Hack@Brown's Unity 3D Starter Pack

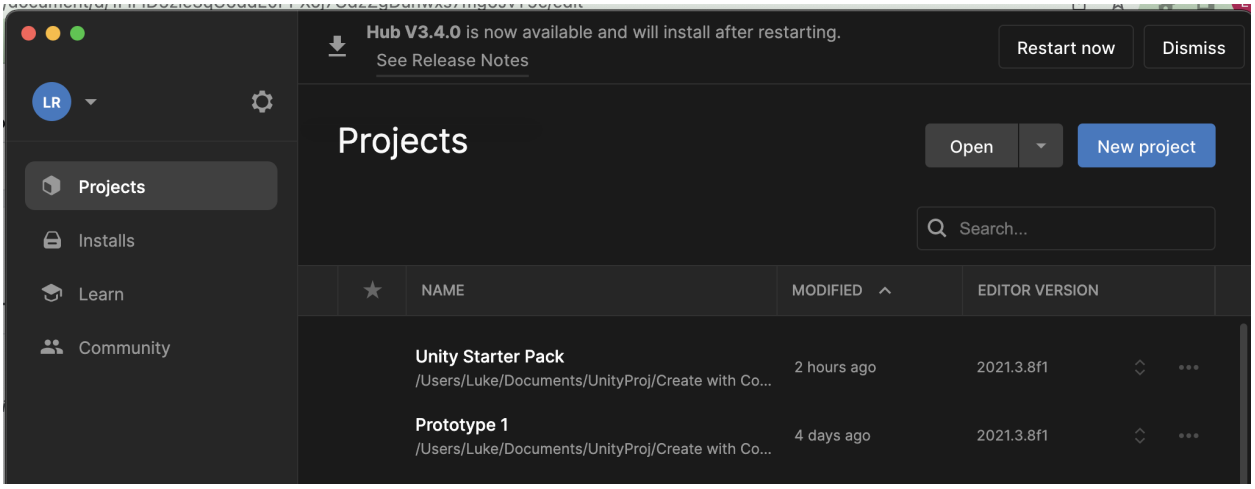
This starter pack includes a minimal demo project, a manual to help navigating the software/project, and a list of resources to help.

This demo project is quite minimal – its purpose is to help demonstrate how the different parts of Unity fit together without being overly flushed out or complicated.

[Download Unity]

[Download Demo Project]

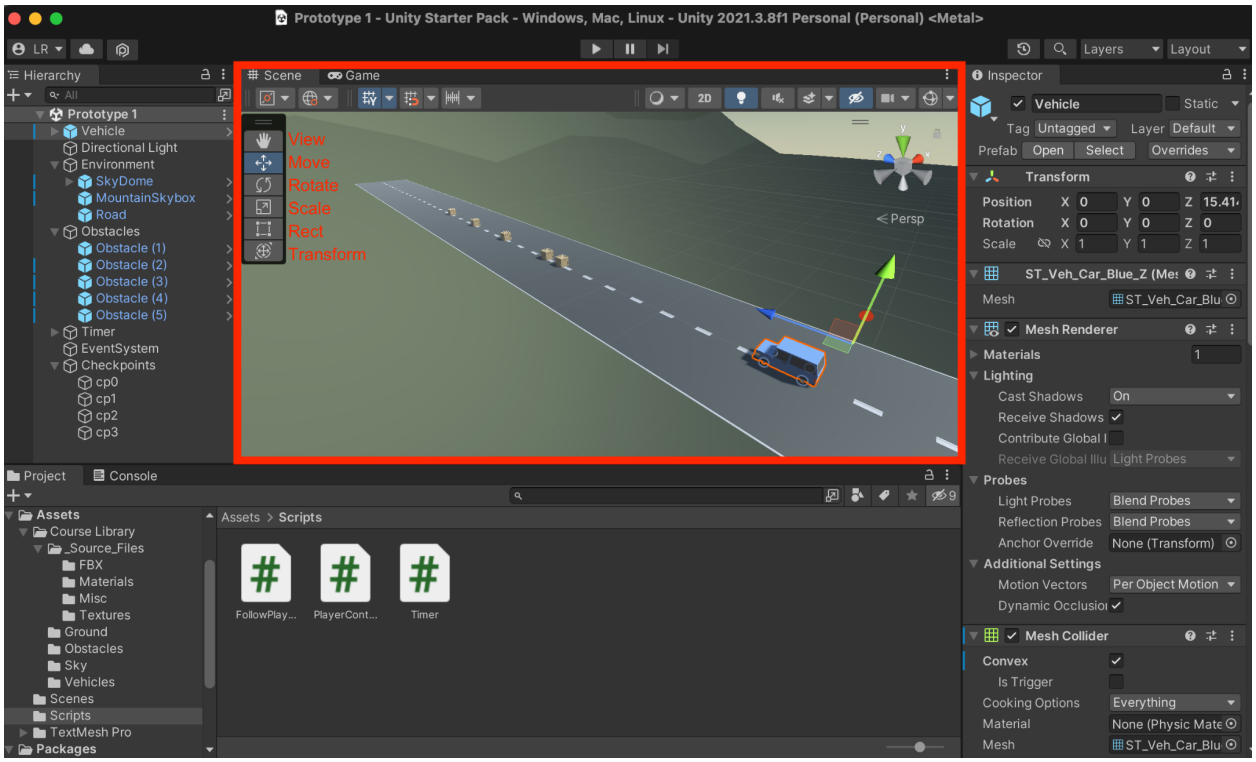
After installing Unity, unzip DemoProject.zip. Open Unity – under Projects, click Open, then clicking on the DemoProject folder to open the project.



