CS50's
Introduction to Game Development

OpenCourseWare

Colton Ogden (https://www.linkedin.com/in/colton-ogden-0514029b/)
cogden@cs50.harvard.edu

David J. Malan (https://cs.harvard.edu/malan/)
malan@harvard.edu
(https://www.facebook.com/dmalan) (https://github.com/dmalan)
(https://www.instagram.com/davidjmalan/) (https://www.linkedin.com/in/malan/)
(https://orcid.org/0000-0001-5338-2522) (https://www.quora.com/profile/David-J-Malan) (https://www.reddit.com/user/davidjmalan)
(https://www.tiktok.com/@davidjmalan) (https://davidjmalan.t.me/)
(https://twitter.com/davidjmalan)

Lecture 10: Portal

Today's Topics

- Holding a Weapon
  - We used a First-Person Controller last week, and we'll explore that a bit further today with "Parenting."
- Raycasting
  - We'll see how to shoot a ray from our weapon!
- RenderTexture
  - This is essentially a texture that we're rendering to with a camera.
- Texture Masking
  - How will we actually create the portal effect? This is where Texture Masking comes in.
- Decals
  - Decals are just something that you fix to a surface - in this case, our portals will be

decals.

- Teleporting
  - We'll see how to implement teleporting, which will require us to overwrite some default behavior for the First-Person Controller.
- ProBuilder and ProGrids
  - We'll take a look at some additional tools that Unity offers us, which will allow us to model geometry in each scene.

Downloading demo code

- github.com/games50/portal (https://github.com/games50/portal)

Holding a Weapon

- The main issues to tackle when it comes to having our first-person character hold a weapon are: getting a weapon Prefab, actually holding the weapon, and keeping the weapon in front of the FPS camera.
- As you might've guessed, we can easily find a weapon Prefab from the Asset Store, and holding the weapon is as easy as positioning it in front of our character... but how might we track the weapon's movement such that it's consistent with our character's camera?
  - The answer is quite simple, actually, as one solution might be to simply make the weapon object a Child of the character object. This automatically assigns the Transport component of the weapon to mirror that of the character's Transport component, which includes translations, rotations, etc.

Raycasting

- Raycasting is a nice feature that Unity provides for free.
- It's part of the physics namespace and scripting API, and what it allows us to do is shoot "rays" (i.e., straight lines), given that we provide a Transform as the source for the ray's direction vector (in our case, we're using `Transform.forward`, which is a straight line in the forward direction from the center of our camera).
  - The casting is done by using `Physics.Raycast`, which requires the declaration of a `RaycastHit` struct object to store the collision information (e.g., was there a hit? Where was the hit? What was the collision angle?, etc.).
- Additionally, Unity provides a `Debug.DrawRay` function that is helpful for debugging purposes, as it allows us to actually draw the rays onto the scene and check whether

they're working properly.

RenderTexture

- Recall that a RenderTexture is simply a texture in Unity (can be created as an asset) that can additionally be rendered to.
    - For example, a camera can be attached to a RenderTexture. This is how we create our portals!
- Once we create a RenderTexture using the Unity editor, we can render a camera to it by selecting the camera we'd like to render to it from our Scene dropdown, which will display the camera's settings on the right-most column, and then drag our RenderTexture from the Textures folder to the camera's "Target Texture" setting.

Texture Masking

- As we see in our game, our portals are round, but almost everything else in the game world is rectangular.
- Since our portals are supposed to reflect back an image of the game world, we have to use Texture Masking in order to properly render our rectangular world within the camera view of our round portals.
- This means that we must somehow select pixels from our other textures that we do want to render in our portal textures, as well as pixels that we don't want to render.
- To do this, we have to cheat a little bit and select a rectangular texture for our portals, such that we make the edges invisible using a shader and display only an oval-shaped interior.
- This makes it easy to map which pixels we want to display on our portal camera.

Teleporting

- To implement teleporting when the player enters a portal, we simply create a Mesh Collider on the PortalMesh.
- If we then define this collider as a trigger, we can implement the behavior we want within an `OnTriggerEnter` callback in the `Portal` script.
- It's then simple enough to simply change the player's position to "teleport" them to the desired location, making sure to specify a `linkedPortal` if we want to teleport from one portal to another, and setting a `portalActive` flag so that we're not stuck infinitely teleporting from one place to the other.

ProBuilder and ProGrids

- ProBuilder and ProGrids are two new tools that you can use to create geometry within the Unity scene editor, which eliminates the need for using external 3rd party applications when developing games in Unity.
- As you might imagine, this allows you to immediately test out your creations within the Unity scene without needing to waste time importing them from somewhere else.
- ProBuilder tutorial:
  - youtube.com/watch?v=PUSOg5YEflM (https://www.youtube.com/watch?v=PUSOg5YEflM)
- ProGrids tutorial:
  - youtube.com/watch?v=UtNvtIrJcNc (https://www.youtube.com/watch?v=UtNvtIrJcNc)
- To install ProBuilder and ProGrids, a quick search in the Asset Store will suffice.
- ProBuilder and ProGrids are particularly good for creating levels for 3D games, so play around with them to see all the features they have to offer!

Shader Graph

- In a similar vein, do check out Shader Graph, another new tool offered in Unity 2018.1, which you might find useful moving forward!
  - blogs.unity3d.com/2018/02/27/introduction-to-shader-graph-build-your-shaders-with-a-visual-editor (https://blogs.unity3d.com/2018/02/27/introduction-to-shader-graph-build-your-shaders-with-a-visual-editor/)