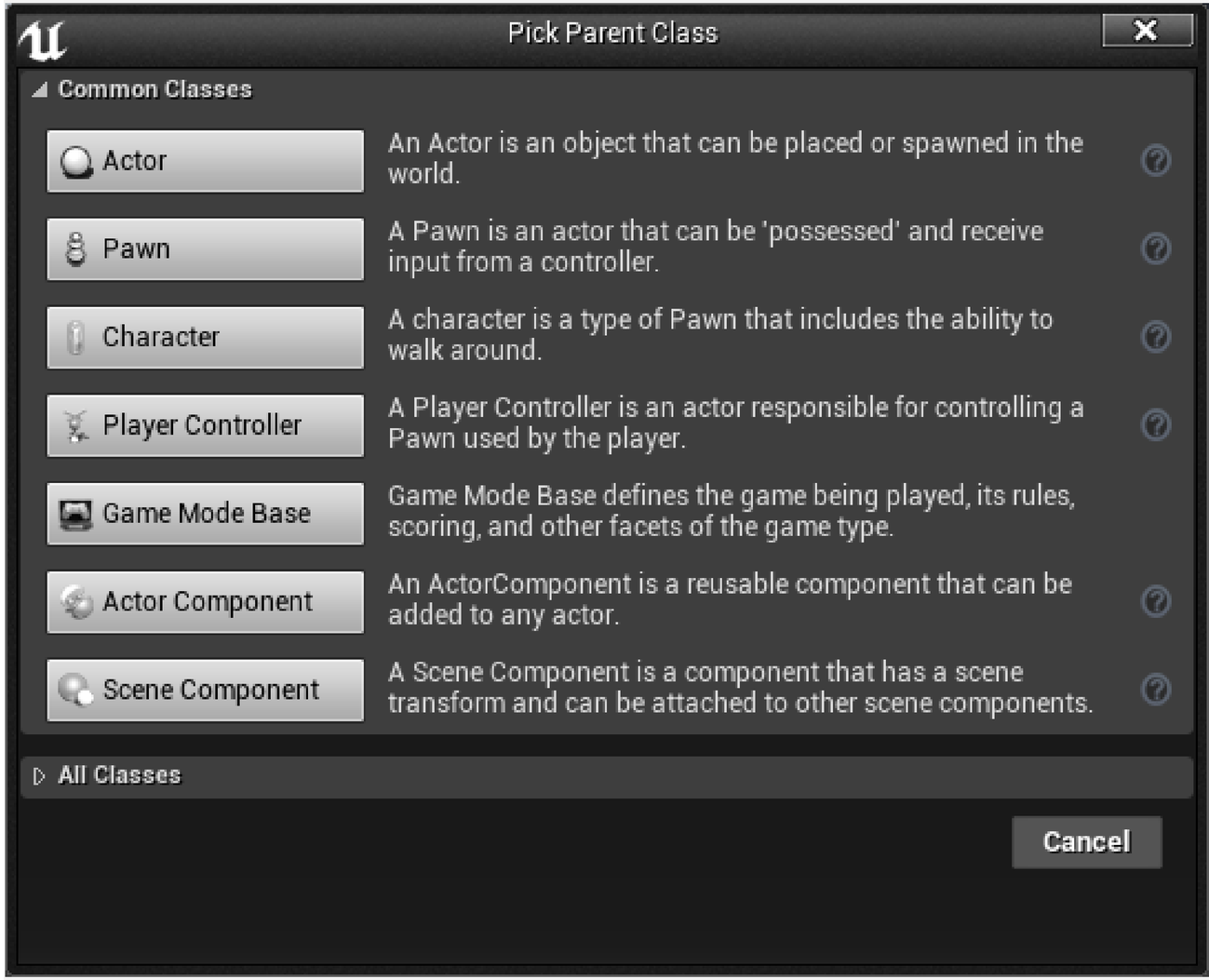
To create a new Blueprint class in the **Content Browser**, click the green **Add New** button and select “**Blueprint Class**”.



# CREATING BLUEPRINT CLASSES: PICK PARENT CLASS WINDOW

The next step is to choose the parent class of the new Blueprint.

The image on the right shows the most commonly used classes listed in the **Pick Parent Class** window. Other parent classes can be found by expanding the **All Classes** option.

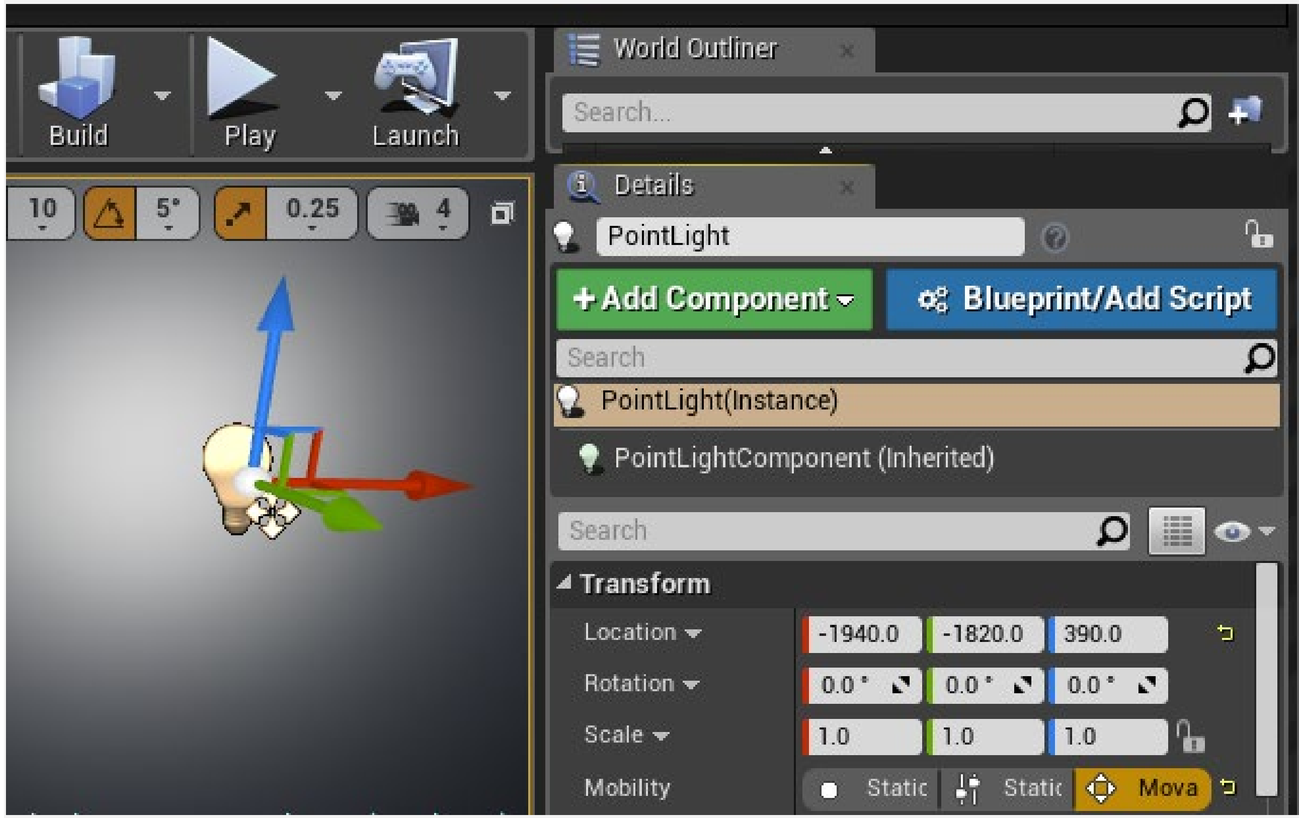


# CREATING BLUEPRINT CLASSES: ACTORS IN THE LEVEL

It is possible to create Blueprint classes based on Actors that have been placed in the Level.