The scripts folder in this repository are located in both the demo project zip and in a separate folder, if you wish to use them in your own projects.

Navigating Unity

Scene



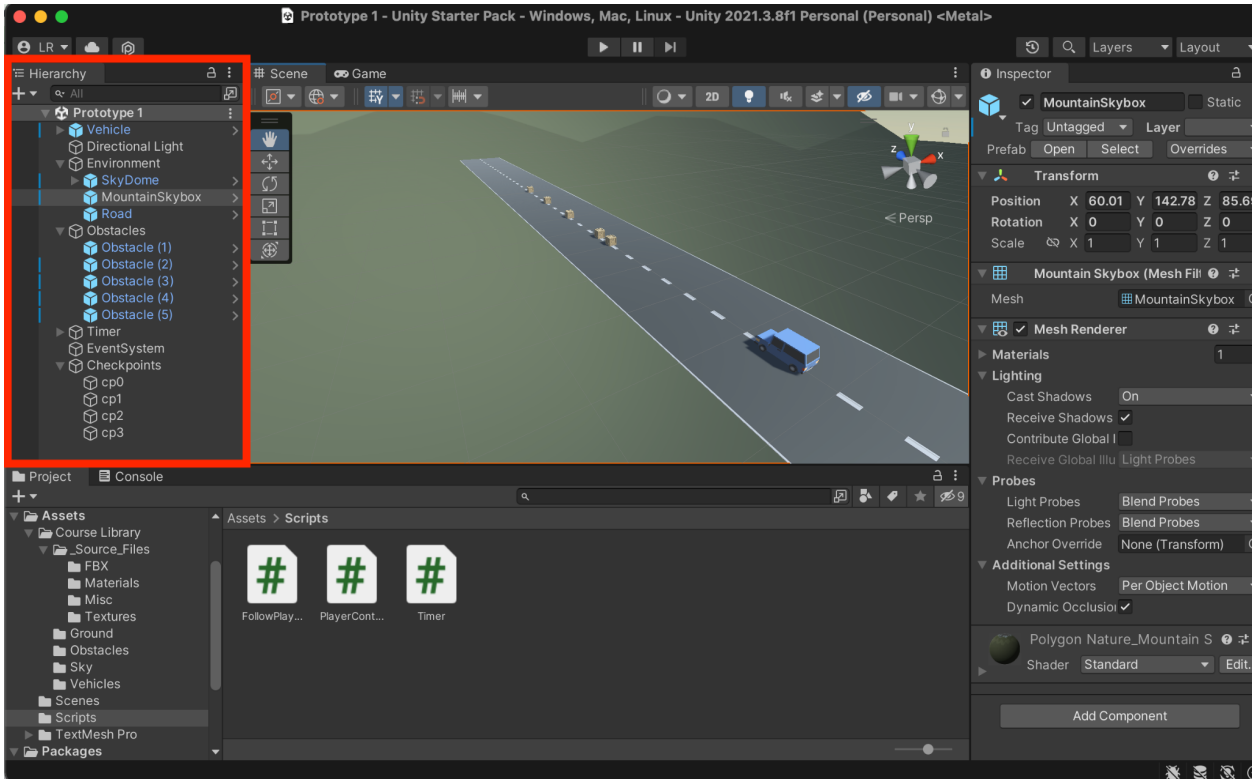
When you first open the demo project, your eyes will likely first go to the Scene view. The scene view displays the internal camera's perspective of the current scene (think of scenes like levels).

