Courtesy of AltSpace

UNREAL FAST TRACK
WORKSHOP TWO

Workshop Two: Diving into Blueprints, Powerups, and Obstacles

Learning Points

Test Drive (75 minutes)

- Deeper Blueprint learning
- Creating Blueprints
- Understanding the different kinds of Blueprints
- Blueprint casting

Grand Prix (1 hour)

- Making powerups and obstacles
- Using event dispatchers in Blueprints
- Creating multiple types of variables

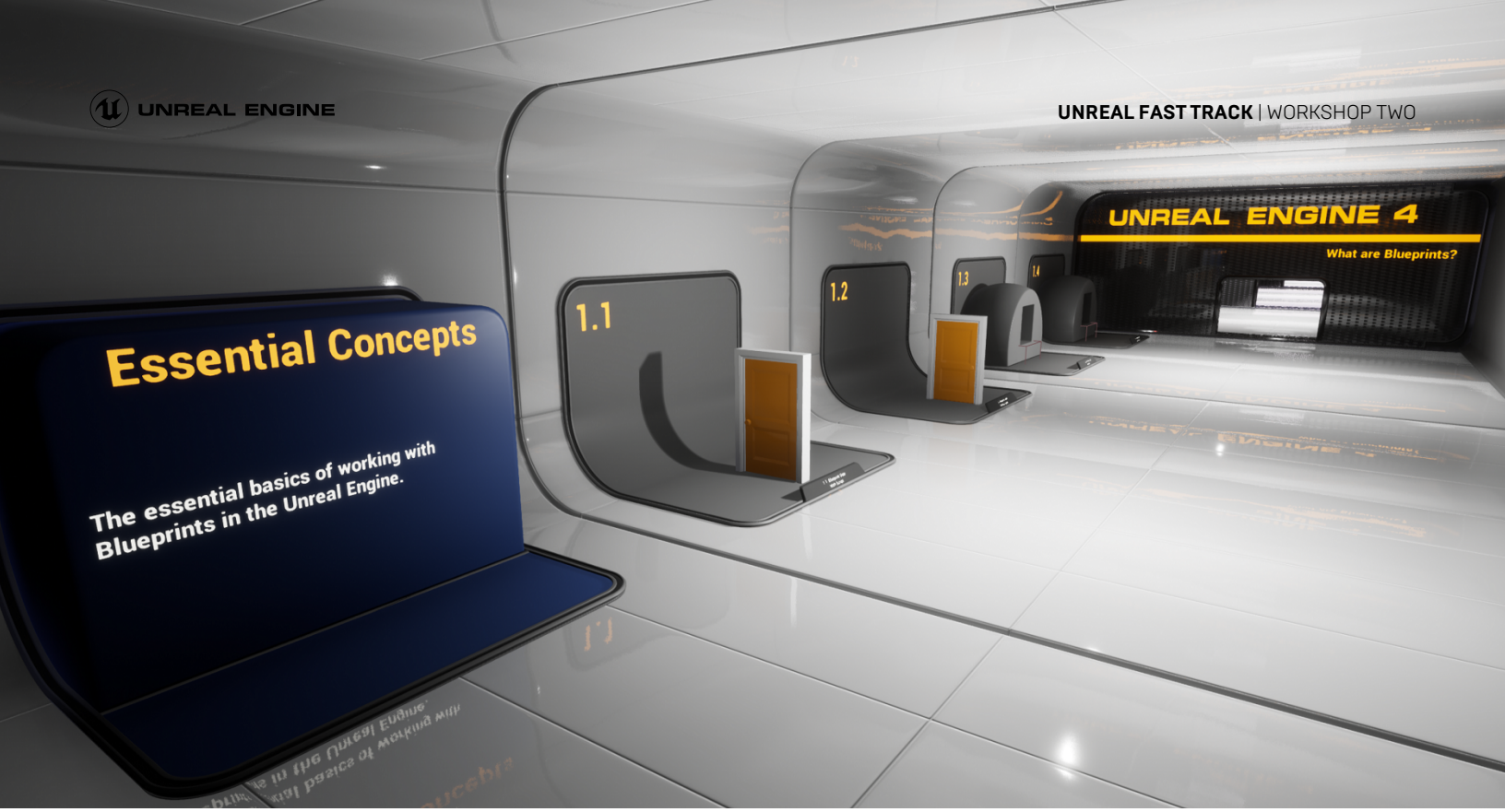
Off-Roading and Discussion (75 minutes)

- Creating Blueprints from scratch
- Modifying existing content

Test Drive

For the second week, we are going to learn more about Blueprints and how they work. The Unreal Online Learning course for this week is called "Blueprints—Essential Concepts." The course can be found at https://www.unrealengine.com/en-US/onlinelearning-courses/blueprints---essential-concepts. Blueprint is a visual C++ scripting system within Unreal Engine and it is an extremely important tool to know how to use. No prior coding experience is required for learning Blueprints. Note that once you begin the course, you will be prompted to download an Unreal Engine project to work with as you follow alongside the instructor.

