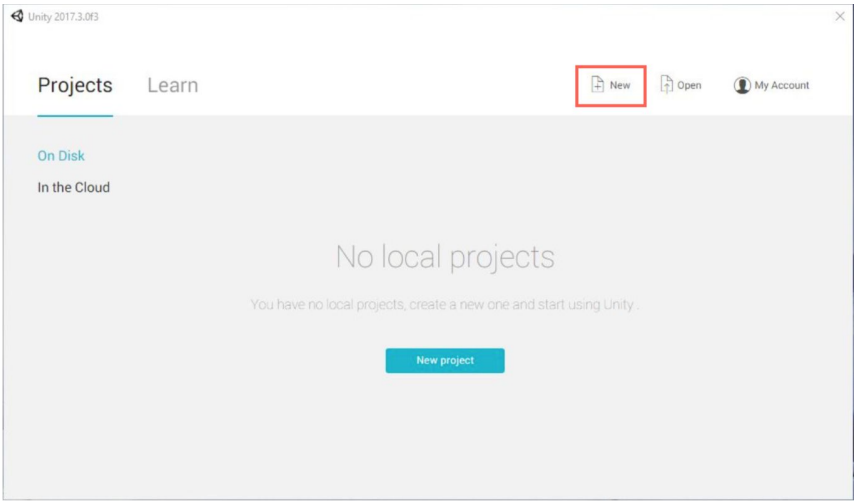
Intro to Unity Basics

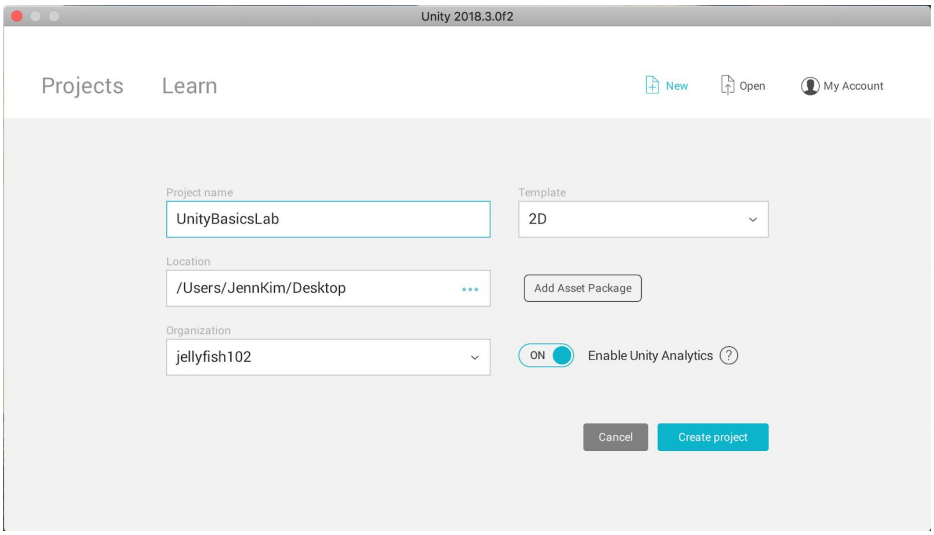
In this lab, you will make a small game from scratch using Unity Playground. You don't need to know any scripting or how to make art; this is an introduction to the basics of the Unity engine.

Setting up your project

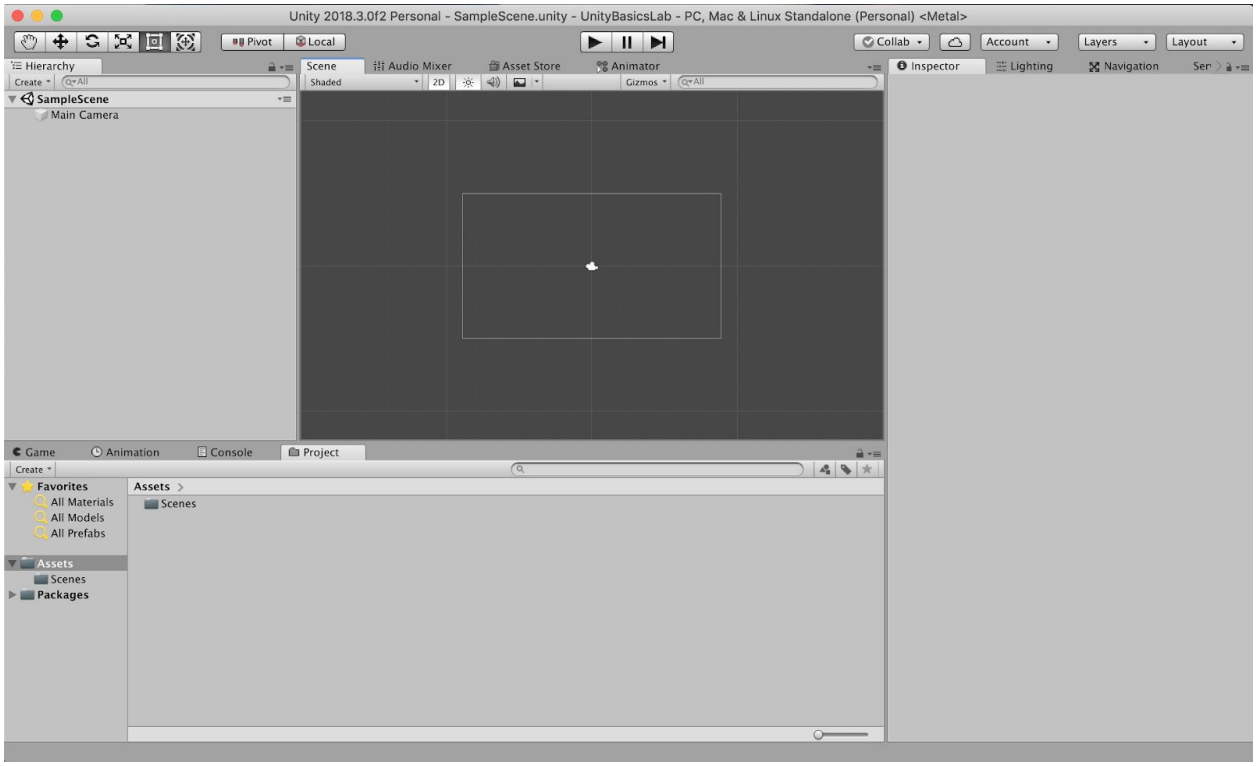
When you open up Unity, you should see something like this. Click on the New button on the top right hand corner:



Now you should see a screen like the one below. Title your project something along the lines of Unity Basics Lab in the project name field. Make sure the Template is in 2D. Choose whichever location you want the project to be in your computer. The organization should already be filled in for you and you can choose to either keep Enable Unity Analytics on or off. This is Unity's built in system for analytics on your game. Since we won't be releasing this, you won't need it; I like to keep it off (if you turn it off the Organization field should disappear. Once you've filled everything in, click create project



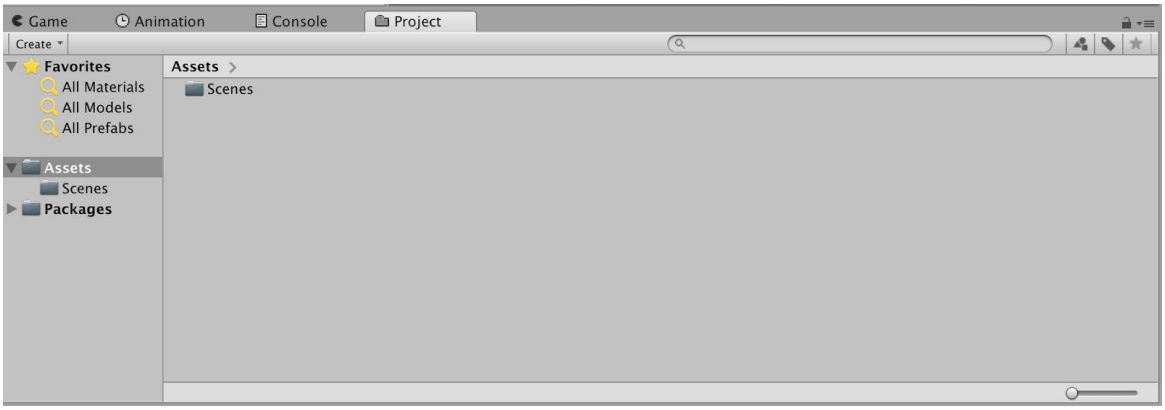
After waiting for a bit, you should see a screen like this. It's okay if it doesn't look exactly like this. As you continue developing you will figure out which setup works best for you.



Let's talk about how to navigate through Unity. The editor has a lot of different windows and if you don't see one that is mentioned here, you can look at the window tab in the toolbar and click on it to open it up.

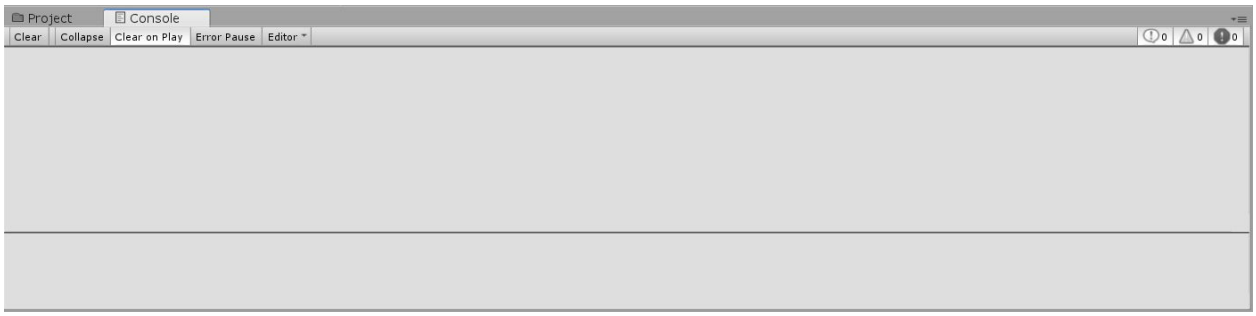
Project and Console

The first window we will look at is the Project window.



This window shows all of the files and folders that are usable assets for your project. Everything you need, from sprites to scripts will be here. In a new project, the Assets folder will not have anything in it except for the scenes folder. Scenes are different files that are made using all of the assets, think of them as different levels. They all draw from the same pre-existing assets, but have a different format and layout. We'll start adding more to this once we start the lab.

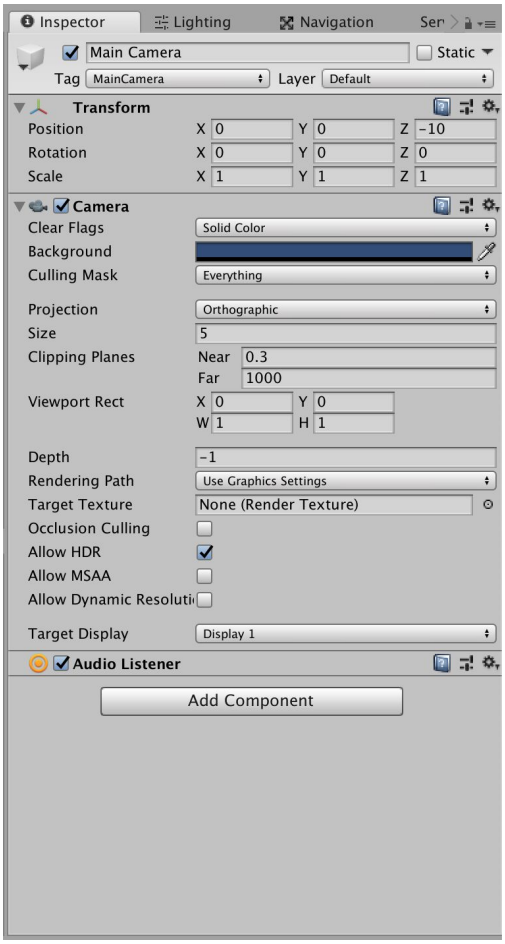
Next to the Project window is the Console window.



