HOUR 16, LECTURE 1

Blueprint Classes:
Blueprints as Assets

INTRODUCTION

Level Blueprints are great for creating Level-specific event sequences, but they are bound to the Level they are created in.

Blueprint classes can be used in any Level and even in other Blueprints. They are treated like other assets and are created, stored, and edited in the Content Browser.

This lecture introduces you to working with Blueprint classes.

LECTURE GOALS AND OUTCOMES

Goals

The goals of this lecture are to

- Learn about Blueprint classes
- Explore the concept of inheritance
- Learn about components

Outcomes

By the end of this lecture you will be able to

- Create Blueprint classes based on existing classes
- Add components to Blueprint Actors

UNDERSTANDING CLASS INHERITANCE

The Building Blocks of Object-Oriented Programming

Terminology

Class: A class is a *template* for an object that contains all the information needed to create the object.

Object: An object is the basic building block of an object-oriented system. It is a self-contained set of code, data, variables, and functions. Objects can be as simple as a single variable or as complex as a game character.

Actor: Actors are a *UE4-specific* class that defines a special kind of object that can be placed in a Level. Actors are the base class for all classes that can be placed in the Level, including lights, cameras, and Static Meshes.

Parent class: A parent class is a class that a child class inherits from. It can also be called a *superclass* or *base class*.

- For example, the LightActor class is the parent class of the PointLightActor class.
- Each parent class can have *many* child classes.

Child class: A child class is a class that inherits from another class. It can also be called a *subclass*.

- For example, the LightActor class is a child class of the Actor class.
- Each child can only have a *single* parent class.

Inherit: To inherit is to make a child class from a class. This is also called *deriving*.

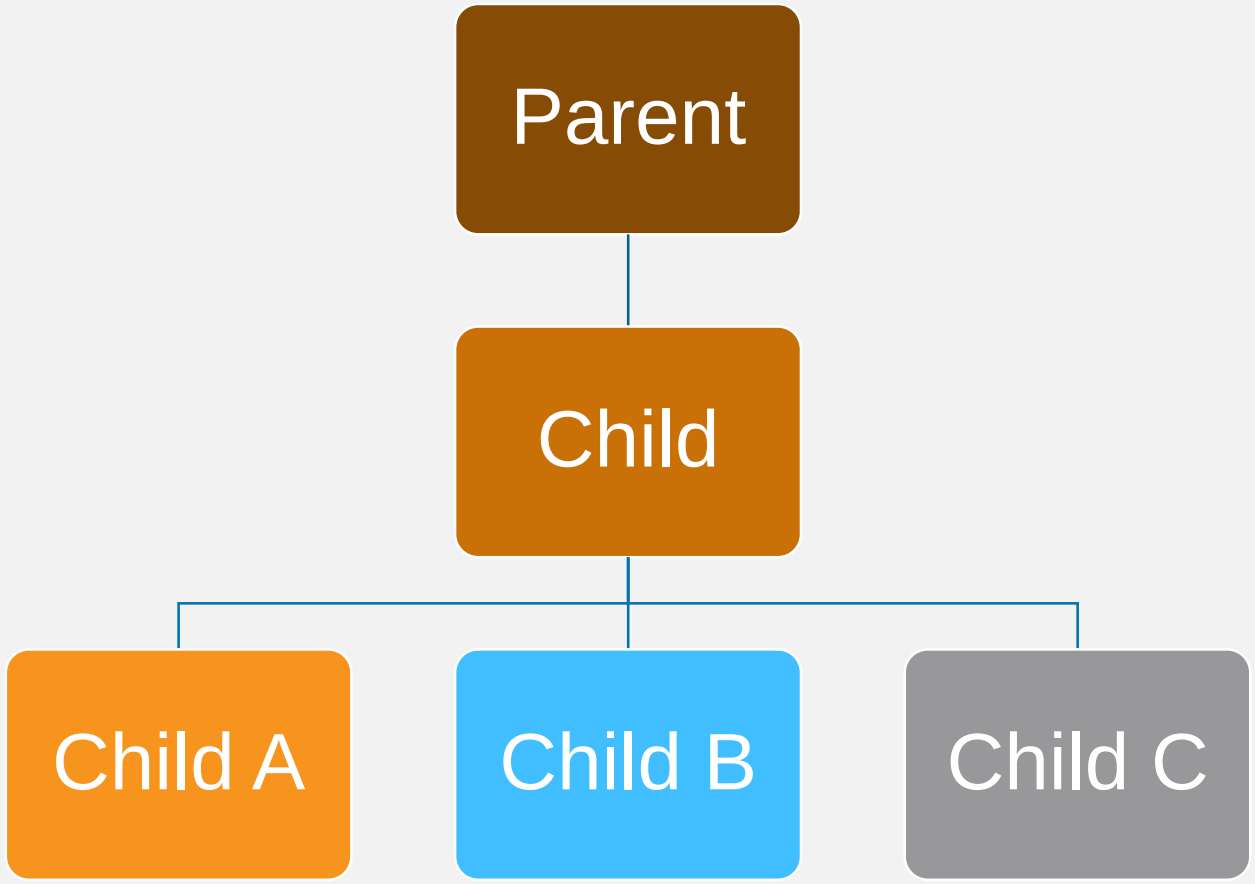
Override: To override is to modify the properties or functionality of a parent class.

