How to: The Talos Principle puzzle kit

Finn Slutter

Unreal Engine 5.1

Project version: 1.3

Contents

Introduction

Figure 1

The Talos principle is a first-person object-based puzzle game created by CroTeam and this puzzle creation set was heavily inspired by their game. To learn more about the game see their fan wiki: https://talosprinciple.fandom.com/wiki/The_Talos_Principle_Wiki. To play their game you can find it here: http://www.croteam.com/talosprinciple /. This toolset includes a set of the objects found in the game, with the aim of working from object traits to give an easy route to expand and include more level elements.

This how-to document is based on the V1.0 of this project and future versions could differ from this document

Migration and setup

This project is built using Unreal Engine 5.1 using the first-person character template.

Project set up

- Right click the folder: TheTalosPrincipleToolset found in the content folder of this project
- Select migrate and find the content folder of your project
- Find your game mode and make the default pawn class the BP_TalosFirstPersonPlayerCharacter
- In the inputs section of your project settings add two new action mappings called: Interact, and Sprint
- In the collision section of your project settings create two new trace channels (the order of these is important)
    - The first trace channel is called EnergyWall and its default response should be ignore
    - The second trace channel is called BlocksSelecting and its default response should be block

(The project can look a bit weird with lumen enabled, mainly the laser beams. To turn of global illumination find it in the rendering section of the project settings)

Level set up

- Make sure the level has a player start
- Make sure the level has a post processing volume and adjust its settings in the following way

- In the details panel under post processing materials click add and select the M_Highlight material
- Under Post Process Volume Settings set: Infinite Extend (Unbound) to true (or scale up the post process volume to include all the playable area)

Building a level with existing content

This project contains a base set of puzzle elements that can be used to build puzzles. This includes the objects to create laser puzzles and puzzles including pressure plates. In this section, an explanation of all puzzle elements will be given to allow the user to build puzzles with the existing content.

Gameplay elements

Laser objects

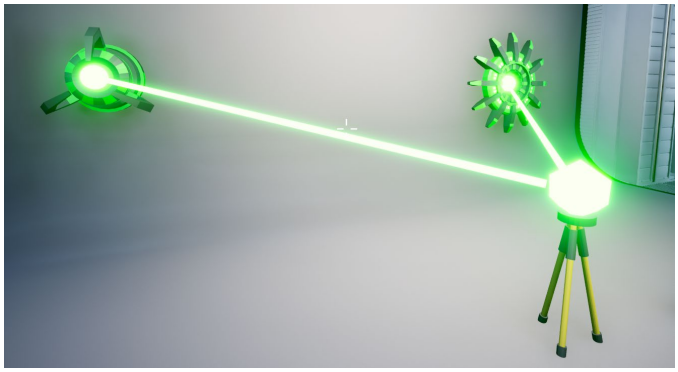

Figure 2

All laser objects are inherited from the parent **object: BP_Laser_Base** *(TheTalosPrincipleToolSet\Blueprints\BP_Laser_Base)*.

This base handles all the functionality for **creating connections** between laser objects, disconnecting from exiting connections, recalculating the laser path, and the logic of what happens when an object is hit by a laser.

This object also includes the different options for laser colors that are used by all children to update their visuals accordingly.

Generator

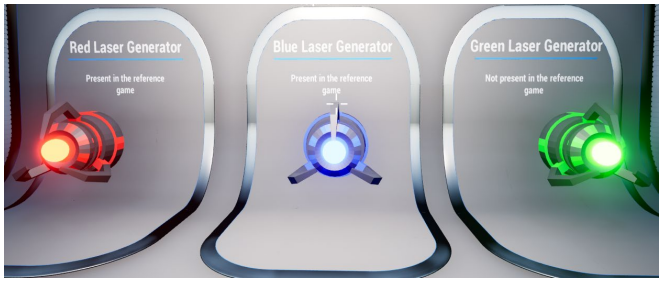

Figure 3

Introduction

Generators are the origin points of lasers. Generators can be different colors; the chosen color affects the laser beam that emanates from this object. Generators will only start generating a laser beam when at least one connector is connected to it by the player or by the level designer.

Add to level

- Find the **BP_LaserGenerator** in the content browser *(TheTalosPrincipleToolSet\Blueprints\BP_LaserGenerator)*
- Drag this object into the level
- Move it to the desired position

(!Make sure the glowing ball is not obstructed by other objects!).

Change *settings*

Changing the color of a generator:

- Select the desired generator
- Look in the details panel for the "**Color Enum**" variable (figure 4)
- Select the color for the generator from the drop down menu

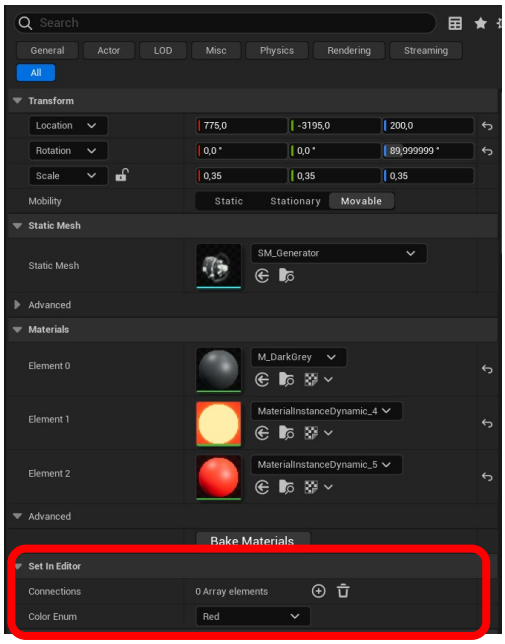

Figure 4

Best practice:

To make sure lasers will always intersect with each other, place the generators 100 units above the floor. This will mean that laser beams between generators and connectors are always straight.