In this document, we've included tips and changes that have been created for some of the videos since the course first came out. If needed, the tips and changes for each video are timestamped so you know when they are relevant. If there is no timestamp, you can wait until the end to read them.



3: What Is Blueprint?
2:57: The Level he is editing is in the **StartAssets** folder found here: **Content → Fundamentals → Blueprints → EssentialConcepts → StartAssets**. The Level is called "**EssentialConcepts_Start**".

13: Additional Resources
Be sure to click on **Additional Resources** after the quiz to complete the course and obtain the badge!

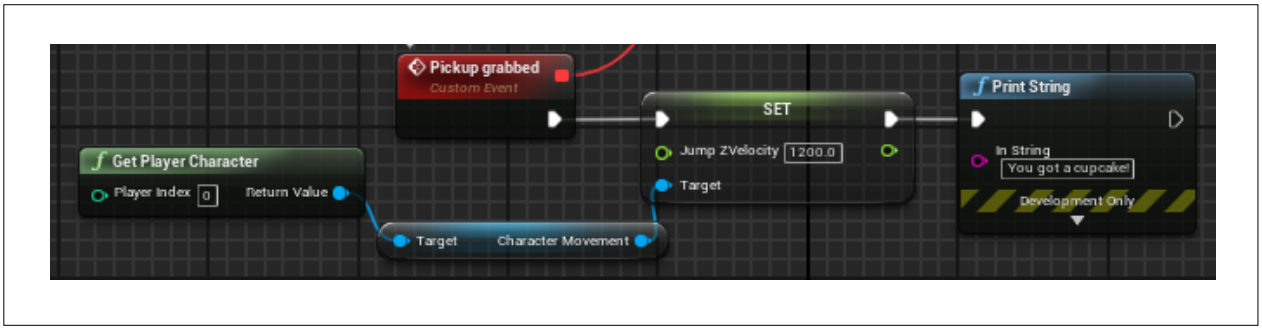
Grand Prix

For this week, we're going to add powerups and fire to your third-person platformer. This will give you the opportunity to change Levels as the player becomes more powerful and has more obstacles to avoid.

First, we'll be adding powerups.

1. Select the following Blueprints in the **FundamentalsClasses** project you downloaded for this course. You can find them in the Blueprints folder found here: **Content → Fundamentals → Blueprints → EssentialConcepts → FinishedAssets → Blueprints**.
    a. **BP_Pickup_Parent**
    b. **BP_Pickup_Child_Coin**
    c. **BP_Pickup_Child_Cupcake**
    d. **BP_Pickup_Child_Health**
2. Right-click on **Assets** after selecting them and choose "**Asset Actions → Migrate**".
3. All the assets referenced by the Blueprints will automatically be collected in the **Asset Report**. Click **Ok**.
4. Navigate to the **Content** folder for the third-person platformer you made and click **Select Folder**.
5. Check to make sure the pickups migrated successfully and close the **FundamentalsClasses** project. We won't be needing that project anymore at this point.
6. Add the three child Blueprints to your scene and test them to make sure they work. Run into them and see that they get smaller, go into your character, and pass a message in the upper-left corner of the screen.
7. After you see that the Blueprints work, delete them from your scene. They were placed temporarily just to ensure that everything migrated correctly.

8. Open the **BP_Pickup_Child_Cupcake** Blueprint.
9. In the Event Graph, delete the **Destroy Actor** node, and break the link between the **Pickup Grabbed** node and the **Print String** node. Using **Alt + Left-click** on a pin will break the connection between two nodes.
10. Place a **Get Player Character** node.
11. Drag a wire from the **Get Player Character** node's **Return Value** pin and place a **Get Character Movement** node.
12. Drag a wire from the **Character Movement** pin and place a **Set Jump Z Velocity** node. Connect the **Pickup Grabbed** node to the **Set Jump Z Velocity** node. Then connect the **Set Jump Z Velocity** node to the **Print String** node.
13. The default setting of **Jump Z Velocity** is "**600**". Assuming you didn't change it in the last exploration, set **Jump Z Velocity** to "**1200**". If you did change it, set **Jump Z Velocity** to double the value of what you changed it to.

