

- Developer
- / Documentation
- / Unreal Engine ▾
- / Unreal Engine 5.4 Documentation
- / Programming and Scripting
- / Class Creation Basics
- / C++ Only

C++ Only

Introductory information for gameplay programmers getting started with Unreal Engine.



Using the [C++ Class Wizard](#), you can easily add new C++ classes to your project. After you choose the class you would like your new class to derive from, the wizard sets up the necessary header file and source file. If this is the first code you have added to the project, the project is converted to a code project, which creates the game module that contains your source code. It also lets Unreal Editor know that the game module exists, so it can load your C++ changes from Visual Studio or Xcode, and minor code changes can be compiled within Unreal Editor.

The LightSwitch class set up using only C++ is named `LightSwitchCodeOnly`, and is explained below.

Class Setup

The **LightSwitchCodeOnly** class was created by using the [C++ Class Wizard](#), and selecting **Actor** as the parent class.

The header file, `LightSwitchCodeOnly.h`, contains the class declaration.

```
1 UCLASS()
```

```

2 class [PROJECTNAME]_API ALightSwitchCodeOnly : public AActor
3 {
4     GENERATED_BODY()
5
6 };
7

```

 Copy full snippet

Class declarations created with the **C++ Class Wizard** are automatically preceded by the `UCLASS()` macro. The `UCLASS()` macro makes the engine aware of the class, and can also be used with keyword specifiers to set the class's behavior within the engine.

The class declaration contains any variable and/or function declarations. These can be preceded with `UPROPERTY()` and `UFUNCTION()` macros, respectively, which serve similar functions to the `UCLASS()` macro. Components are set up with `UPROPERTY()` macros as well.

In `LightSwitchCodeOnly.h`, C++ is used to:

- Declare the `PointLightComponent` and the `SphereComponent`. The `PointLightComponent` and the `SphereComponent` are made `VisibleAnywhere`. This means that their properties will be visible in the **Details** tab of a `LightSwitchCodeOnly` Actor.

```

1 public:
2     /** point light component */
3     UPROPERTY(VisibleAnywhere, Category = "Switch Components")
4     class UPointLightComponent* PointLight1;
5
6     /** sphere component */
7     UPROPERTY(VisibleAnywhere, Category = "Switch Components")
8     class USphereComponent* Sphere1;

```

 Copy full snippet

- Declare your constructor, so you can set default values for your components and variables:

```

1 ALightSwitchCodeOnly();
2

```

 Copy full snippet

- Declare `OnOverlapBegin` and `OnOverlapEnd`, the functions that will be called when another Actor enters or leaves the SphereComponent. Note that these have different signatures.

```
1  /** called when something enters the sphere component */
2  UFUNCTION()
3  void OnOverlapBegin(class UPrimitiveComponent* OverlappedComp, class
    AActor* OtherActor, class UPrimitiveComponent* OtherComp, int32
    OtherBodyIndex, bool bFromSweep, const FHitResult& SweepResult);
4
5  /** called when something leaves the sphere component */
6  UFUNCTION()
7  void OnOverlapEnd(class UPrimitiveComponent* OverlappedComp, class
    AActor* OtherActor, class UPrimitiveComponent* OtherComp, int32
    OtherBodyIndex);
8
```

 Copy full snippet

- Declare `ToggleLight`, a function that toggles the visibility of the PointLightComponent.

```
1  /** Toggles the light component's visibility*/
2  UFUNCTION()
3  void ToggleLight();
4
```

 Copy full snippet

- Declare the `DesiredIntensity` variable and make it visible anywhere with the `VisibleAnywhere` specifier. It will show up in the category **Switch Variables** in the **Details** tab of a LightSwitchCodeOnly Actor. For variables that are not subobjects, like this float value, the `VisibleAnywhere` specifier just makes the variable show in the **Details** tab. There is an `EditAnywhere` specifier that can be used as well, but because the `DesiredIntensity` variable is only used right as the Actor is added to the level, the variable does not need to be editable.

```
1  /** the desired intensity for the light */
```

```
2 UPROPERTY(VisibleAnywhere, Category="Switch Variables")
3 float DesiredIntensity;
```

 Copy full snippet

The resulting header file looks like:

LightSwitchCodeOnly.h

```
1 // Copyright 1998-2018 Epic Games, Inc. All Rights Reserved.
2
3 #pragma once
4
5 #include "GameFramework/Actor.h"
6 #include "Components/PointLightComponent.h"
7 #include "Components/SphereComponent.h"
8 #include "LightSwitchCodeOnly.generated.h"
9
10 /**
11 *
12 */
13 UCLASS()
14 class [PROJECTNAME]_API ALightSwitchCodeOnly : public AActor
15 {
16 GENERATED_BODY()
17 public:
18 /** point light component */
19 UPROPERTY(VisibleAnywhere, Category = "Switch Components")
20 class UPointLightComponent* PointLight1;
21
22 /** sphere component */
23 UPROPERTY(VisibleAnywhere, Category = "Switch Components")
24 class USphereComponent* Sphere1;
25
26 ALightSwitchCodeOnly();
27
28 /** called when something enters the sphere component */
29 UFUNCTION()
30 void OnOverlapBegin(class UPrimitiveComponent* OverlappedComp, class AActor*
    OtherActor, class UPrimitiveComponent* OtherComp, int32 OtherBodyIndex, bool
    bFromSweep, const FHitResult& SweepResult);
31
```

```

32 /** called when something leaves the sphere component */
33 UFUNCTION()
34 void OnOverlapEnd(class UPrimitiveComponent* OverlappedComp, class AActor*
    OtherActor, class UPrimitiveComponent* OtherComp, int32 OtherBodyIndex);
35
36 /** Toggles the light component's visibility*/
37 UFUNCTION()
38 void ToggleLight();
39
40 /** the desired intensity for the light */
41 UPROPERTY(VisibleAnywhere, Category = "Switch Variables")
42 float DesiredIntensity;
43
44 };

```

 Copy full snippet

The **C++ Class Wizard** also creates the source file for the class, in this case `LightSwitchCodeOnly.cpp`. By default, the source file has basic includes set up.