INHERITANCE

A child class inherits all the variables, components, functions, and events of the parent class.

The child class can then override the settings and functions of its parent, changing the fundamental behavior.

A child class can also implement new functionality in addition to what it inherits from its parent class.

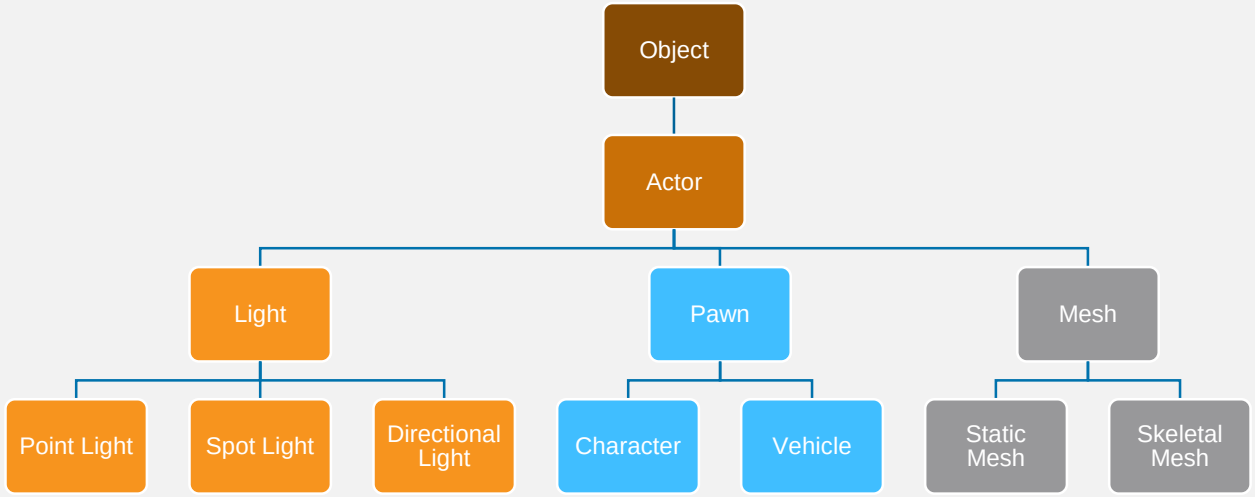


INHERITANCE

UE4 uses inheritance in almost all aspects of its design.

The Actor class is the base class for all objects that can be placed or spawned in the Level.

Lights, Pawns, and meshes are all child classes of the Actor superclass.



Inheritance

Objects can be modified after spawning, but they will always spawn with the values defined by their class.

If you change a property on a parent class, any child classes will also inherit that change—unless the child class has overridden that functionality or property, in which case the child’s implementation will be used.