Connector

Figure 5

Introduction

Connectors are pick-ups that the player uses to create networks of laser beams. When picked up the player can select laser objects by looking at them and clicking the left mouse button. When the player wants to place down the connector they can use the left mouse button when not looking at a selectable object. A placed down connector with connections will create a network of laser beams between it and its connections.

Add to level

- Find the object **BP_LaserConnector** in the content browser *(TheTalosPrincipleToolSet\Blueprints\BP_LaserConnector)*
- Drag the object into the level
- Move the connector into the desired location

(!Make sure that the connector is not intersecting with any other static mesh!)

Change settings

Establish existing laser connection on begin play

- Select a **BP_LaserCconnector** in the level editor
- Find the **connections array** in the details panel
- Find the object you want to connect to in the viewport
- Click on the plus and then the eye dropper icon
- Click on it using the eye dropper tool (figure 6 & 7)

To create multiple connections, add more object to the connections array.

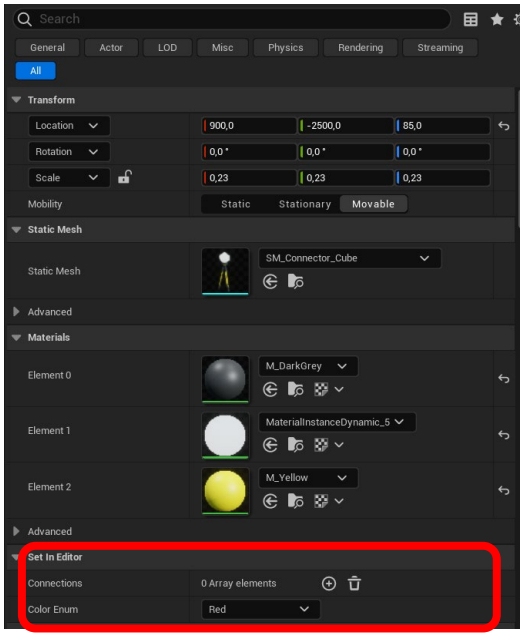

Figure 7

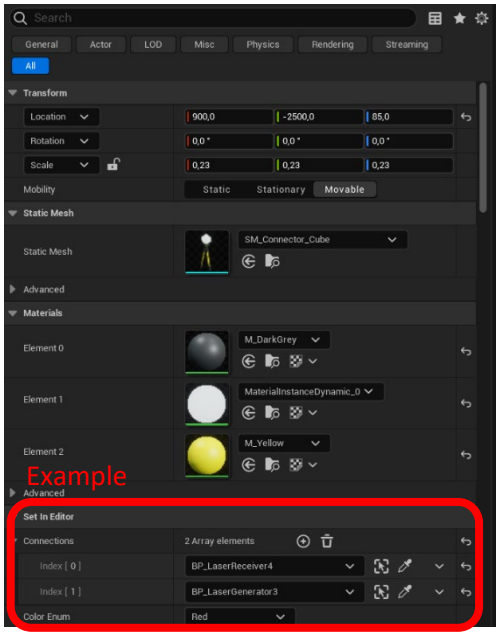

Figure 6

Receiver

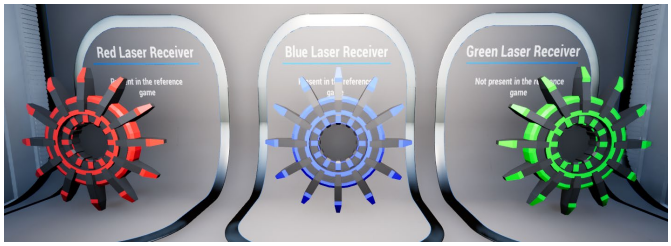

Figure 9

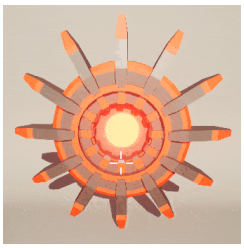

Figure 8

Introduction

The receivers are the end point of laser beams. When a laser with the same color as the receiver hits this object, it will turn on. The laser receiver has a charge time that is showcased with a circular animation. If the laser connection to a receiver breaks the charge resets, only when the receiver is fully charged will it turn on any objects that have this receiver as an input. Receivers in this demo can open and close energy walls.

Add to level

- Find the object **BP_LaserReceiver** *(TheTalosPrincipleToolSet\Blueprints\BP_LaserReceiver)*
- Drag the object into the level
- Move the receiver into the desired location

(!Make sure the object is not obstructed by other collision objects!)

Change settings

Change the color of the receiver

- Select the receiver in the level editor
- Look in the details panel for the **“Color Enum”** variable
- Select the color from the drop-down menu

Change the charge-up time

- Look for **“ReceiverConnectionTime”** in the details panel of the receiver
- Set the float value to the desired charge up time in seconds (1.0 equals 1 second)