To do this, select the Actor in the Level and click the blue **Blueprint/Add Script** button in the **Level Editor**'s **Details** panel.

The class of the Actor will be used as the parent class of the new Blueprint.



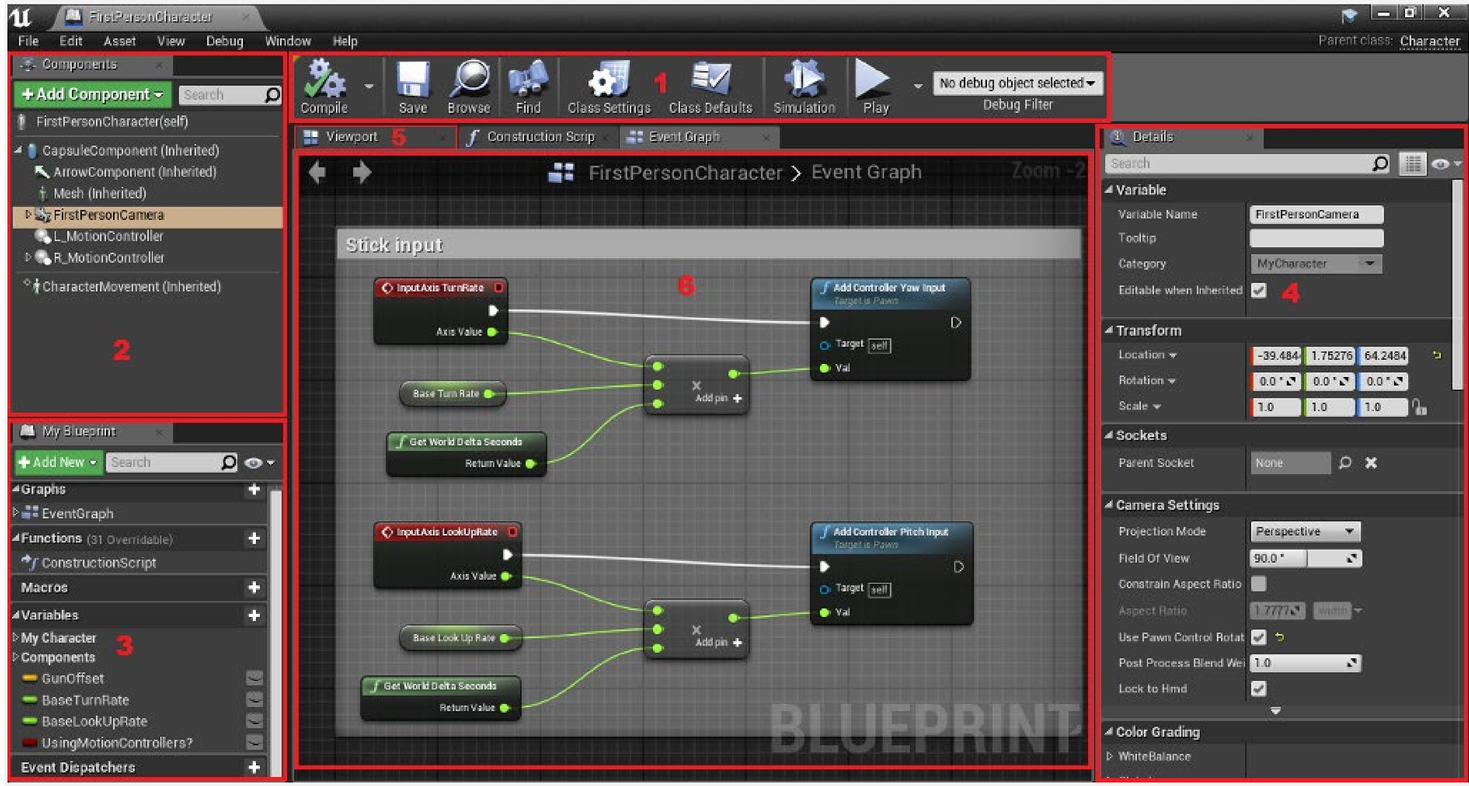
# BLUEPRINT EDITOR INTERFACE

# BLUEPRINT EDITOR: OVERVIEW

To open the **Blueprint Editor**, double-click on a Blueprint or right-click and choose "**Edit...**".

The main parts of the Blueprint Editor, highlighted in the image to the right, are as follows:

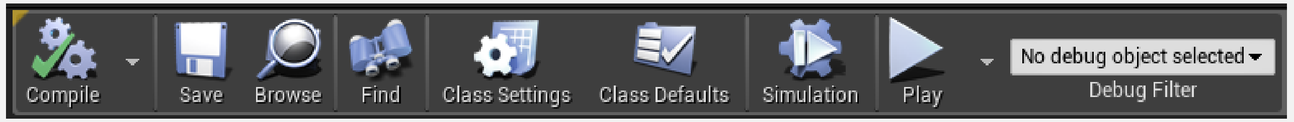
1. Toolbar
2. Components panel
3. My Blueprint panel
4. Details panel
5. Viewport
6. Event Graph



# BLUEPRINT EDITOR: TOOLBAR

The **Toolbar**, located at the top of the Editor, has a few essential buttons for editing Blueprints:

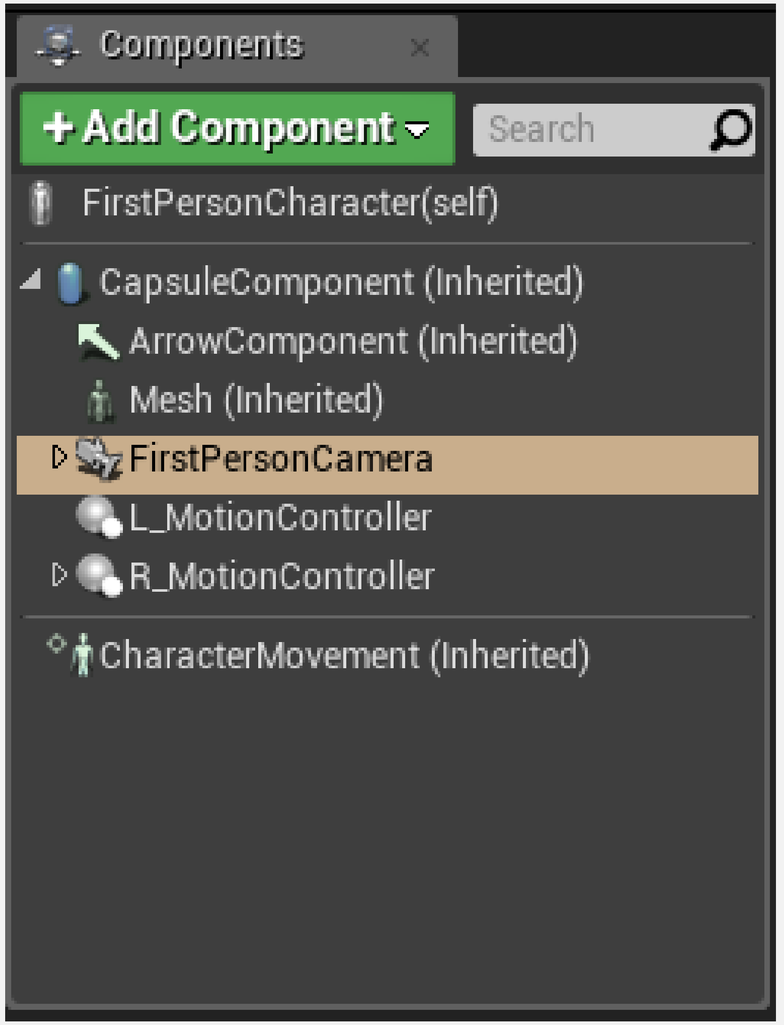
- **Compile**: “Compiles” the Blueprint, necessary for validating the code and applying modifications.
- **Save**: Saves all changes made to the current Blueprint to disk.
- **Browse**: Shows the current Blueprint class in the Content Browser.
- **Find**: Searches nodes within a Blueprint.
- **Class Settings**: Opens Blueprint properties.
- **Class Defaults**: Allows for modification of the initial values of the Blueprint variables.



# BLUEPRINT EDITOR: COMPONENTS PANEL

In the **Components** panel, various types of components can be added to the current Blueprint.

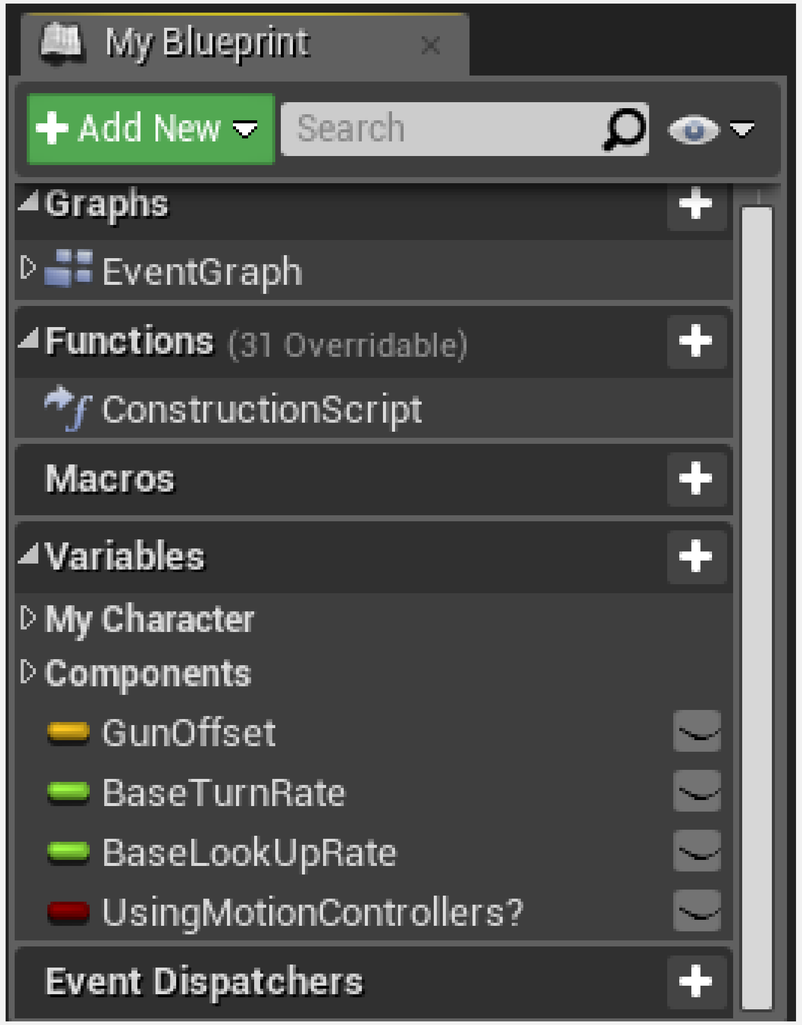
Examples of components are Static Meshes, lights, sounds, and geometric volumes used in collision tests.



# BLUEPRINT EDITOR: MY BLUEPRINT PANEL

The **My Blueprint** panel is used to manage the variables, macros, functions, and graphs of the Blueprint.

It is separated into categories, and each category has a “**+**” button for adding new elements.

