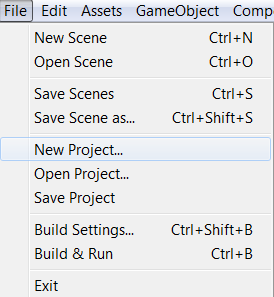
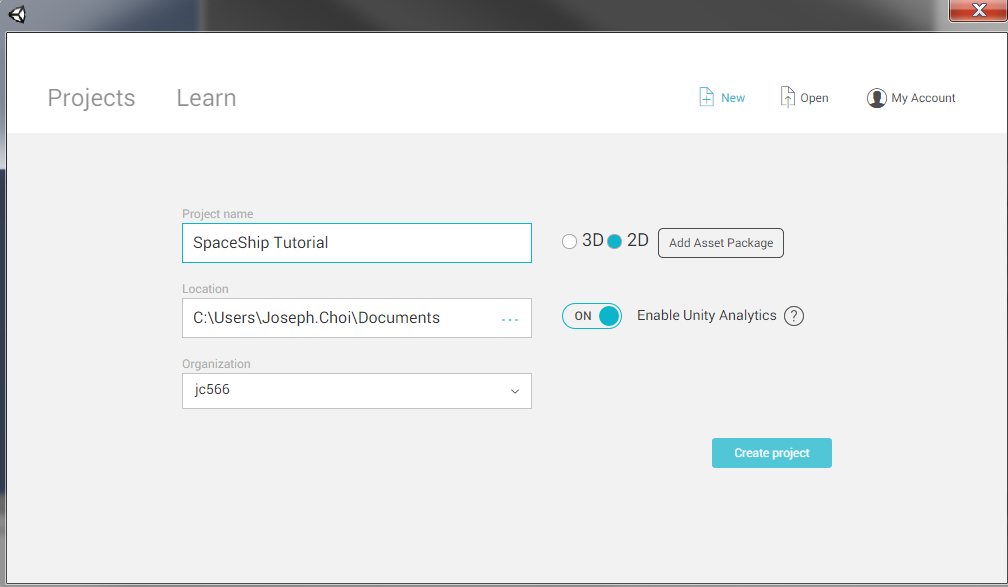
**Introduction**

1. Begin by selecting “New Project”

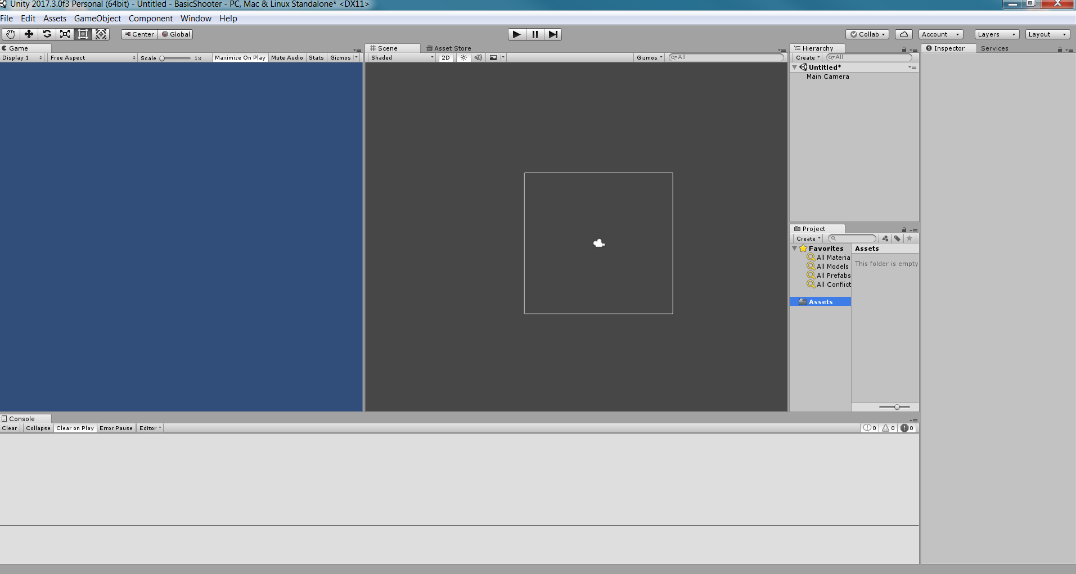


1. Type in the name of your project and ensure that you toggle from 3D project to 2D project.

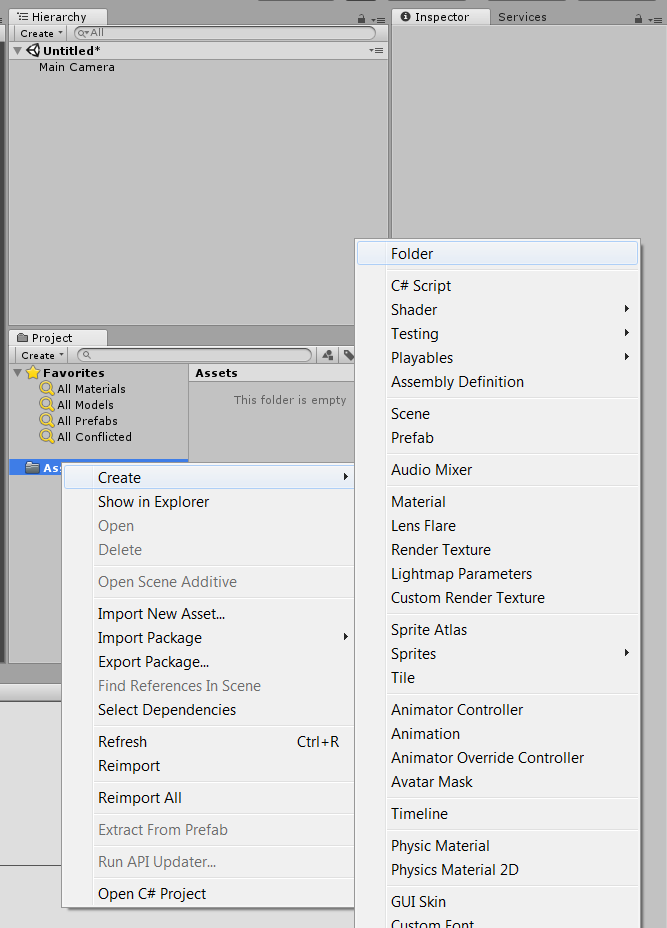


**Lets Begin…**

1. You should see a screen like the image below as a blank project

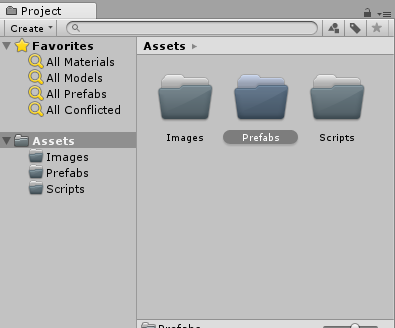


1. Start by making a new folder called “Scripts”. To make a new folder find the “Assets” Folder under the Project tab and right click on it. A new window should appear, hover over Create, and then Left Click on Folder. Repeat these steps to make the other folders listed below.