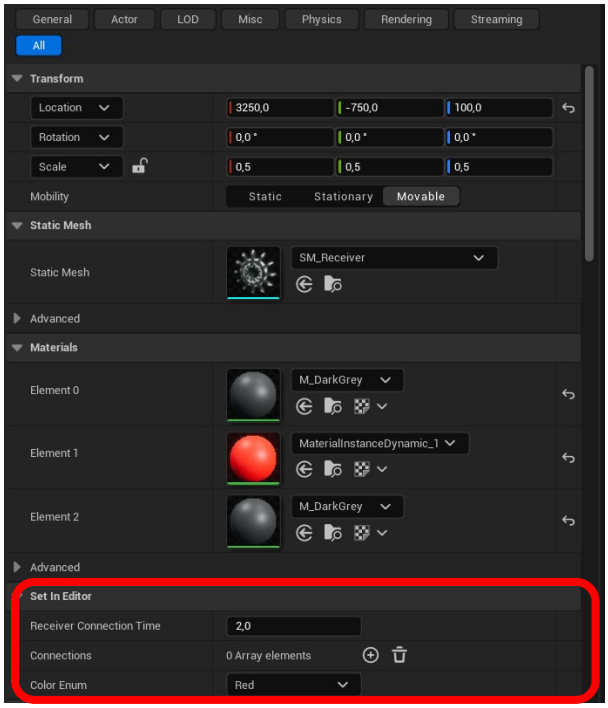

Figure 10

Energy wall

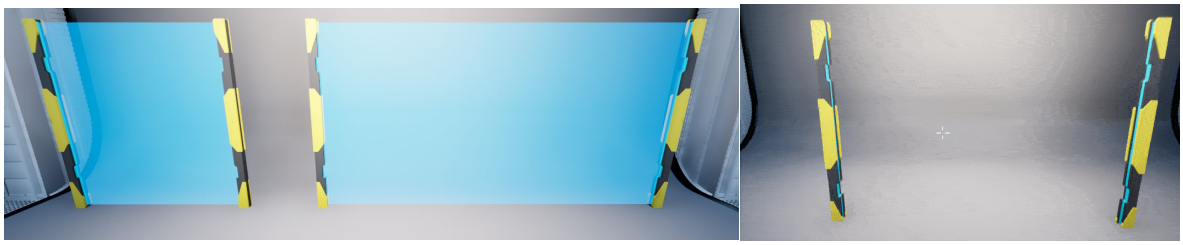

Figure 11

Introductions

Energy walls are the doors in The Talos Principle. Energy walls will switch states when they are turned on or off. These objects do not allow the player, laser beams, and objects to pass through. But the player can select laser objects through these transparent walls. Energy walls can be switched on and off in this demo by either pressure plates or laser receivers.

Add to level

- Find the object **BP_EnergyWall** *(TheTalosPrincipleToolSet\Blueprints\BP_EnergyWall)*
- Drag the object into the level editor
- Move the energy wall in the desired location
- Adjust the size if needed (see “change settings” in this document and then look for “Change Energy Wall width)
- Add input switch(es) (see “change settings” in this document and then look for “Adding a switch input”)

Change settings

Adding a switch input

- Find the desired switch input in the level editor
- Select the energy wall in the level editor
- Find the **Inputs** array in the details panel
- Click on the plus icon and then select the eye dropper
- Select the switch input in the level editor with the eye dropper

Switch inputs included in this project are the **BP_LaserReceiver** and **BP_PressurePlate**

Set to start open

- **Select an energy wall** in the level editor
- In the details panel look for the **"StartOpen"** boolean and set it to true

This will make sure the energy will open up on begin play and that it will close when powered instead of open.

Changing input modes

- Select an energy wall in the level editor
- In the details panel look for **"OnlyOneInputNeeded"** Boolean
- Set this to true to set the mode to Only One Input Needed

Energy walls can be in two modes. The default mode requires all of its inputs to be on before it will turn on itself. The other mode called "Only One Input Needed" turns the energy wall on if at least one if its inputs is on and will turn off only if all of its inputs are off.

Change energy wall width

- Select an energy wall in the level editor
- In the details panel look for **"EnergyWallWidth"** float
- Adjust this float (A value of 1 is equal to a width of 100 units)

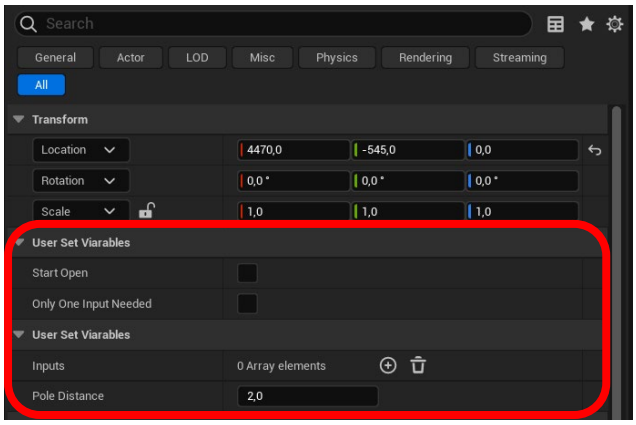

Figure 12

Pressure plate

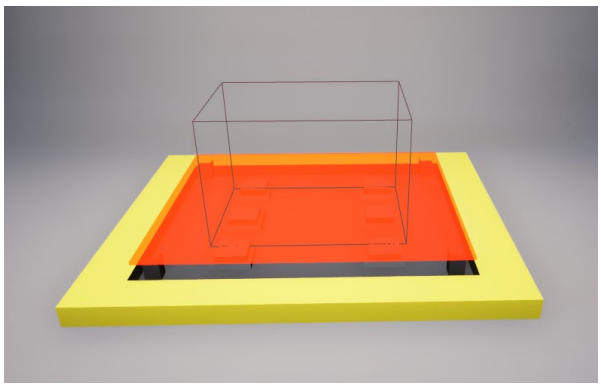

Figure 13

Introduction

Pressure plates like laser receivers are inputs. When either the player or an object enters a trigger box above the pressure plate it will activate. Pick-ups like connectors can snap to pressure plates to make it easier to place them on it. Unlike receivers, pressure plates do not have a charge time and instantly activate and deactivate when objects enter and leave their trigger box.

Add to level

- Find the object **BP_PressurePlate** (TheTalosPrincipleToolSet\Blueprints\BP_PressurePlate)
- Drag the object into the level editor
- Move the pressure plate to the desired location

Best Practice

Make sure that if a pressure plate is placed directly in front of the energy wall it opens there are at least 200 units of space between the two to make sure the player can't get through by brute forcing it. If the player run speed is adjusted this distance also has to be adjusted.