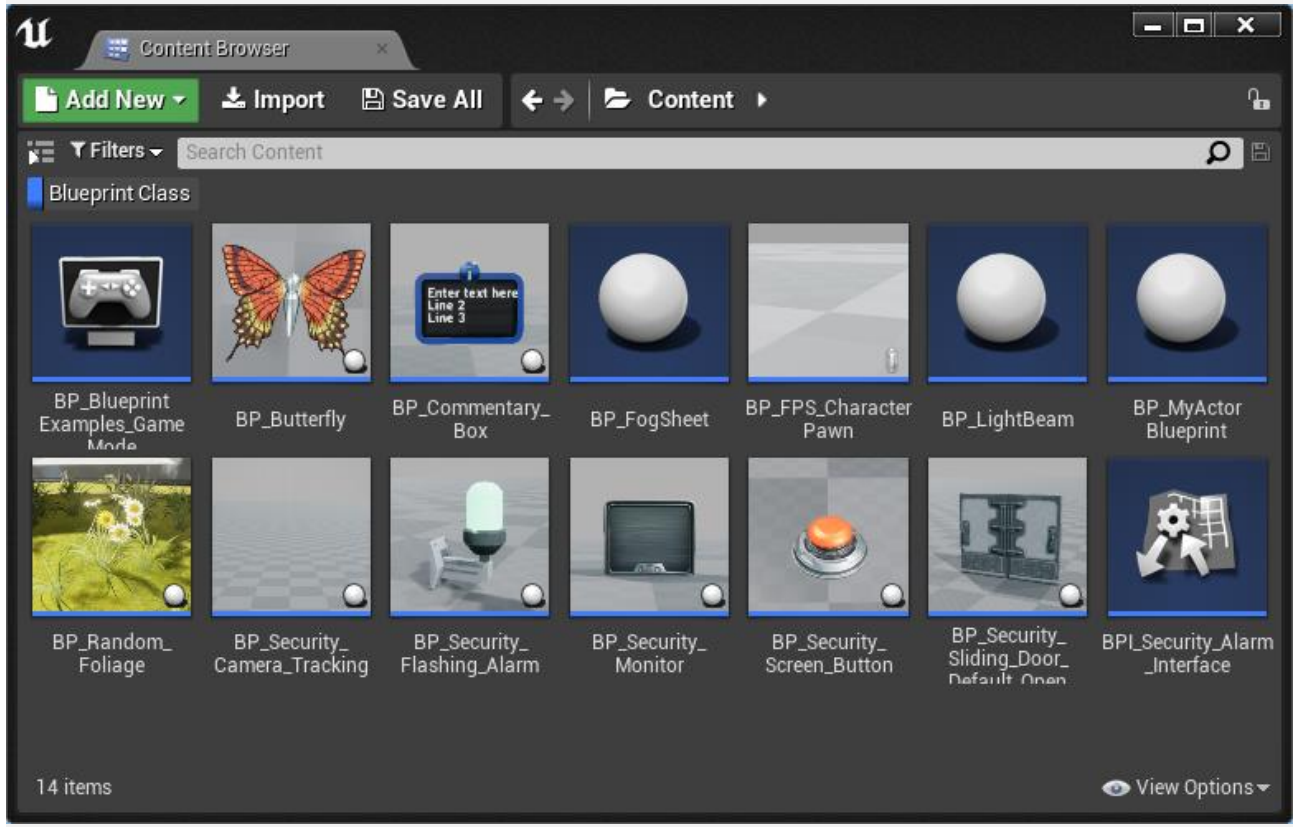
CREATING A BLUEPRINT CLASS

BLUEPRINT CLASSES

Blueprint classes are assets that live in the Content Browser. They exist entirely in your project's content folder.

Blueprint classes can inherit from any number of parent classes such as Actor, Object, and Game Mode.

The Blueprint class's parent class determines what properties and capabilities the Blueprint class will have by default.

