Module 2.0

Getting Started With VR

Introduction

We'll now get started exploring some VR elements with Unity. We'll be introducing you to VR functionality using the SteamVR toolkit by incorporating some of its functionality into your Roll-A-Ball game.

1 VR-izing Roll-a-Ball

To get started with VR development, we'll add some VR interactions and elements to your Roll-a-Ball game.

1.1 Importing SteamVR

1. Open your Roll-a-Ball project
2. Go to the Asset Store and search/download/import SteamVR
3. Setup the SteamVR Inputs by going to `Window` → `SteamVR Input`
   a) Have it generate a default `actions.json` file
   b) Click `Save and generate`
      We'll come back to this system later.
4. Make your game a VR game:
   a) You may want to rename your ball to `Ball` instead of `Player` (in fact, we recommend reorganizing all of your Roll-a-Ball game objects under one empty parent object to organize them better).

b) Drag and drop the SteamVR Player Prefab into your scene:

   `SteamVR/InteractionSystem/Core/Prefabs`

   This is an all-in-one prefab that includes all the necessary element for the headset, left and right "hands" and other SteamVR elements.

c) Disable (or remove) the `Main Camera`. As a VR game, you no longer need a camera for the *screen* since the headset will now be our camera.

You can try to test your game at this point by playing it. If you don't have a headset (or you just want to not use it), you can fall back to "2D Debug" mode in which you can still interact with the scene in Unity:

- Right click/drag can move your head
- Arrow keys can move you around in the VR space
- Left click can act as your *interactive "hand"*
- You can use T to teleport (but we need to do the next step first)

1.2 Moving Around

Various VR games and experiences allow you to move around in the environment in several ways. Some allow you to "walk" by shuffling your arms, others by using the touch pad. However, the easiest and least vomit-inducing way is to allow a user to teleport. Let's make your game's play area teleportable.

1. Drag the Teleporting Prefab into your scene:
   `SteamVR/InteractionSystem/Teleport/Prefabs/Teleporting`
2. Make your game "plane" teleportable. To do this, we recommend:
   a) Create a transparent plane (as a child) to your game plane. To do this, click your game plane (Ground) and add a new plane to it. Since it is a child, the default 1-1 scale should cover the entire surface.

   b) Rename the child plane `TeleportArea`

   c) Make the plane transparent by disabling the Mesh Render

   d) "Float" this plane by changing its y position to be 1 centimeter (0.01 units) above the plane (we do it this way to avoid clipping/render anomalies when the teleporting surface is rendered).

   e) Drag the TeleportArea script:
   `SteamVR/InteractionSystem/Teleport/Scripts/TeleportArea.cs` onto your floating plane.
3. You can now teleport in VR mode and in 2D Debug mode (though this is not all that useful as you can only ever teleport a few meters directly in front of you).

1.3 Play With Your Ball

Now let's interact with one of the game elements. When your "hands" interact with VR elements you have many different ways they can interact. By default, your hands can interact with elements that have a rigid body *unless* it is kinematic. For example:

- Your hand can touch the player ball and even move it (in fact, the hand will animate slightly so it doesn't clip with the ball).
- Your hand will not interact with (and will simply move through the pickup items (which are kinematic)
- Your hand *will* interact with the walls but in a non-intuitive way: the rendered hand object will not go through the wall even though your *real* hand moves down/into the wall. Once the controller has been removed from the wall, the rendered hand snaps back to it. This is because the wall has no rigid body, but *does* have a box collider that your hand is interacting with.

Let's make it so that the player can pick up and throw the ball. To do this simply drag and drop the Throwable script:

`SteamVR/InteractionSystem/Core/Scripts/Throwable.cs`

onto the player ball. In 2-D Debug mode, you can use right-click to pick it up (but not really throw it as a mouse has no 3-D direction you can only move it up/down and left/right). In full VR mode, you should be able to pick it up and throw it.

Adding this script actually adds two scripts. The other script is an `Interactable` and is a basic script used by many others to make a game element interactive with the VR player.

1.4 Move Your Ball

In this section we'll make it so we can move our ball using the VR wand instead of the keyboard arrow keys. To do this, we're going to steal some assets from the SteamVR toolkit.

1. Open the showcase/demo scene:

   `SteamVR/InteractionSystem/Samples/Interactions_Example`

   There's lots of cool stuff in here, but for now let's focus on the "Joe Jeff" controllable robot. If you're in VR, go ahead and try it out (the controller can be picked up and used to make a little robot "walk" around)

2. Under the `Remotes` game object, highlight `Table_Round`, `JoeJeffController` and `JoeJeffControllerSnapLocation` and copy them (control-c).

3. Go back to your Roll-a-Ball scene and create an empty game object named `Controller`

and paste the three assets as children of this element. Move the parent to somewhere within your game space.

4. The controller currently controls the Joe Jeff robot, not our ball. We'll need to modify the stolen script to interact with our ball instead of the (now non-existent) robot.

   However, we don't want to screw up the original script. In general you **do not want to modify any library scripts in SteamVR**. You'll want to make a copy and modify the script instead.

   a) Find the `JoeJeffController` script and drag a *copy* into your own script folder.

   b) Rename the file `BallController`

   c) Open the script and edit it so that it looks like the following: https://gist.github.com/739df91f286ae4fccfab4158d45f82d3 (in fact, you can copy-paste the contents if you like). Read through the script comments to understand what it does and how it does it.

   d) Click the controller (you can also rename that if you wish) and disable (or remove) the original script and add the new `BallController` script.

   e) Drag/drop the Joystick child into the script's joystick reference.

   f) Try testing your script in VR

2 Interactive System

The SteamVR toolkit provides the vast majority of what you'll ever need to develop in VR, but the documentation is not all that great and tutorials not that plentiful.[1] Valve's philosophy has been instead to provide examples in the Interactive scene:

`SteamVR/InteractionSystem/Samples/Interactions_Example`

with the expectation that you borrow, adapt and reverse engineer what you need.

One thing that does have some decent documentation is the Input system which is an attempt to abstract away from lower-level functionality (such as a button press) and use more general *gestures* such as "Move" and "Jump" used in our example above. To get more familiar with the input system:

1. Read through the provided tutorial:

   https://valvesoftware.github.io/steamvr_unity_plugin/tutorials/SteamVR-Input.html

2. Watch the video version of this tutorial:

[1] You'll find most are obsolete anyway, dealing with SteamVR 1.2 or earlier.

https://www.youtube.com/watch?v=qo-9CmcKWlY

3 Teaming Up

To get some practice borrowing and adapting from SteamVR, team up with another partner. This should be a different partner than you teamed up with in Module 1.0 in order to get to know more people and to cross-pollinate knowledge.

Once teamed up, work on only one of your projects. Choose which of your projects to continue working on and add your partner as a collaborator as in module 1.0). Then, work together to brainstorm a different VR-oriented game mechanic and implement it. Be creative and build something cool. If you're at a loss for ideas, here are some suggestions to get you started:

- Integrate the bow and arrow to shoot pickup items instead of using the ball. Have pickups regenerate or move to make it more challenging
- Winterize! Create a pile of (regenerating) snow balls that the player throws at the pickup items instead of using the ball. Animate them to "splat" if the user misses and be sure to destroy the game objects after animation or after a certain period of time (think coroutines!)
- Change the mechanics of the controller to tilt the ground instead of the ball to move it (you'll need to cover the surface with "glass" or limit the tilt so the so the ball doesn't roll out)

Show your final project to the instructors and move on to module 3.0!