Level ingredients

Fence

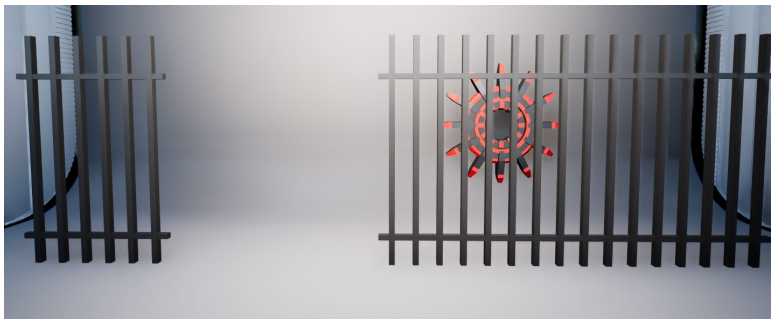

Figure 14

Introduction

A fence is an object that is very similar to objects with default collision settings (these should be used as walls, floors, and ceilings). Fences block the player, laser beams, and objects. However, they don't completely block vision and allow the player to select laser objects.

Add to level

- Find the object **BP_Fence** *(TheTalosPrincipleToolSet\Blueprints\BP_Fence)*
- Drag the object into the level editor

- Move the fence to the desired location
- Adjust size if necessary (see “Change settings”)

Change settings

Change the length of the fence

- Select a fence object in the level editor
- In the details panel look for the **“Length”** float variable
- Adjust this float (1 equals 100 units in length)

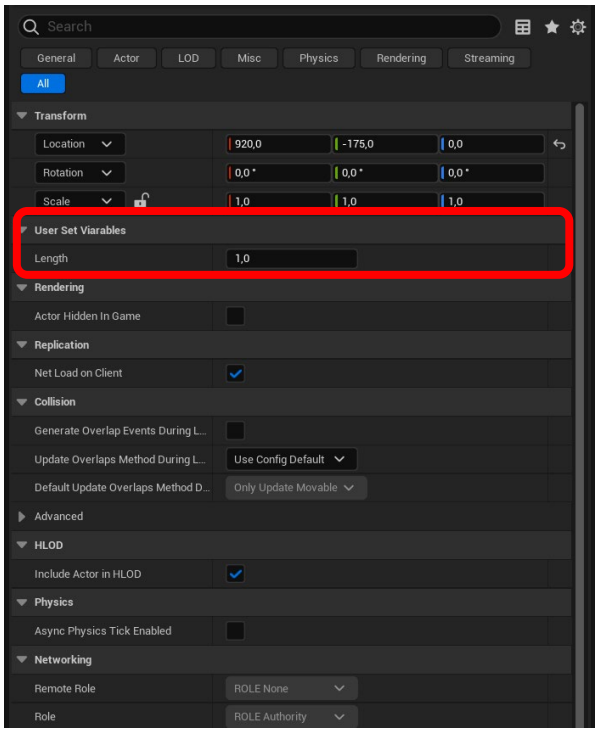

Figure 15

Exclusion field

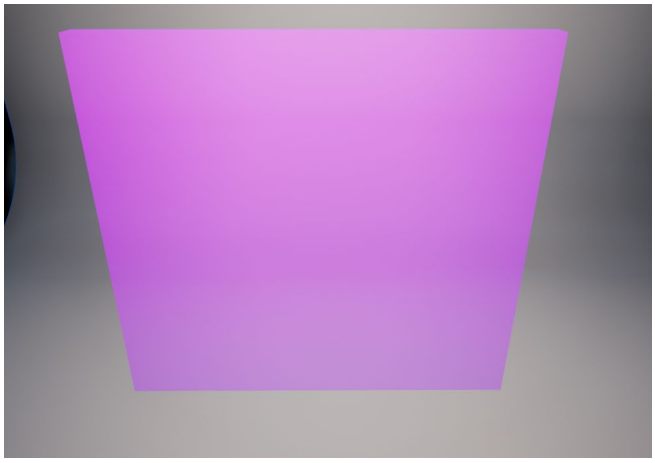

Figure 16

Introduction

Exclusion fields are volumes that will **only allow the player and laser beams to pass through**. No objects are allowed through this volume. This means that if the player is holding an object and passing through an exclusion field the item will be dropped from the player's hand.

Add to level

- Find the object **BP_ExclusionField** *(TheTalosPrincipleToolSet\Blueprints\BP_ExclusionField)*
- Drag the object into the level editor
- Move the object to the desired location
- Scale the exclusion field using normal scale properties if necessary

Best practice

This object can be scaled to fit into doorways without hindering its functionality.

Place exclusion fields after or before a puzzle to make sure the player can not use objects from one puzzle to solve the other.

Use exclusion fields in a puzzle to keep certain objects locked in a specific area.

Adding new content

Laser colors

If you want to create puzzles using more then the 3 colors included, or if you want to change the color of the lasers follow this guide:

- Find the **LaserColors Enumeration** (*TheTalosPrincipleToolSet\Blueprints\E_LaserColors).*
- Open the enumerator and click the plus icon to add a new color, here you can also adjust the names of already exiting colors. (figure 17)
- Find the **BP_Laser_Base** blueprint (*TheTalosPrincipleToolSet\Blueprints\BP_Laser_Base).*
- Open the blueprint and navigate to the construction script
- Look for the node **Set:ObjectColor** (figure 18)
- Click on the newly created Enum and change the color to your liking