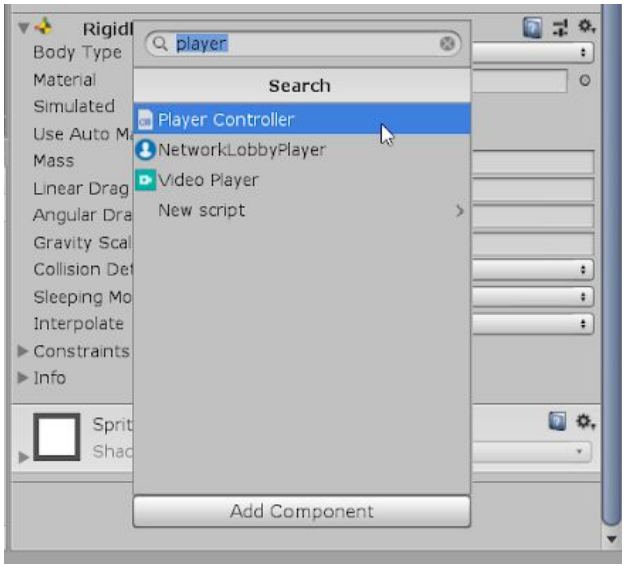
If there are any errors that occur within the project, like compiling errors in code, missing components or invalid code, the errors will show up here. Also, if you print anything in code (using Debug.Log() or print()), it will also show up here. This window can be extremely useful in debugging code.

Hierarchy and Inspector

The Hierarchy window contains all of the Game Objects that are currently being used in a scene. Game Objects are the basic building components in Unity. Every object you see is basically some sort of Game Object, and each Game Object can have extra components and scripts attached to it, giving each of them their own properties and behaviors, like colliders and scripts. Right now, your Hierarchy should only have the Main Camera



The Inspector window (see left image) allows you to view and edit the properties and components of Game Objects. If you click on a game object in the Hierarchy window, you can see all of the components currently attached to that game object. Right now, there aren't any game objects other than the main camera, which should have the fields for Transform, camera, and Audio Listener.



To add new components or scripts, press Add Component, which is located at the bottom of the Inspector. You can look for specific components in the search bar whether it was made by you (like a script) or something that is built in Unity such as a rigid body (See top right image). If you want to add the component, you just click on it and it should show up under everything else. Another way to add a script is to simply locate it in the project window and drag it into the inspector.

Scene and Game

Here, you can organize the layout of this particular scene. The Game window shows what the game will look like through the camera when it is running, and the Scene window next to it is where you will actually move and manipulate each item in the scene. There is a toolbar in the top left corner that will help you move around in the scene.

1. Moves the scene view
2. Moves objects vertically or horizontally
3. Rotates objects
4. Changes scale of objects
5. Moves objects freely
6. A combination of 2-5

Above the Scene and Game windows are the play, pause, and step buttons.

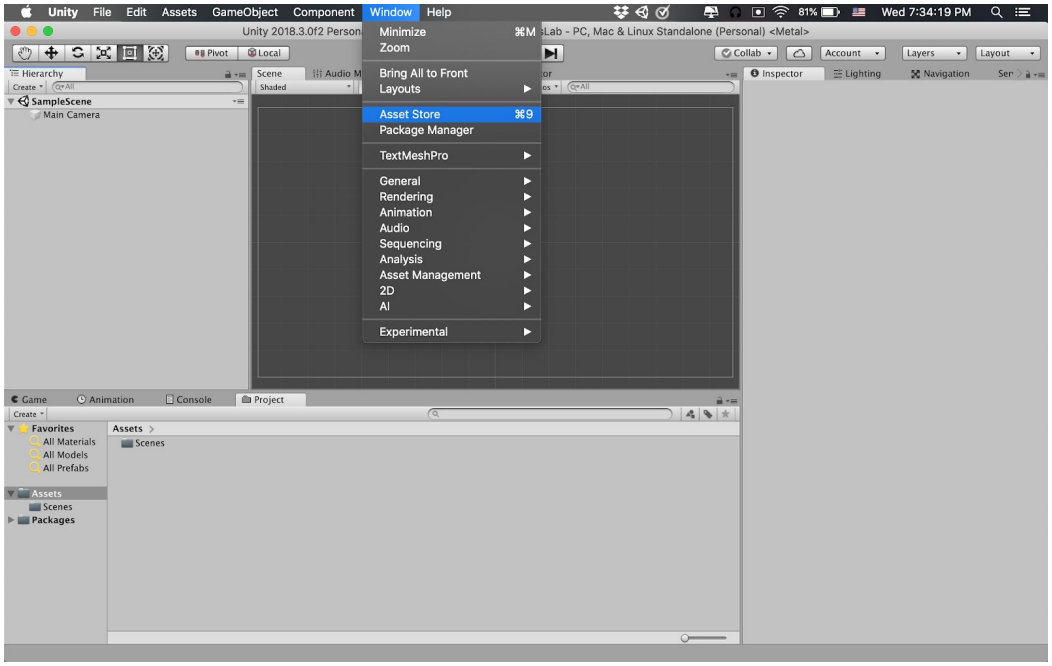
1. The play button will start running the game inside the editor so that you can test the game. Hitting the play button while the game is running will stop the game.
2. The pause button will pause the game on a frame when you are playing it, which is useful when you want to be able to tweak something or debug a problem
3. The step button allows you to skip forward a frame. It will pause if the game isn't already paused. This is also useful if you want to debug since it allows you to go frame by frame

In addition, you can change variables in the inspector while the game is running, but when you stop playing the game, your changes will be reverted back to what they were before you started playing. This is important to keep in mind while testing your game.

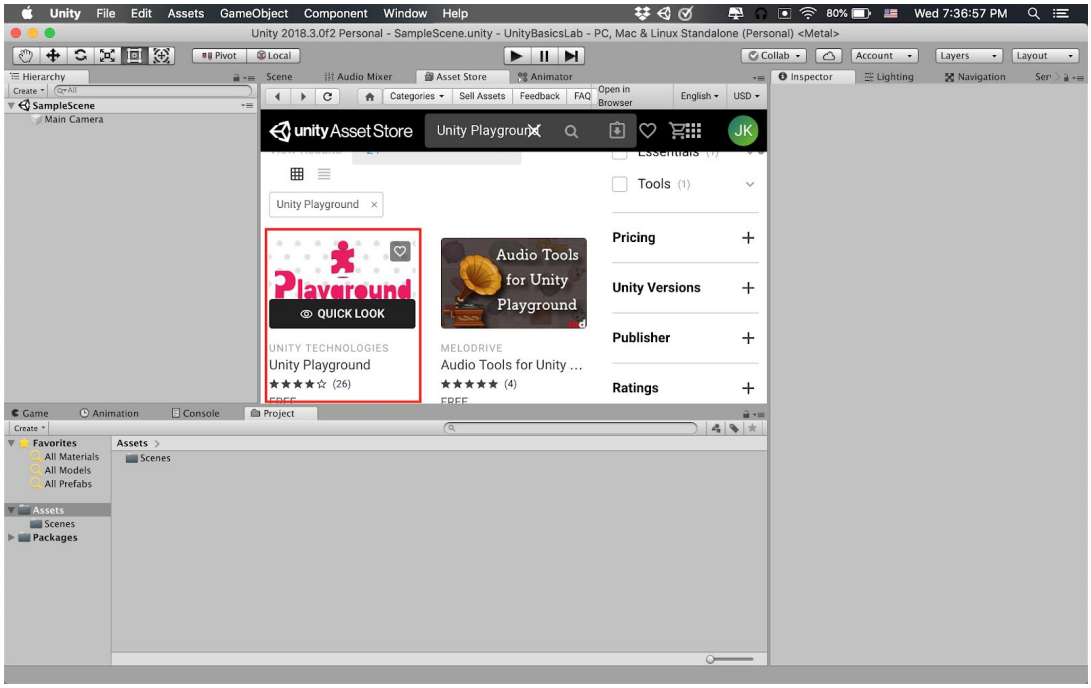
Now that you have learned some of the basics, let’s start making a game!

Getting Unity Playground

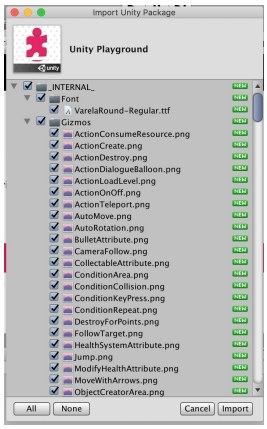
Look for the Asset store tab (it is probably somewhere next to the scene tab) on your editor. If it is not there, click on the Window tab in your toolbar and click on the asset store option (pictured below)



In the search bar in the Asset Store tab, type in Unity Playground and press enter. Click on the Unity Playground option (the one with a pink puzzle piece as the image and made by Unity Technologies)

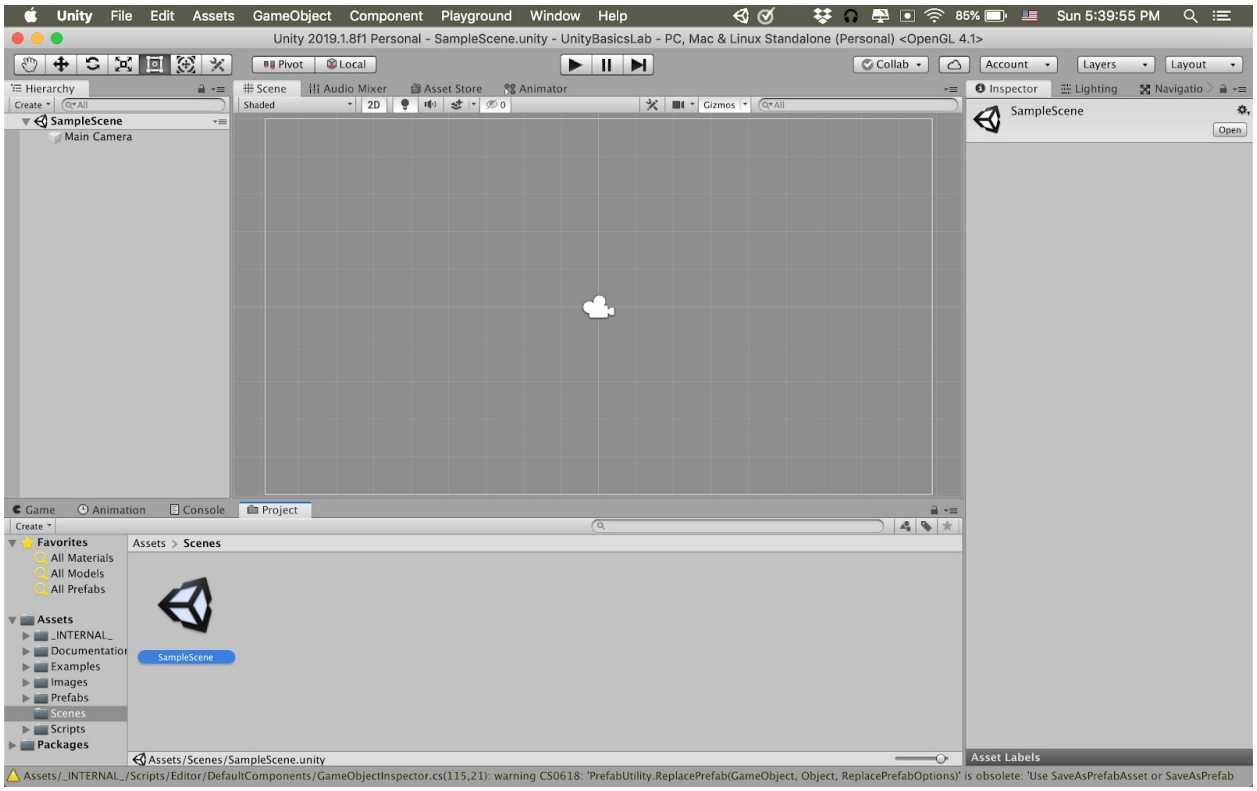


Once it loads, scroll down until you see the download button, and download it, and accept the terms and conditions when prompted. Once it finishes downloading, the button will change to say import. Click on it and a warning should pop up saying that importing a complete project will overwrite your current project. This is fine since we haven't even started our project, click on the import button. The image shown below should pop up afterwards, make sure everything is checked before clicking import and restart your project if prompted to.