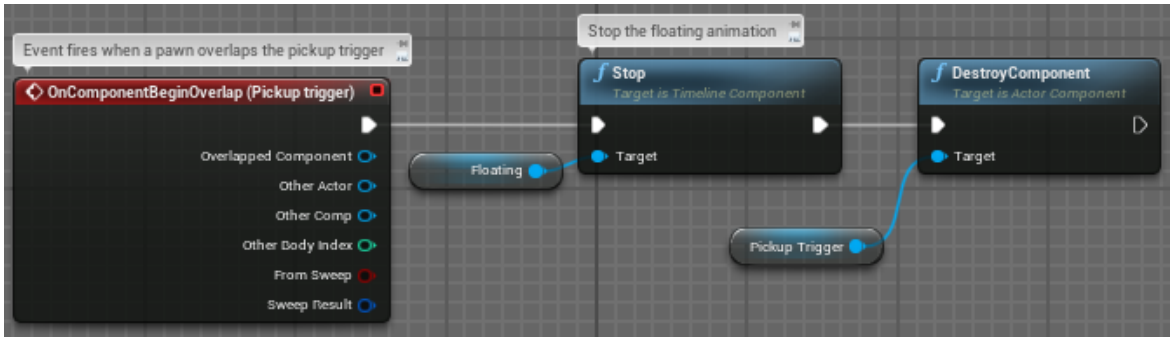


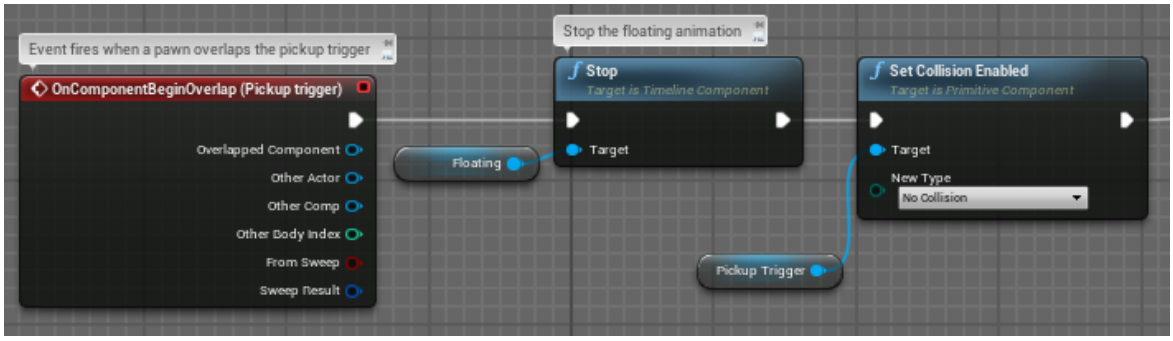
14. Compile the Blueprint and drag and drop the cupcake pickup into your scene. Test it! Your jump height should have doubled.
15. Drag a wire from a **Get Player Character** node's **Return Value** pin and type "**character movement**" in the search box to look at the list of variables you can set to change the movement of your character. Play around with other variables and see what kinds of powerups you can create.

At this point, some of you might notice a problem. When the player falls and respawns, our powerups don't respawn! If those powerups are needed to finish the Level, they need to respawn every time the player falls. To make that happen, we are going to use a new type of node: an **event dispatcher**. Event dispatchers can be found below the **Variables** section of a Blueprint Event Graph. Event dispatchers are a way for a Blueprint to listen for events that happen on other Blueprints. We actually used a special event dispatcher in the first Grand Prix—we bound the player respawning to an event dispatcher of the player being destroyed. When the player is destroyed, the event dispatcher **On Destroyed** sends out a message to all of the other Blueprints who are listening; because we bound the **On Destroyed** event in the **Game Mode** Blueprint, it listens when the player is destroyed and spawns a new player. We'll be building another event dispatcher here.