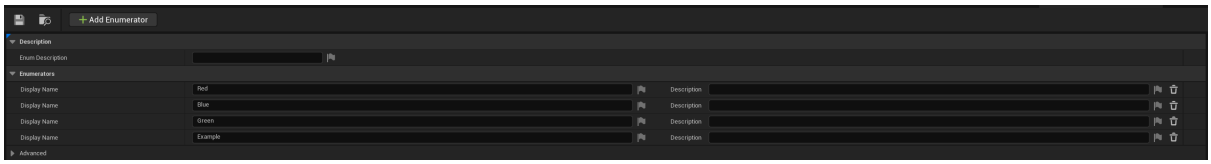

Figure 17

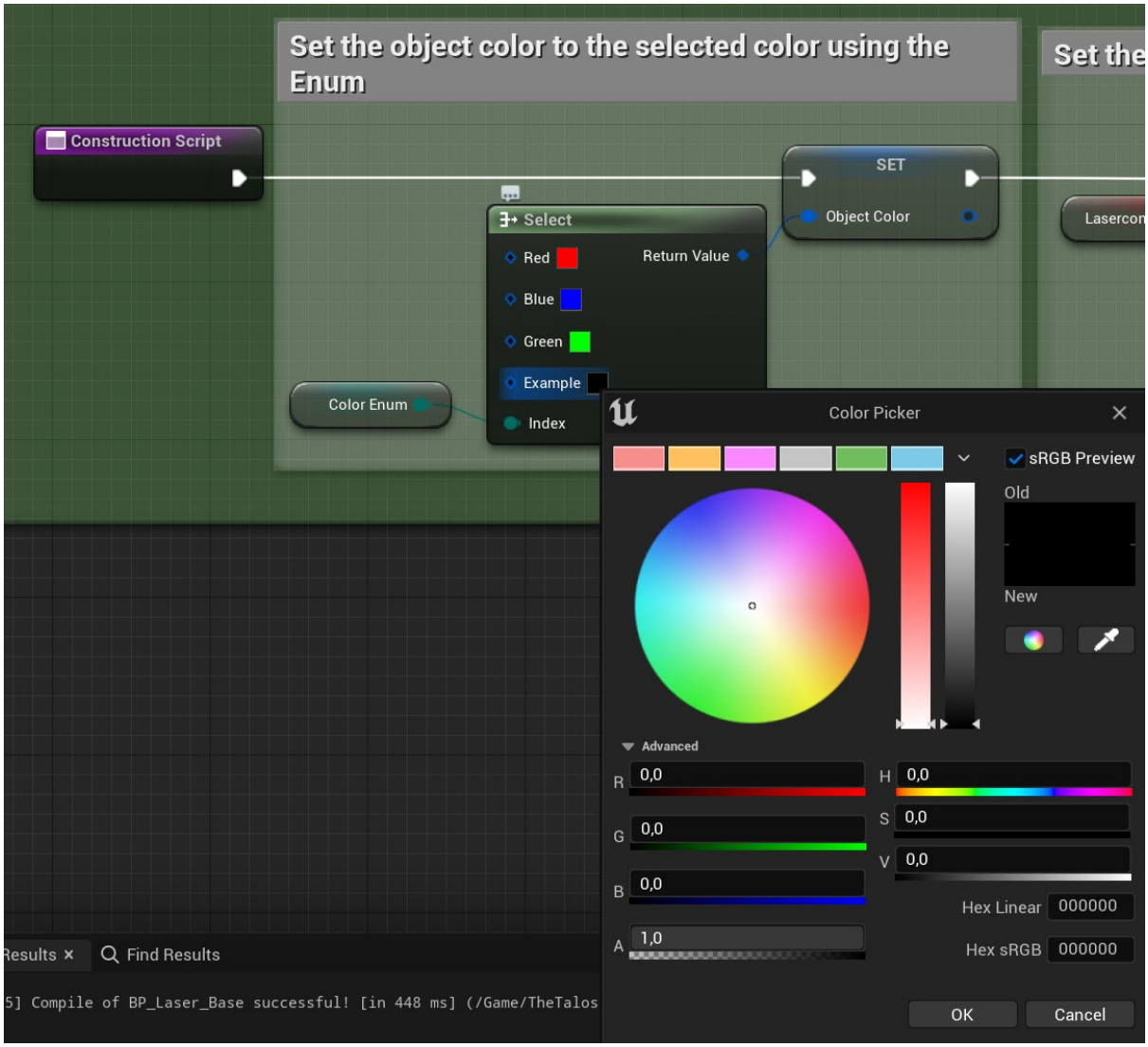

Figure 18

The newly created color is now available in the receiver and generator, and laser beams will show up in the newly created color if it is produced from a newly created generator.

Creating a laser object

All laser objects are children of the laser base. If you want to add a new object that interacts with the laser beams, follow these steps:

- Find the **BP_Laser_Base** (*TheTalosPrincipleToolSet\Blueprints\BP_Laser_Base)*
- Right click this blueprint and select "**create child blueprint class**" (give it a appropriate name)
- All events in the parent or the BPI_Laser can be overwritten to create new functionality for your object