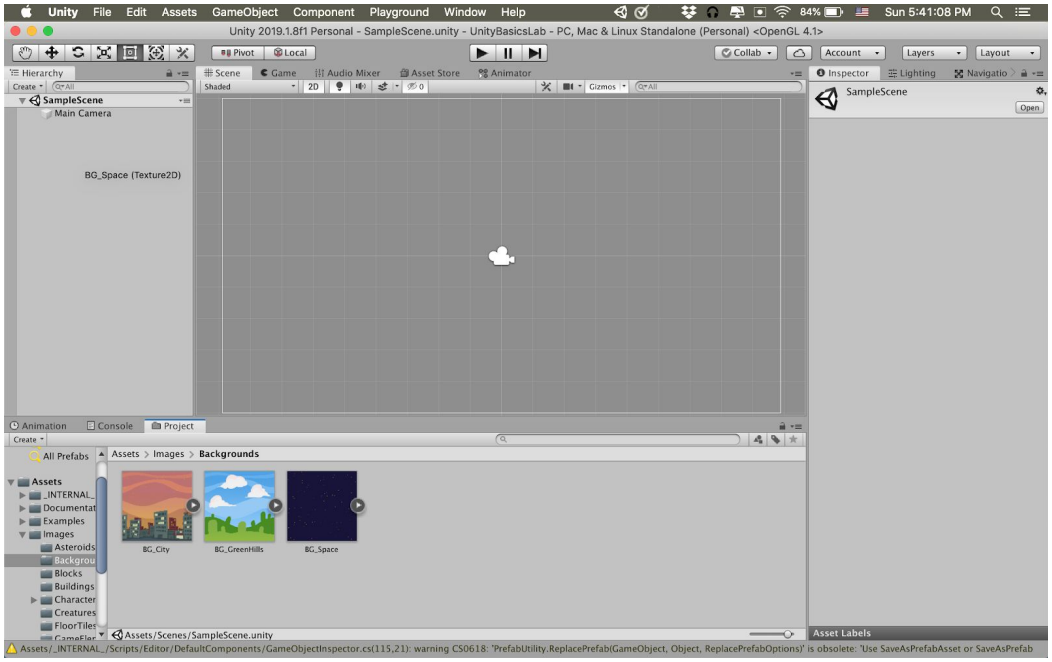


Setting it up

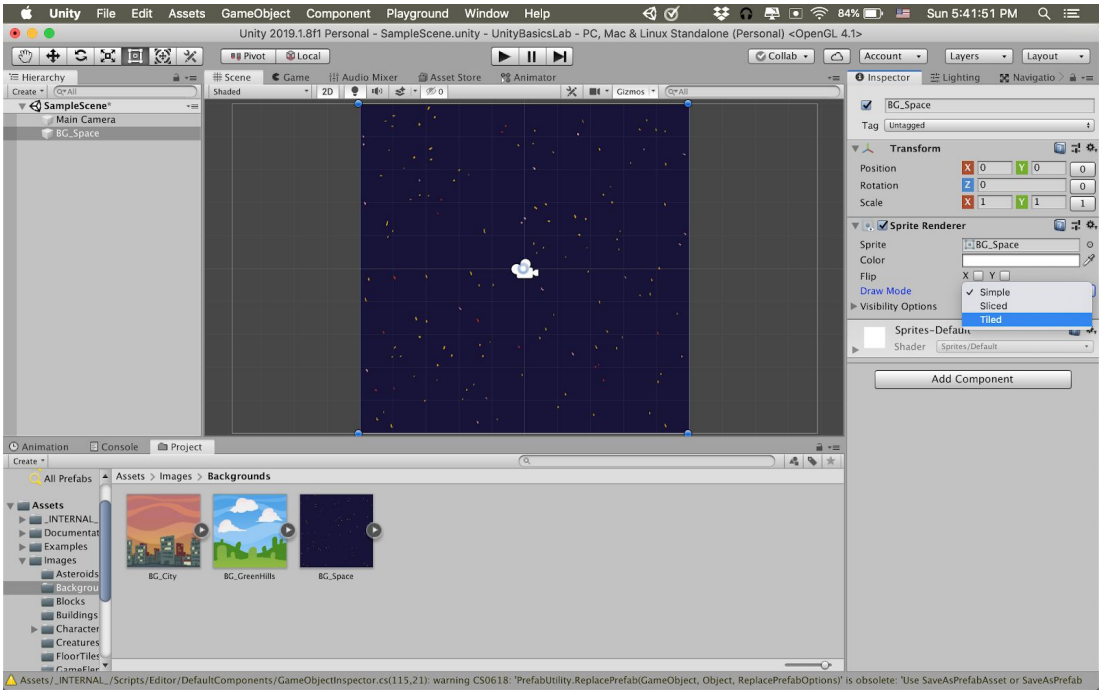
Let's start. Make sure that you are in the SampleScene by double clicking on it. It should be within your scenes folder.



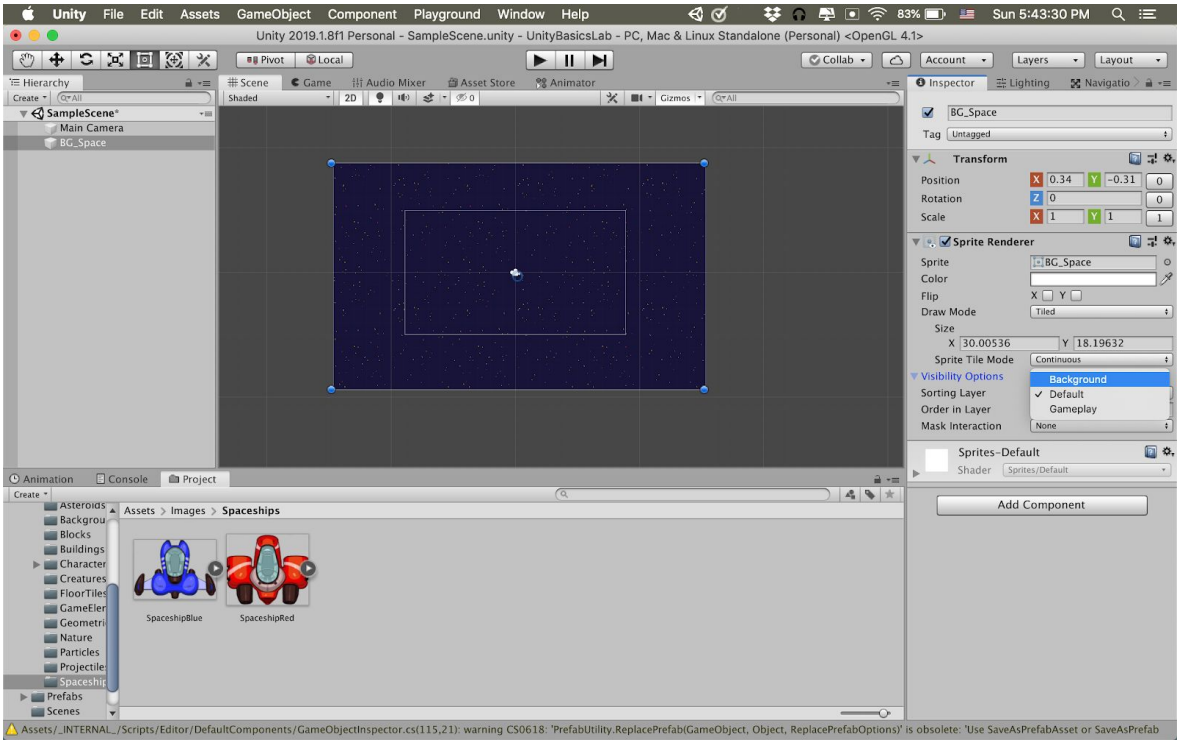
Go into the images folder and in there you should be able to see a folder called background. In there, select BG_Space and drag it into the hierarchy (just click and drag). This is the background for the game.



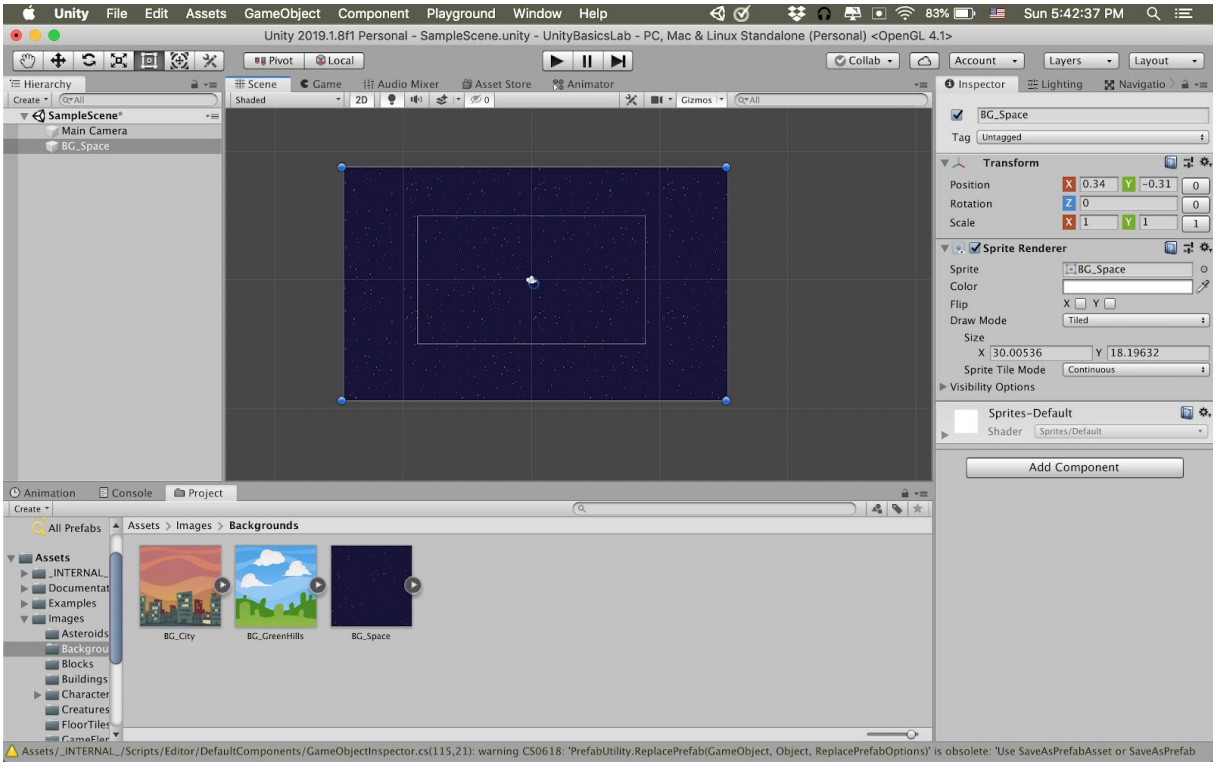
Now look in the inspector for this image (make sure BG_Space is highlighted in the hierarchy) and changed the Draw Mode to Tiled. Most Unity Playground assets support tiling which simplifies the number of assets you need.



Also expand the Visibility Options and set the sorting layer to background. Once you do this, the background will always be behind the other game objects you put in.

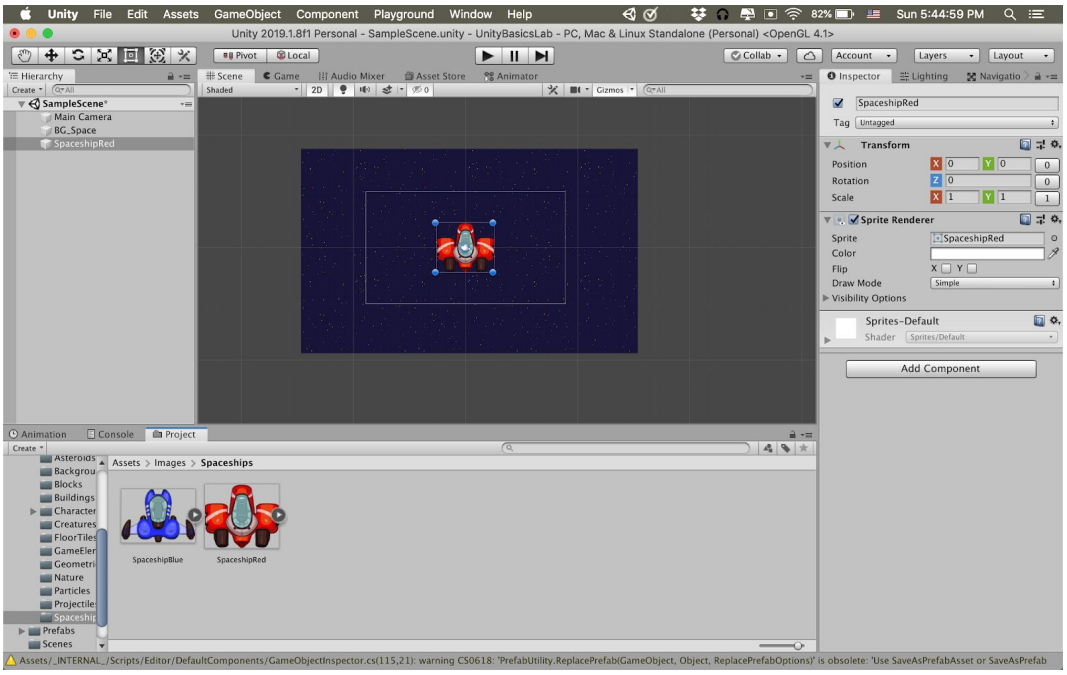


Now expand your image by clicking on it in the scene (Make sure you are in the scene tab rather than the game tab) and dragging the corners so that it extends past the gray box that indicates what the camera can see.

