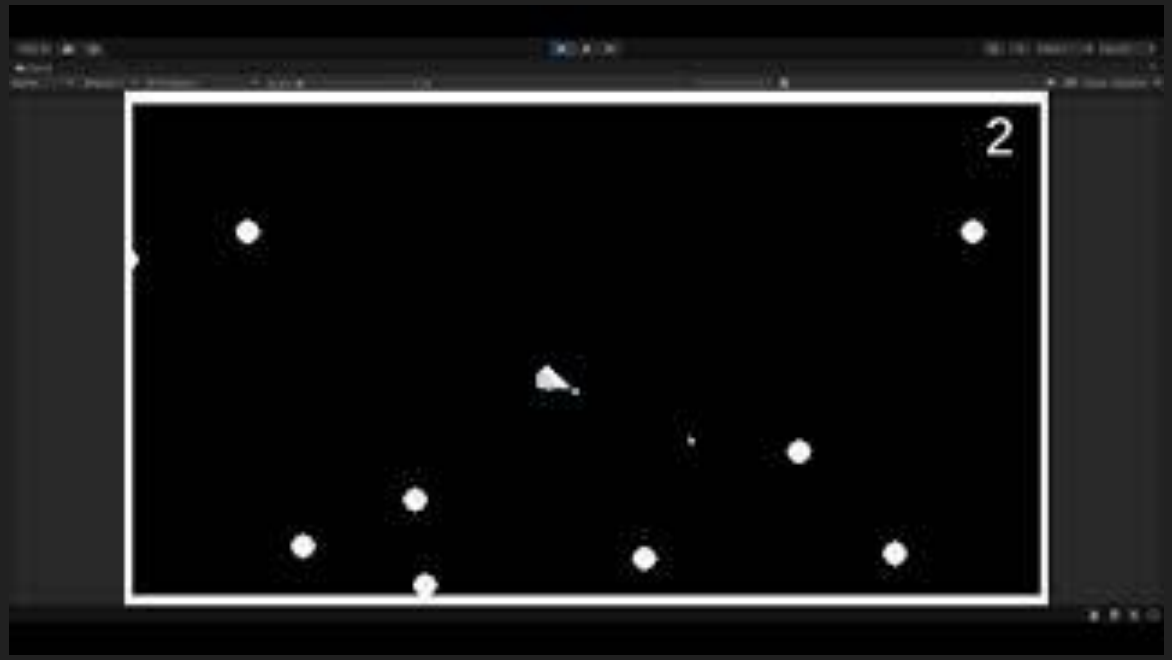
Homework 4 - Programming

End Goal

The goal of this homework of this homework is for you to create several scripts that will allow the player to move around and interact with the world in this asteroids game.

