unity

4.1 Watch Where You're Going

Steps:

Step 1: Create project and open scene

Step 2: Set up the player and add a texture

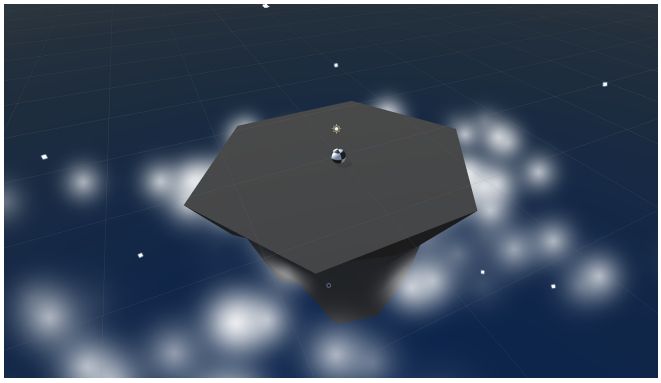
Step 3: Create a focal point for the camera

Step 4: Rotate the focal point by user input

Step 5: Add forward force to the player

Step 6: Move in direction of focal point

Example of project by end of lesson



Length: 60 minutes

Overview: First thing's first, we will create a new prototype and download the starter files! You'll notice a beautiful island, sky, and particle effect... all of which can be customized! Next you will allow the player to rotate the camera around the island in a perfect radius, providing a glorious view of the scene. The player will be represented by a sphere, wrapped in a detailed texture of your choice. Finally you will add force to the player, allowing them to move forwards or backwards in the direction of the camera.

Project Outcome: The camera will evenly rotate around a focal point in the center of the island, provided a horizontal input from the player. The player will control a textured sphere, and move them forwards or backwards in the direction of the camera's focal point.

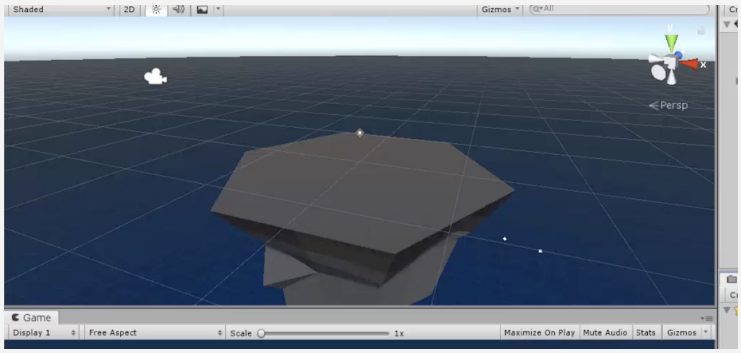
Learning Objectives: By the end of this lesson, you will be able to:
- Apply Texture wraps to objects
- Attach a camera to its focal point using parent-child relationships
- Transform objects based on local XYZ values

Step 1: Create project and open scene

You've done it before, and it's time to do it again... we must start a new project and import the starter files.

1. Open **Unity Hub** and create an empty "Prototype 4" project in your course directory on the correct Unity version.
   If you forget how to do this, refer to the instructions in Lesson 1.1 - Step 1
2. Click to download the Prototype 4 Starter Files, **extract** the compressed folder, and then **import** the .unitypackage into your project.
   If you forget how to do this, refer to the instructions in Lesson 1.1 - Step 2
3. Open the **Prototype 4 scene** and delete the **Sample Scene** without saving
4. Click **Run** to see the **particle effects**

- **Don't worry:** You can change texture of floating island and the color of the sky later
- **Don't worry:** We're in isometric/orthographic view for a reason: It just looks nicer when we rotate around the island

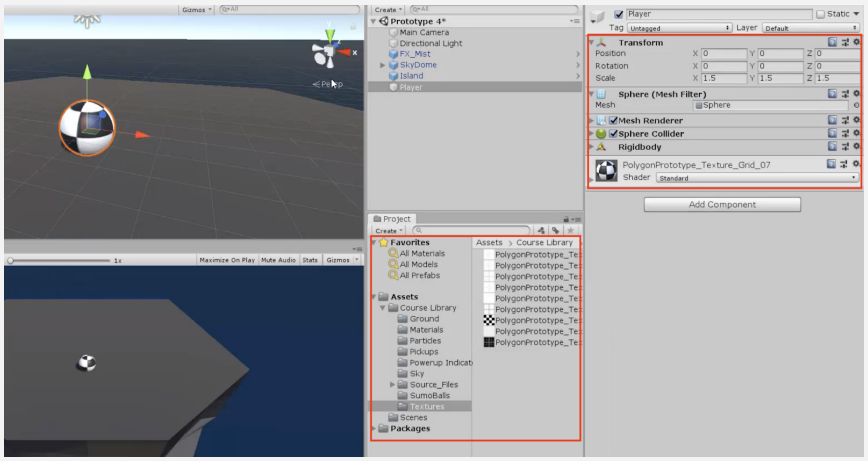


Step 2: Set up the player and add a texture

We've got an island for the game to take place on, and now we need a sphere for the player to control and roll around.