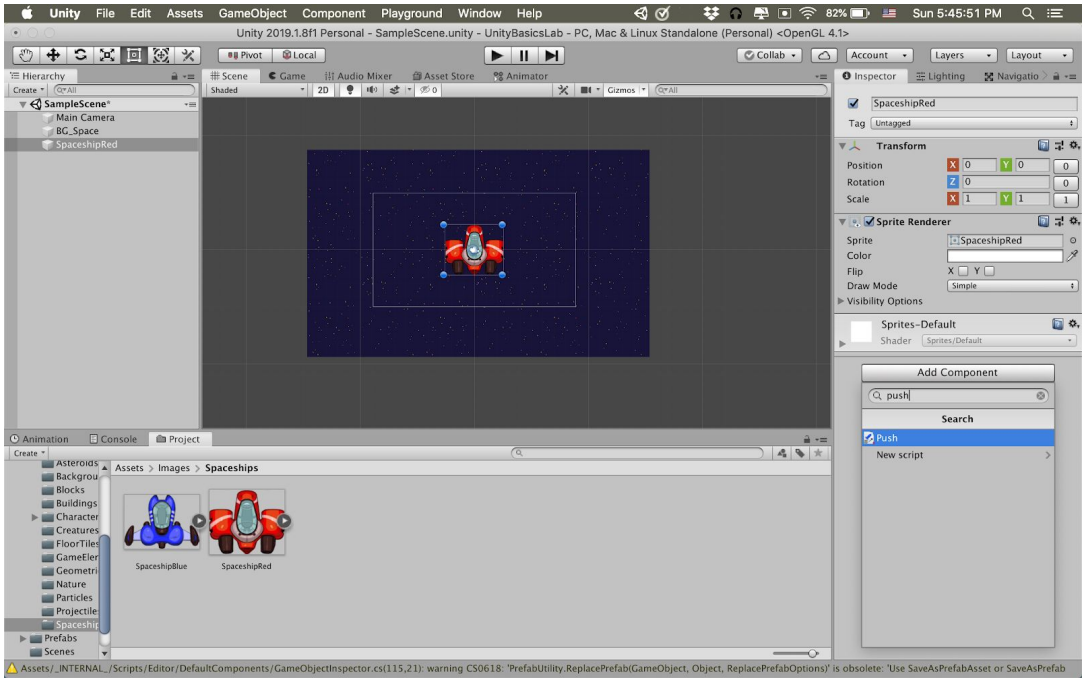


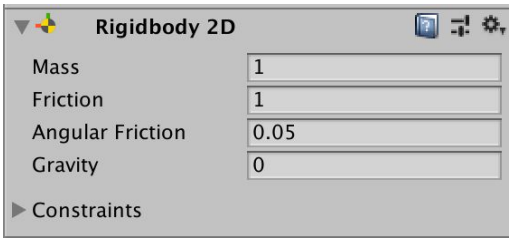
Character

Now let's add a character. Go into the images folder again, but this time, go into the spaceships folder and choose a spaceship to be the character. Drag it into the hierarchy like we did for the background. For this tutorial we will be using the red one. This creates our player game object and sets it into the scene.



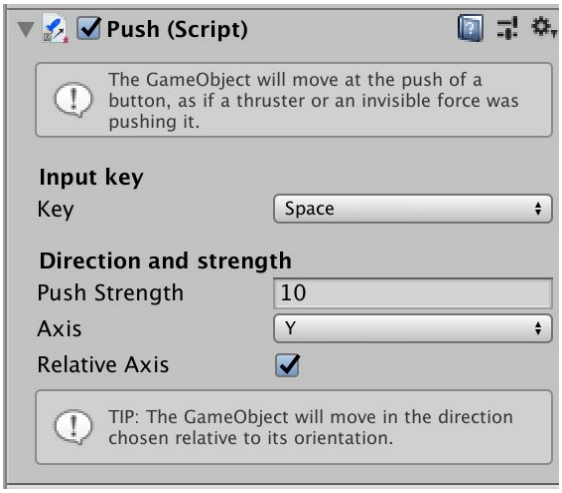
Let's add interactivity by adding some scripts for movement. Click on the player and in the inspector click add component. Type in Push and select the push script.

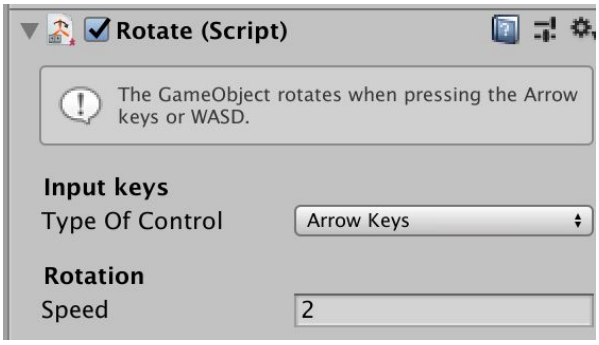




When you add this script, another component should be added to our player game object, a rigid body component. A rigid body object allows a game object to be governed by physics and most of the Unity Playground movement scripts rely on physics. Let's change the gravity in the Rigidbody 2D for the spaceship to be 0, since it is in space, there is no gravity and set the friction to 1 so that our spaceship won't float forever.

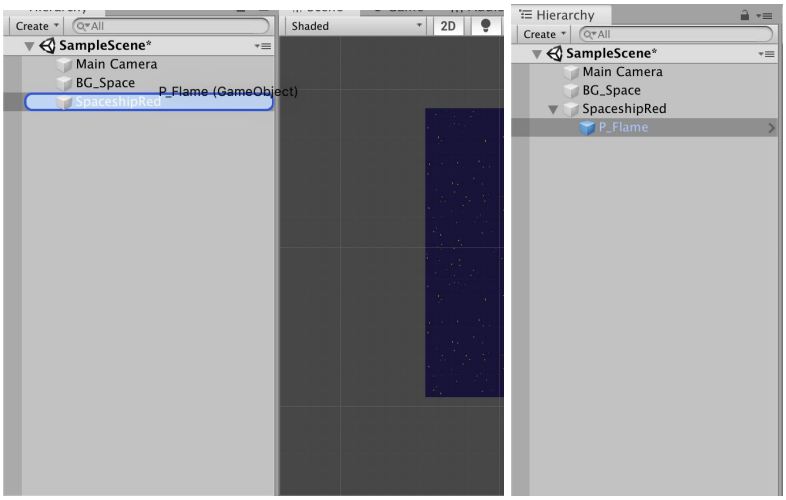
In the scene, you should see a green arrow coming out of your spaceship. This arrow helps indicate the direction and force at which your game object will go. Now lets play to test the game. Click on the play button near the top of the editor and your game should start. Press your space bar to see how far the character moves. You can adjust the settings in the fields of the Push script. Push Strength is how far the character will go, while Axis is the direction it will move in. Remember, that the changes made in this mode will not be saved, so once you stop playing the scene, you will need to change them again to save them.



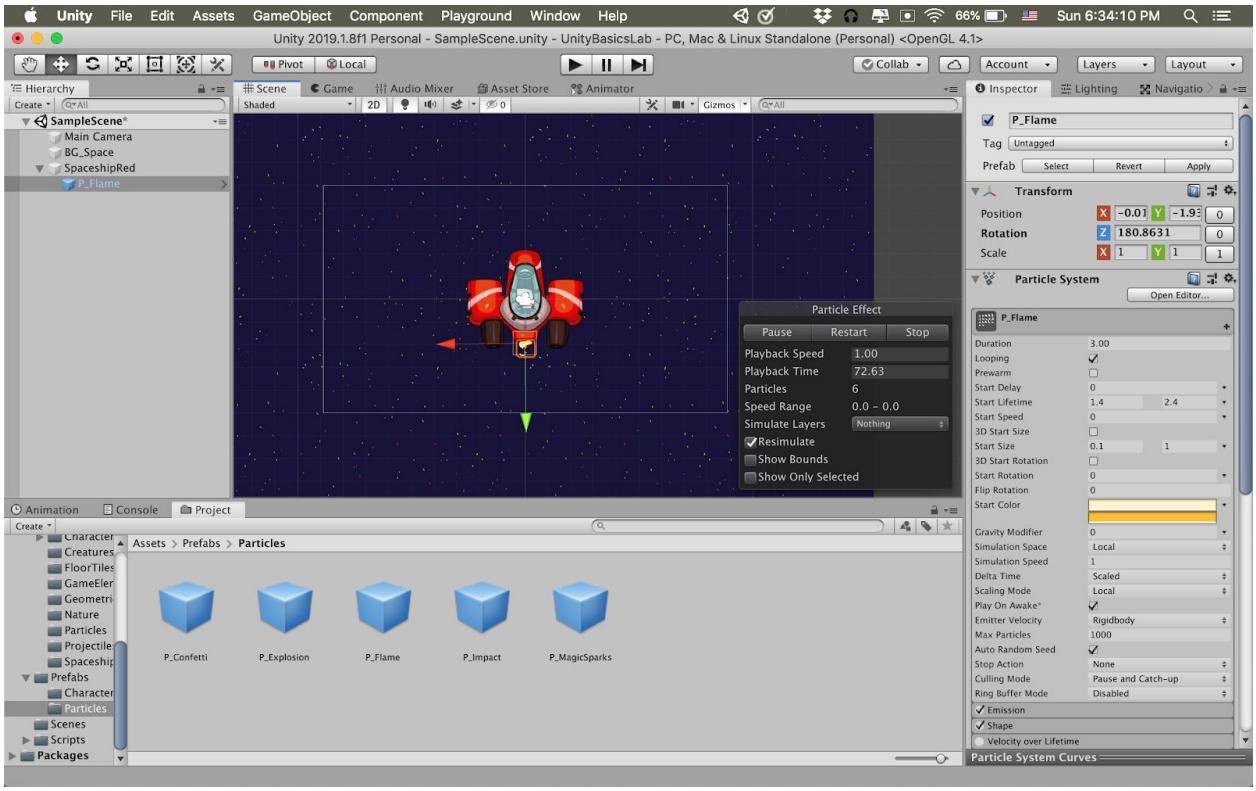


Now let's add the Rotate component so we can steer our spaceship around. Again click on the Add Component, but this time, type in Rotate. When you play the scene now, you should be able to rotate the spaceship around. You can play around with the rotation of the speed, until you get to a speed you like. (Again, remember changes in this mode are not saved).

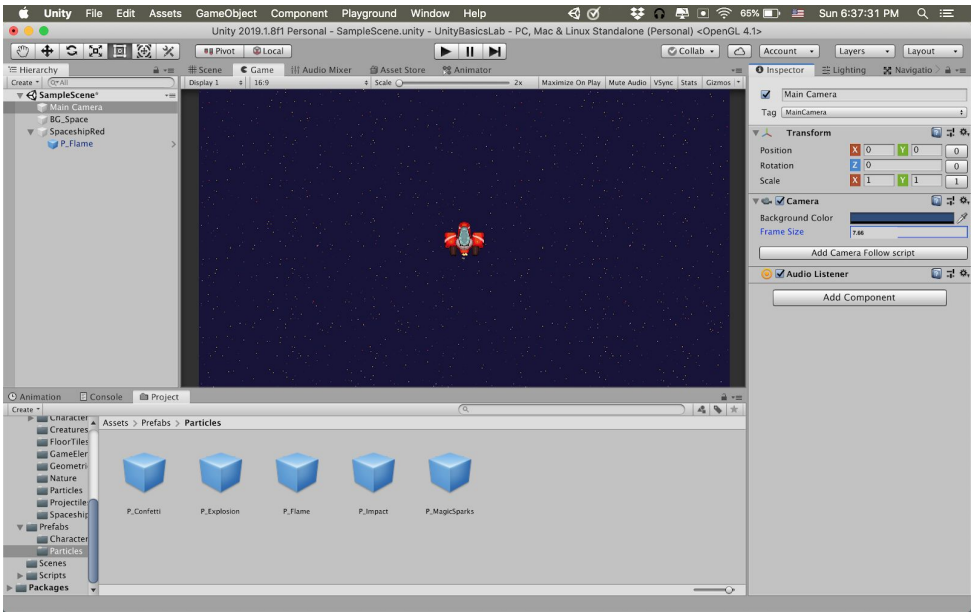
Let's add a bit of animation and flare, by adding thrusters to our character. Go into the prefabs folder, and in that folder, go into Particles. There you should see a Prefab called P_Flame. Drag this into the heirarchy and make it a parent of the ship, so that it will move with the ship. You can do this by dragging the P_Flame in the hierarchy onto the Ship game object.