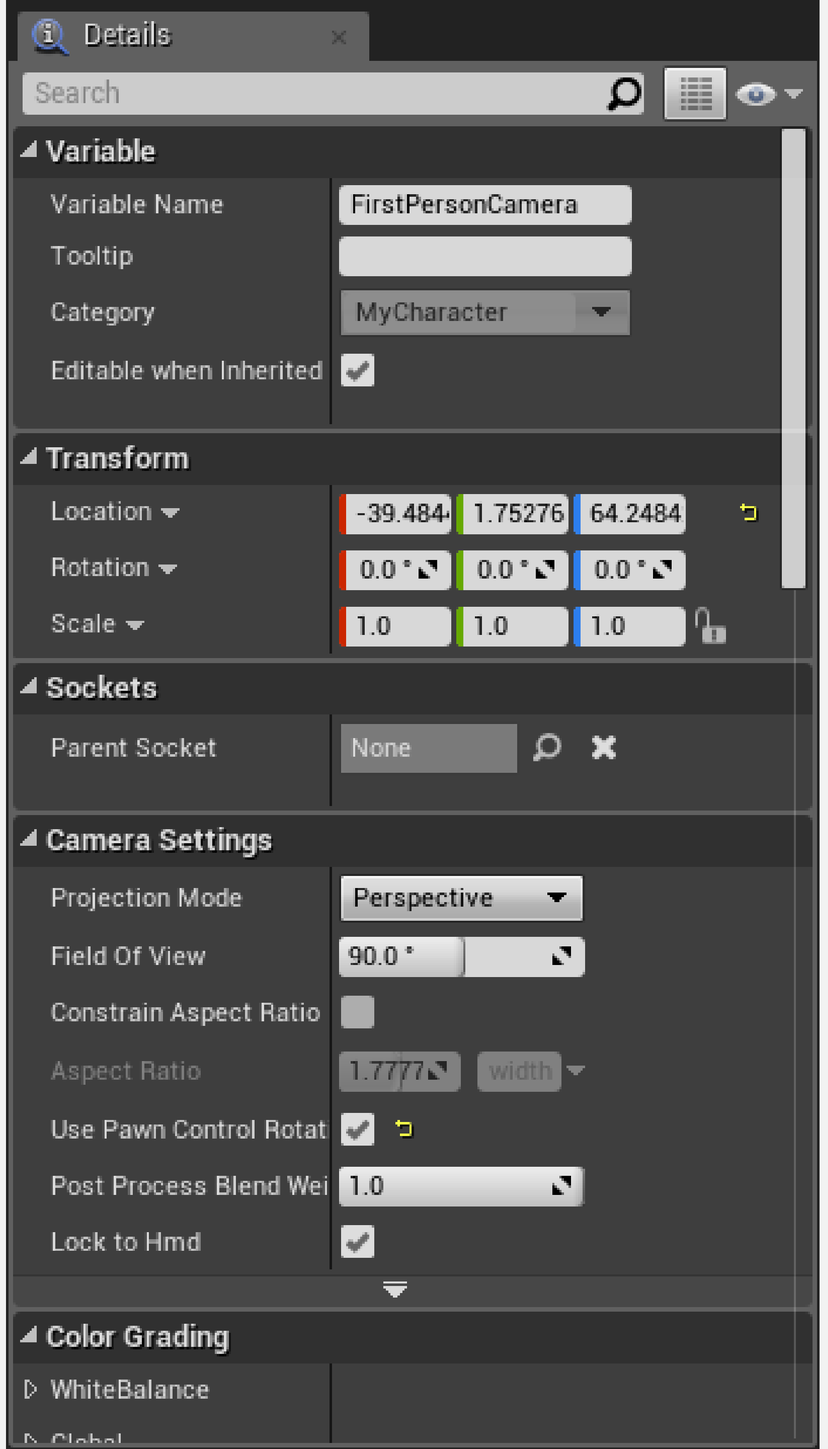


# BLUEPRINT EDITOR: DETAILS PANEL

The **Details** panel shows the properties of the currently selected element of the Blueprint class, which can be a variable, function, or component.

These properties are organized into categories, and their values can be modified.

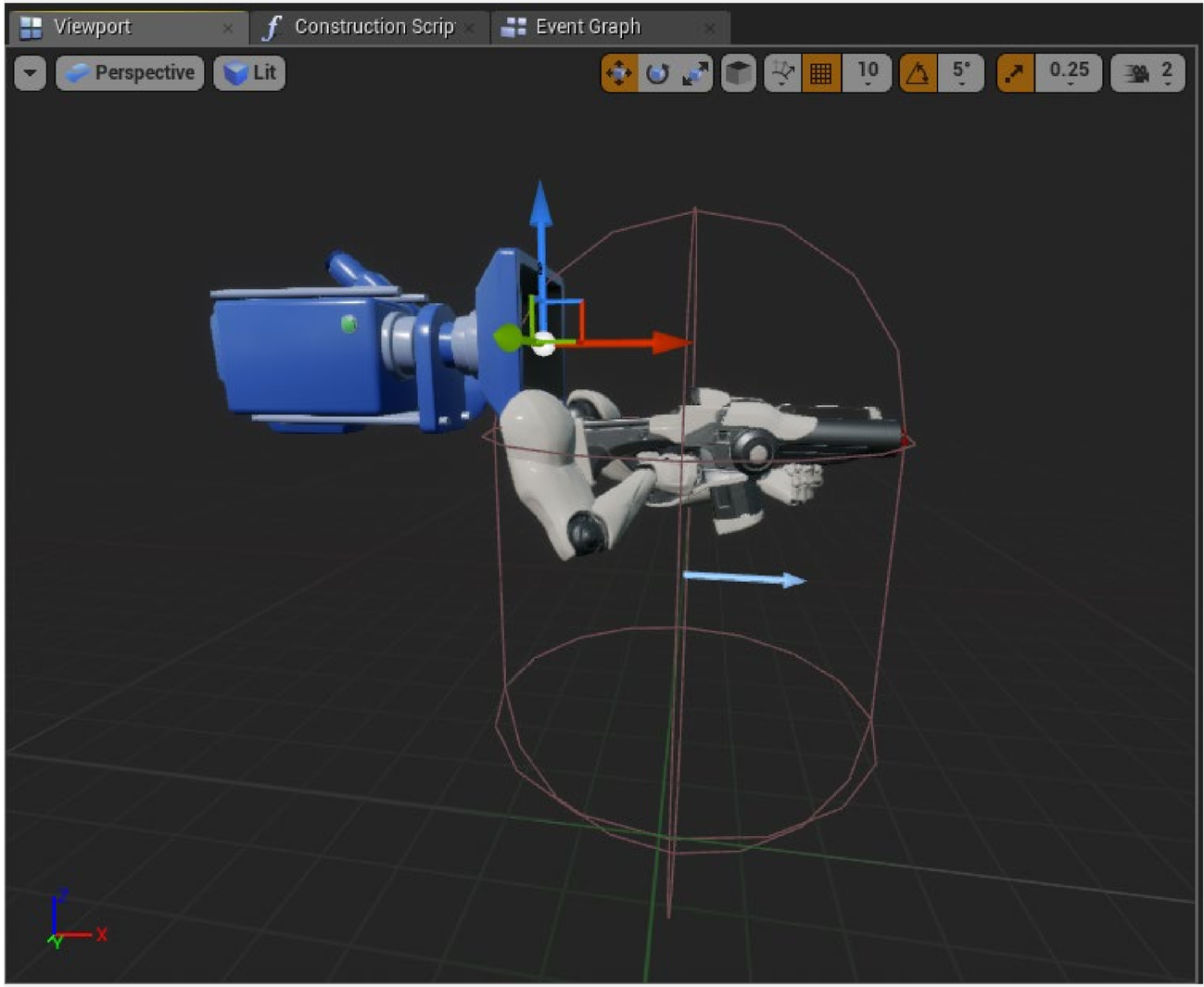
At the top of the panel, there is a search box that can be used to filter the properties.



# BLUEPRINT EDITOR: VIEWPORT

The **Viewport** contains the visual representation of the components that are part of the Blueprint.