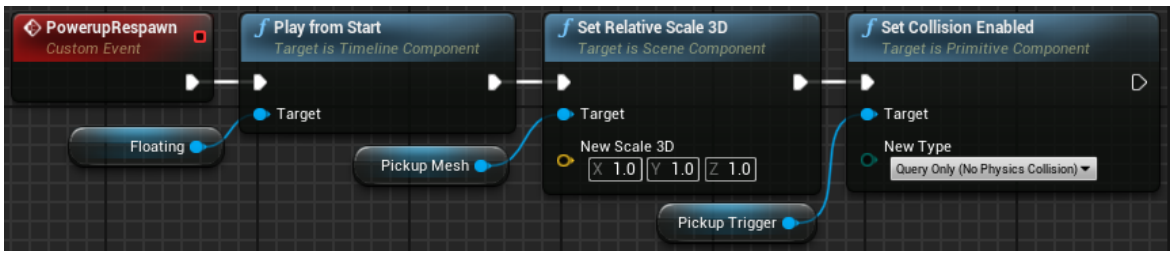
1. In the **ThirdPersonGameMode** Blueprint, find the **My Blueprint** panel on the bottom-left side of the window. Click the plus sign next to **Event Dispatchers**.
2. Name the event dispatcher "**Player Fell**".
3. In between the **OnDestroyed_Event_0** and the **Delay** nodes, add a node called "**Call Player Fell**". You'll notice this node has an envelope in the top-right corner. This indicates that it is sending out an event dispatcher. Don't forget to reattach the input execution pins.
4. Compile the Blueprint.
5. Open the **BP_Pickup_Parent** Blueprint. Look at the Event Graph and find the two animations: **Floating**, an animation for bobbing and rotating, and **Pickup**, an animation for when the player overlaps with the trigger. We need to reset these animations when the player dies.
6. In the bottom-left corner, in the **My Blueprint** panel, find the **Variables** section and click on the triangle to the left of **Components** to see a drop-down of all the components. Drag and drop **Floating** from that list and select "**Get**". This creates a reference to the Timeline animation.
7. Find the **Add Custom Event** node. Select it and name the event "**PowerupRespawn**".
8. Drag a wire from the **Floating** pin and add a **Play from Start** node. Connect the **Play from Start** node to the **PowerupRespawn** event you just created.

9. Next, add a **Set Relative Scale 3D** node and connect it to the **Play from Start** node. Set the **X**, **Y**, and **Z** values of New Scale 3D to "**1**, **1**, **1**". From the **Components** panel, drag and drop the **Pickup Mesh** into the Events Graph and connect it as the target.
10. Finally, we need to change the collision. Find the **Delete Component** node that's part of the **On Component Begin Overlap** event. Delete that node and replace it with a **Set Collision Enabled** node. Reattach the **Pickup Trigger** mesh and make sure **New Type** is set to "**No Collision**". Now the collision component won't be deleted but instead will just not register collision again unless it is re-enabled.

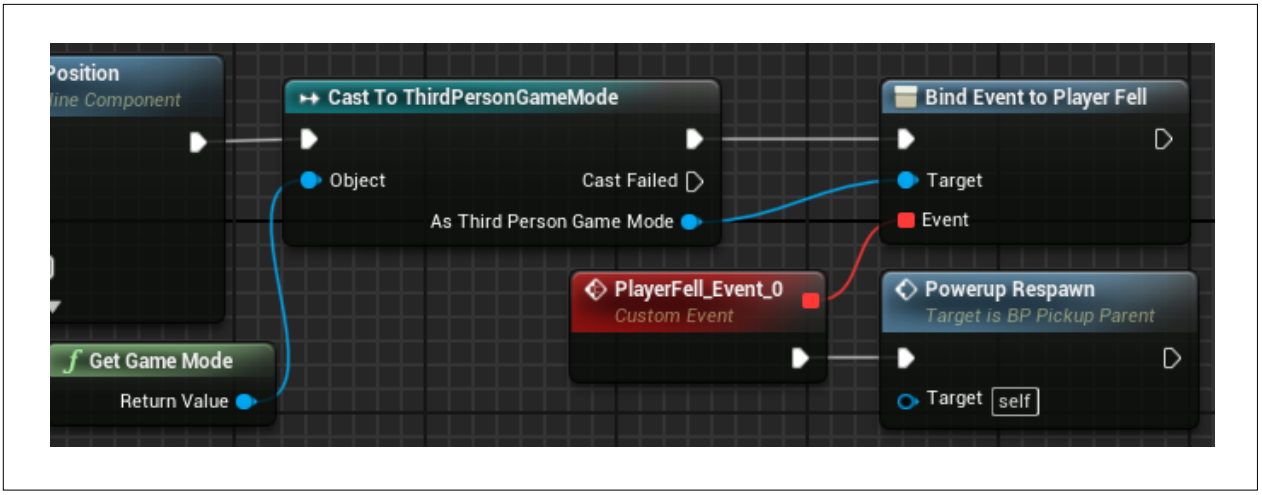




11. Add a **Set Collision Enabled** node to the end of the custom event you created. **New Type** should be set to "**Query Only**".