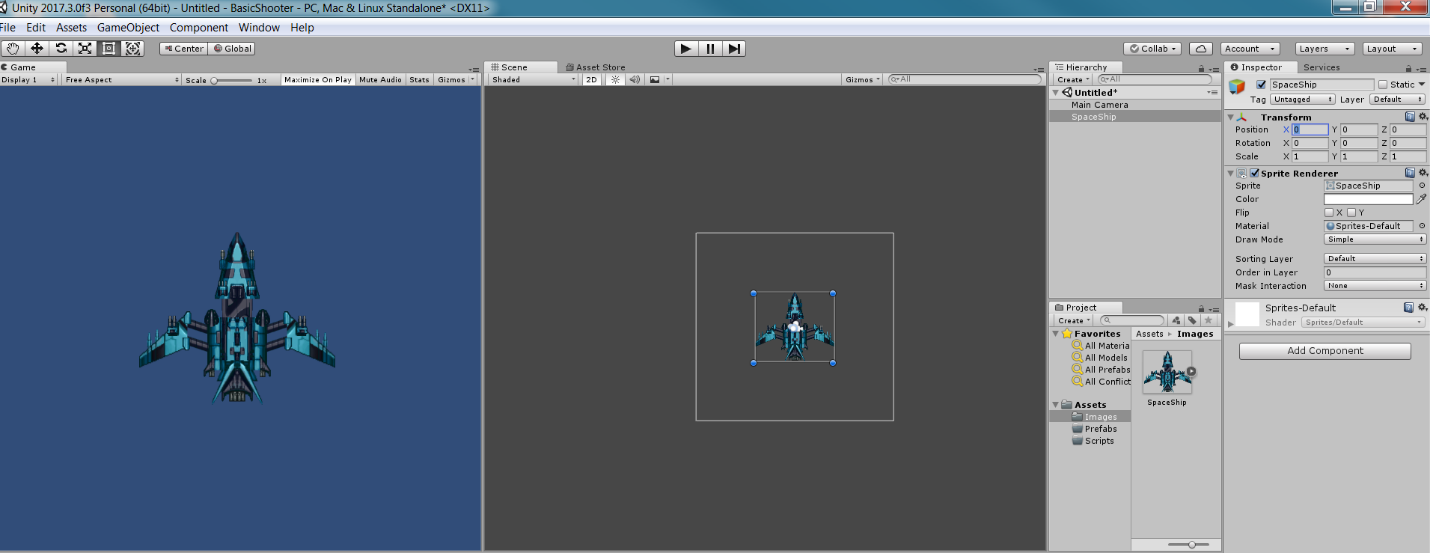


1. Create a new folder called “Images”
2. Drag and Drop the Spaceship image into the “Images” Folder
3. Create a new folder called “Prefabs”

**\*Your Project Folder should look like the image below by the end of this Intro section\***

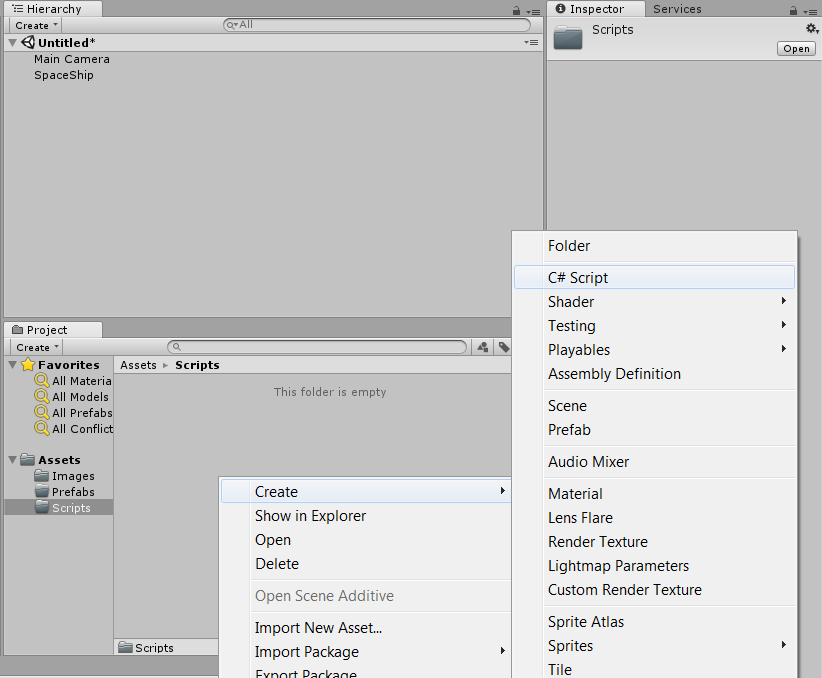
****

1. Goto <https://opengameart.org/content/spaceship-fighter-ipod1> to get a free image of a space ship to use for this tutorial or import your own image.
2. To import an image to the project, goto where your image is located and Drag and Drop the image from the folder window to the “Image” folder.
3. Drag and drop the Space Ship image from your “Images” folder in Unity to the Scene.
4. Finally, click on the game object in the scene, navigate to the Inspector window and reset its Position X,Y,Z to 0s.

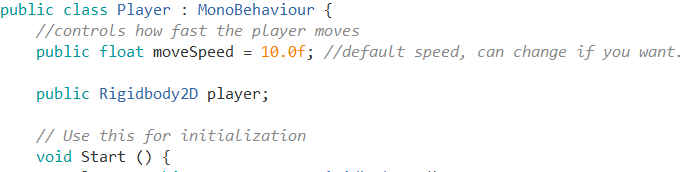


**Scripting – Make the Player Move**

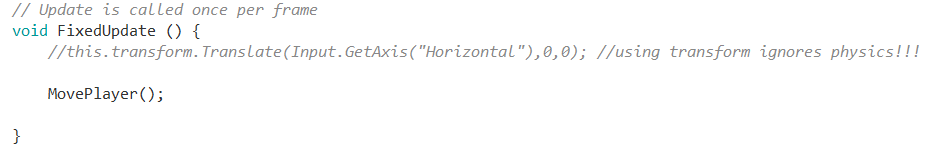
1. Navigate to the “Scripts” folder, it should say “This folder is empty”. Right click anywhere in that space and Create a new C# script. Give this new script the name, **Player**.