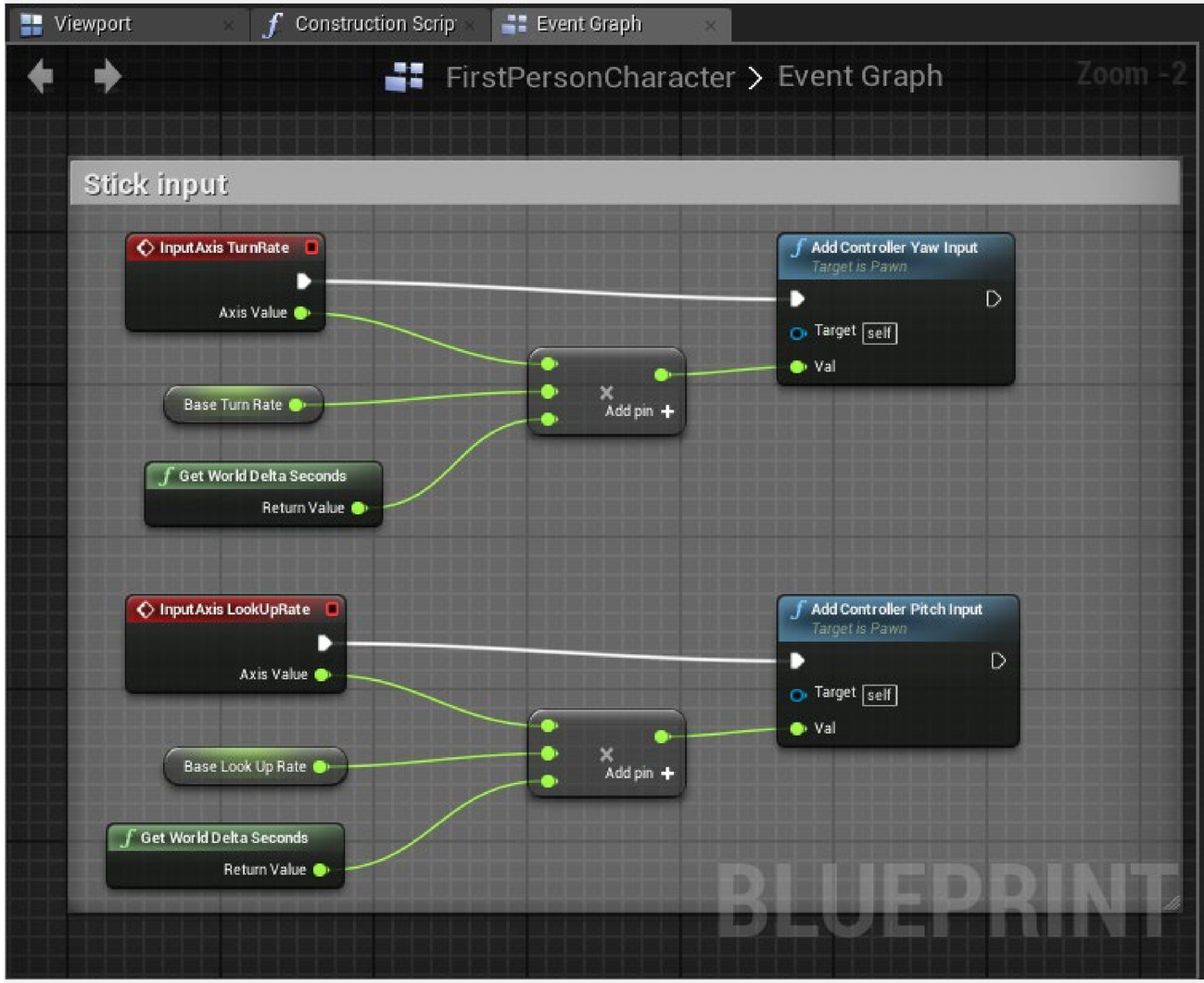
The components can be manipulated in the Viewport using the transformation widgets in the same way as in the Level Editor.



# BLUEPRINT EDITOR: EVENT GRAPH

The **Event Graph** is where the gameplay logic that determines a Blueprint class's behavior during gameplay appears.

The Event Graph contains events and actions represented by a node graph.