The toolbar on the left gives you access to the following tools:

- View tool: when selected, you can drag your mouse to pan around the scene. Holding right click while moving your mouse lets you look around the scene. Scrolling your mouse wheel zooms in and out. Another way to navigate through the scene is using WASD keys like arrow keys to fly around, while holding right click to point look around.
    - Tip: using Unity with a mouse instead of a trackpad makes things much easier because of this!
- Move tool: when selected, you can click on any GameObject and move it around the scene on all 3 axes.
- Rotate tool: when selected, you can click on any GameObject and rotate it on all 3 axes.

- Scale tool: when selected, you can click on any GameObject and scale (size it up or down) on any or all 3 axes.
- Rect tool: when selected, you can scale any 3D object on 2 axes at once.
- Transform tool: when selected, lets you move, rotate, and scale a selected GameObject all at once.

Hierarchy

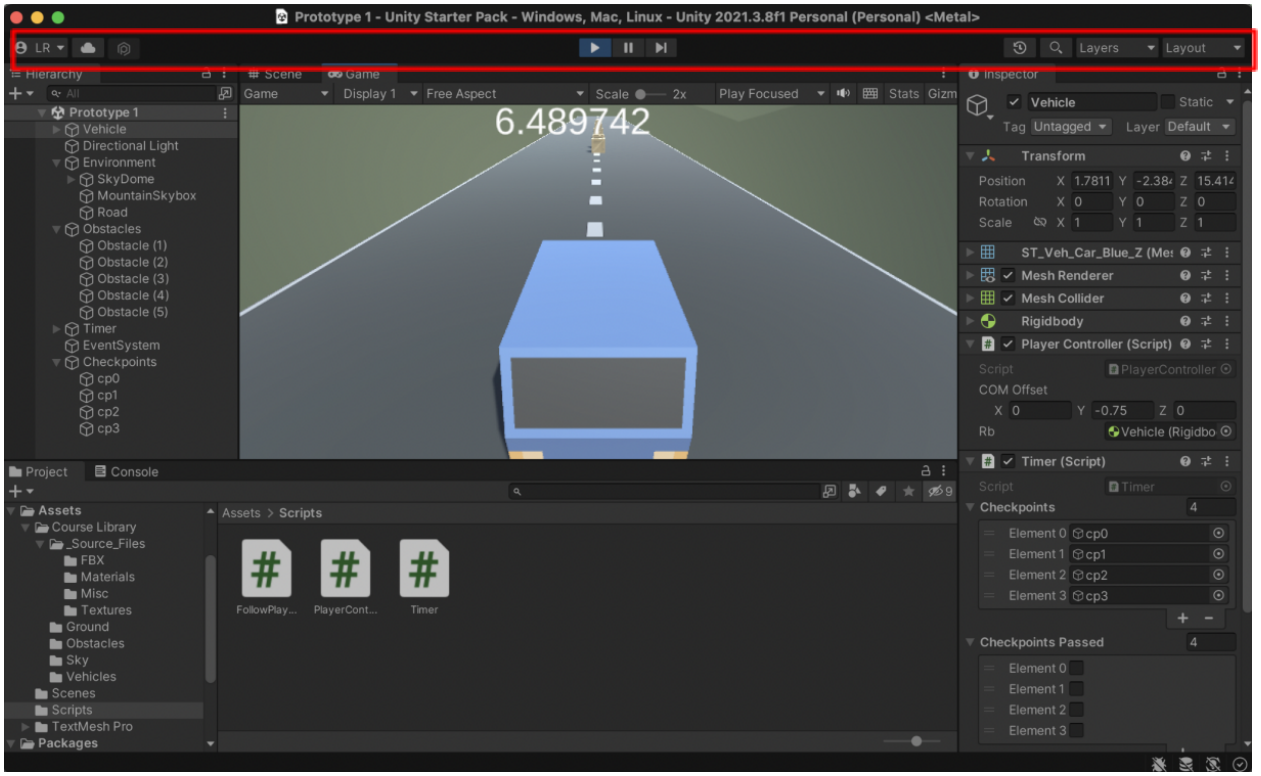


Your hierarchy displays every GameObject in your current scene. Here, the demo project's only scene is "Prototype 1". Clicking on an element in the hierarchy selects it in your scene view as well. This is helpful when you can't find an element in your scene view. Additionally, pressing F while having an object selected brings the camera right to it in your scene.

Also note that it's organized in a, well, hierarchical structure: you can drag any object onto another to be group it under that element. Clicking the triangle next to an object's name reveals all the elements grouped under it.

The '+' sign in the top left lets you add a new element to your scene.