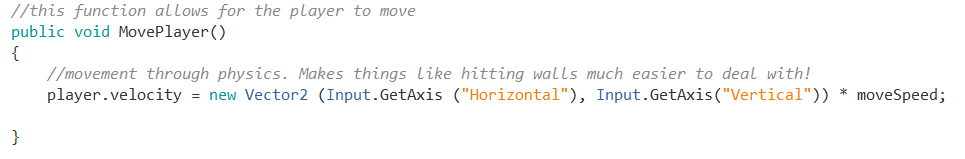
1. If you have Visual Studio installed along with Unity it will open that, otherwise it will open MonoDevelop. Either is fine to develop.
2. Double click on the **Player** script to open it. You should see a new window that looks like the below image(s).
3. Under the “public class Player : Monobehavior {“ and before “void Start() {“, type the following :



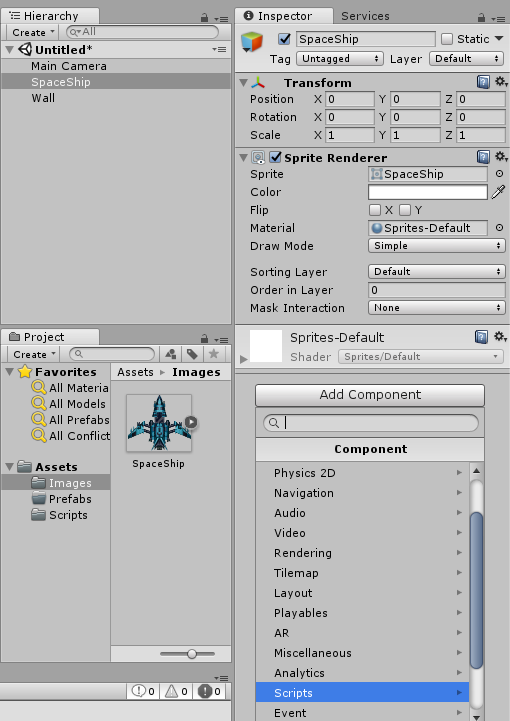
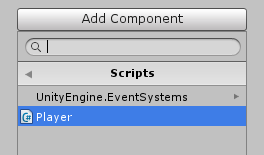
1. Under “void Start() {{“ and before “void Update(){“ type the following :



1. Change “void Update()” to “void FixedUpdate”.
2. At the very end of the file but before the final “}”, type the following :

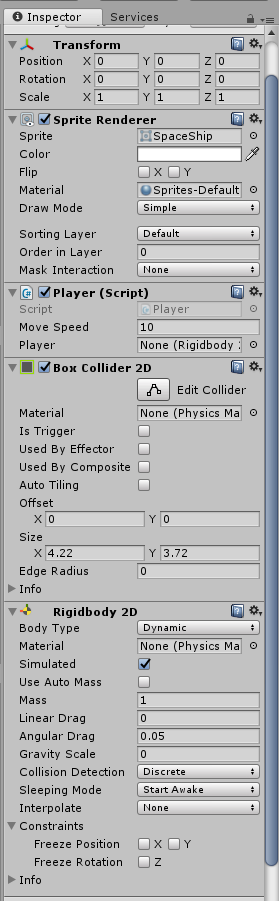


1. Now go back to Unity and attach the script to the Player Game Object. Click on the game object, goto A”Add Component”, find Scripts, and find the Player script and click on it.

1. Now we will add a few more vital components to make it all work!
2. Once again, with the Spaceship (Player) game object selected, find “Add Components” and navigate to “Physics 2D”.
3. While under “Physics 2D”, add **Box Collider 2D.**
4. Repeat step 10, and add **Rigidbody 2D**.
5. Now under the Rigidbody2D, modify the settings :
   1. Under the “Gravity Scale” textbox, change the value from 1 to 0.
6. Now press play and use the Arrow keys to move the player (Space Ship) around!

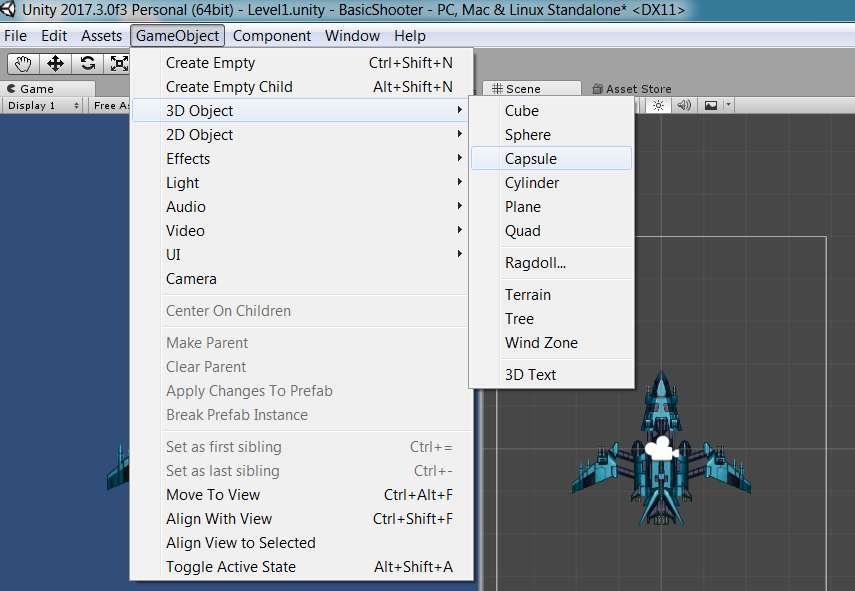
**Your final “Inspector” for Player aka the Space Ship should look like the following image below :**

****

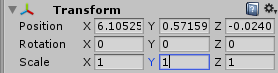
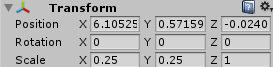
**Congrats! You got the player object to move around with Physics!!!**

**Scripting - Make the Player Shoot!**

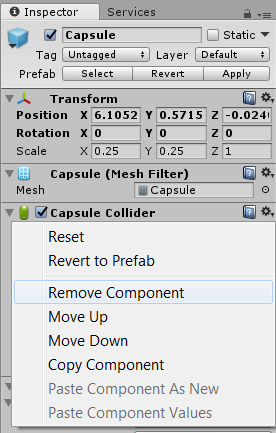
1. Navigate to the toolbar at the top of Unity and click on “GameObject”. Then navigate to 3D Game Object and finally left click on Capsule. This will add a capsule to your project.