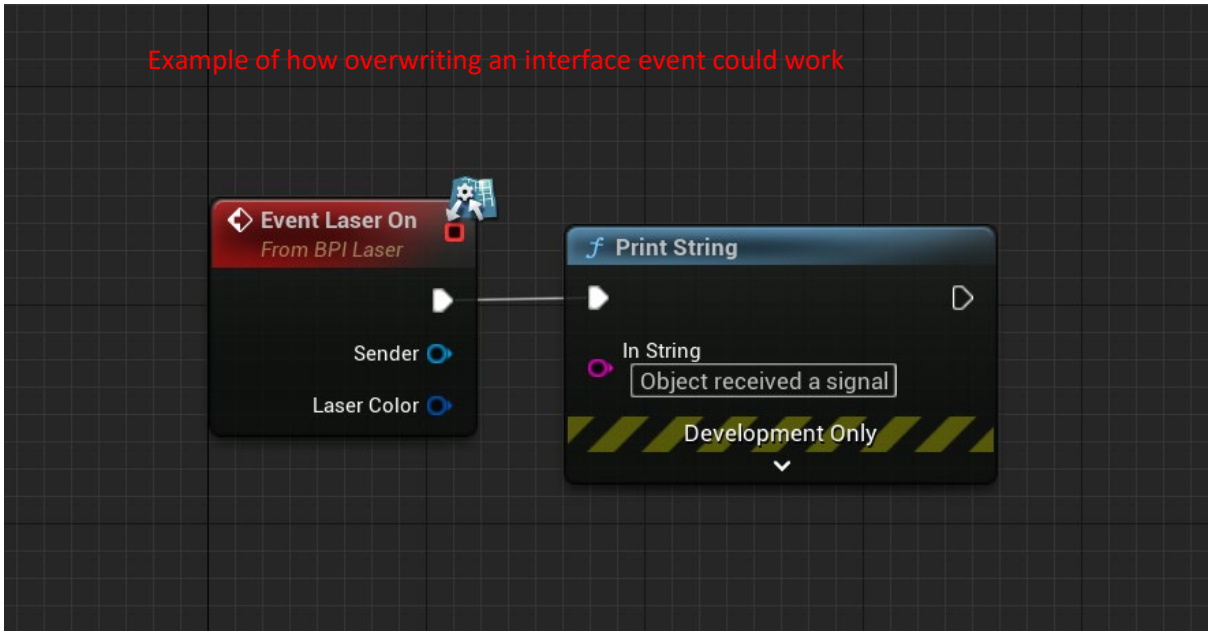

Figure 19

To make sure you can select this object with a connector follow these steps:

- Find the **BP_LaserConnector** (*TheTalosPrincipleToolSet\Blueprints\BP_LaserConnector)*
- Click on the **AC_PickUp** in the Components panel
- In the details panel look for "**Select Objects**", click on the plus and in the drop down look for your newly created laser object.

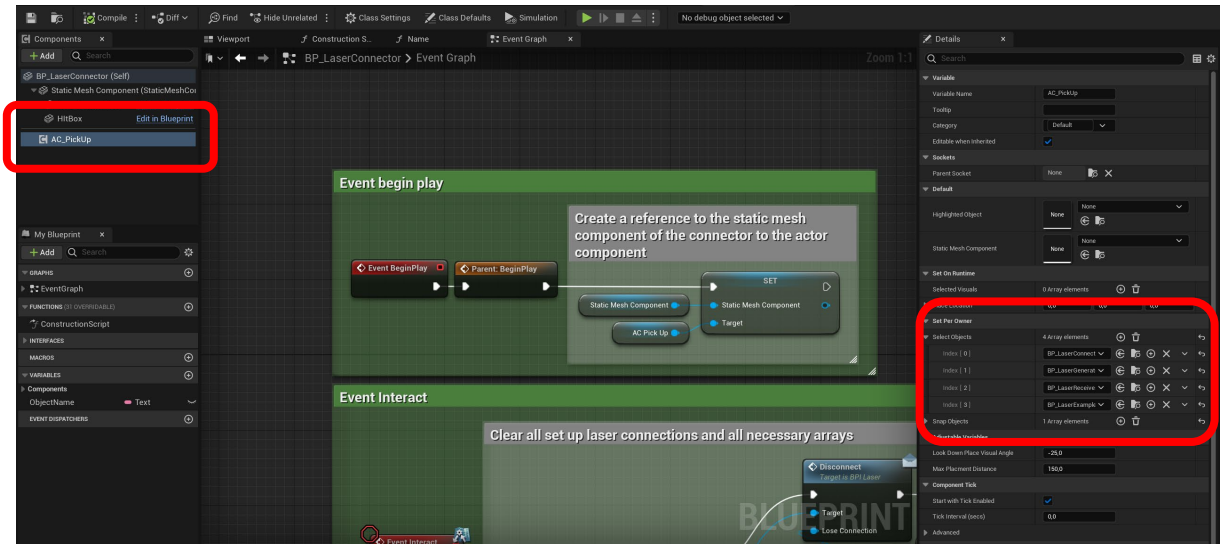

Figure 20

Look through the already exiting laser objects to see how events are used and how they could be changed.

Creating interactable objects

Interactable object are only shared with pickups in this demo, but in the reference game there are level elements like the lever and the play back machine that are purely intractable. To set up these objects follow these steps:

- Create a new actor blueprint
- Click on **Class Settings** and find the inherited interfaces in the details panel
- Add the **BPI_Interact** to the blueprint
- Set up the blueprint as seen in the picture below
- Add custom functionality when the player presses interact when in range of this object

Figure 21

Creating a pickup object

Pickup objects are created using an actor component. This actor component can be found here (*TheTalosPrincipleToolSet\Blueprints\ActorComponents\AC_PickUp)* this actor component handles things like highlighting, selecting picking up, and placing down objects.

To create a **new pick up** follow these steps:

- Create a new actor blueprint
- In the components panel click on the plus and look for "**AC_PickUp**"
- Click on the "**Class Settings**", and in the details panel under inherited interfaces add the "**BPI_Interact**"
- Make sure the actor blueprint contains a static mesh

For a basic set up for an object that needs the ability to be picked up, placed down the minimal blueprint set up looks like the picture in figure 22.

Before or after calling the events or functions in the actor component custom logic can be added for the specific pick up that will be created.