1. In the **Hierarchy**, *create 3D Object* > ***Sphere***
2. Rename it "Player", reset its **position** and increase its XYZ **scale** to 1.5
3. Add a **RigidBody** component to the **Player**
4. From the *Library* > *Textures*, drag a **texture** onto the **sphere**

- **New Concept:** Texture wraps

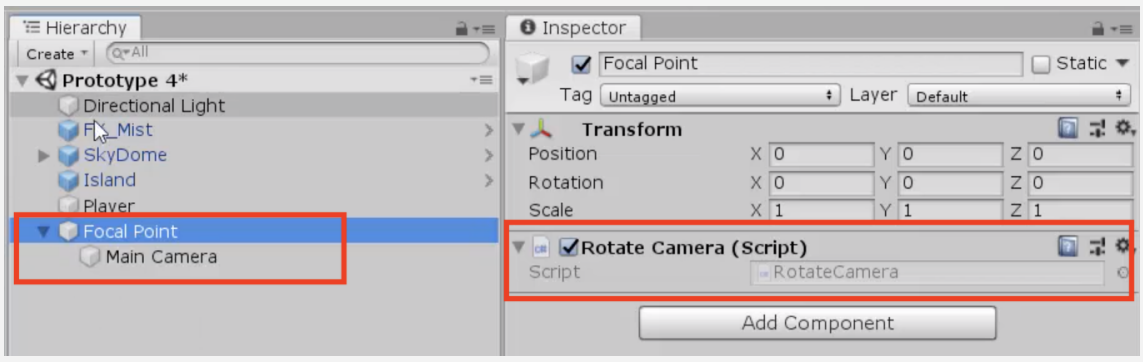


Step 3: Create a focal point for the camera

If we want the camera to rotate around the game in a smooth and cinematic fashion, we need to pin it to the center of the island with a focal point.

1. Create a new **Empty Object** and rename it "Focal Point",
2. Reset its position to the origin (0, 0, 0), and make the Camera a **child object** of it
3. Create a new "Scripts" folder, and a new "RotateCamera" script inside it
4. **Attach** the "RotateCamera" script to the **Focal Point**

- **Don't worry:** This whole "focal point" business may be confusing at first, but it will make sense once you see it in action
- **Tip:** Try rotating the Focal point around the Y axis and see the camera rotate in scene view



Step 4: Rotate the focal point by user input

Now that the camera is attached to the focal point, the player must be able to rotate it - and the camera child object - around the island with horizontal input.

1. Create the code to rotate the camera based on ***rotationSpeed*** and ***horizontalInput***
2. Tweak the ***rotation speed*** value to get the speed you want

- **Tip:** Horizontal input should be familiar, we used it all the way back in Unit 1! Feel free to reference your old code for guidance.

```
public float rotationSpeed;

void Update()
{
  float horizontalInput = Input.GetAxis("Horizontal");
  transform.Rotate(Vector3.up, horizontalInput * rotationSpeed * Time.deltaTime);
}
```

Step 5: Add forward force to the player

The camera is rotating perfectly around the island, but now we need to move the player.

1. Create a new "PlayerController" script, apply it to the **Player**, and open it
2. Declare a new ***public float speed*** variable and initialize it
3. Declare a new ***private Rigidbody playerRb*** and initialize it in ***Start()***
4. In ***Update()***, declare a new ***forwardInput*** variable based on **"Vertical"** input
5. Call the **AddForce()** method to move the player forward based ***forwardInput***

- **Tip:** Moving objects with RigidBody and Addforce should be familiar, we did it back in Unit 3! Feel free to reference old code.
- **Don't worry:** We don't have control over its direction yet - we'll get to that next

```
private Rigidbody playerRb;
public float speed = 5.0f;

void Start() {
  playerRb = GetComponent<Rigidbody>(); }

void Update() {
  float forwardInput = Input.GetAxis("Vertical");
  playerRb.AddForce(Vector3.forward * speed * forwardInput); }
```

Step 6: Move in direction of focal point

We've got the ball rolling, but it only goes forwards and backwards in a single direction! It should instead move in the direction the camera (and focal point) are facing.

1. Declare a new ***private GameObject focalPoint;*** and initialize it in ***Start()***: *focalPoint = GameObject.Find("Focal Point");*
2. In the ***AddForce*** call, Replace ***Vector3.forward*** with ***focalPoint.transform.forward***

- **New Concept:** Global vs Local XYZ
- **Tip:** Global XYZ directions relate to the entire scene, whereas local XYZ directions relate to the object in question

```
private GameObject focalPoint;

void Start() {
  playerRb = GetComponent<Rigidbody>();
  focalPoint = GameObject.Find("Focal Point"); }

void Update() {
  float forwardInput = Input.GetAxis("Vertical");
  playerRb.AddForce(~~Vector3.forward~~ focalPoint.transform.forward
  * speed * forwardInput); }
```

Lesson Recap

New Functionality	• Camera rotates around the island based on horizontal input • Player rolls in direction of camera based on vertical input
New Concepts and Skills	• Texture Wraps • Camera as child object • Global vs Local coordinates • Get direction of other object
Next Lesson	• In the next lesson, we'll add more challenge to the player, by creating enemies that chase them in the game.