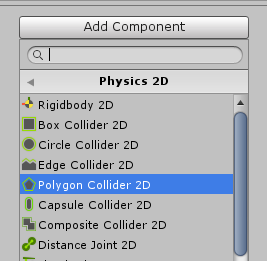
1. Next, select the Capsule in the “Heirarchy” and goto the “Inspector” window. Change the Scale settings for x,y,z from 1,1,1, to 0.25, 0.25, 1.

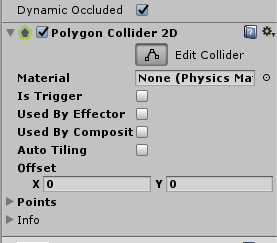
1. Next, goto the Capsule and check out the Inspector. Remove the “Capsule Collider”.



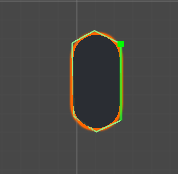
1. Goto Add Components, navigate to Physics 2D, and select Polygon Collider 2D. You should see something like the image on the right below.

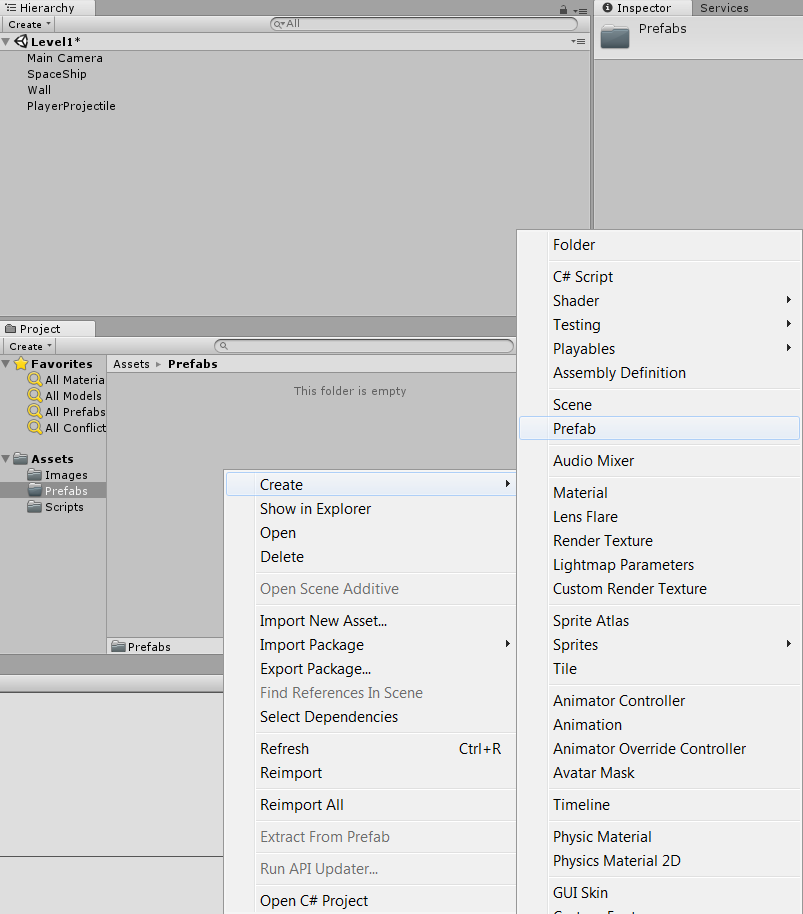
1. Now click on “Edit Collider”, and make a custom shape that wraps around the capsule as best and accurately as possible.



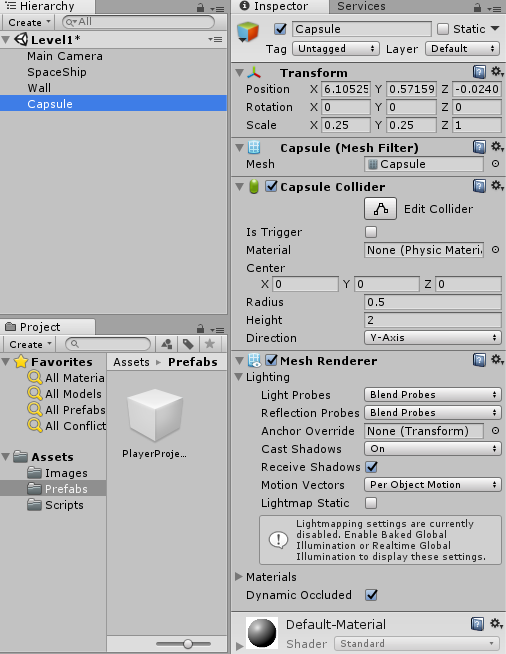
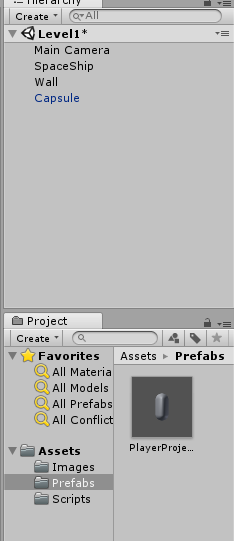
1. Your final Collider box should look something like the image below.