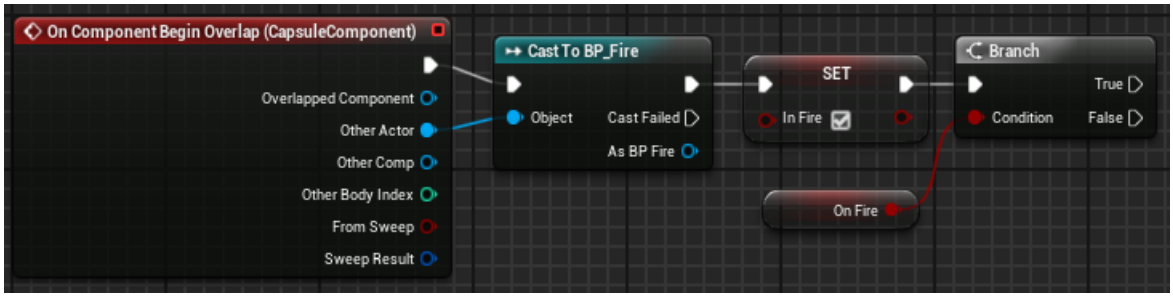
12. We're almost there. Now we just need to bind the event. Off the **Set Playback Position** node, which is connected to the **Event BeginPlay** node, place a **Cast To ThirdPersonGameMode** node. Drag a wire from the **Cast To ThirdPersonGameMode** node's **Object** pin and place a **Get Game Mode** node. Off the **Cast To ThirdPersonGameMode** node, place an **Assign Player Fell** node. Connect the **Bind Event to Player Fell** node's **Target** pin to the **Cast To ThirdPersonGameMode** node's **As Third Person Game Mode** pin.
13. Finally, off the **PlayerFell_Event_0** node, place a **Powerup Respawn** node. Compile the Blueprint.



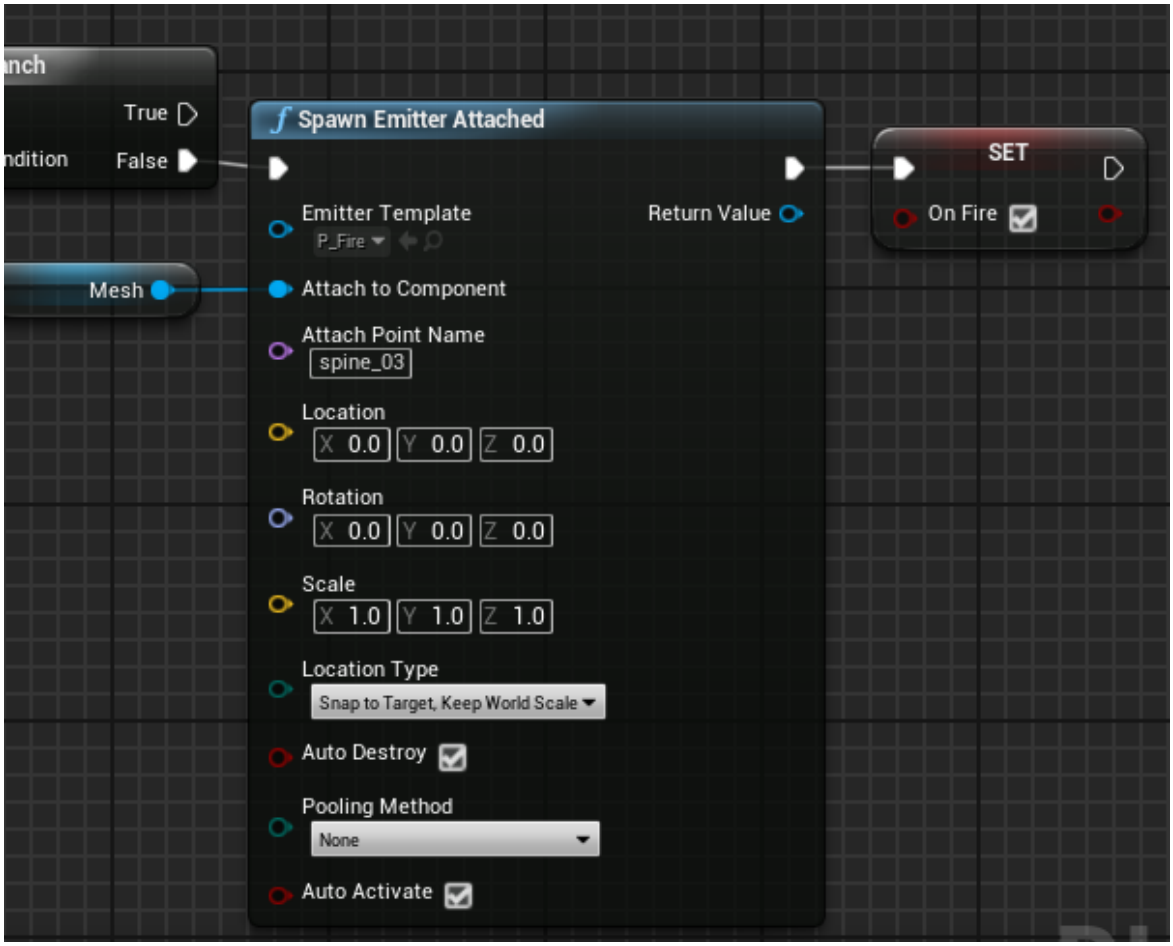
Now, whenever the player falls, the event dispatcher goes off and the parent pickup Blueprint hears it and runs the custom event **PowerupRespawn**.