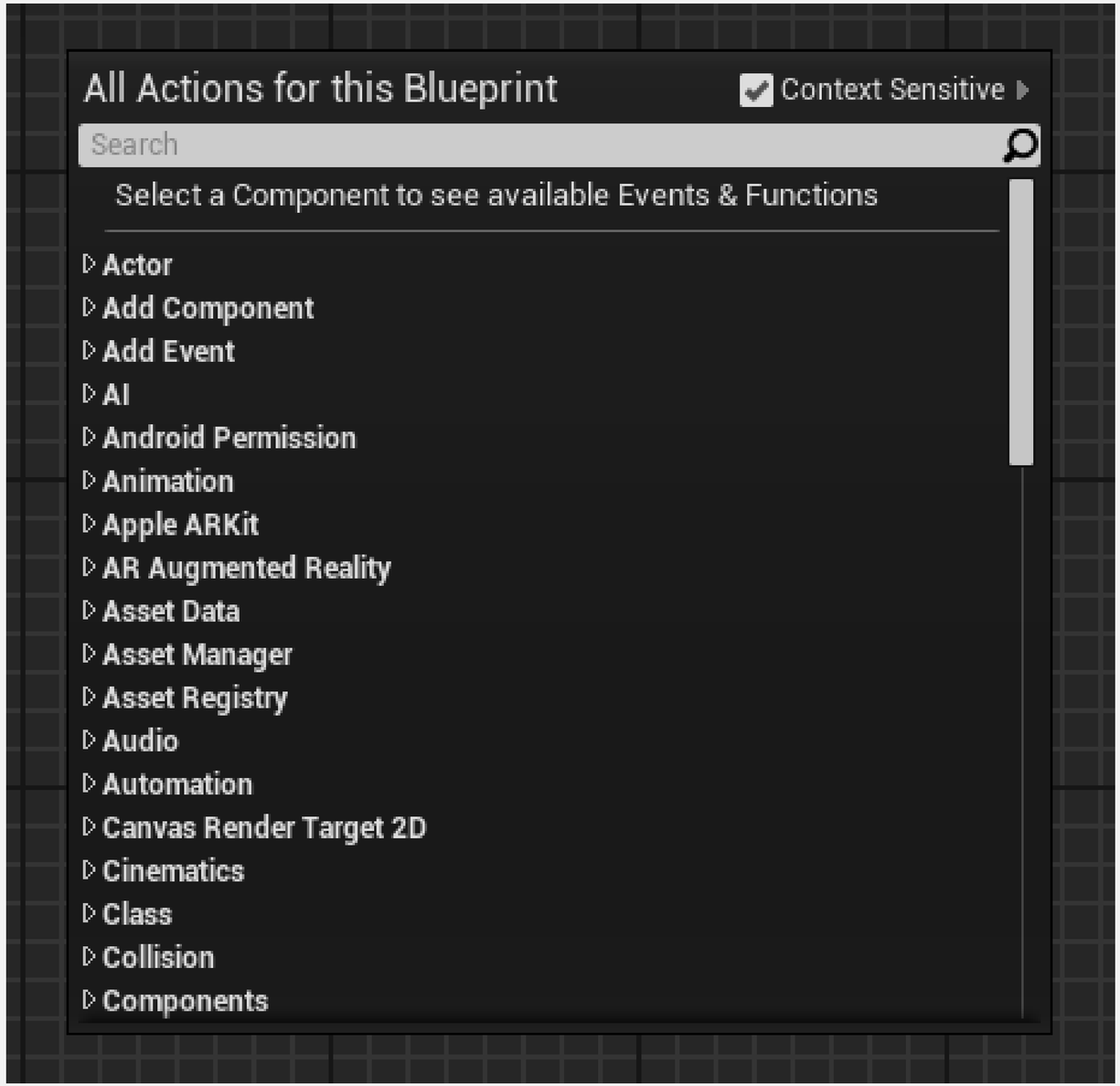


# PLACING NODES

# PLACING NODES: CONTEXT MENU

The **context menu** is used to add nodes to the graph. The nodes represent variables, operators, functions, and events.

To open the **context menu**, right-click in an empty area of the **Event Graph** or drag a wire from a pin of a node and release the mouse button.

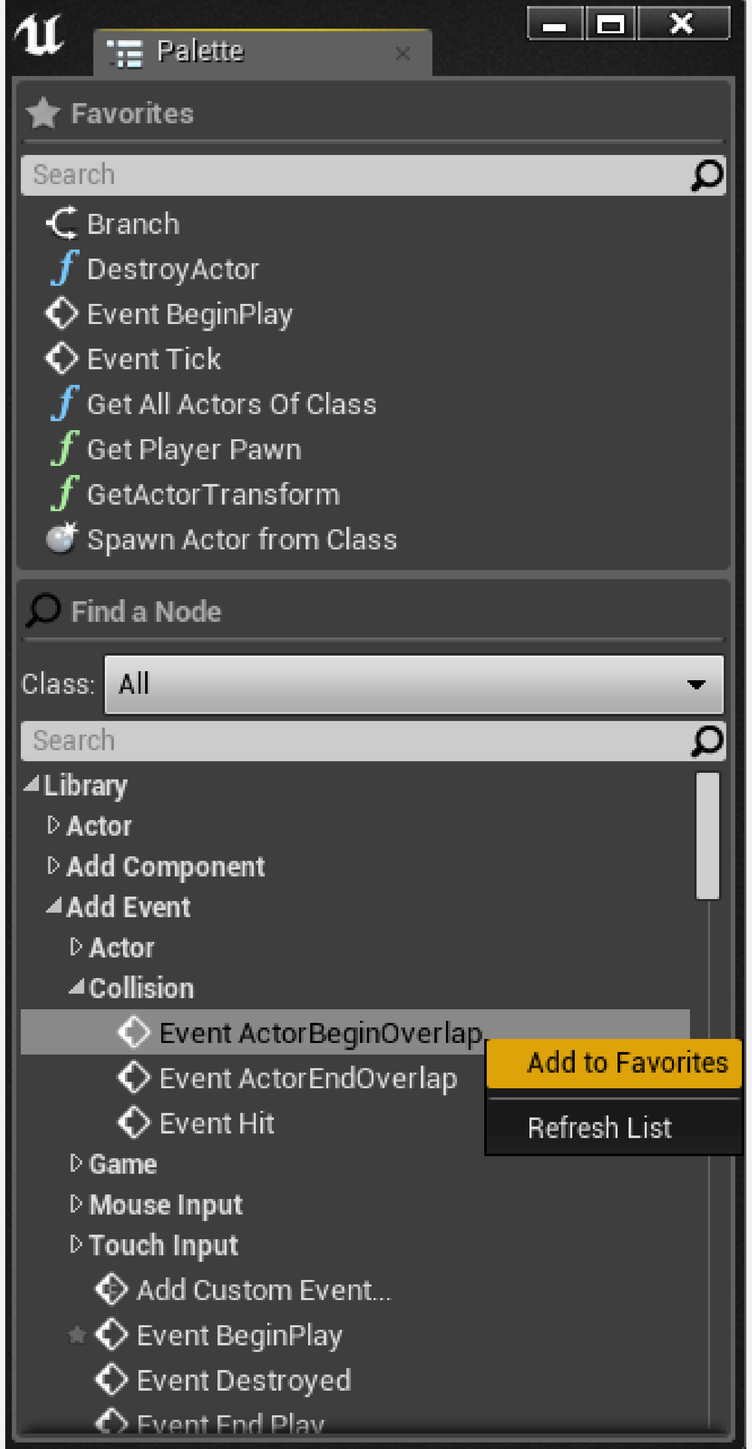


# PLACING NODES: PALETTE PANEL

There is a **Palette** panel that can be opened by going to **Window > Palette** in the **Blueprint Editor**.

The Palette panel contains a list of all nodes that can be used in Blueprints.