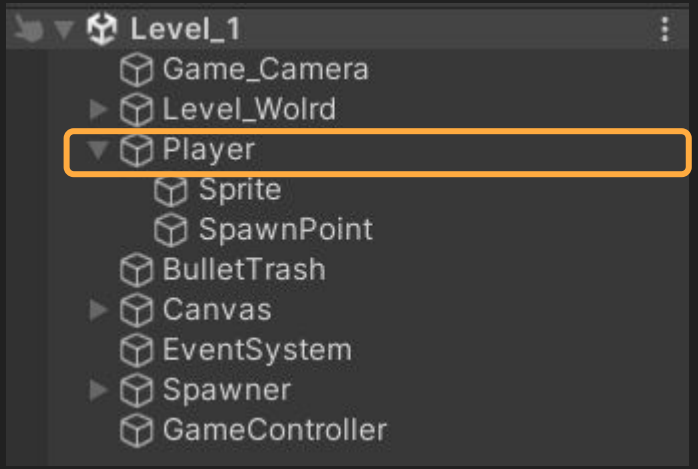


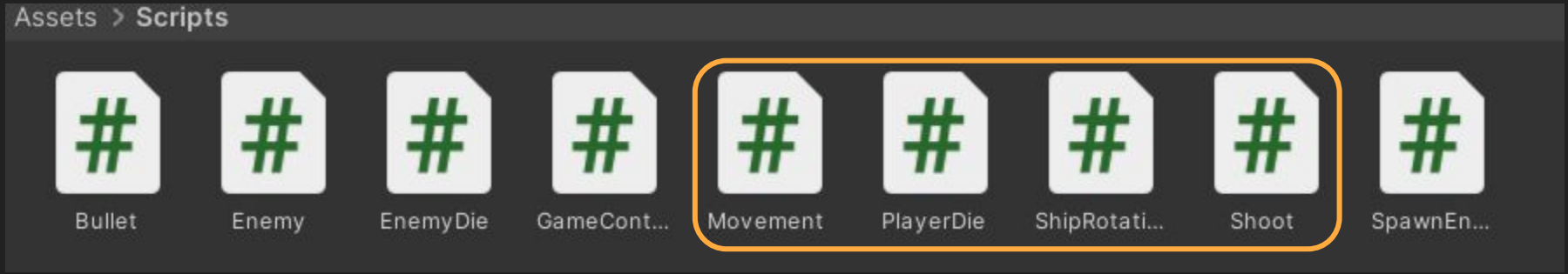
The Level

There are many game objects in the level, feel free to look through them but the only one you will need to work with is the Player parent object. No need to edit the Sprite or SpawnPoint.

Inside the Scripts Folder you will have all the files already made, you will edit Movement, PlayerDie, Ship Rotation and Shoot.

All the other scripts are there to make the rest of the game happen.





Movement

Follow this code inside the Movement script.

This will allow you to make it move using the RigidBody component.

The Input.Get Axis will collect data from WASD and Arrow Key for you action.

The speed variable will allow you to adjust how fast you want your ship to be.

Assembly-CSharp | Movement | _rigidbo

```
using UnityEngine;

Unity Script (1 asset reference) | 0 references
public class Movement : MonoBehaviour
{
    private Rigidbody2D _rigidbody2D; //Rigidbody component that will move player
    private float _xVelocity;   //Keeps track of player input on horizontal velocity
    private float _yVelocity;   //Keeps track of player input on vertical velocity
    public float speed = 3;    //Speed at which player can move horizontally and vertically

    // Start is called before the first frame update
    Unity Message | 0 references
    void Start()
    {
        //Connects to the components
        _rigidbody2D = GetComponent<Rigidbody2D>();
    }

    // Update is called once per frame
    Unity Message | 0 references
    void Update()
    {
        _xVelocity = Input.GetAxis("Horizontal") * speed;
        _yVelocity = Input.GetAxis("Vertical") * speed;
        _rigidbody2D.velocity = new Vector2(_xVelocity, _yVelocity);
    }
}
```

Player Die

The Player Die script will use the built in OnCollisionEnter2D method to check if the object it hit is the “Enemy” asteroid and if it is so will disable the player and turn on the Game Over canvas.

Assembly-CSharp | PlayerDie

```
using UnityEngine;

Unity Script (1 asset reference) | 0 references
public class PlayerDie : MonoBehaviour
{
    //The UI
    public GameObject canvas;

    //Checks if enemy has collided with player if so end the game
    Unity Message | 0 references
    private void OnCollisionEnter2D(Collision2D collision)
    {
        if(collision.gameObject.tag == "Enemy")
        {
            //Turns on the game over UI
            canvas.gameObject.SetActive(true);
            //Turns off the player
            gameObject.SetActive(false);
        }
    }
}
```

Ship Rotation

Here we're going to grab a reference to the camera, and using that and the position of the mouse we will rotate the ship to look in the direction that you're point.