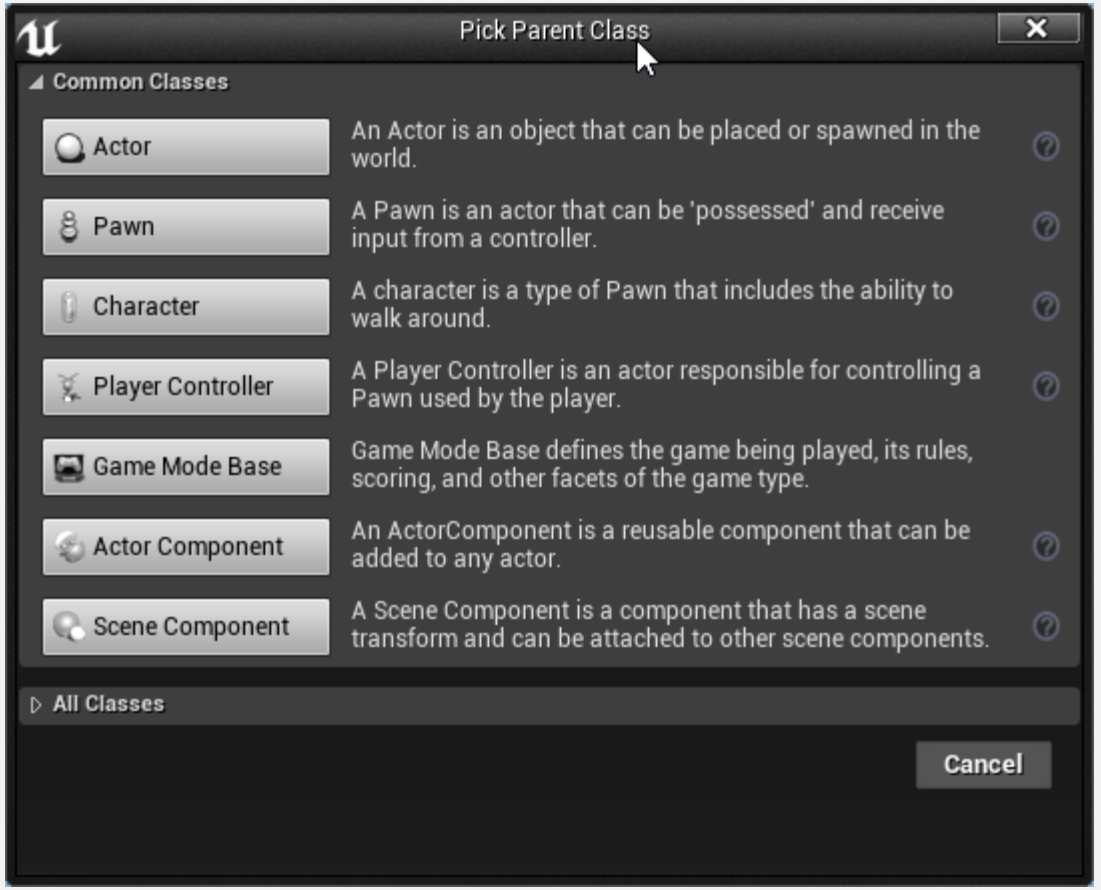


CREATING BLUEPRINT CLASSES

To create a Blueprint class, use either the Add New button or the context menu that appears when you right-click in the Content Browser. Select Blueprint Class from the list.

You will be presented with the Pick Parent Class dialog. Here you can select the class that your Blueprint will inherit from.

The most common types are listed at the top. All available parent classes are listed in the All Classes list.