You will begin by adding the class constructor.

```

1 ALightSwitchCodeOnly::ALightSwitchCodeOnly()
2 {
3
4 }
5

```

 Copy full snippet

In the `LightSwitchCodeOnly` constructor, C++ is used to:

- Set the `DesiredIntensity` variable's value to 3000.

```
DesiredIntensity = 3000.0f;
```

 Copy full snippet

- Create the `PointLightComponent`, set its variables (including setting its intensity to the value of `DesiredIntensity`), and make it the root component.

```

1 PointLight1 = CreateDefaultSubobject<UPointLightComponent>
  (TEXT("PointLight1"));
2 PointLight1->Intensity = DesiredIntensity;
3 PointLight1->SetVisibility(true);
4 RootComponent = PointLight1;
5

```

 Copy full snippet

- Create the SphereComponent, set its variables, and attach it to the PointLightComponent.

```

1 Sphere1 = CreateDefaultSubobject<USphereComponent>(TEXT("Sphere1"));
2 Sphere1->InitSphereRadius(250.0f);
3 Sphere1->SetupAttachment(RootComponent);
4

```

 Copy full snippet

- Designate the `OnOverlap` function as a delegate called when an Actor overlaps or leaves the SphereComponent.

```

1 Sphere1->OnComponentBeginOverlap.AddDynamic(this,
  &ALightSwitchCodeOnly::OnOverlapBegin); // set up a notification for when
  this component overlaps something
2 Sphere1->OnComponentEndOverlap.AddDynamic(this,
  &ALightSwitchCodeOnly::OnOverlapEnd); // set up a notification for when
  this component overlaps something
3

```

 Copy full snippet

The source file is also where you can define any functions you declared for your class. For instance, `LightSwitchCodeOnly.cpp` has implementations of `OnOverlapBegin` and `OnOverlapEnd`, which toggle the visibility of the PointLightComponent by calling `ToggleLight`. Combined with the class constructor, the source file looks like:

LightSwitchCodeOnly.cpp


```

1 // Copyright 1998-2018 Epic Games, Inc. All Rights Reserved.

```

```
2
3 #include "BasicClasses.h"
4 #include "LightSwitchCodeOnly.h"
5
6 ALightSwitchCodeOnly::ALightSwitchCodeOnly()
7 {
8     DesiredIntensity = 3000.0f;
9
10    PointLight1 = CreateDefaultSubobject<UPointLightComponent>
        (TEXT("PointLight1"));
11    PointLight1->Intensity = DesiredIntensity;
12    PointLight1->SetVisibility(true);
13    RootComponent = PointLight1;
14
15    Sphere1 = CreateDefaultSubobject<USphereComponent>(TEXT("Sphere1"));
16    Sphere1->InitSphereRadius(250.0f);
17    Sphere1->SetupAttachment(RootComponent);
18
19    Sphere1->OnComponentBeginOverlap.AddDynamic(this,
        &ALightSwitchCodeOnly::OnOverlapBegin); // set up a notification for when
        this component overlaps something
20    Sphere1->OnComponentEndOverlap.AddDynamic(this,
        &ALightSwitchCodeOnly::OnOverlapEnd); // set up a notification for when this
        component overlaps something
21
22 }
23
24 void ALightSwitchCodeOnly::OnOverlapBegin(class UPrimitiveComponent*
    OverlappedComp, class AActor* OtherActor, class UPrimitiveComponent*
    OtherComp, int32 OtherBodyIndex, bool bFromSweep, const FHitResult&
    SweepResult)
25 {
26     if (OtherActor && (OtherActor != this) && OtherComp)
27     {
28         ToggleLight();
29     }
30 }
31
32 void ALightSwitchCodeOnly::OnOverlapEnd(class UPrimitiveComponent*
    OverlappedComp, class AActor* OtherActor, class UPrimitiveComponent*
    OtherComp, int32 OtherBodyIndex)
33 {
```

```
34 if (OtherActor && (OtherActor != this) && OtherComp)
35 {
36     ToggleLight();
37 }
38 }
39
40 void ALightSwitchCodeOnly::ToggleLight()
41 {
42     PointLight1->ToggleVisibility();
43 }
44
```

 Copy full snippet



`BasicClasses.h` is referring to the name of the project that the class has been set up in.

If this is the first code class you have added to a blank project, you will need to close Unreal Editor, compile the project in Visual Studio or Xcode, and then open Unreal Editor and re-open the project to ensure that the game module is created and loaded properly. Also, it is important to make sure that the **Build Configuration** matches the version of the Unreal Editor executable you are using to open the project. Read more about build configurations and compiling projects in the [Compiling Game Projects](#) documentation.

If you are adding code to an existing C++ project, you can use the Hot Reload functionality to compile the new code within Unreal Editor.

C++ classes can be extended with C++ classes as well as with Blueprint Classes - they show up after checking the **Show All Classes** checkbox in the **C++ Class Wizard** and in the **Custom Classes** section of the **Pick Parent Class** window during Blueprint Class creation.

The LightSwitchCodeOnly class is in the [Class Viewer](#), and can be dragged into the level from there. For more information on placing Actors in a level using the Class Viewer, see the [Placing Actors](#) documentation.