Assembly-CSharp | ShipRotation

```
using UnityEngine;

Unity Script (1 asset reference) | 0 references
public class ShipRotation : MonoBehaviour
{
    //Camera + Mouse
    private Camera _camera;    //Camera Game Object
    private Vector3 _mousePos; //Keeps track of where the mouse cursor is

    //Connects Game Objects
    Unity Message | 0 references
    public void Start()
    {
        //Connects Components
        _camera = GameObject.Find("Game_Camera").GetComponent<Camera>();
    }

    // Update is called once per frame
    Unity Message | 0 references
    void Update()
    {
        //Gets the player mouse in respect to the camera
        _mousePos = _camera.ScreenToWorldPoint(Input.mousePosition);
        //Gets where the based on where the mouse and character are
        var pos = _mousePos - transform.position;
        //Gets the rotation based on this position difference
        var rotZ = Mathf.Atan2(pos.y, pos.x) * Mathf.Rad2Deg;
        //Updates the rotation based on previous calculations
        transform.rotation = Quaternion.Euler(0, 0, rotZ - 90);

```

Shoot

We're going to create many variables to shoot.

PreFab will hold the prefab of the bullet.

Bullet Trash will be the game object the bullets are parented under.

Bullet Spawn is where the bullet will appear.

Timer will keep track of how often the player can shoot and we will do that using two methods we create called BulletSpawnTimer and SpawnBullet.

```
using UnityEngine;

Unity Script (1 asset reference) | 0 references
public class Shoot : MonoBehaviour
{
    //Game Objects
    public GameObject preFab;        //The Bullet PreFab we will spawn
    public Transform bulletTrash;  //Where the bullets will be placed upon spawning
    public Transform bulletSpawn;  //Where the bullet will be spawned at

    //Bullet Spawning Timers
    private const float Timer = 0.5f;  //How long should it take till player can next bullet
    private float _currentTime = 0.5f; //Counter to allow player to shoot
    private bool _canShoot = true;     //Tells us if we can shoot or not

    // Update is called once per frame
    Unity Message | 0 references
    void Update()
    {
        BulletSpawnTimer();
        SpawnBullet();
    }

```

Bullet Spawn Timer

In Bullet Spawn Timer you will count down the timer, if the timer reaches 0 or less time the player is allowed to shoot another bullet.

```
//Checks if the player can shoot, if they can't counts down till they can again
1 reference
private void BulletSpawnTimer()
{
    //If player can shoot don't do anything else, count down
    if (_canShoot) return;
    _currentTime -= Time.deltaTime;
    //If timer is less than 0 allow player to shoot and reset the counter
    if (!(_currentTime <= 0)) return;
    _currentTime = Timer;
    _canShoot = true;
}
```

Spawn Bullet

Checks if the player can shoot or has clicked the Left Mouse Button.

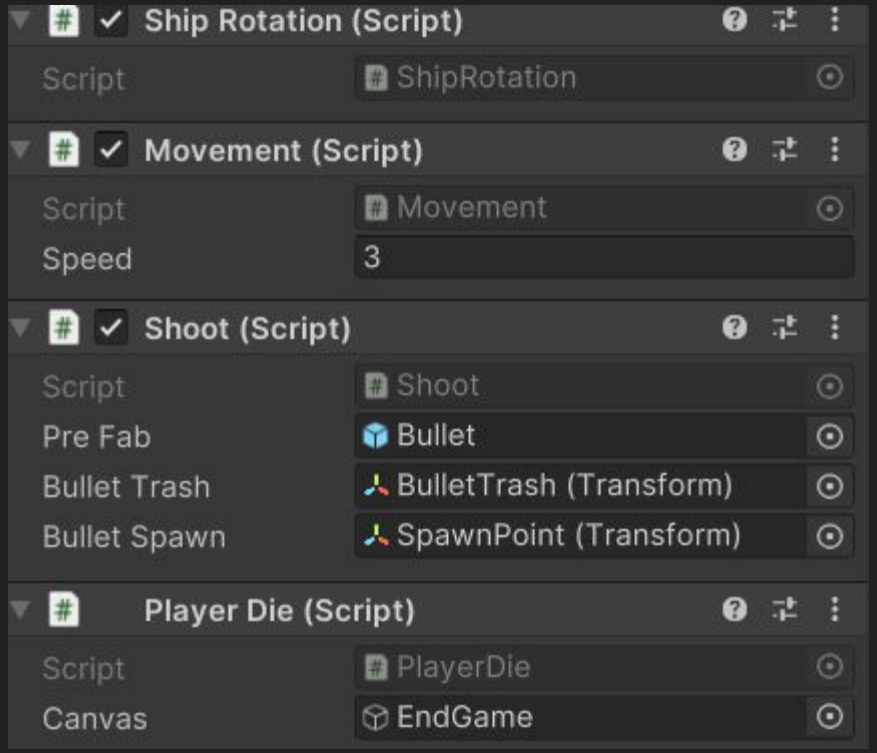
If they have does so you Instante a bullet, connect it to the trash collector and prevent the player from shooting till the timer reset.

```
    //Creates the bullet, puts it in a trash game object and stops player from shooting further
    1 reference
    private void SpawnBullet()
    {
        //Checks if player can shoot/clicked input to shoot
        if (!Input.GetKey(KeyCode.Mouse0) || !_canShoot) return;
        //Make bullet
        var bullet = Instantiate(preFab, bulletSpawn.position, Quaternion.identity);
        //Attach to trash
        bullet.transform.SetParent(bulletTrash);
        //Stop player from shooting
        _canShoot = false;
    }
}
```