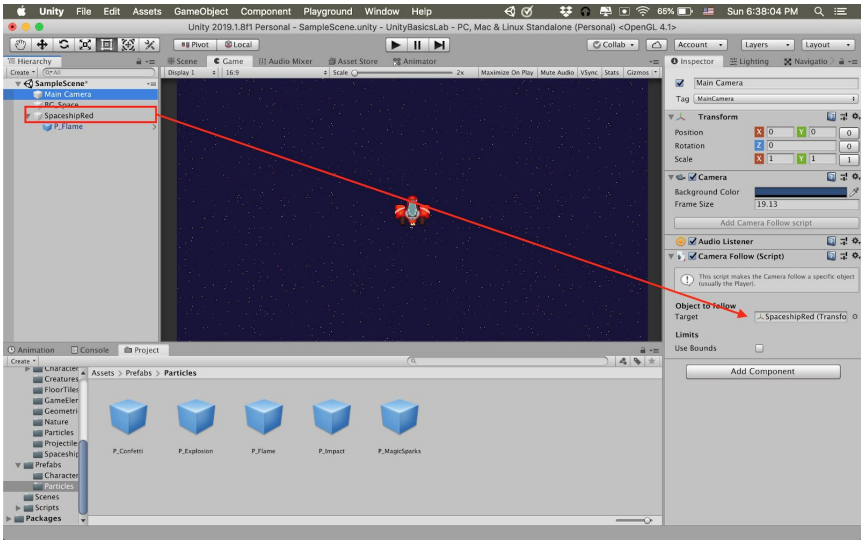
Once you do this, you can adjust how the flame looks and where it is relative to the ship. Fiddle around until it looks nice and then click play. You should see that the flame move with the space ship and also has it's own animation

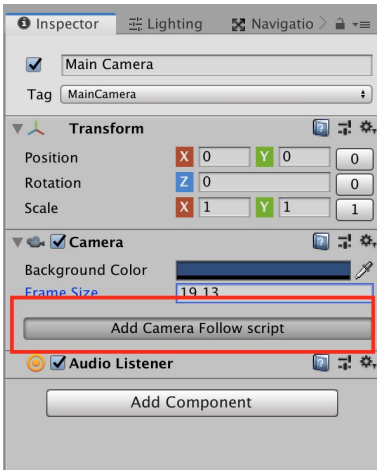


Before we move on, let's save our scene. You can do this by pressing Ctrl + S (Windows) or Cmd + S (Macs). If you don't save a scene before exiting it, all your data will be lost. Now let's move the camera to scale the image you see. Make sure to select the Main Camera object in your hierarchy and fiddle around with the Frame Size variable (I put mine around 7.66) until you like how small the ship looks on the screen (be in the game scene when you do this). Another way to do this, is to scale the player ship down.



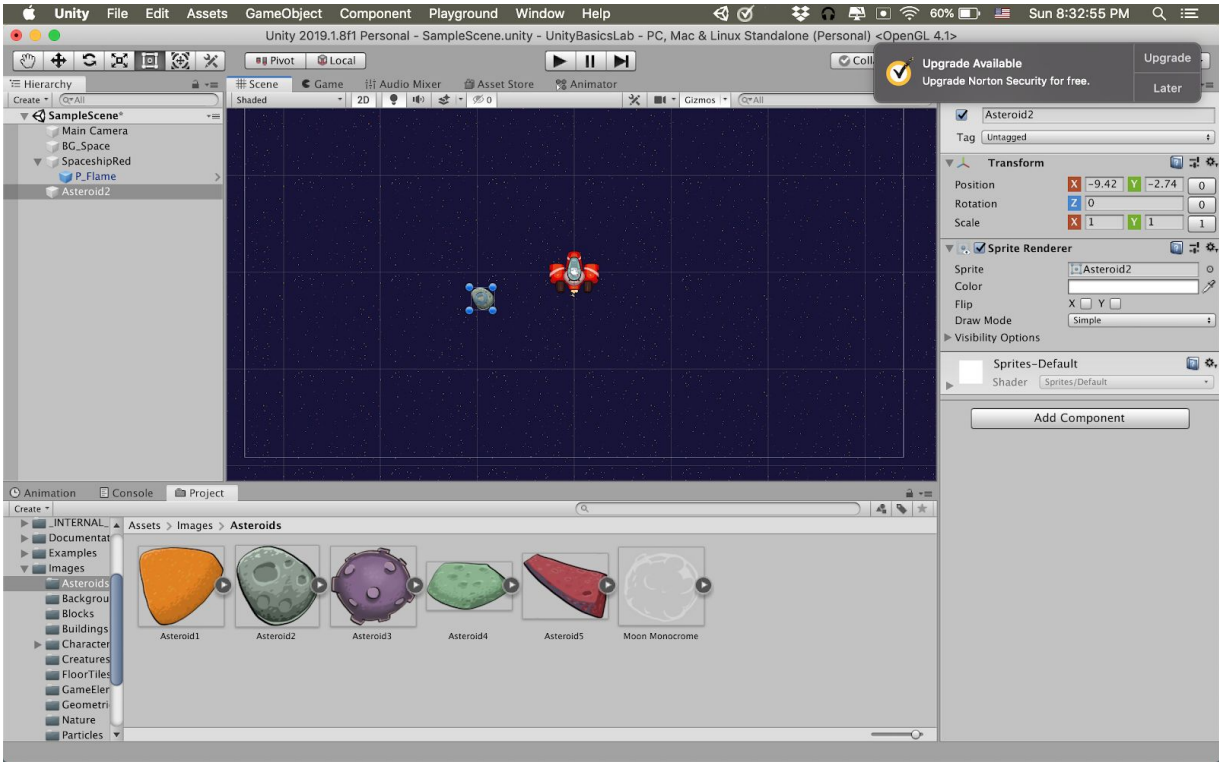
Now that our scene looks good, let's make the camera follow the ship. Under the Camera component, there should be an Add Camera Follow Script button. Click on it and now a new script should show up in the inspector. Drag the player ship in the hierarchy and drag it into the Target field under the Camera follow script. This will let the camera know which object to follow. Once you press play you can see that the camera follows the player



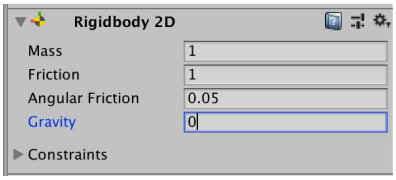


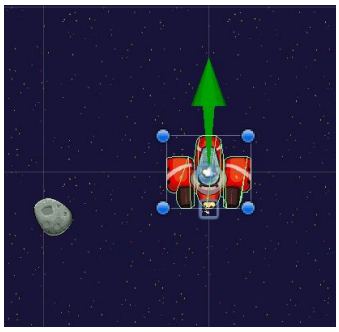
Asteroid

Now let's add something that the player can interact with. Add another image from the Asteroid folder by dragging it into the hierarchy - like we did for the player.



Add the rigidbody 2D component so that it can move and set the gravity for it to 0 and friction to 1. This will allow the asteroid to slowly stop moving if it is ever touched.

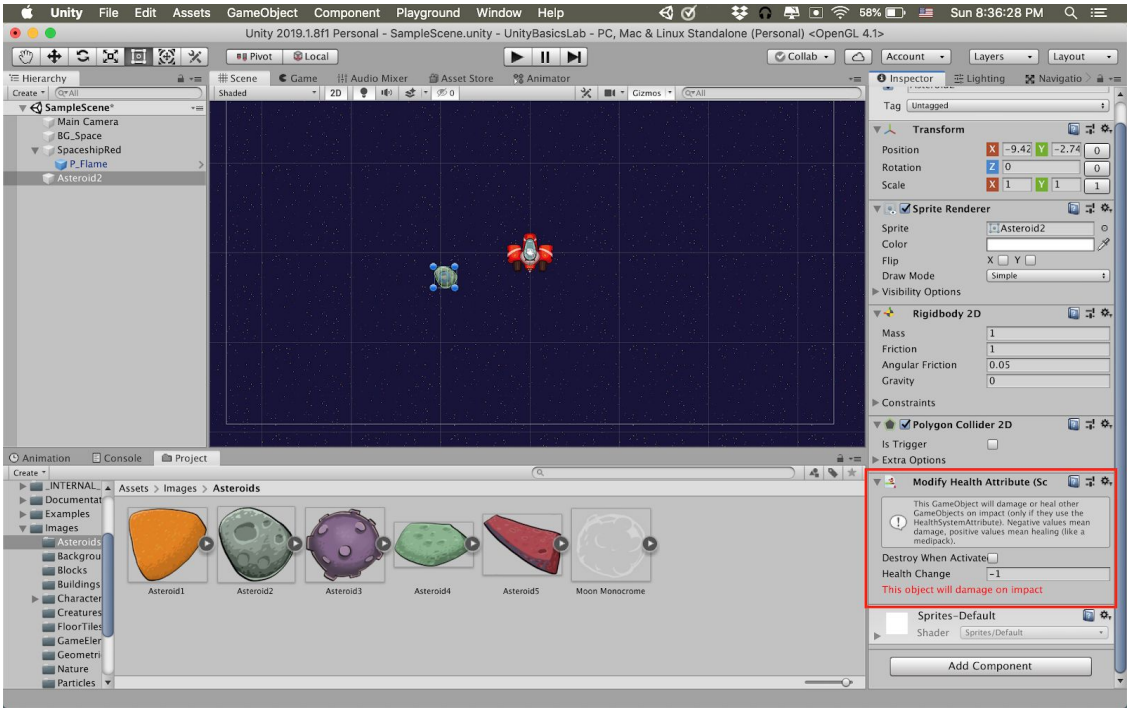




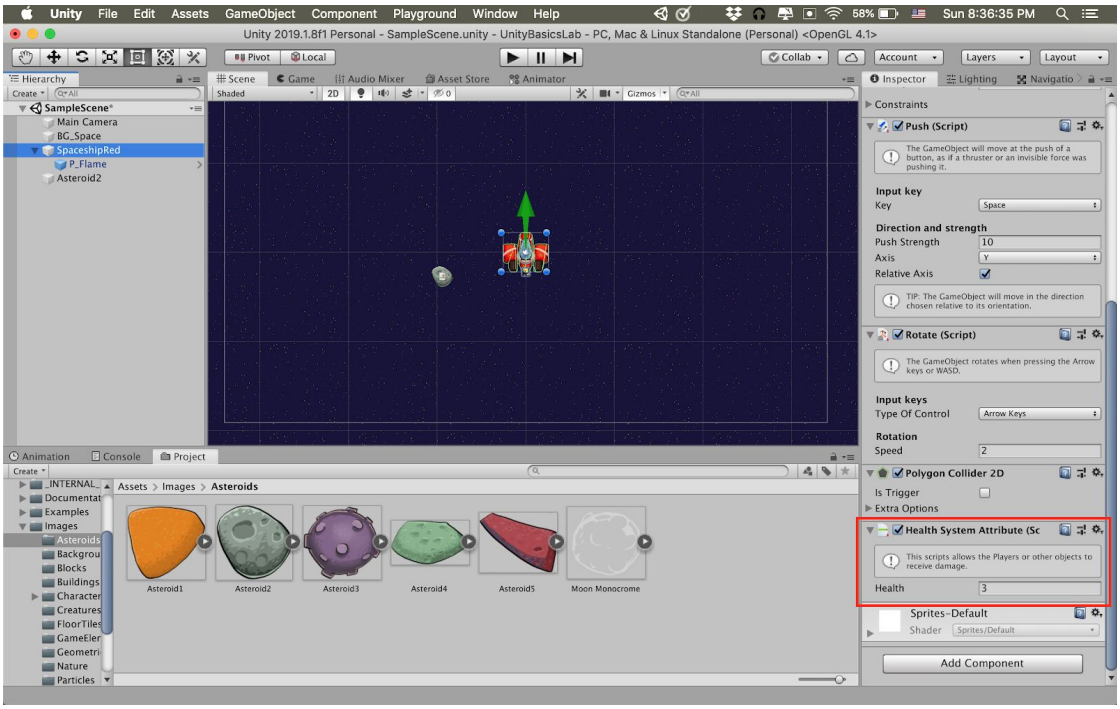
Now to create interactivity between the objects, let's add polygon colliders. Add a component called Polygon Colliders 2D onto both the asteroid and the player. If you look in the scene, you should be able to see that the colliders automatically scale to fit the graphics for the game object. When you hit play, the rock and player now push each other.