With all of the new powerups you added, you now need more obstacles to avoid! Let's add fire to your game. Search the **StarterContent** folder for "**P_Fire**". The "P" here stands for "particle effect."

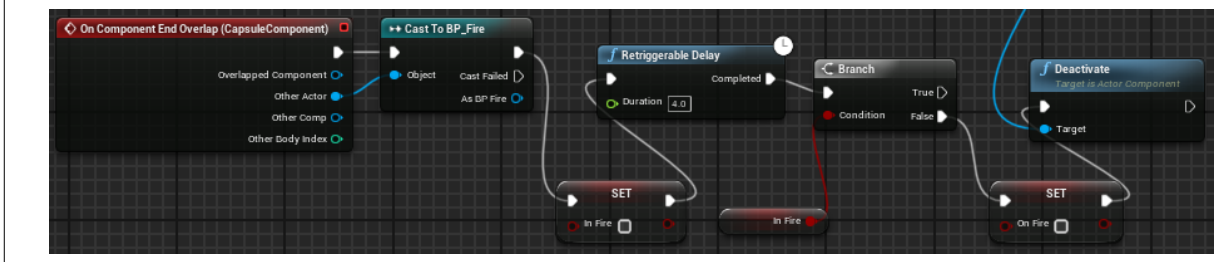
1. Create a Blueprint and name it "**BP_Fire**". Open it up and add a **Particle System** component.
2. In the **Details** panel, under "**Particles**", set **Template** to "**P_Fire**".
3. Add a **Sphere Collision** component to the Blueprint and set its position and size to encompass the fire.
4. Compile the Blueprint.
5. Drag the Blueprint onto the map to see it burn!
6. Open the **ThirdPersonCharacter** Blueprint.
7. Select the **Capsule** component in the **Components** panel, and then add an **On Component Begin Overlap (CapsuleComponent)** node to the Event Graph.
8. We want to check that the **Capsule** component is overlapping with the fire. Drag a wire from our newly created event and create a **Cast To BP_Fire** node, and then attach the **Overlap** event's **Other Actor** pin to the **Cast** node's **Object** pin.
9. We're going to add a variable. In the **My Blueprint** panel, find the **Variables** section and click the plus sign. Name the variable "**InFire**".
10. In the **Details** panel, set **Variable Type** to "**Boolean**". There are many different variable types available for us to use, but we just need a Boolean now. Compile the Blueprint.
11. Off the **Cast** node, we're going to set the **InFire** Boolean variable. Select "**InFire**" from the **Variables** list and drag and drop it into the Event Graph. Choose "**Set**" and check the box next to **InFire** so we are setting the variable to "**true**".
12. Create a Boolean variable called "**OnFire**". Don't forget to compile the Blueprint after creating the variable.
13. Off the **Set InFire** node, place a **Branch** node. A branch is the same as an IF statement. Drag and drop the **OnFire** variable into the Event Graph and select "**Get**". Attach the **OnFire** pin to the **Branch** node's **Condition** pin.