Add the Scripts

Make sure to save all your changes to the scripts and when they are ready drag and drop them to the Player Object.

You should see 4 scripts and 2 of them you will need to configure by connect game objects and assets.

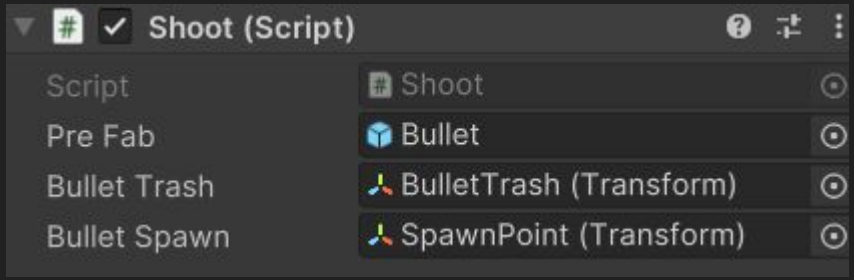


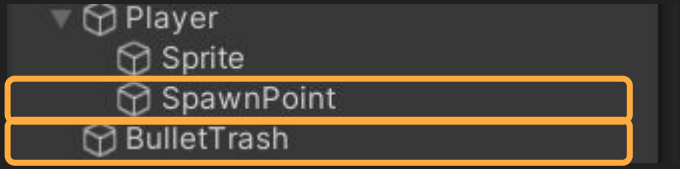
Shoot Component

Drag the Bullet PreFab found in PreFabs folders into the slot. This will now have a reference to what a bullet is and how it should act like.

From your Hierarchy drag the Bullet Trash, this is where the bullet will be peranted under.

From your Hierarchy drag Spawn Point this is where the bullet will be Instantiated.

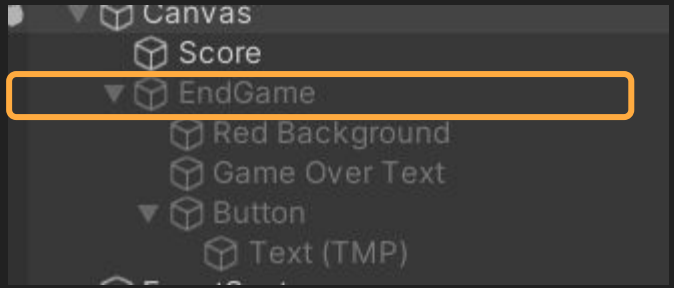




Player Die

From you Hierarchy drag the EndGame object to the Canvas slot in Player Die.

As you can see it's grey out, so not actively on in the game. What the script will do is turn it on with all of the UI object that are connected to it.



You’re Done Enjoy the Game