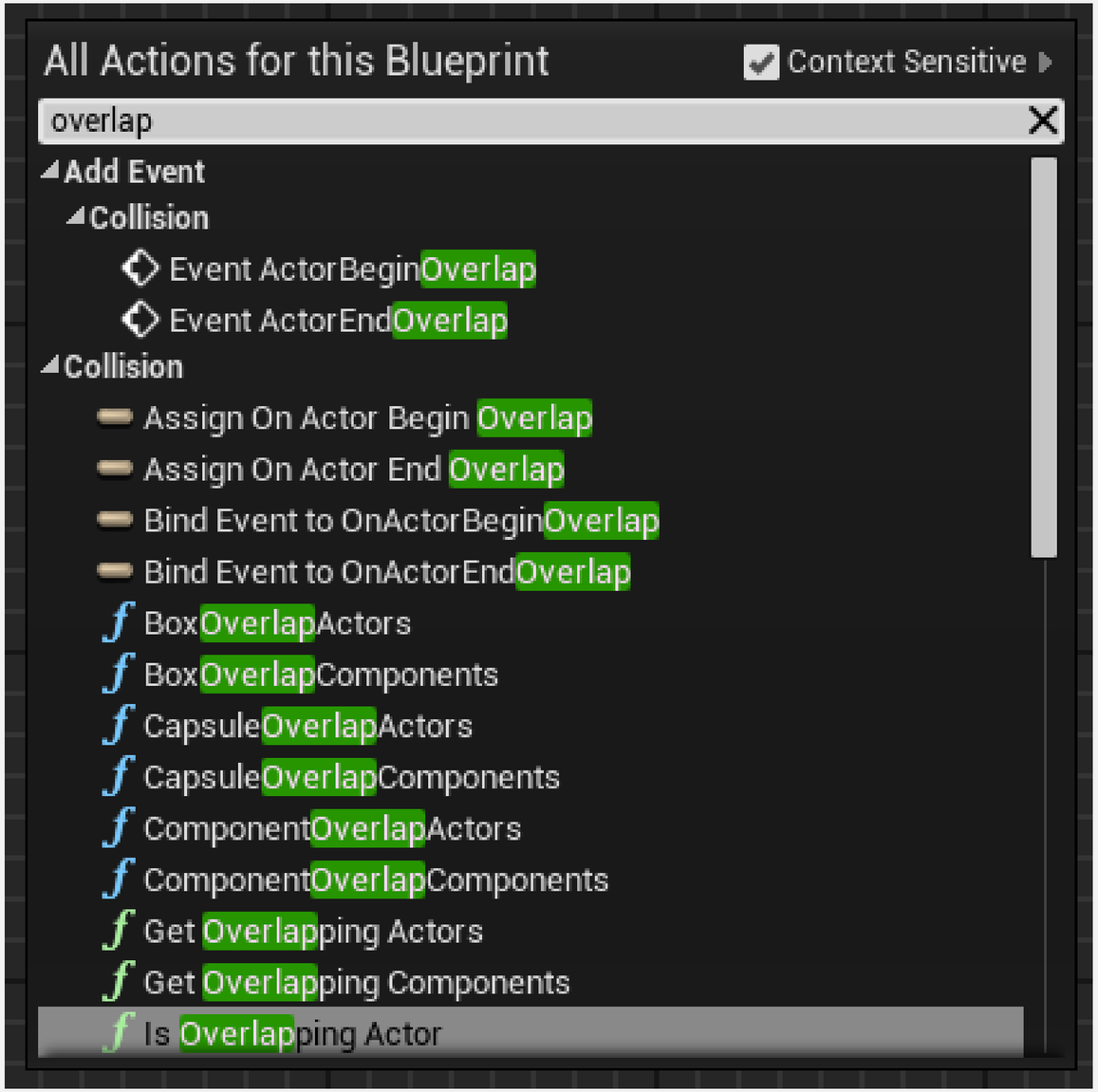
At the top of the panel there is a **Favorites** section that shows the favorite and most frequently used nodes. To set a node as a favorite, right-click on it in the **Palette** panel and choose “**Add to Favorites**”.



# PLACING NODES: SEARCH BARS

The **context menu** has **a search bar** that filters the nodes list when the user is typing.

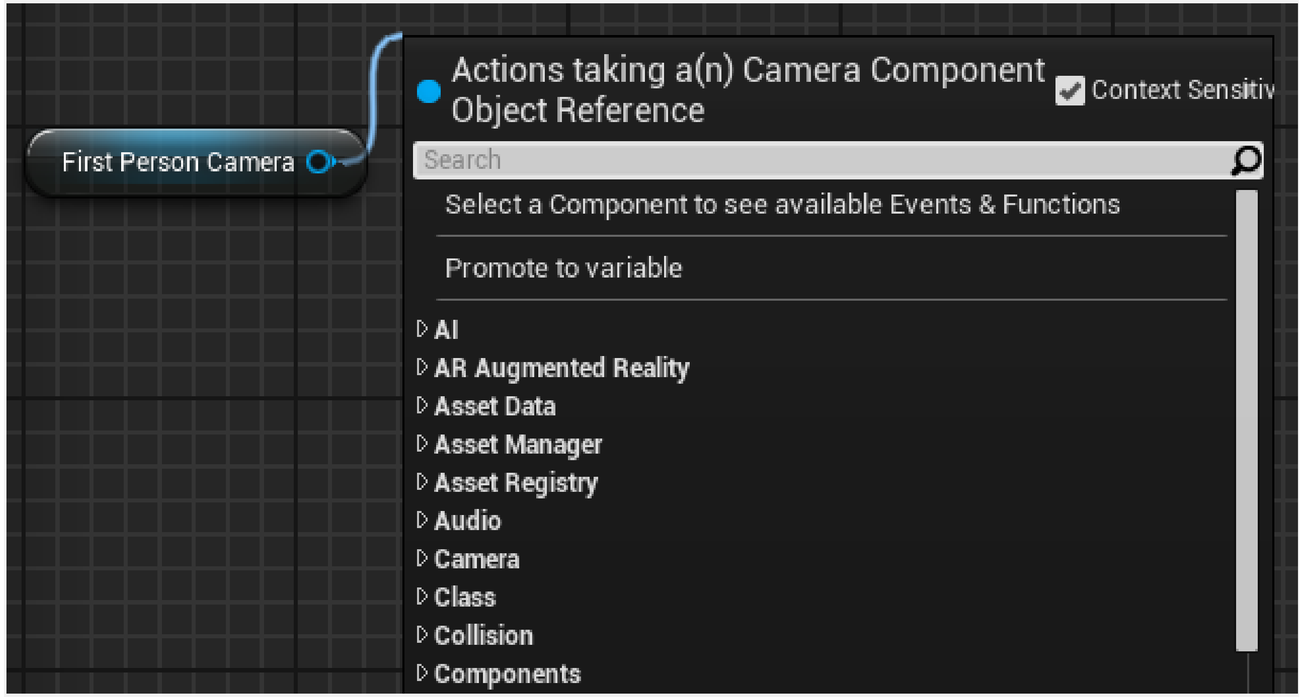
The **Palette** panel has **two search bars**, one for the **Favorites** section and one for the **Palette** list.



# PLACING NODES: CONTEXT SENSITIVE

At the top of the context menu, there is a check box named “**Context Sensitive**”. If it is checked, the list of nodes will be filtered to actions that can be used in the current context.

If the context menu was opened by right-clicking in the Event Graph, the context will be the current Blueprint class. If it was opened by dragging a wire from a pin of a node, the context will be the pin type.