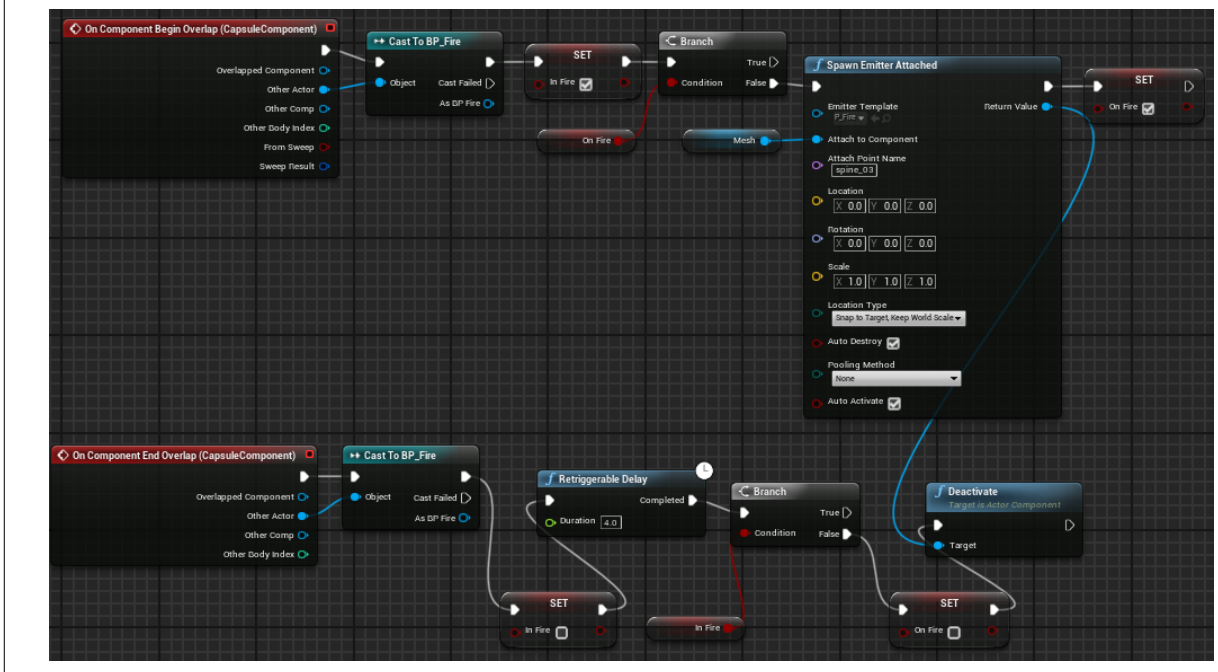
14. Drag a wire from the **Branch** node's **False** pin and add a **Spawn Emitter Attached** node.
15. Select and drag the **Mesh** from the **Components** panel into the Event Graph and attach the **Mesh** pin to the **Spawn Emitter Attached** node's **Attach to Component** pin.
16. Below the **Attach to Component** pin is the **Attach Point Name** pin. Instead of attaching a pin to it, click "**None**" and type in "**spine_03**". That is the socket on the skeleton where the fire will attach.
17. Set **Location Type** to "**Snap to Target, Keep World Scale**" and check **Auto Destroy** and **Auto Activate**.
18. Drag a wire from the **Spawn Emitter Attached** node and create a **Set OnFire** node. Check the box.