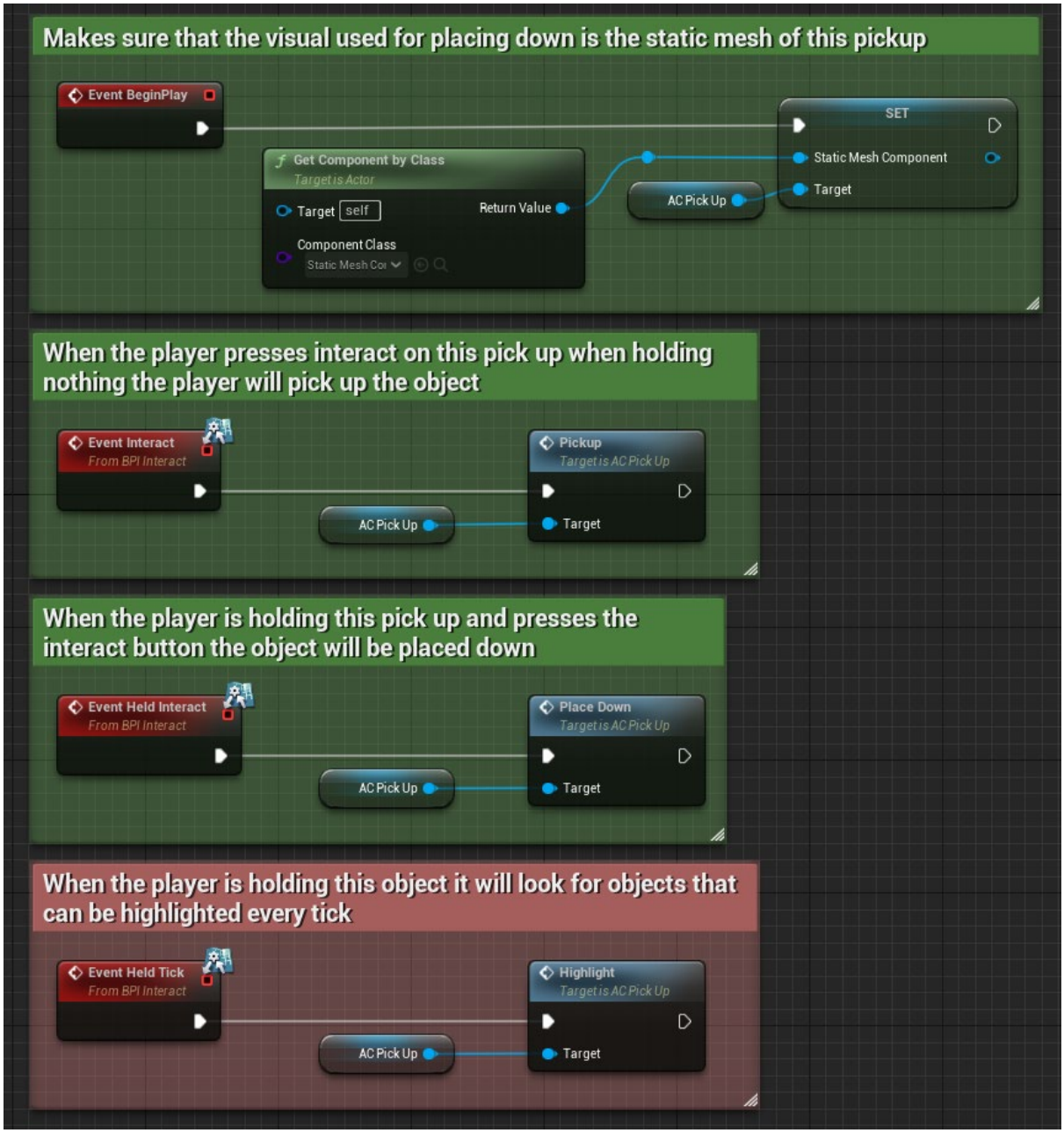

Figure 22

Highlighting and Selecting: If the object needs to be able to select objects when it is being held by the player, follow these steps:

- Click on the **AC_PickUp** in the components panel of the created pick up
- Look for the **Select Objects array** in the details panel.
- Add classes that need to be able to be selected when holding the created pick up

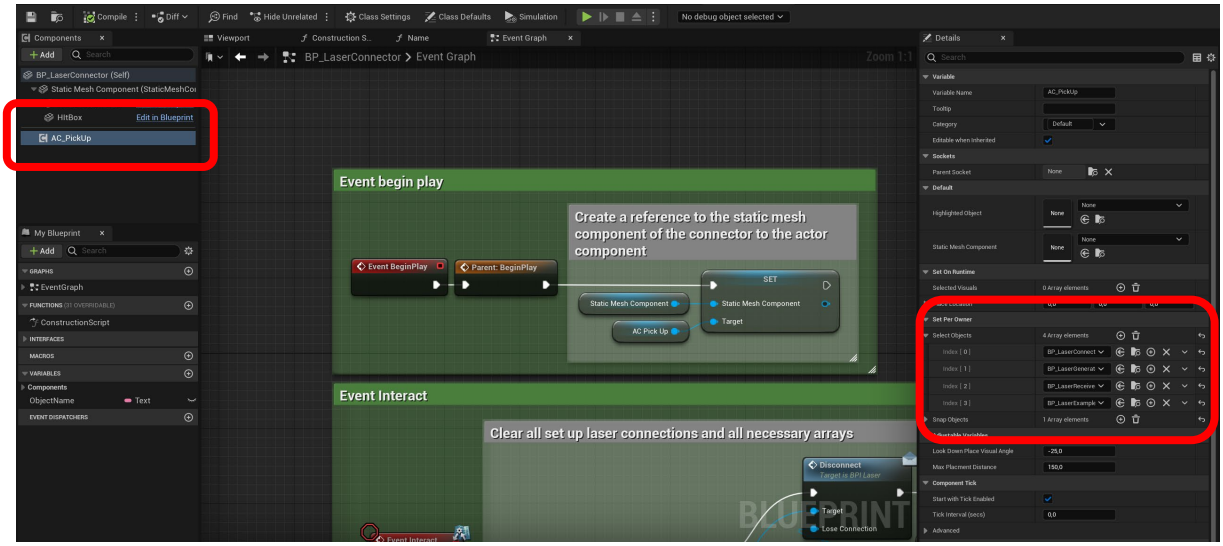

Figure 23

Snapping objects for placing down: If the object needs to be able to snap to certain classes when placing them down (like how the connector snaps to the pressure plate for example) follow these steps:

- Click on the **AC_PickUp** in the components panel of the created pick up
- Look for the **Snap Objects** array in the details panel.
- Add classes that the pickup needs to be able to snap too.

