LEVEL BLUEPRINTS

Contents

1 Introduction to Level Blueprints

A **Level Blueprint** in Unreal Engine is a special Blueprint that controls logic specific to a single map (`.umap`).

Unlike Blueprint Classes (which are reusable), the Level Blueprint:

- Exists only once per level
- Can directly reference actors placed in the viewport
- Is ideal for environment-specific interactions

Best Use Case: Room-specific logic, cinematic triggers, environmental events.

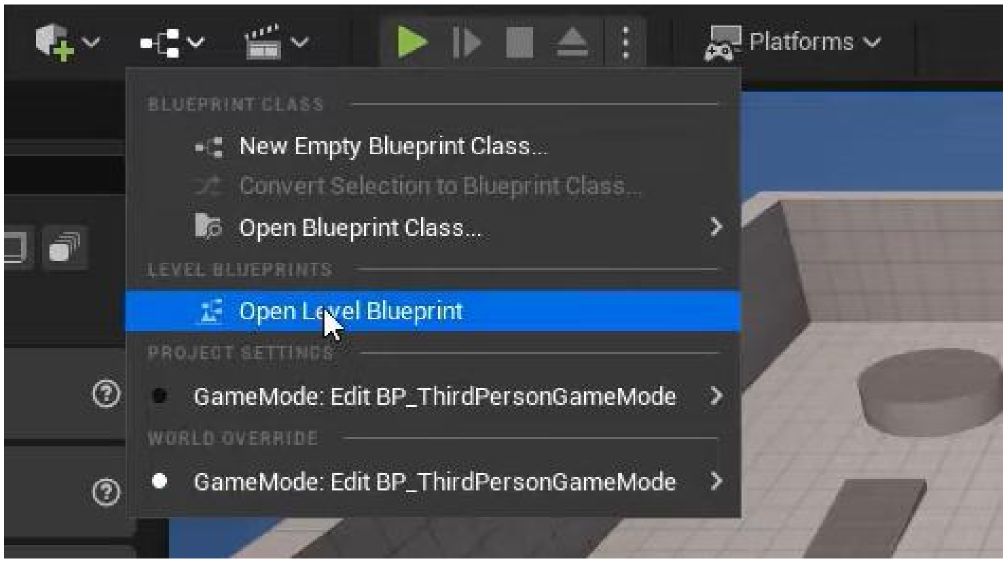
Platforms
BLUEPRINT CLASS
New Empty Blueprint Class...
Convert Selection to Blueprint Class...
Open Blueprint Class...
LEVEL BLUEPRINTS
Open Level Blueprint
PROJECT SETTINGS
GameMode: Edit BP_ThirdPersonGameMode
WORLD OVERRIDE
GameMode: Edit BP_ThirdPersonGameMode

Figure 1: Opeing the Level Blueprint

2 Essential Blueprint Concepts

Nodes: Functional building blocks (events, functions, variables).

Execution Pins (White): Control the order of operations.

Data Pins (Colored): Transfer values (Vector, Bool, Float, etc.).

Compile: Required after every change.

3 PART I – Light Color Change After Delay (No Trigger Box)

This section demonstrates a simple automatic event: *When the level starts, the light changes color after a delay.*

3.1 Level Setup

- Place a **Point Light** in the level.
- Set Intensity (e.g., 5000).
- Set Mobility to **Movable**.

3.2 Blueprint Logic

1. **Reference the Light**
   (a) Select the Point Light in the Viewport.
   (b) Open Level Blueprint.
   (c) Right-click → Create Reference to PointLight.

2. **Add BeginPlay Event**
   Right-click in Graph:
   `Add Event → Event BeginPlay`

3. **Add Delay Node**
   From BeginPlay execution pin:
   `Add Delay` (Set Duration = 5 seconds)

4. **Change Light Color**
   From Point Light reference:
   `Set Light Color`

   Choose any color (e.g., Red).

3.3 Final Node Flow

Event BeginPlay → Delay (5s) → Set Light Color

$$\text{LightColor}(t) = \begin{cases} \text{Default Color} & t < 5 \\ \text{Red} & t \geq 5 \end{cases} \tag{1}$$

Result: After 5 seconds, the light changes color automatically.