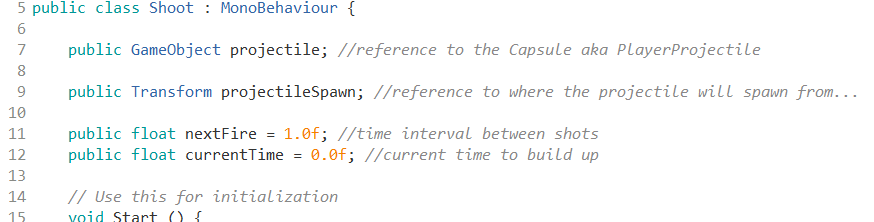
1. Now, goto the “Project” window and select the Prefabs folder. Right click, goto Create, and select Prefab. Name this new prefab **PlayerProjectile.**



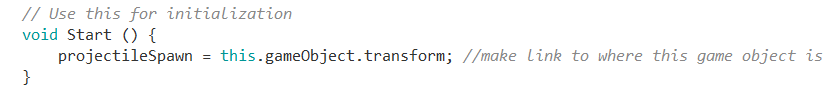
1. Now drag the Capsule object that you changed the Scale of and drop it inside the newly created Prefab you named **PlayerProjectile.**

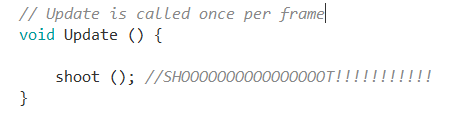
1. Notice how the Capsule is now in blue text and the empty square PlayerProjectile prefab now has the Capsule in it. Next we move on to more scripting so hang tight!
2. Return to the “Scripts” folder and create a new C# script called **Shoot**.
3. Inside the **Shoot** script, after “public class Shoot : Monobehavior {“ and before “void Start() {“ enter the following :



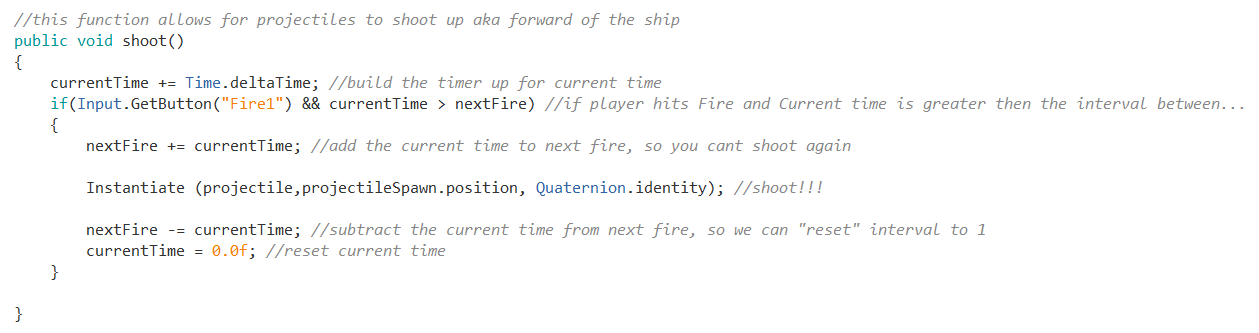
1. Inside “void Start() { }” enter the following :



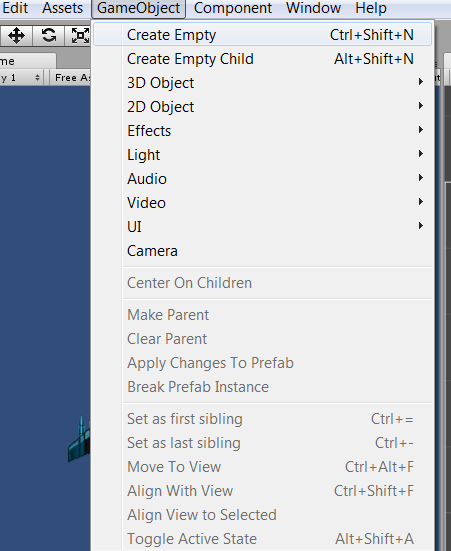
1. Inside “void Update() {}” enter the following :



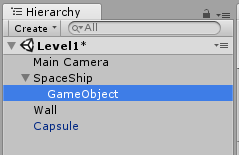
1. After “void Update() {}” create a new function. Enter the following :



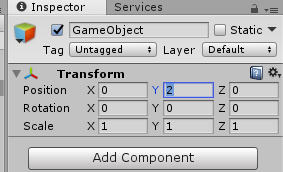
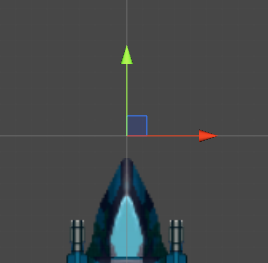
1. Now go back to Unity and attach your script to the player.
2. Under the same toolbar found in step 1, select “Create Empty”.



1. Now select the Empty Gameobject and set all of its Positions in the transform to 0,0,0.
2. Then drag the empty game object and drop it on top of the SpaceShip (Player) game object. SpaceShip should now look like the following picture in the Heirarchy.



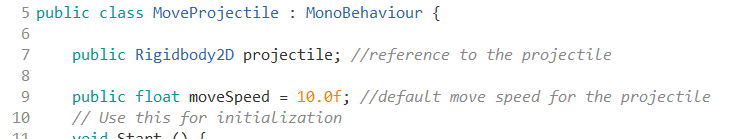
1. Next, goto the empty game object and set its position in the Y axis to 2.

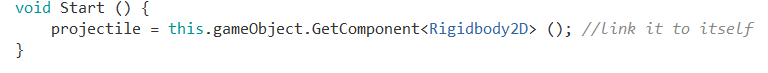
1. AWESOME! You did it! Now we need to give the projectile the ability to move.

**Scripting – Making the projectile move**

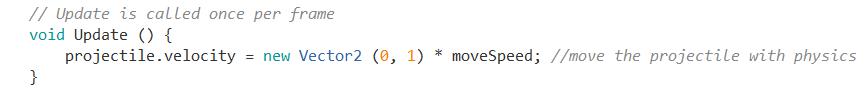
1. Select the **PlayerProjectile** prefab and goto “Add Component”, navigate to Physics 2D and add a Rigidbody 2D.
2. Goto the settings in Rigidbody2D and change the Gravity Scale from 1 to 0 like we did earlier for the player.
3. Now create a new C# script and give it the name **MoveProjectile**.
4. Create the following variables :



1. Place the following under “Void Start()” :



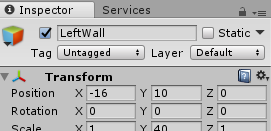
1. Place the following under “void Update()” :



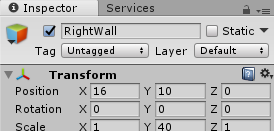
1. Return to Unity and attach the **MoveProjectile** script to the **PlayerProjectile** prefab.
2. Run the game and test it. Now you are creating the projectile and the projectile is MOVING!