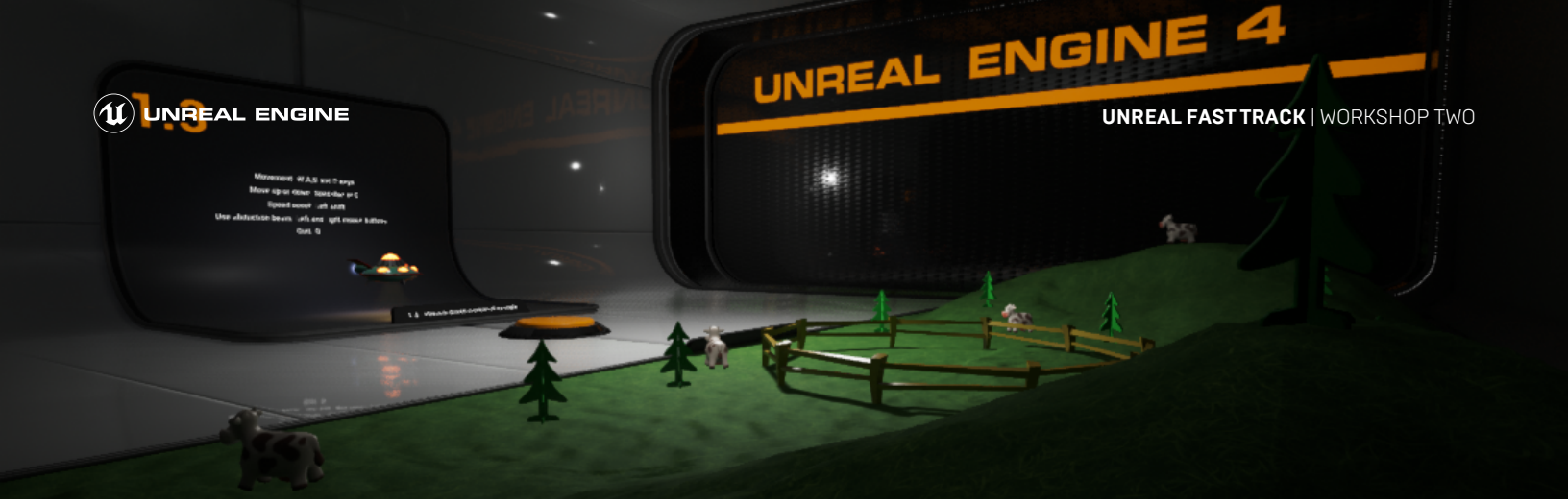


19. Now let's set up what happens when the player exits the fire. Select the **Capsule** component in the **Components** panel again, and this time add an **On Component End Overlap (CapsuleComponent)** node to the Event Graph.
20. Create a **Cast To BP_Fire** node and attach it in just the same way as you did with the **On Component Begin Overlap** node above. Then create a Set InFire node, but don't check the box this time. Now we are setting **InFire** to "**false**".
21. Off the **Set InFire** node, place a **Retriggerable Delay** node and set **Duration** to "**4**".
22. Place a **Branch** node off the **Retriggerable Delay** node, and then connect the **Branch** node's **Condition** pin to a **Get InFire** node.
23. Create a **Set OnFire** node off the **Branch** node's **False** pin and set **OnFire** to "**false**" by not checking the box.
24. Finally, drag a wire from the **Spawn Emitter Attached** node's **Return Value** pin and place a **Deactivate** node. Connect the **Deactivate** node to the **Set OnFire** node. Compile the Blueprint.



25. Place some fire in your Level and test what happens by running into it.
26. Add the powerups and fire to your Level, changing the design to suit the new powers and obstacles the player now has. In the next workshop, we will make sure that fire does some damage.





Off-Roading and Discussion

In the **Content Browser**, search for the Level called "**Indoor**". In it, you will find a blank demo room, with the stands and nothing else. Applying what you've learned and with a bit of exploration, try making your own Blueprint demos. Explore using different nodes and components in the **EssentialConcepts_Start** Level, and even poke around at the other Levels within the project. Don't be afraid to copy and paste node networks between Blueprints, try things, break things, and put things back together. You can find most of the Blueprints you have seen in the course in the folder found here: **Content** → **Fundamentals** → **Blueprints** → **EssentialConcepts** → **FinishedAssets** → **Blueprints**. You can find some more Blueprints in the folder at **Content** → **DemoRoom** → **BluePrint**. The point here is not to create a stunning perfect masterpiece, but to explore the system.

Once you've created a variety of Blueprints in your demo room, show them to your teammates and discuss what you made. As usual, use the three **D**'s to drive the discussion.

- **Difficulties**: What was the most difficult or confusing part of this week's workshop?
- **Discoveries**: What were some interesting, fascinating, or exciting things you discovered while making your demo Blueprints?
- **Dreams**: What ideas for mechanics and systems have popped into your mind while doing this week's workshop? Discuss how you might be able to create those with your current knowledge.

CONGRATULATIONS!

You've finished Workshop Two of the Fast Track!

Dauntless | Phoenix Labs