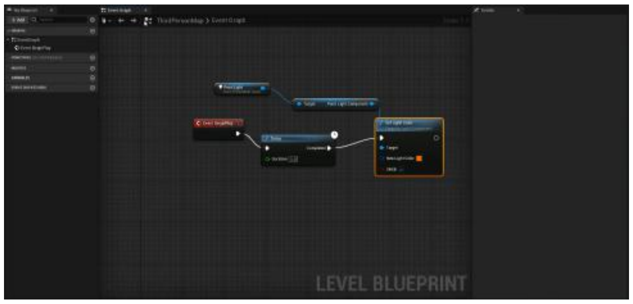

Figure 2: Changing the color of the Light using the EventBeginPlay

4 PART II – Interactive Trigger Box (Light + Sound)

Now we extend functionality:

- Player enters trigger → Light turns ON + Sound plays
- Player exits trigger → Light turns OFF + Sound plays

4.1 Level Preparation

- Place a **Point Light**
- Place a **Trigger Box** (Place Actors > Volumes)
- Add a sound asset to project by importing it

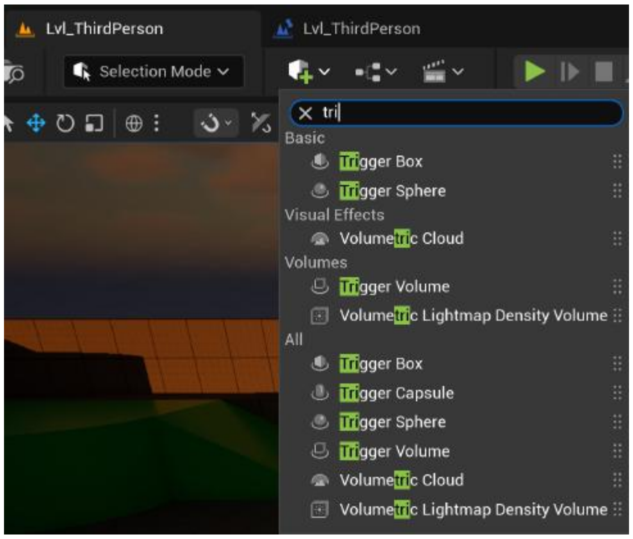

Figure 3: Trigger Box

Things to note:

- The sound cue when imported is saved in content drawer as the form of a **Sound Cue** or **Sound Wav**
- Trigger should cover a walkable area

4.2 Adding Overlap Events

1. Select Trigger Box
2. Open Level Blueprint
3. Add:
    - OnActorBeginOverlap
    - OnActorEndOverlap

4.3 Referencing the Light

Select Point Light → Create Reference
Drag from blue pin: `Set Visibility`

4.4 Adding Sound Playback

To play sound:
Right-click in graph: `Play Sound at Location`
Set:

- Sound = Your Sound Cue

- Location = Get Actor Location (Trigger or Light)

You may use different sounds for ON and OFF.

4.5 Complete Logic Flow

Player Enters Trigger

- OnActorBeginOverlap
- Cast To Character
- Set Visibility = TRUE
- Play Sound (Light On Sound)

Player Exits Trigger

- OnActorEndOverlap
- Cast To Character
- Set Visibility = FALSE
- Play Sound (Light Off Sound)

4.6 Blueprint Implementation

The below shows the the blueprint implementation for the toggling lights with the use of a trigger box.

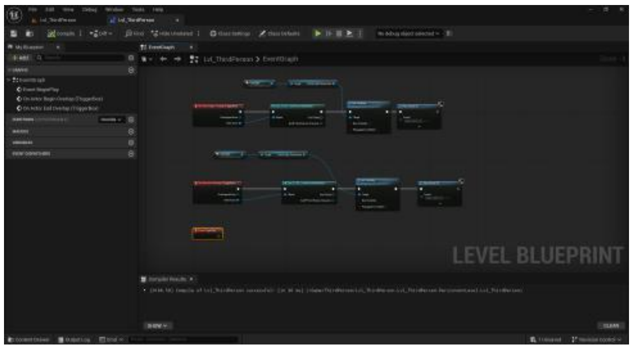

Figure 4: Toggling the Lights with the Trigger Box

4.7 Boolean State Representation

$$\text{Light State} = \begin{cases} 1 & \text{if Player Inside Trigger} \\ 0 & \text{if Player Outside Trigger} \end{cases} \tag{2}$$

4.8 Important Technical Notes

- Use **Cast To Character** to prevent physics objects triggering logic
- Avoid heavy logic in Event Tick
- Compile after every modification

5 Level Blueprint vs Blueprint Class

Feature	Level Blueprint	Blueprint Class
Scope	One Level Only	Reusable Anywhere
Direct Referencing	Yes	No
Best For	Scene Logic	Reusable Objects
Example	Door in this room	Spinning coin actor

6 When Should You Use Level Blueprints?

Use Level Blueprints when:

- Logic is unique to that environment
- You need quick references to placed actors
- You are prototyping interactions

Do NOT use Level Blueprint if:

- You need reusable functionality
- You are building scalable systems

Conclusion

Level Blueprints are ideal for:

- Environmental triggers
- Delayed events
- Scene-specific sound and light control

Understanding when to use Level Blueprint vs Blueprint Class is critical for scalable Unreal Engine architecture.