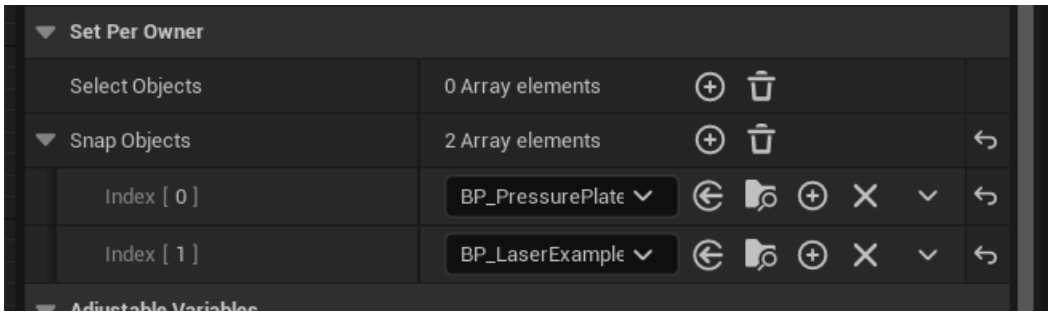

Figure 24

Creating a switch input object

Switch inputs like the pressure plates and laser receivers are created using an interface. To create a new switch input, follow these steps:

- Create a new actor blueprint
- Click on **Class Settings** and find the inherited interfaces in the details panel
- Click on add and add the **BPI_Switch**.

To set up the basic functionality for a switch input follow these steps:

- Create an actor reference array variable (here called ConnectedSwitches)
- Set up the blueprint as seen in the picture below (figure 25)
- Add functionality to call these events

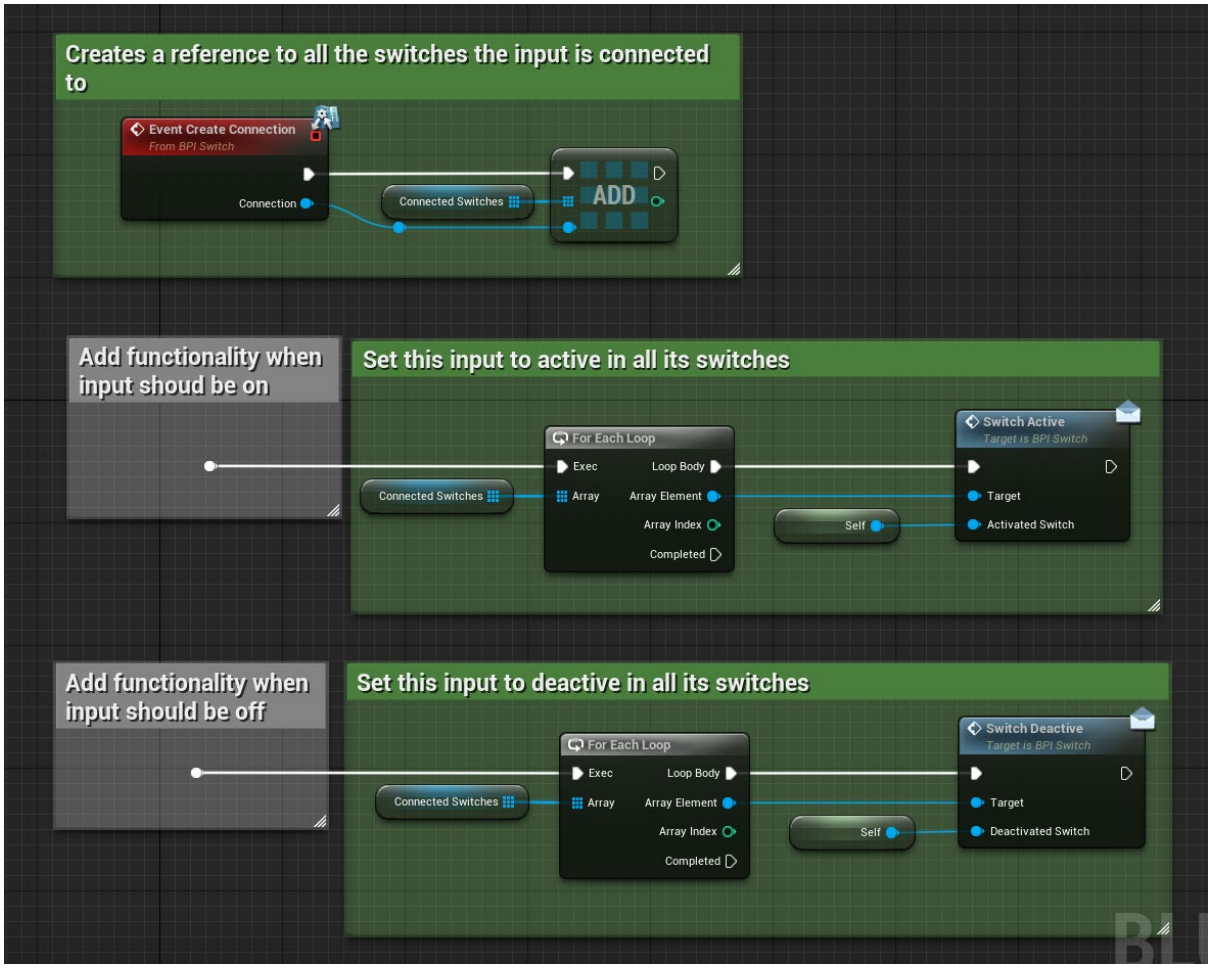

Figure 25

Creating a switch object

Switches like the energy walls are objects effected by switch inputs. To set one up in the project follow these steps:

- Create a new actor blueprint
- Click on **Class Settings** and find the inherited interfaces in the details panel
- Click on add and add the **BPI_Switch**

To set up a basic functioning switch follow the following steps:

- Create an actor reference array variable (here called Inputs)
- Make the variable instance editable
- Create an actor reference array variable (here called ActivatedInputs)
- Set up the blueprint in the picture below
- Add custom functionality when the switch activates and deactivates

Figure 26

Known issues

- It can be difficult to select laser objects from far away
- Connectors sometimes don't show the laser color that it is reflecting