**Level Design – Creating some restrictions**

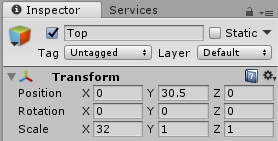
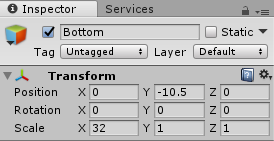
1. Remember that first cube “wall” we made to show the difference between moving with physics and moving with transform.translation? Well its time to repeat that on a bigger scale.
2. Goto Gameobject, navigate to 3D objects, select cube. Remove the Box collider and a Physics 2D box Collider 2D. Give this cube aka the wall the name **Wall.**
3. You should have 2 **Wall** objects, one for left side, the other for the right side.
4. For the left side **Wall** rename it **LeftWall** and give it the following dimensions :



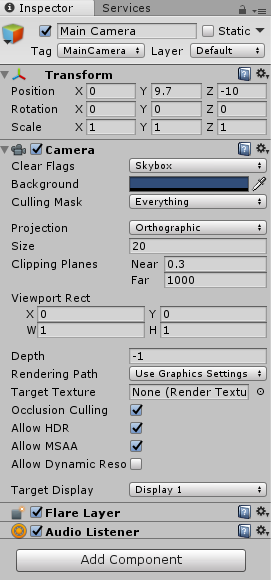
1. For the right side **Wall** rename it RightWall and give it the following dimensions :



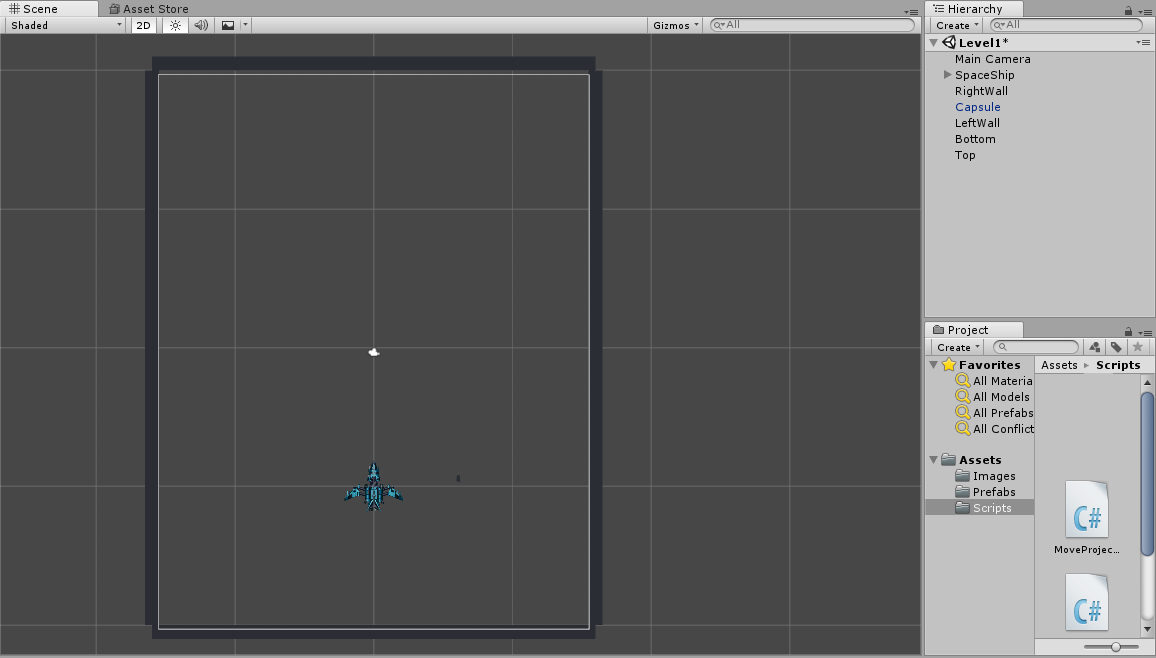
1. Now, repeat step 2 and create two more Cube objects.
2. Give these two new Cube objects the following names : **Top, Bottom**
3. Give **Top** and **Bottom** the following dimensions :

1. These 4 cubes will be our movement restrictions!
2. Next, select the **Main Camera** and change its dimensions and its settings to the following :