Health

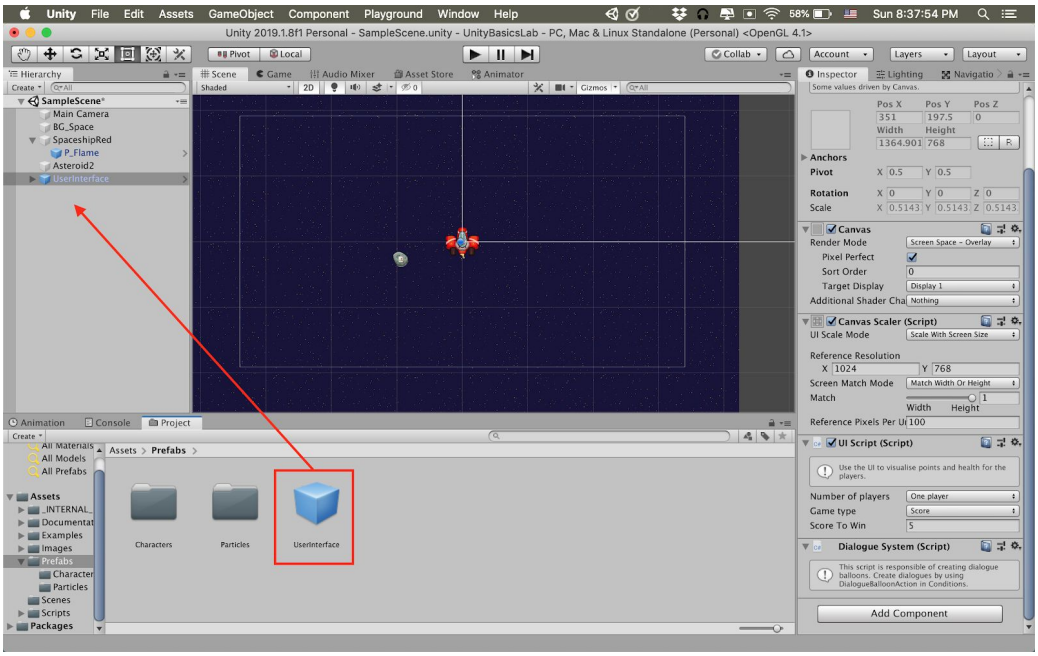
Now let's add something more interesting. Let's make it so that the player has health and every time it touches an asteroid, it's health goes down. Add the Health System Attribute script to the ship game object. This will help keep track of the health for the player

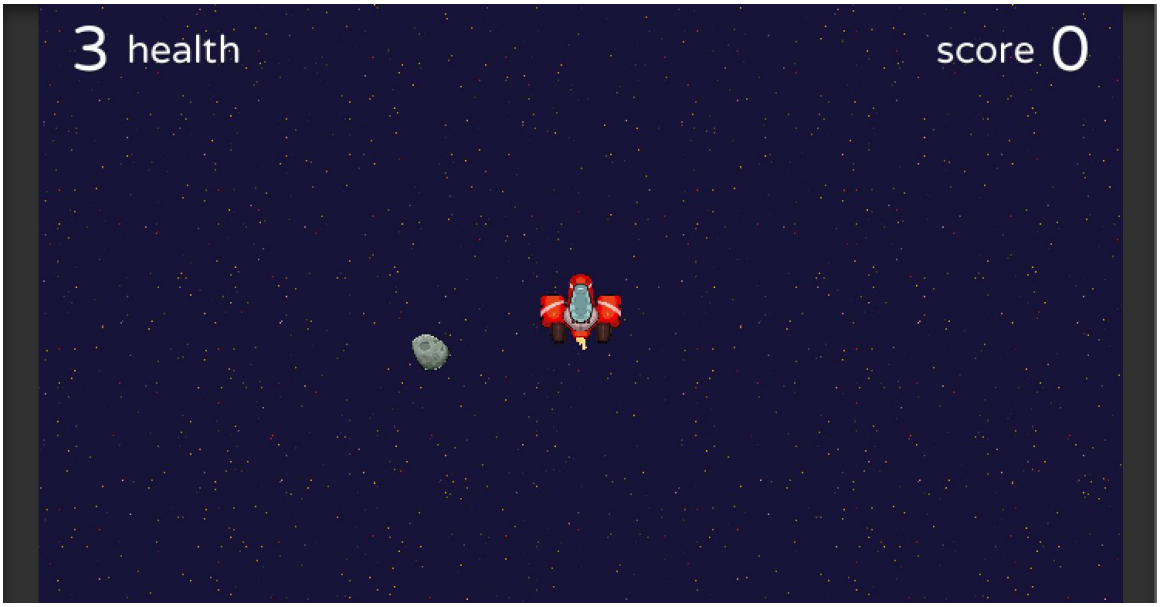


Now add the Modify Health Attribute script to the asteroid. This makes it so that every time the player hits an asteroid, it loses health based on the number in the health change field.

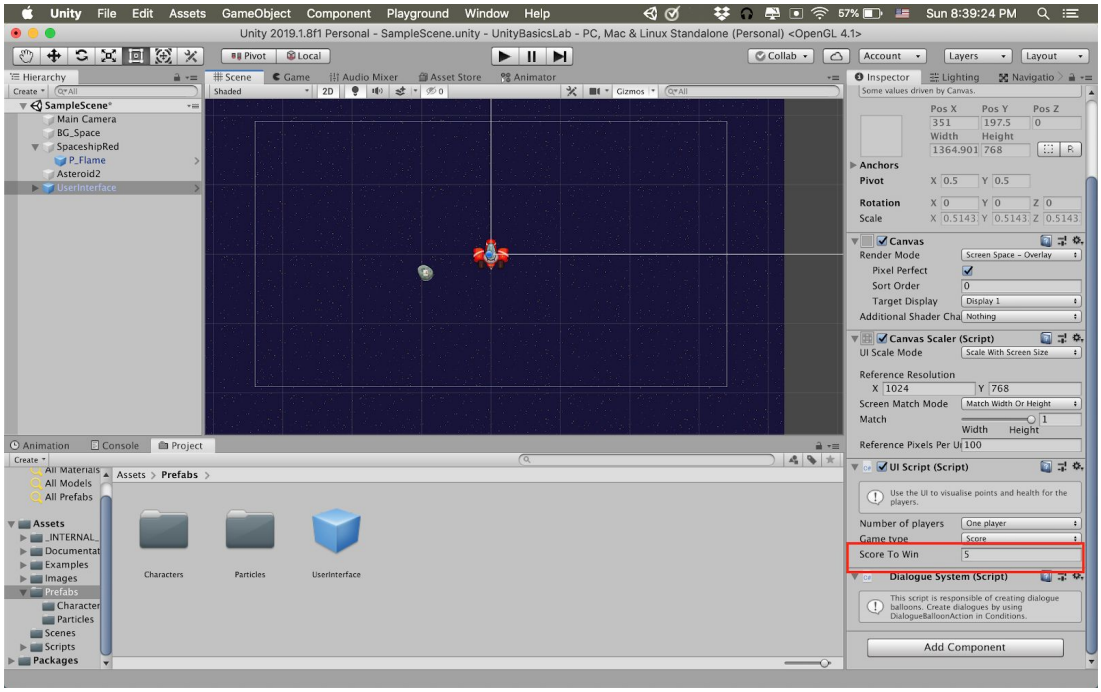


To add some UI to see the health of the player, look in the prefabs folder and drag the UserInterface prefab into the hierarchy. You should be able to see the health and score when you click play. You can't see it in the scene because UI overlays on a UI canvas. If you zoom out, you should be able to see the words, or if you look in the game tab.

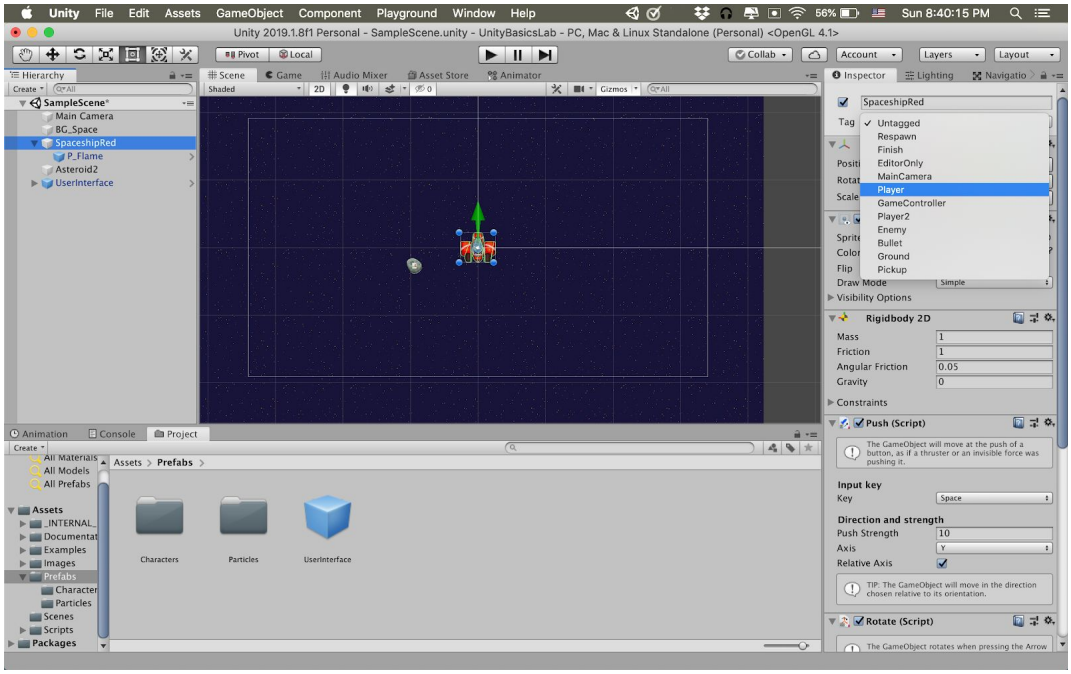




The UserInterface also takes care of the win condition of the game. Right now, you need a score of 5 to win, as you can see in the score to win field of the UI Script.

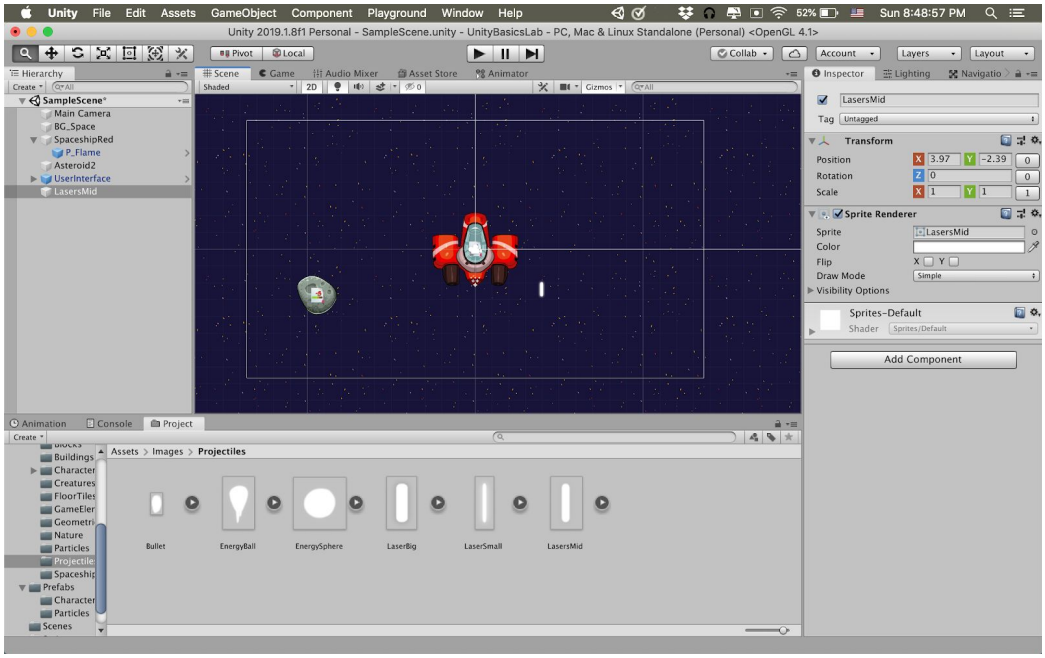


To make the system work, you need to do one more thing. Select the player in the hierarchy and change the tag for the player to Player. You can do this by looking at the top of the inspector and selecting player as the tag. With tags you can easily keep track of what types of object, game objects in your scene are. In this case, the UI needs to know which object's health to keep track of. You can click play and hit the asteroid 3 times to see that the player loses all its health and the player loses the game.