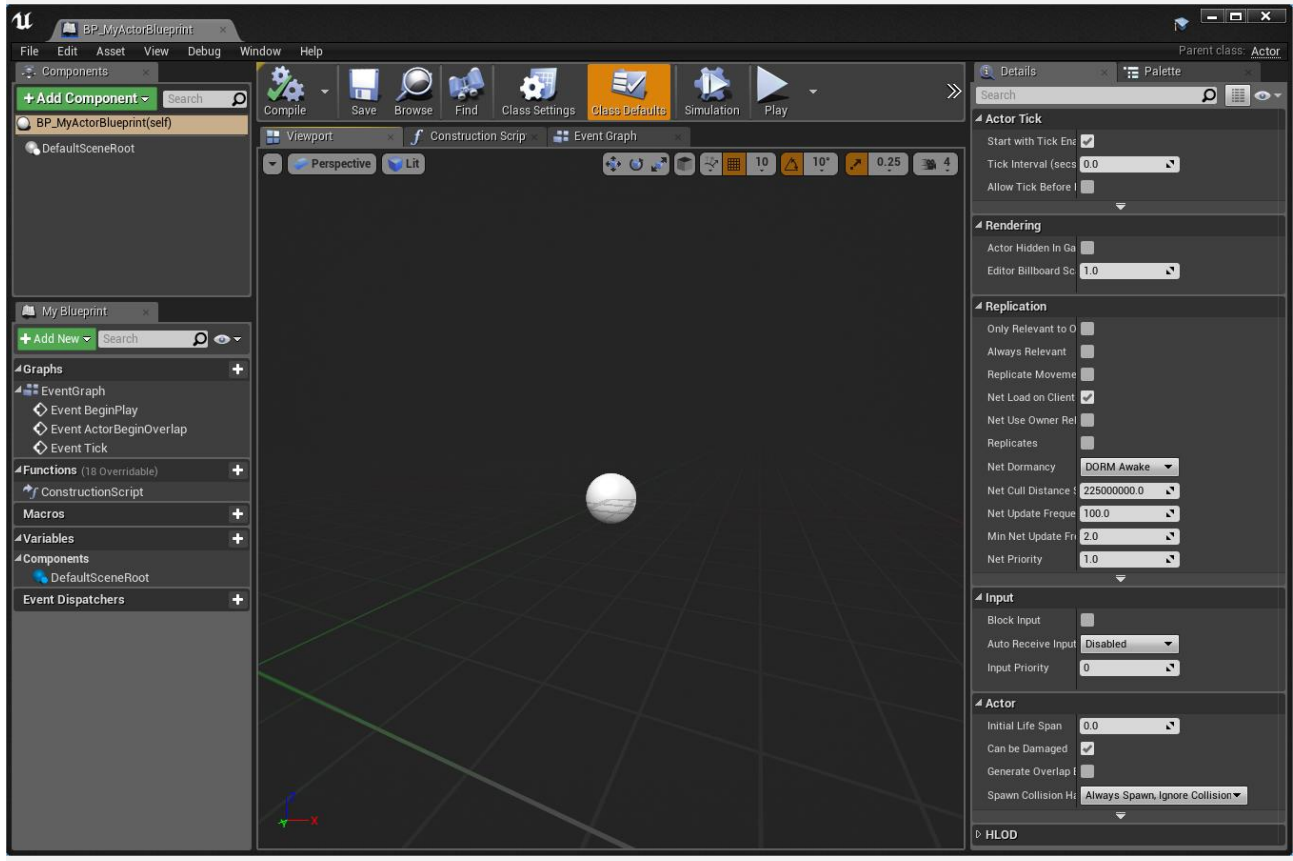
ACTOR BLUEPRINT CLASSES

Actor Blueprints can be placed in a Level.

By default, Actor Blueprints contain almost nothing but a blank scene component and a 3D transform.

Actor Blueprints add additional functionality to the Blueprint Editor interface not found in Level Blueprints:

- Components panel
- Viewport panel
- Construction Script Graph panel



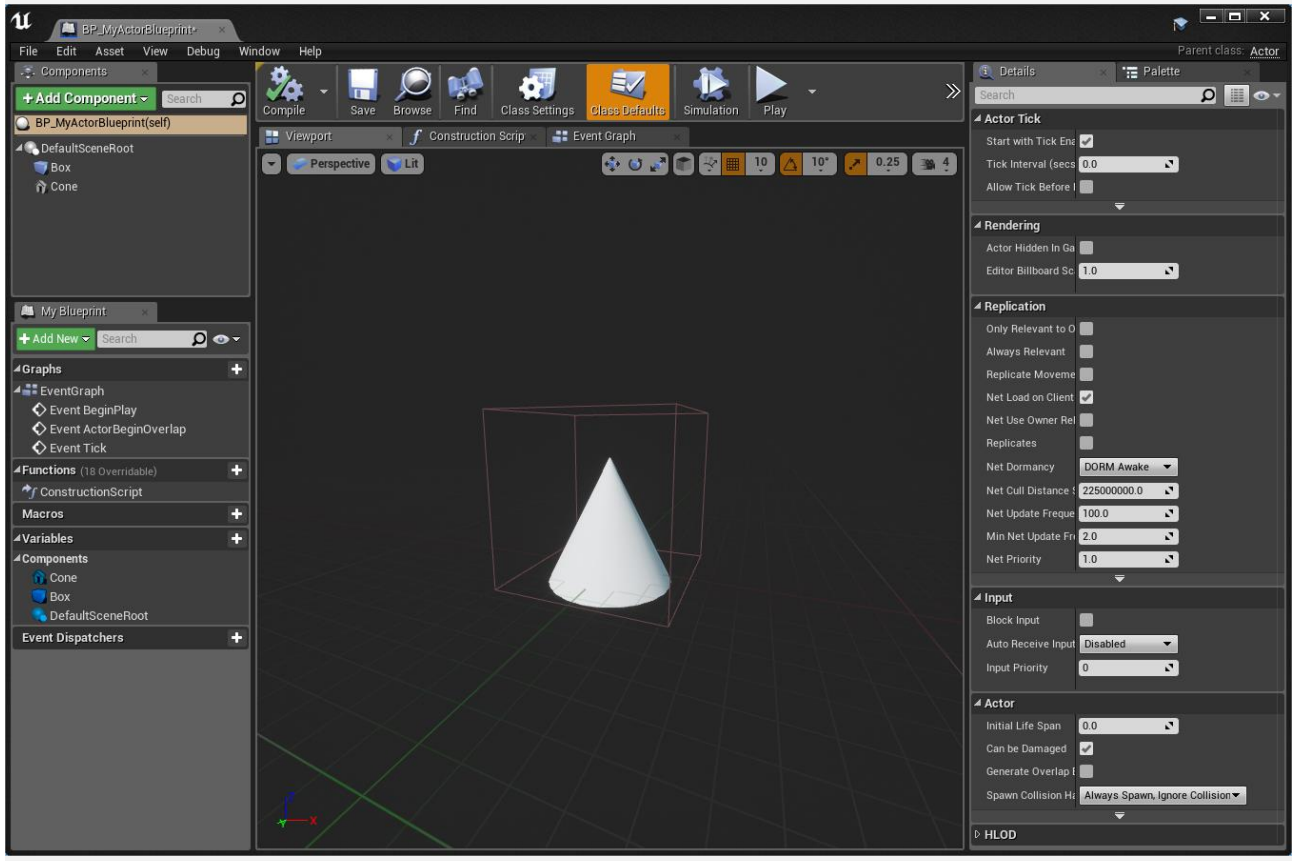
ADDING COMPONENTS

To add things like Static Meshes, lights, and collision volumes, you use the Add Component button in the Components panel.

Like the World Outliner panel, the Components panel allows you to rearrange, rename, and build hierarchies of components.

Components also appear as variables in the My Blueprint panel, allowing you to access them from the Event Graph.

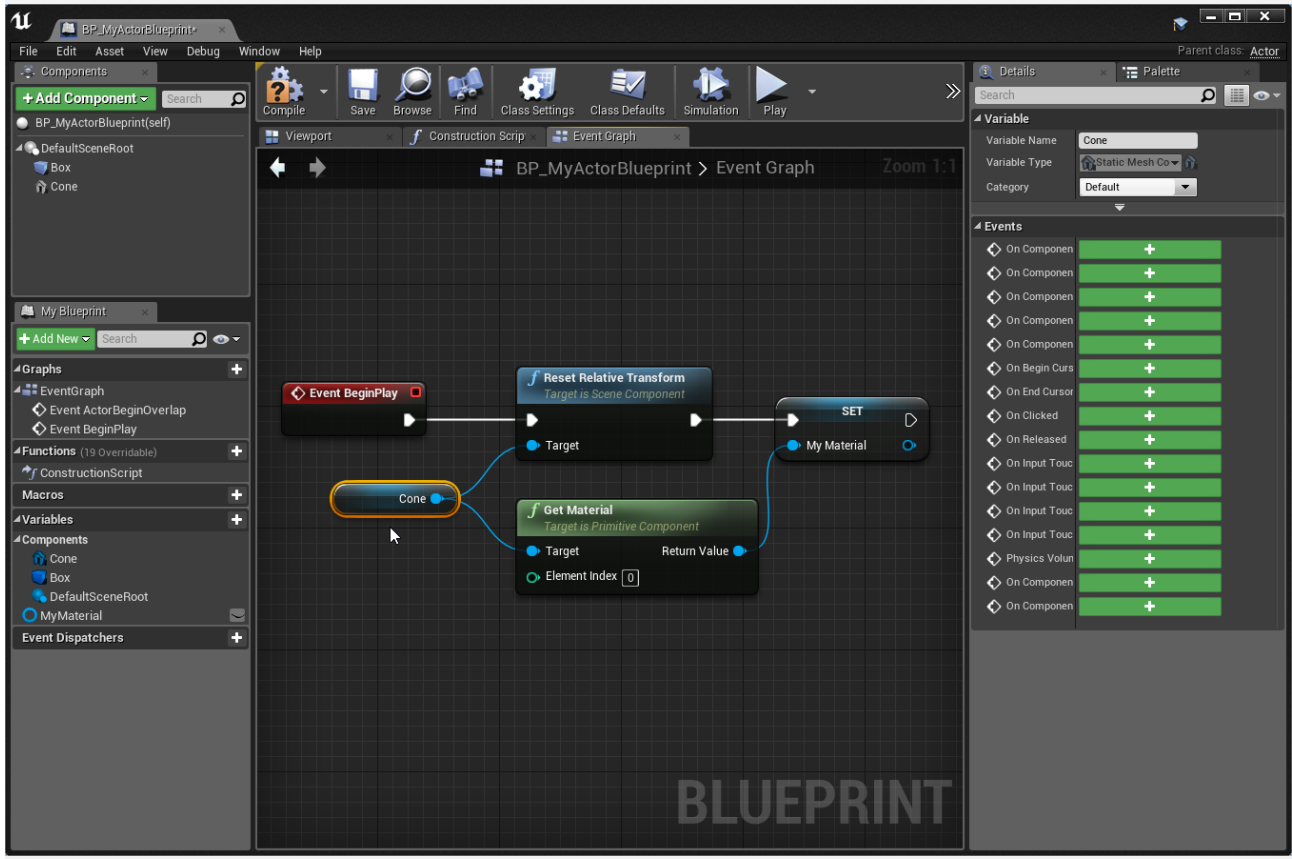
In the example on the right, a box volume and Static Mesh have been added. The Static Mesh component has had a cone mesh applied.