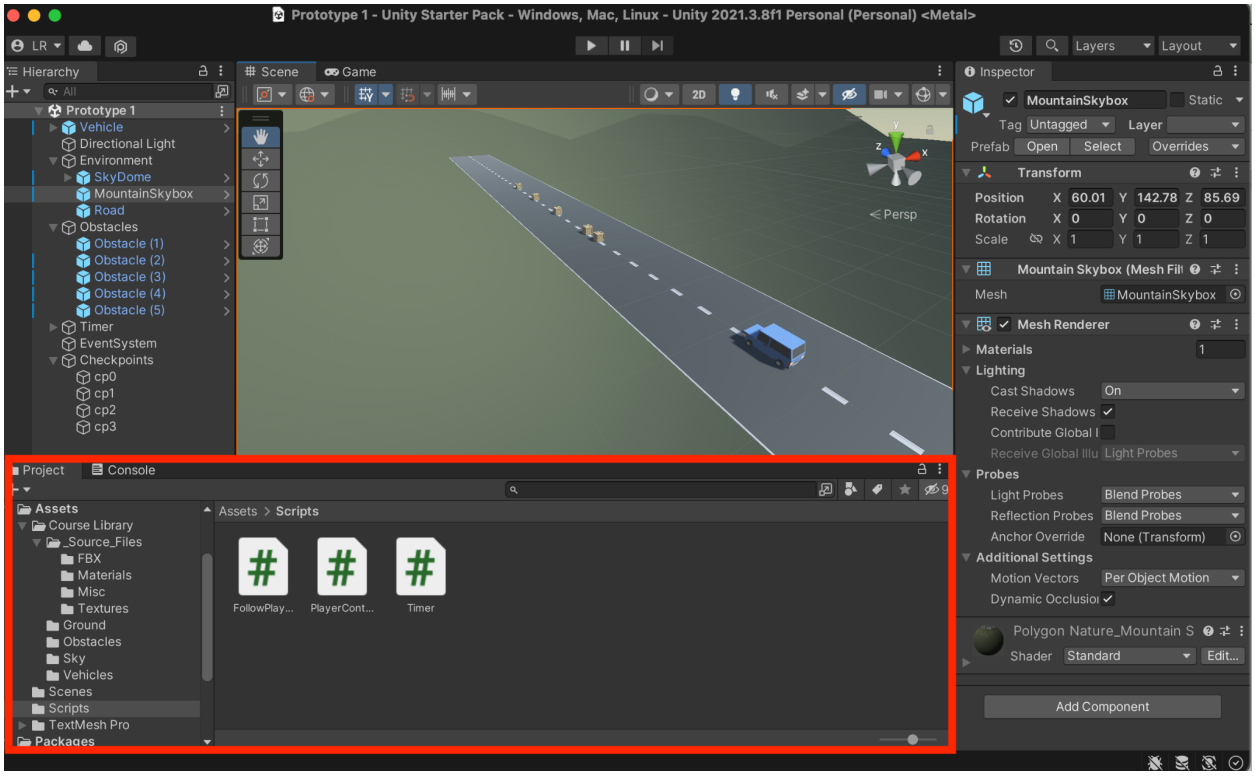
Toolbar Window



This is the most important window in Unity Editor. On the left side has some primary tools to manipulate the scene view and the objects that the scene view contains.

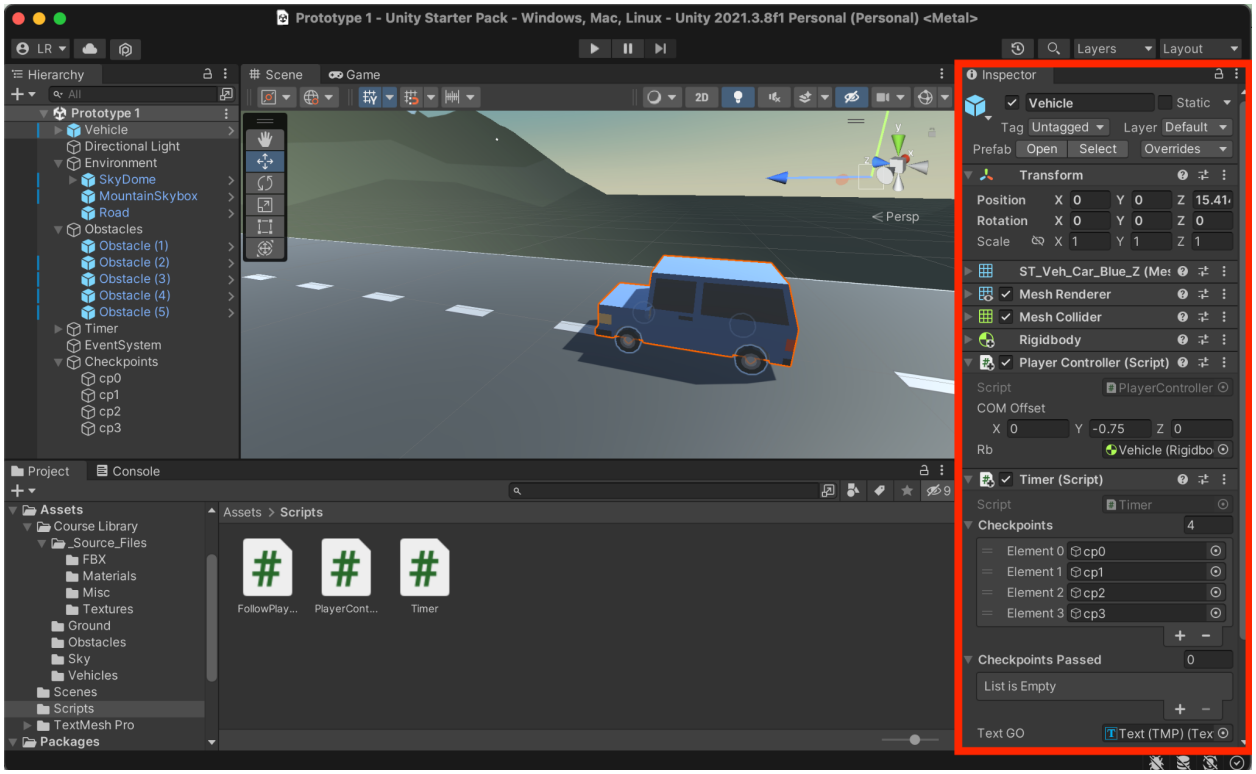
The play, pause and step control buttons are also on this Window. Unity Cloud Services and the Unity Account is also available using the buttons seen on the left of the toolbar window. On the right side of the toolbar, we have access to the visibility menu and editor layout menu that lets us change the editors-window.