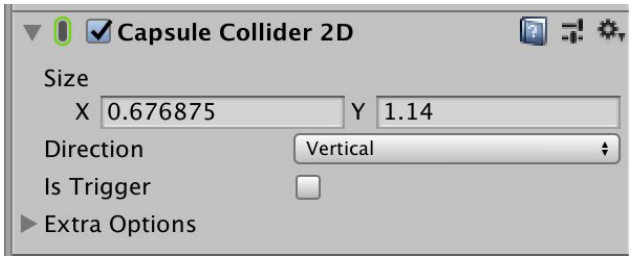


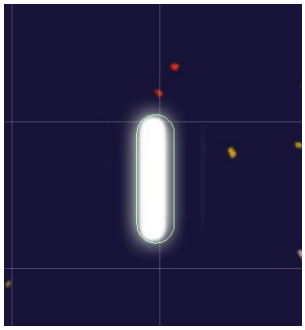
Bullet

Now let's add something to get rid of the pesky asteroids. Look into the images folder once more and find the projectiles folder. Drag the LaserMid image into the hierarchy.

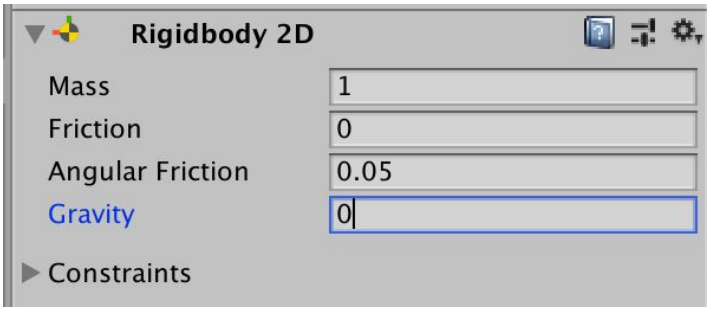


Add a Capsule Collider 2D component to the laser and change the direction to be Vertical. Then fiddle around with the size until the capsule fits the size of the laser. This collider allows us to see what the laser interacts with. Also set the Is Trigger field to be true (click on it so it's checkmarked). You set a collider to be a Trigger collider when you want to know when an object touches or interacts with another object without causing the objects to move each other.

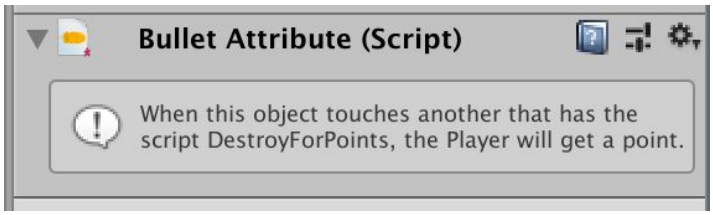




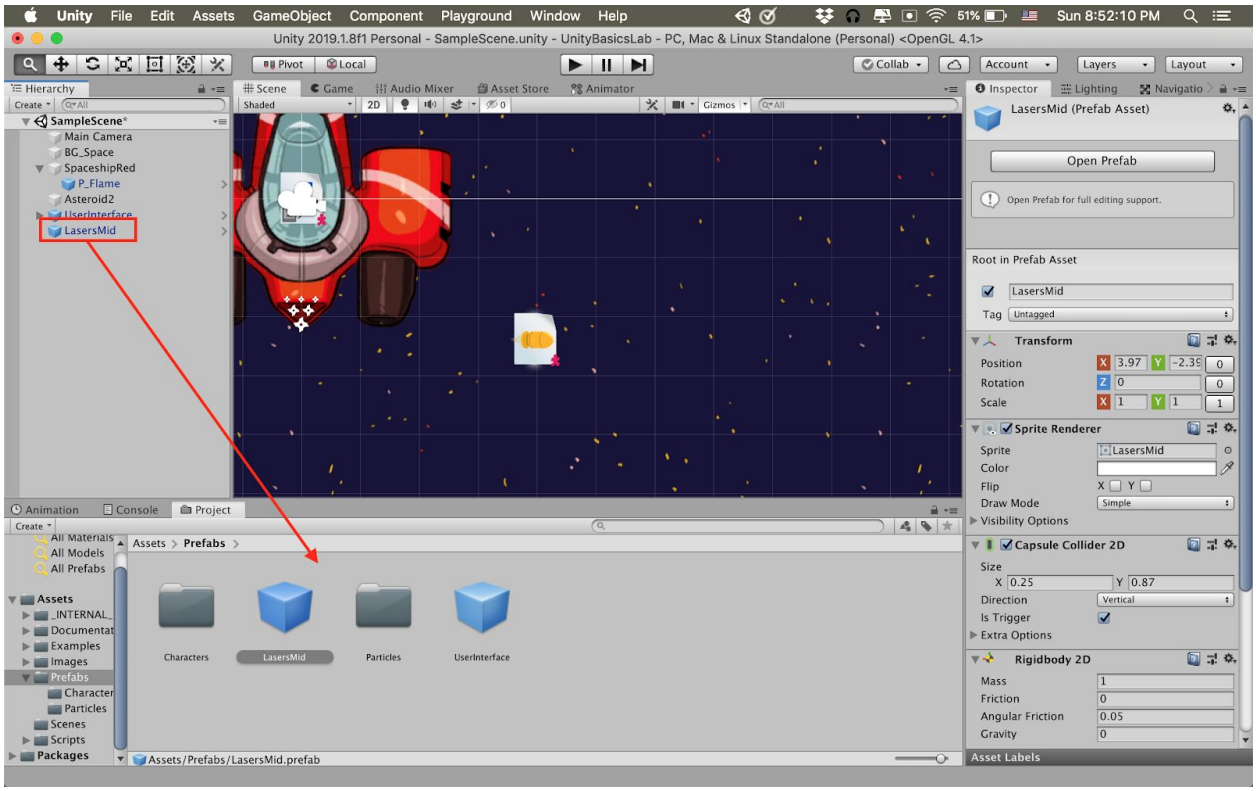
Then add the Rigid Body 2D component to let it move and set the gravity to 0 like we've done before.



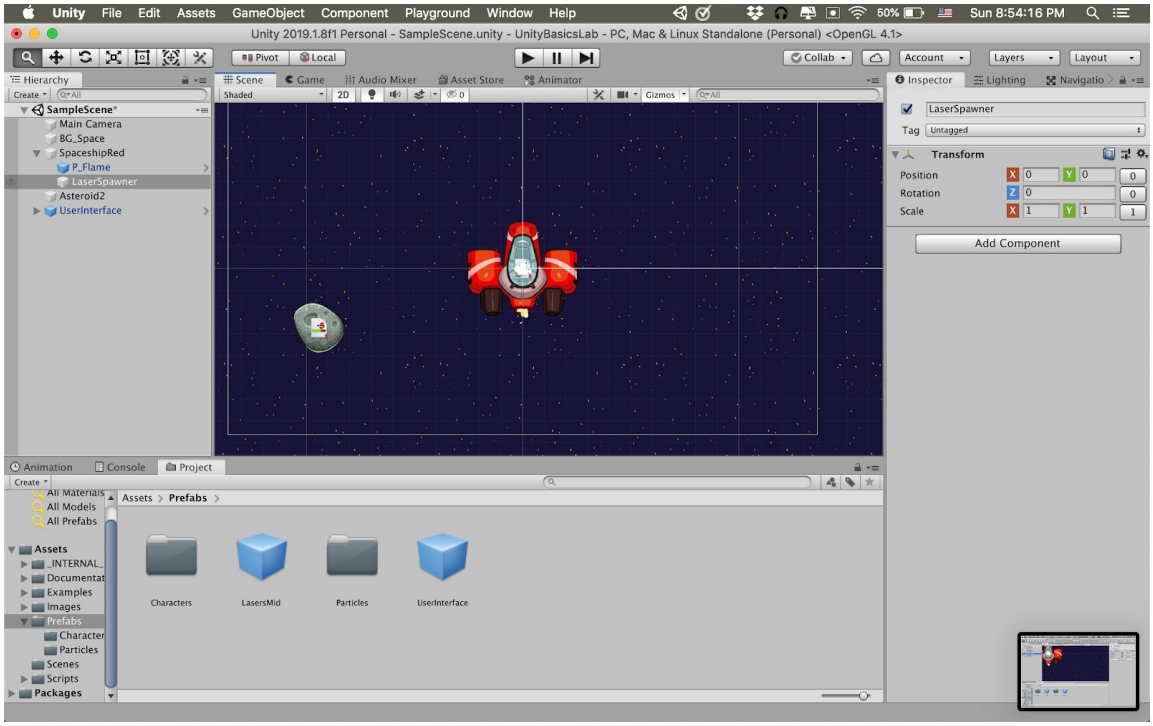
Now add the Bullet Attribute Script. This keeps track of who shot the bullet, which is useful for keeping track of score.

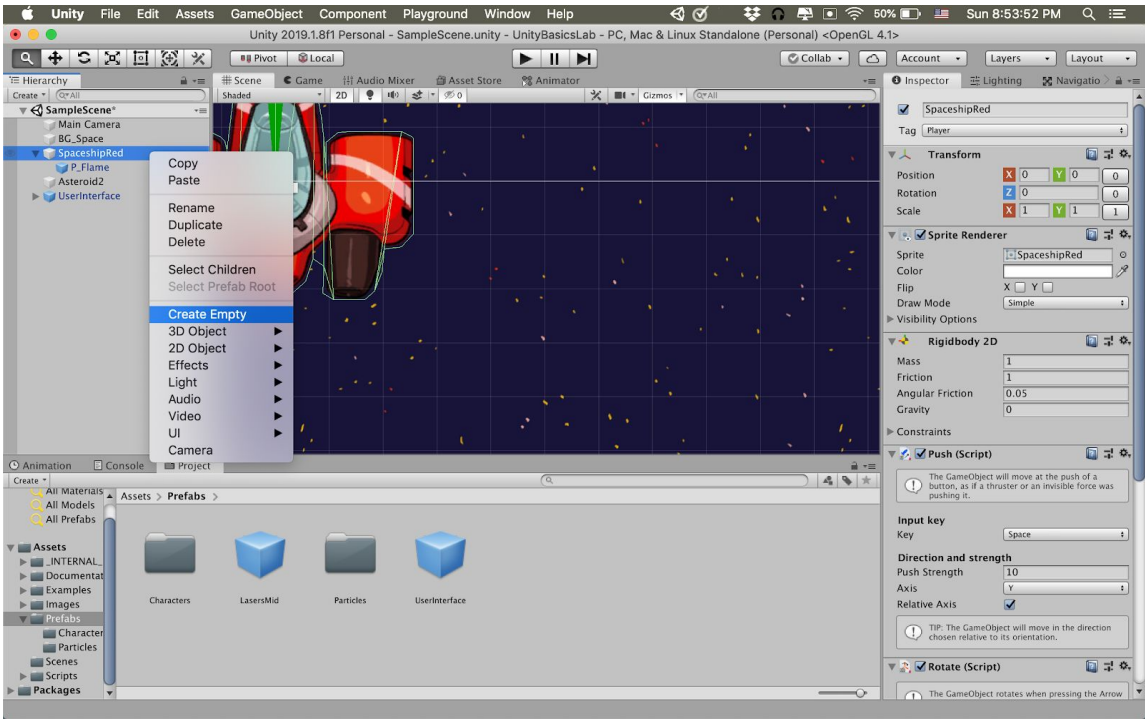


Now we want to make the laser a prefab. A prefab is an instance of a game object, so it saves a game object and all its components and fields, so that you can create multiple instances of the exact same game object. This way we can make copies of the laser shot when the game is running. You can do this by dragging the laser from the hierarchy into the prefabs folder in your Assets.