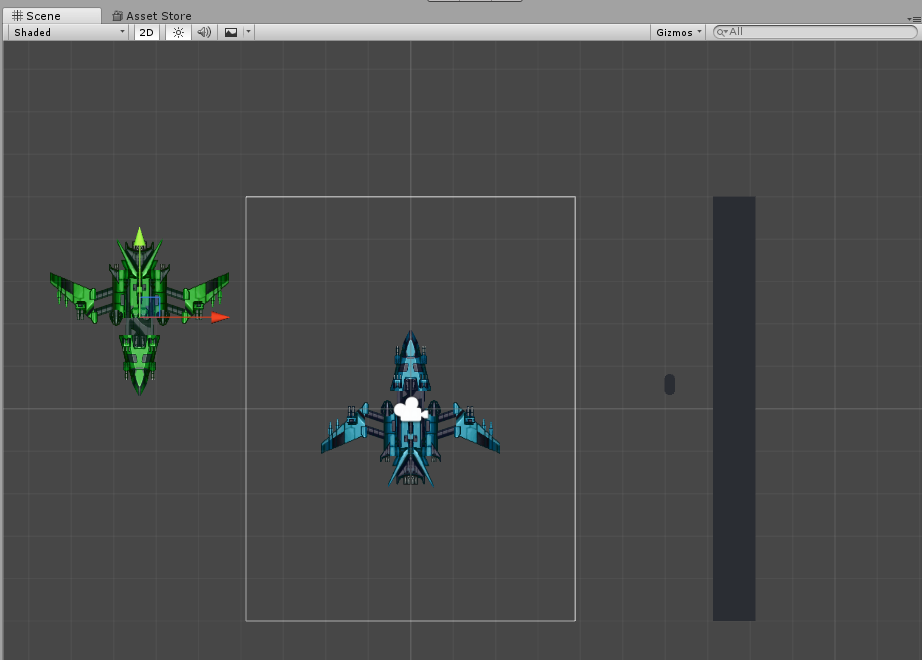


1. Your final outcome should look something like the image below :

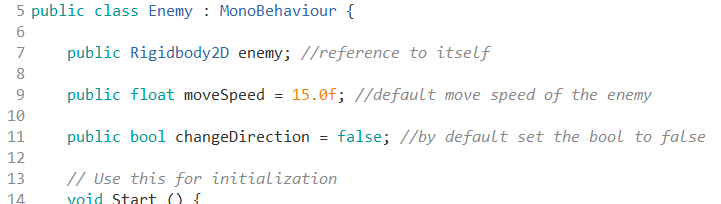


**Scripting – Create an enemy**

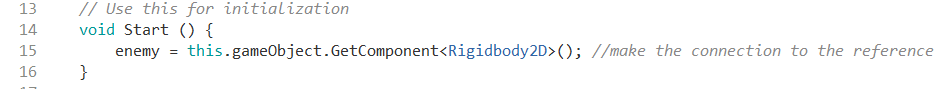
1. Go back to <https://opengameart.org/content/spaceship-fighter-ipod1> and download a different color ship. I chose green for this tutorial. Add it to the “Images” folder just like we did for the player.
2. Rename the new Spaceship to Enemy.
3. Goto the “Prefab” folder and create a new Prefab. Give it the name **Enemy**.
4. Drag and drop the Enemy image onto the Scene.
5. Goto Enemy images transform, and change its position for x,y,z, to 0,0,0 like we did for the player. Then move only the X and Y axis to a desirable location so that it does not overlap with the player space ship.
6. Now goto the Enemy image’s Transform and goto the Rotation for x,y,z and change it to 180,0,0. You should see the ship is now flipped upside down as to face the player ship.



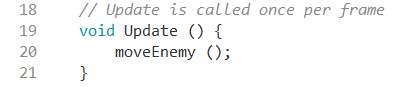
1. Now goto the “Scripts” folder and create a new script called **Enemy**.
2. Inside the **Enemy** script, create the following variables :



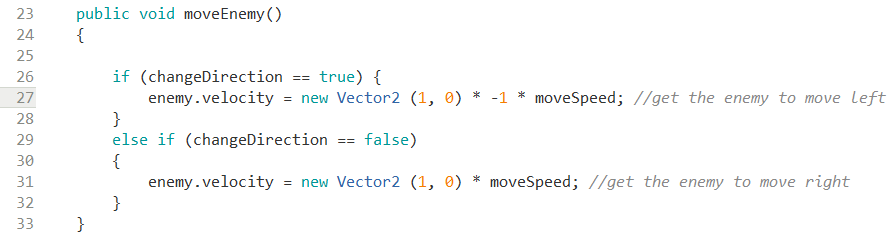
1. Put the following code inside “void Start() {}” :



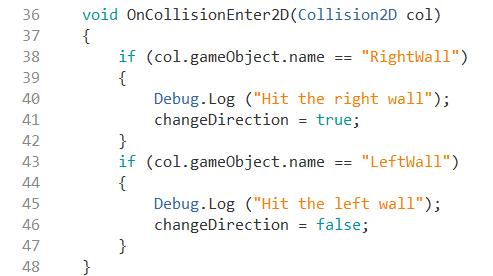
1. Put the following code inside “void Update() {}” :



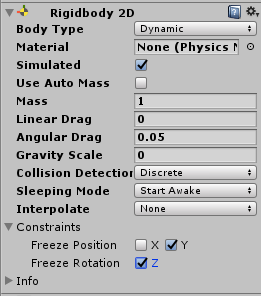
1. Create a new function called “moveEnemy” and put the following code inside it :



1. Finally, create a “OnCollisionEnter2D” function like the below image :



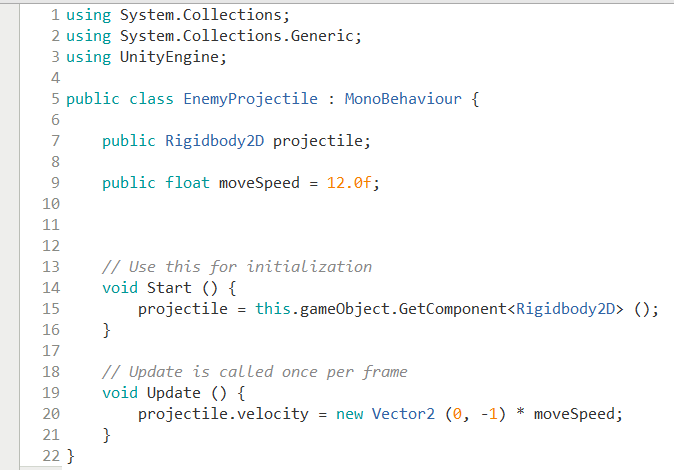
1. Return to Unity and attach the script to the **Enemy.**
2. Additionally, add the following components to the Enemy Prefab :
   1. Box Collider 2D
   2. Rigidbody 2D
3. Inside the Rigidbody 2D attached to the **Enemy** prefab, make the following changes like in the image below :



1. Now we are finished with the movement for the enemy, we can move on to it shooting projectiles.

**Scripting – Enemy Projectile**

1. Create a Capsule game object similar to when we create a capsule for the **MoveProjectile**
2. Make the newly created Capsule a prefab similar to when we made **MoveProjectile,** but this time call it **EnemyProjectile** as a prefab.
3. Goto the “Scripts” folder and create a new C# script called **EnemyProjectile**.
4. The script should look like the following below :



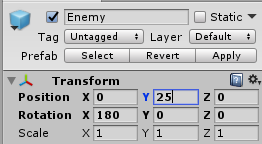
1. Now go back to Unity and attach the script. Next we make the enemy shoot.

**Scripting – Enemy Shooting**

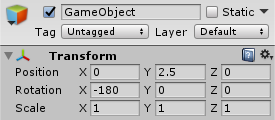
1. Create a new C# script called **EnemyShoot**.
2. The **EnemyShoot** script should look like the below :



1. Then create an empty game object and set its Transform position x,y,z to 0,0,0. Then place the empty game object inside the Enemy in the Heirarchy.
2. Next, set the following settings for the Transform position for x,y,z on the empty game object attached to the enemy. It should be slightly in front of the enemy so it looks like its shooting towards the player. Just in case the position is wrong, reset the **Enemy** in Heirarchy to the following position :



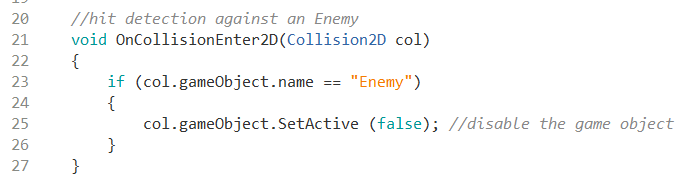
1. Next set the empty game object’s position to this :



1. We are now complete with the enemy shooting. We can move on to hit detection on **EnemyProjectile** to the player and **MoveProjectile** to the Enemy.