Project/Assets



The bottom half of your screen is taken up by the Project window, which includes a file viewer for all of your project's assets. Here, you can drag and drop files to import them into your project, and open up other elements like 3D models and scripts. You can view them through the hierarchical structure on the left or the gallery view on the right.

Inspector

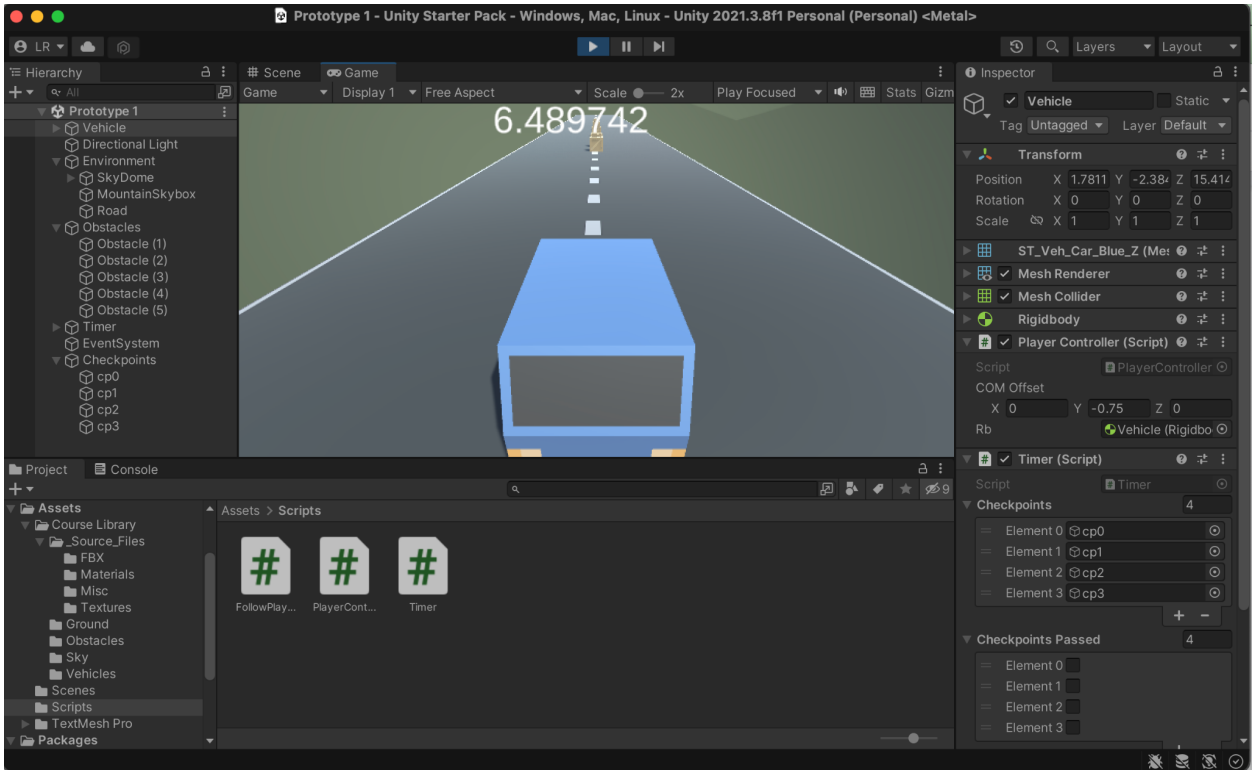


The inspector lets you view the properties of components of your selected GameObject. The ‘Transform’ components lets you view and change properties about the object’s position, rotation, and scale. Other common components of GameObject’s include:

- MeshRenderer: manages the rendering and displaying of an object’s 3D skeleton
- Mesh Collider: gives the object a skeleton that other objects can collide with
- Rigidbody: which simulates physics for an object
- Scripts: dragging a script for your Project window onto an object in the Hierarchy attaches to the object. The script should appear as a component in the Inspector view. In the demo project, the Vehicle object has Timer.cs and PlayerController.cs attached to it.