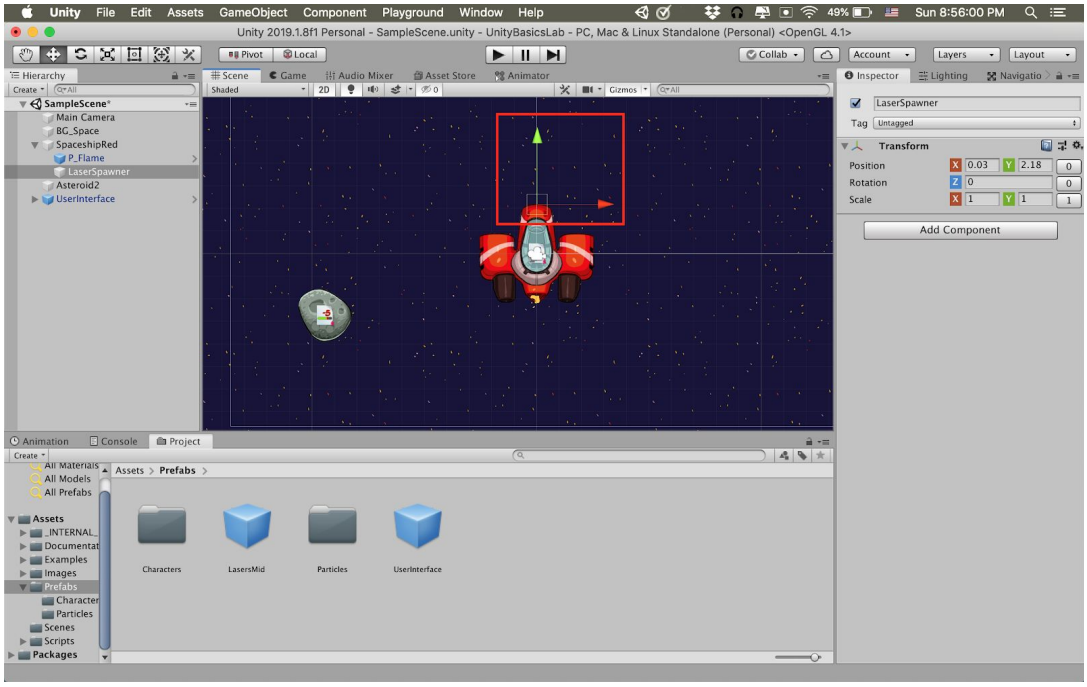


Once you have a copy of it in your prefab, you can delete the one in the hierarchy since we don't need to edit it anymore. Now let's make sure it spawns from the correct location. Select the ship in the hierarchy and create a new game object. You can do this by right clicking on the ship and creating a new empty object. You can name it something like laser spawner.

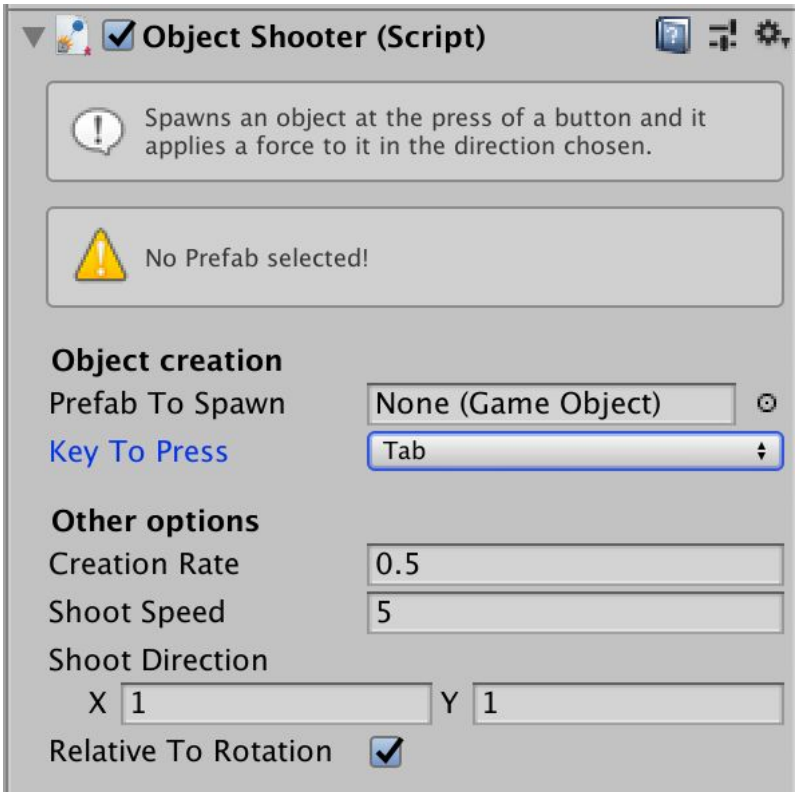




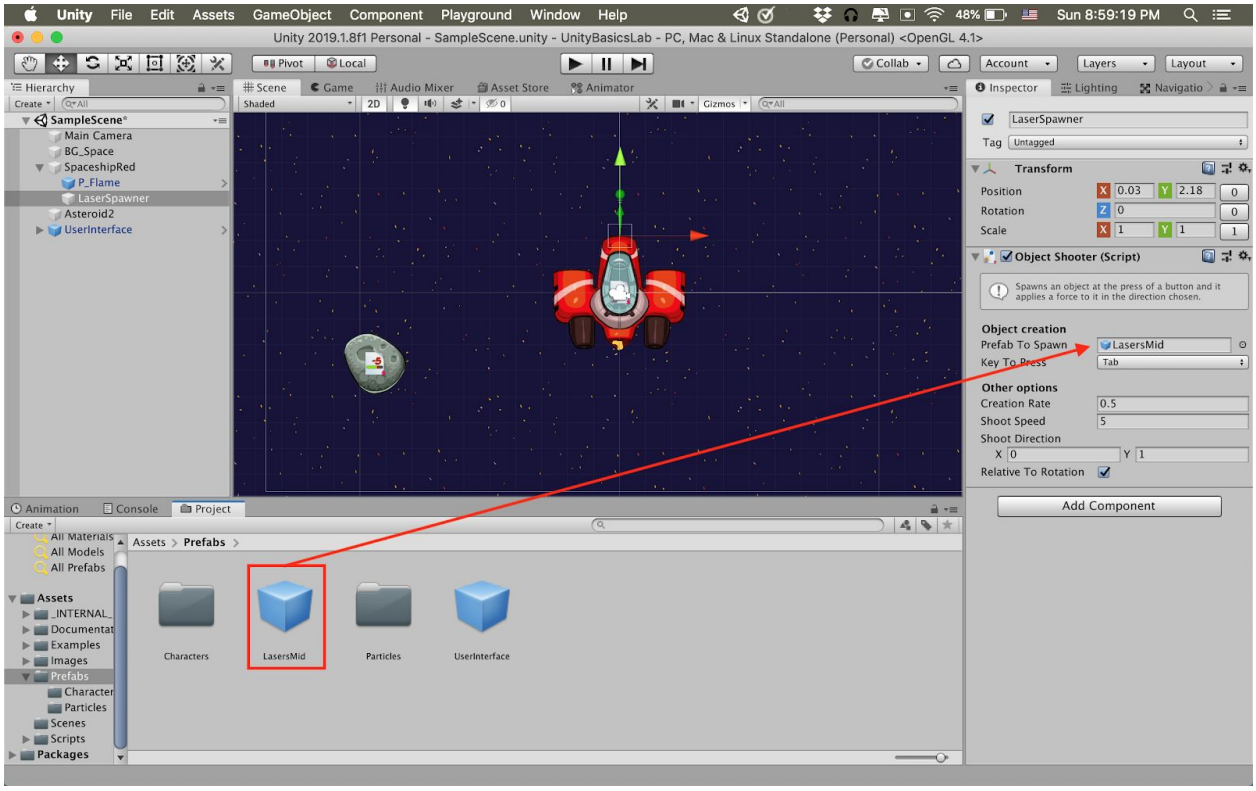
You can move laser spawner onto the position from which you want the laser to be shot. Since I want it to shoot from the tip of the ship, I will move it to the tip, like so.



Now add the Object Shooter Script to the laser spawner. I also changed the shooter key so that it is different from the push key.



Set a reference of the laser to the script by dragging your new laser prefab into the Prefab To Spawn section. You can change the other settings as you wish. With the Shoot Direction, you want to set the x to 0 and y to 1. In the script this is regarded as a vector, and this way the laser will be shot forwards from the ship.

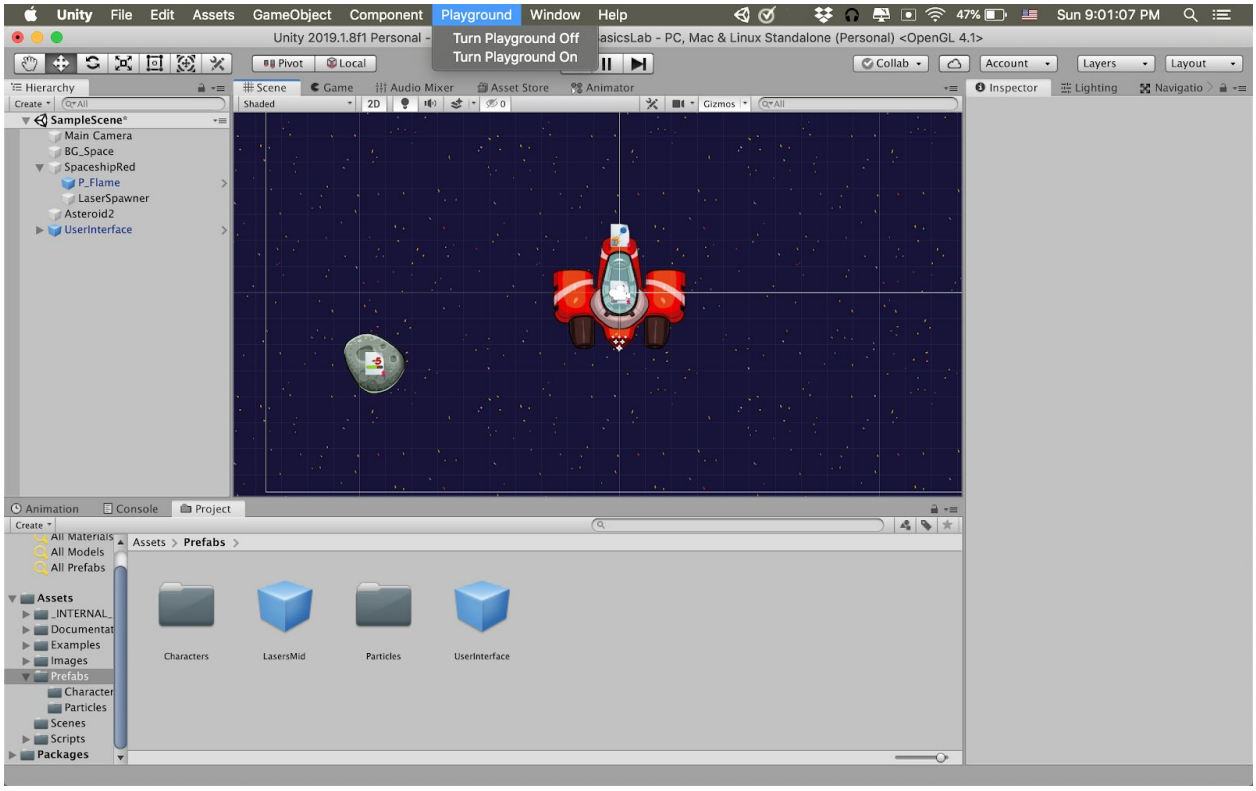


Finally add the Destroy for Points Attribute Script to the asteroid. This will allow the player to gain points for destroying asteroids.

Once you press play, you can destroy asteroids and gain points once you do.

Difference between Playground and non Playground

If you've used Unity before, you can see that most of the components are simplified. This is very easy to see int he Transform and Camera components. If you want to have full control, you can select Playground from the Toolbar and select Turn Playground Off. When you do, you will see the original inspectors. You can toggle between the modes at any time you would like.



Try it yourself

Now that you have some basics, create a small game. Some ideas you can go off of is a precision based maze game, or a racing game, or shooting game. Feel free to use anything, including functionality that we did not go over in this lab. Hint: you can create a prefab of the asteroid and create multiple instances of the same asteroid by dragging the prefab into your scene.

Check off

Show your game to a facilitator to get this lab checked off!