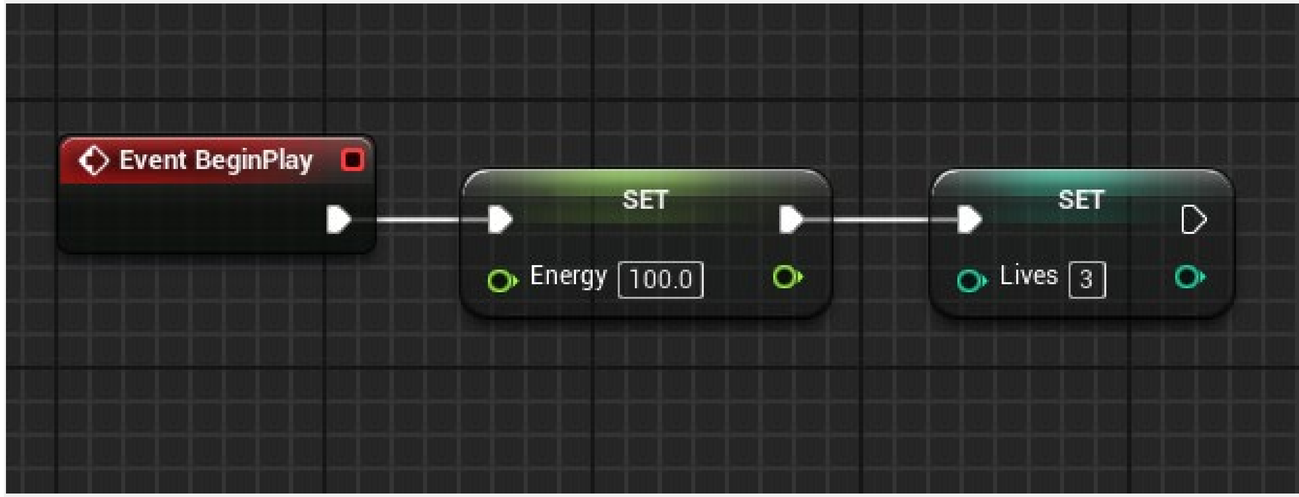
# NODES, PINS, AND WIRES

# BLUEPRINT GRAPH EXECUTION

The **execution** of the nodes of a Blueprint starts with a red **Event** node and follows the **white wire** from left to right until it reaches the last node. After that, it moves on to another Blueprint event that has been triggered.

The white pins of nodes are called **execution pins**. The other colored pins are the **data pins**.

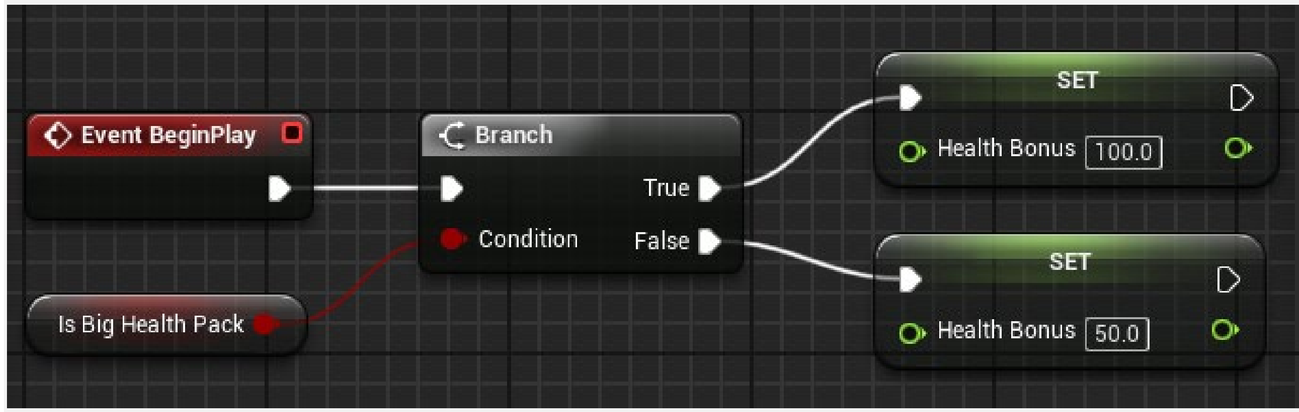
In the image on the right, values are assigned to the **Energy** and **Lives** variables when the **BeginPlay** event is triggered.



# EXECUTION PATHS

There are some nodes that control the flow of execution of the Blueprint. These nodes determine the **execution path** based on conditions.

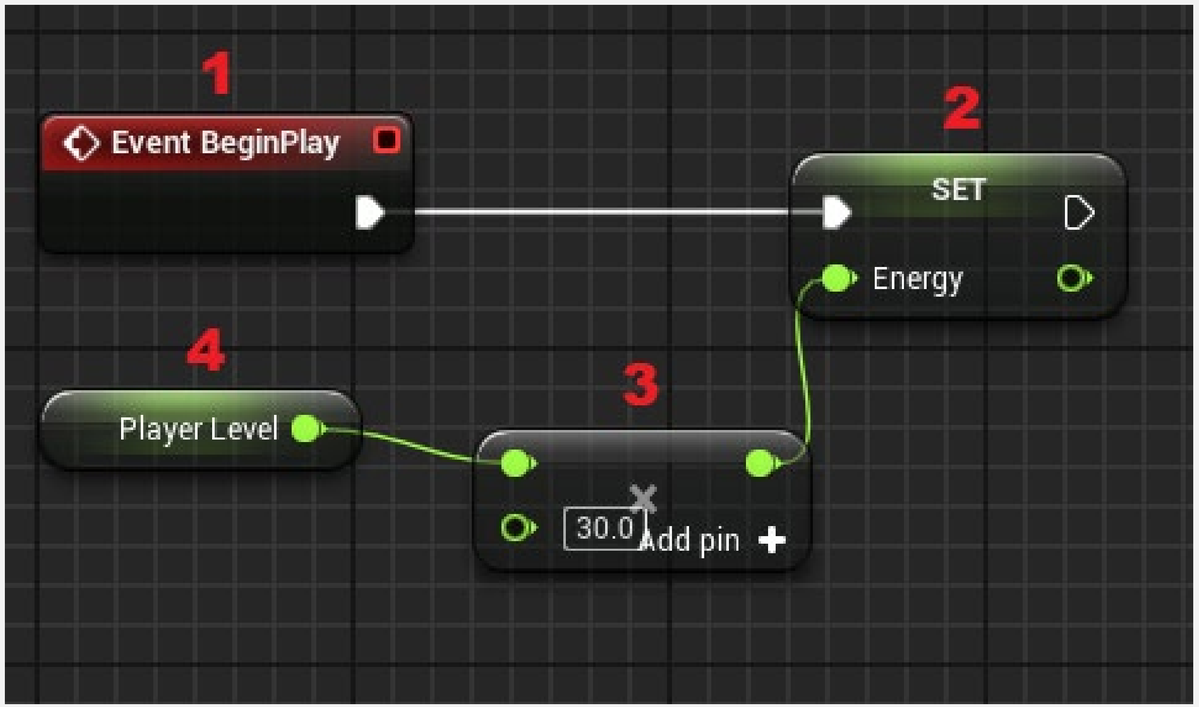
For example, the image on the right shows a **Branch** node that is using as a condition the value of the Boolean variable **Is Big Health Pack**. If the value is “**true**”, then the execution will continue on the **True** pin. If the value is “**false**”, the execution will continue on the **False** pin.



# DATA WIRES

When a node with data pins runs, it fetches the required data using the **data wires** before completing its execution.

In the image on the right, the execution starts with the **BeginPlay** event. The **Set** node assigns a new value to the **Energy** variable, but this value must be obtained using the data wire that is connected to a multiplication node that will need to get the value of the **Player Level** variable using another data wire.



# SUMMARY

This lecture introduced the basic concepts involved in a Blueprint. It explained the difference between Level Blueprints and Blueprint classes and the concept of parent-child inheritance.

It showed the various parts of the Blueprint Editor and explained the flow of execution in a Blueprint graph.