SCRIPTING COMPONENTS

You access components much like you do other variables.

You can drag and drop a reference to the Event Graph from the Variables section of the My Blueprint panel.

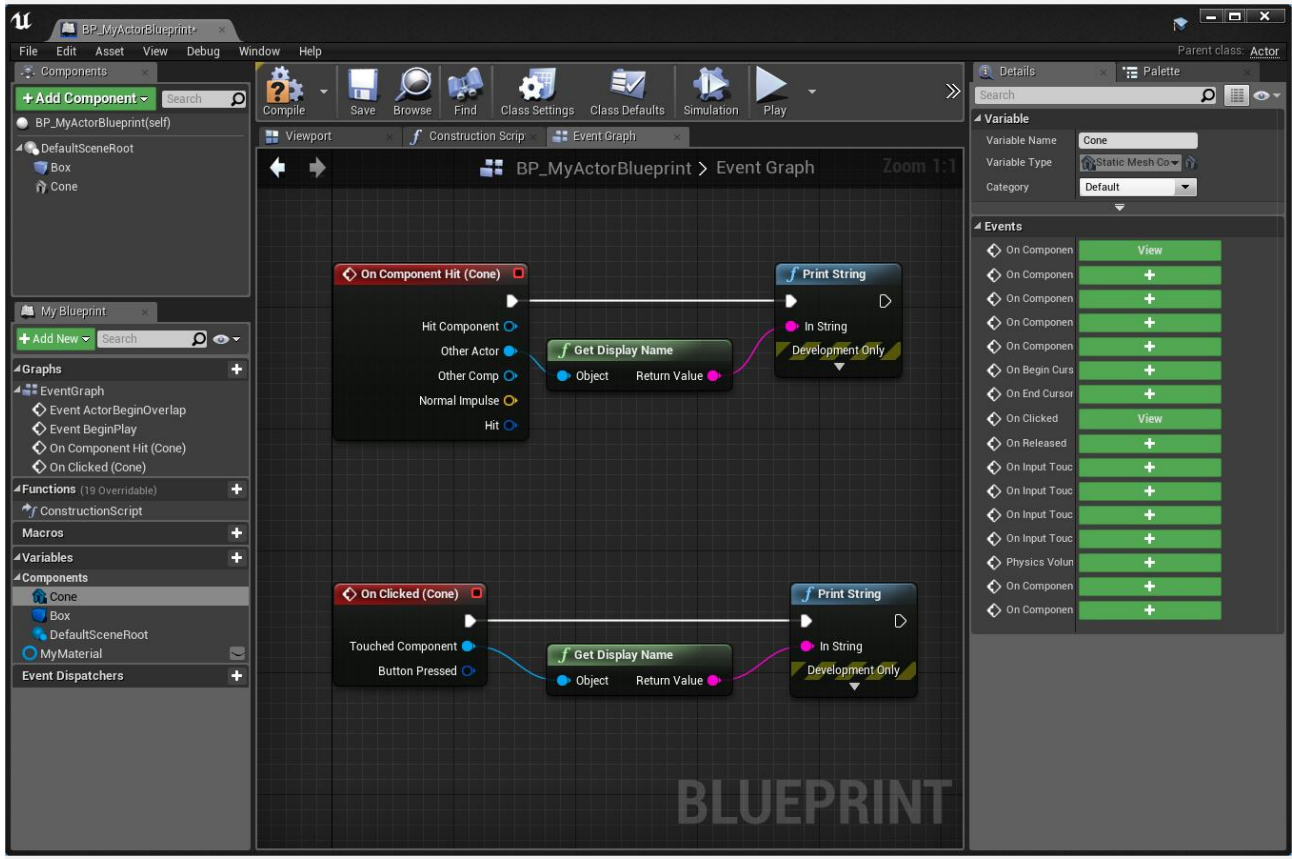


COMPONENT EVENTS

Components can also issue and receive events.