**Scripting –Hit Detection for Player Projectiles**

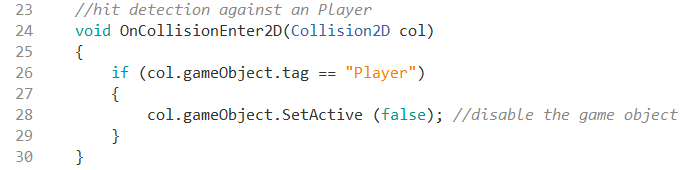
1. Start by editing the **MoveProjectile** script.
2. Add the following code segment to this script :



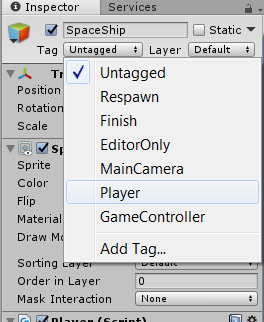
1. Next we move on to editing the **EnemyProjectile** script.

**Scripting – Hit Detection for Enemy Projectiles**

1. Start by editing the **EnemyProjectile** script.
2. Add the following code segment to this script :



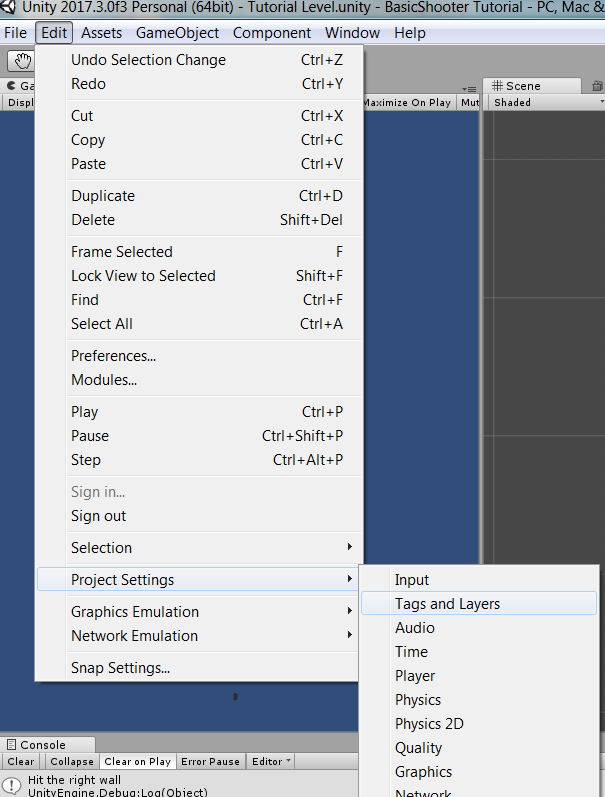
1. Then go back to Unity and find the player SpaceShip. Goto tags, and add the tag “Player” to it.



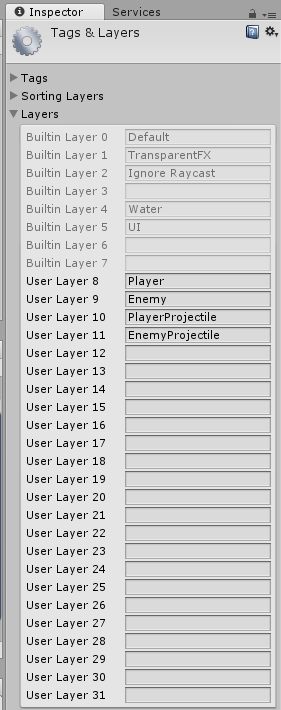
1. Next we move on to LAYERs.

**Game Logic – Layers**

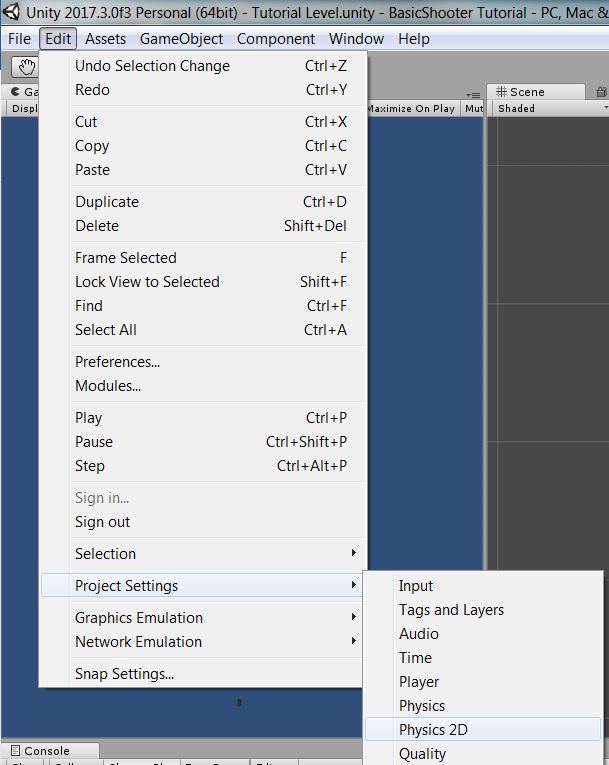
1. Goto the toolbar on the top and click on “Edit” and navigate to “Project Settings”. Hover over it and a new window will pop up and then click on “Tags and Layers”.



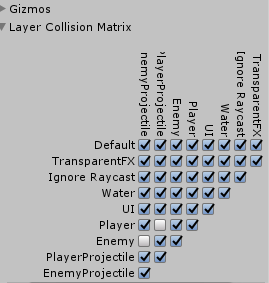
1. Now add 4 new layers called the following :
   1. Player
   2. Enemy
   3. PlayerProjectile
   4. EnemyProjectile



1. Next, goto “Edit” in the toolbar, navigate to “Project Settings” and then in the new window that popped up click on “Physics 2D”.



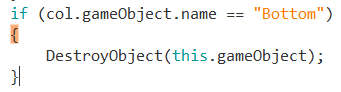
1. Then, make the following changes like in the image below :



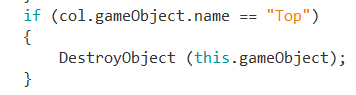
1. Now the Players Projectiles will not collide with the player and the Enemy’s projectiles will not collide with the Enemy.

**Scripting – Destroying the projectiles when they miss**

1. Start by editing the Enemy Projectile and add the following code under the “Void OnCollisionEnter2D” function :



1. Next, edit the MoveProjectile script by adding the following code snippet in the “Void OnCollisionEnter2D” function :



1. Now the Projectiles will disappear when they miss. Whats actually happening is they disappear when they hit the top or bottom wall.