You can add a component from Unity’s library by scrolling to the bottom of Inspector and clicking Add Component.

Game View



Clicking the triangle play button in the top center of the screen will put you into Game view, where you can simulate your game. You will see through the actual camera object in your scene, as opposed to the freecam that you have been looking through up until this point. You can exit this view by clicking the play button again.

Elements of the Demo Project

Vehicle

The Vehicle GameObject is the player object that we can control as the user in this demo. Expanding it in the hierarchy will reveal the Main Camera is placed as a subobject of Vehicle. This makes it so that the camera is always directly behind the vehicle, matching its rotation as well. This is an easy way to make the camera follow the player without any scripts.

Scripts

If you look more closely at the Vehicle's components in Inspector, you can see it has 2 scripts attached to it. Scripts in Unity are C# programming files (with the file extension .cs) that can be dragged onto GameObjects to manipulate them in some way. Unity has a very large library commands that can change any property in an object. Find the right command is usually a matter of looking at the Unity Scripting Docs.

This project has two scripts working under Vehicle. PlayerController.cs manages user input to move the car. Since this demo is a racing game, this is all it takes to move a car realistically. More advanced player movement may involve animation. Timer.cs displays a simple timer that runs until the car has passed over all 4 checkpoints, denoting the end of the race. These checkpoints are invisible to the player but can be found in the hierarchy.

Camera

As mentioned before, you can find the "Main Camera" object in this scene under the Vehicle object (so that it follows the Vehicle). If you don't the camera to directly follow the player, place the Main Camera outside of Vehicle. FollowPlayer.cs is another script available in this demo, although not used. It can also be utilized to follow a GameObject.

Environment

If you expand the "Environment" object in the Hierarchy, you can see the following objects:

- SkyDome: this is made up of a simple Mesh Renderer, which displays a material (think of it like an imported texture or image) across the top half of the screen using a Mesh Filter. This creates the effect of a sky.
- MountainSkyBox: similarly, another Mesh Filter + Mesh Renderer makes up the mountains and the ground texture in the bottom half of the screen.
- Road: this is the road segment that the car drives on, fitted with a Mesh Collider so that the car would not simply fall through it.

Directional Light

This is a light source with a specific direction/rotation, which creates shadows with other objects.

Materials, Shaders, and Textures

The materials, shaders, and textures will be a part of the assets folder in the project/asset window. This demo only contains a textures file, however, below is some more information on materials, shaders, and textures in general.

- Materials describes how a surface should be rendered and includes references to colour tints, tiling information, etc. The shader being used determines many of the materials characteristics.
- Shaders are scripts that contain the algorithms for calculating the colour of each pixel. This is based on the lighting input and the design of the Material.
- Textures are bitmap images. When a material has references to textures, the material's shader uses the textures while calculating the colour of an object's surface. Textures can represent the color, reflectivilty, and roughness of a material's surface.

For normal rendering, such as characters, scenery, environments, surfaces, objects, it is best to use Standard Shader . This is a customisable shader which can realistically render many types pf surfaces.

Physics

The object in a game has to move/accelerate correctly and react to colliding with other objects and forces. Thanks to Unity's built-in physics engines, the physical simulation is already handled. By just adjusting a couple of parameters we can create objects that realistically move. It is possible to control the physics from scripts in the Assets Window. By doing so, we are giving the image the dynamics of a car.

It is important to note that there are actually two physics engines in Unity: one for 3D, and one for 2D physics. We are implementing the physics engine in this game using the Rigidbody component as seen in the Insepctor Window.

Resources

Tutorials

Unity Learn
Learn Unity - Beginner's Development Tutorial
Brackeys Beginner Unity Tutorial
Unity's Official Junior Programmer Tutorial (great for those with no code experience)

3D Models

TurboSquid
Unity AssetStore
CGTrader
SketchFab

Programming in C#

Unity Scripting API Documentation
Unity Docs