Here the Static Mesh component named Cone is selected, and you can see the available events in the Details panel.

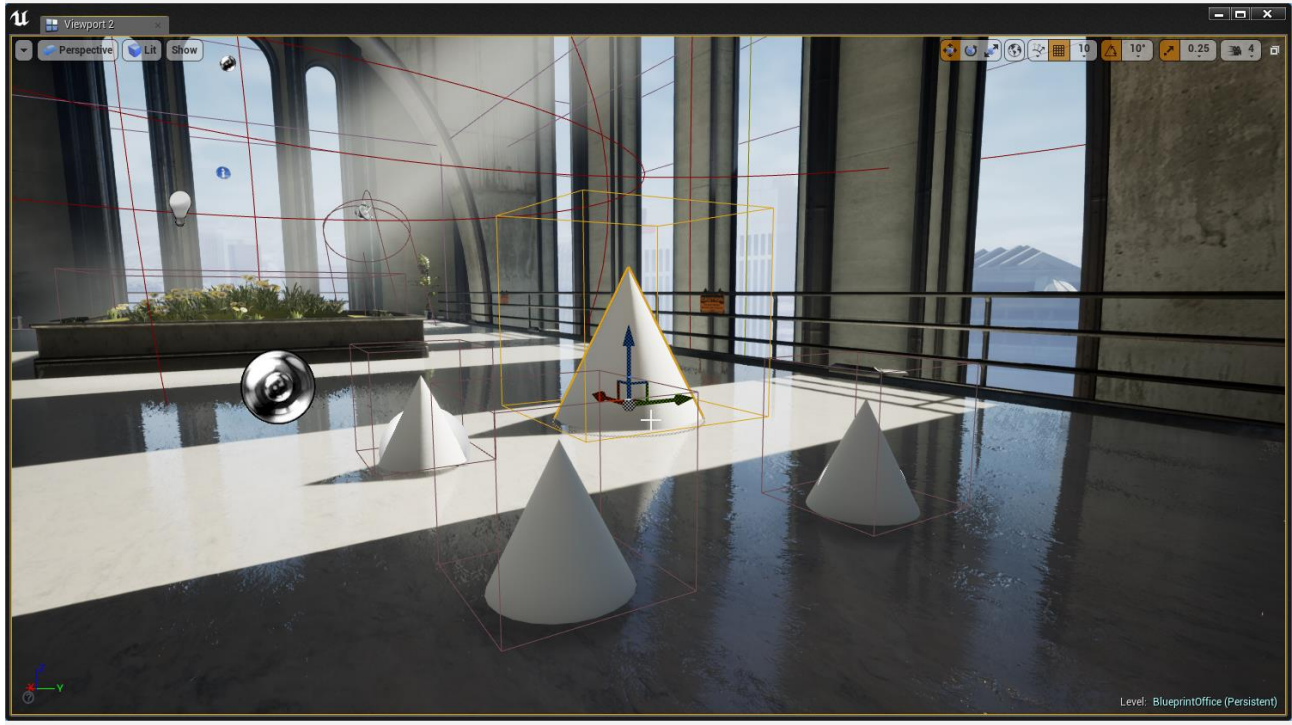
Several events have been wired in the Event Graph.



INHERITANCE

In this image, you can see several instances of the Blueprint Actor class placed in a Level.

Any changes to the underlying Blueprint class will update in the Level.



INHERITANCE

One of the instanced Actors in this figure has had the Static Mesh parameter of the Static Mesh component changed from a cone to a box mesh.

This is an example of a child Actor overriding its parent class.

The parent class is unchanged, and any future instances of the parent class will be created with a default cone mesh.

INHERITANCE

Here you can see that the base class was modified, replacing the cone with a sphere mesh.

The instanced Actor that already overrode the Static Mesh property of the Static Mesh component in the Blueprint did not update.