Games Development Project with Unity

By Edward O'Reilly & Jordan Flanagan

Covering the games Conceptualisation, development, and realisation.

The culmination of various technologies and softwares such as Unity in conjunction with C# programming language and Gimp photo editing software.

General Game Info

The game we wanted to create was one that we would both enjoy not only working on but also playing as in the end if it wasn't an enjoyable player experience then we would feel the project would not have been a success. The overall theme and inspiration was a top down shooter that would become a fast paced almost 'Bullet Hell' type of game with neon theme that would be appealing yet manageable to create and adjust if we needed to pivot the scale and range that our game would reach in terms of its over arching goals and to what level of depth and complexity we could reach with the time constraints we were given.

Player

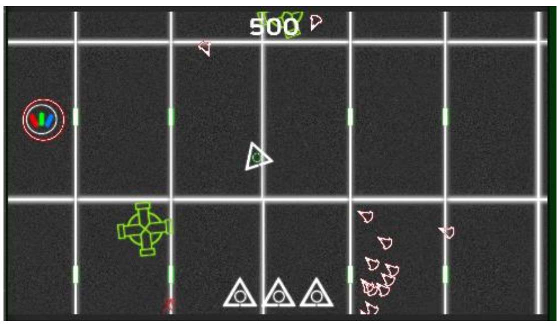
The player design was relatively simple yet unique in that we didn't want it to resemble any existing media and so kept it simple yet effective. The overall movement needed to be smooth and responsive in order for the player to feel in control at all times even with everything going on in the game at once; the varying enemy types and projectiles causing a somewhat chaotic scene.

Camera Follow

Another important aspect was when traversing the map it would be necessary for the camera to follow smoothly, we ran into some difficulty at first as the camera would follow in a jittery manner that was off putting but after reading through some documentation of unity we ended up using a Lerp function that allowed it to more gradually and more smoothly follow the player by attaching the function to the camera and then adding the element of the player to the as a focus point.

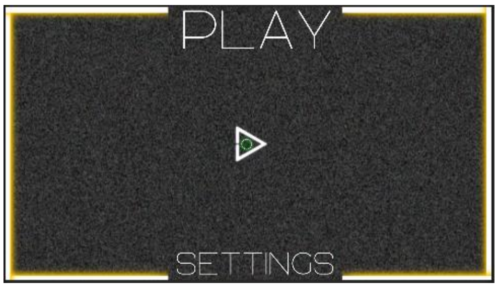
Area Map

The map designed was a simple arena based grid that would allow for the enemies to spawn throughout it the background is a kind of chequered neon pattern that was simple and we were quite fond of. The outer limits of the map is stopped by a wall with a harsh green tint to the outer plains that are unreachable due to the collider system in place to ensure no player go beyond the allocated area and stay within the confines of the arena to ensure the challenge remained it was set to a generous size as to not bombard them entirely with enemies and projectiles overwhelming the player.



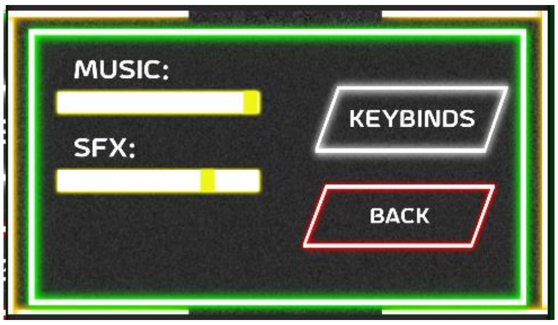
Traversable Menu

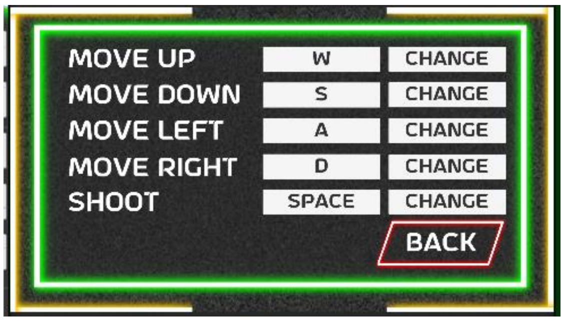
The initial menu the player is met with is one which alike many other games involves the player moving throughout it to get to various desired game states in this case it is rather straight forward giving the option to go the above area which brings the player to a start screen and the below that brings them to the setting of the game.



Settings

The settings of the game are very flushed out given the adjustable volumes of various sound collections of the game via sliding bars and on top of this a very common feature to games released that being remappable keys to change the controls to that of the desired controls the payer wants.





MOVE UP	W	CHANGE
MOVE DOWN	S	CHANGE
MOVE LEFT	A	CHANGE
MOVE RIGHT	D	CHANGE
SHOOT	SPACE	CHANGE

Scoring System

The scores are kept track of, and the highest is saved even after the game. Upon each kill of an enemy the score the player has is incremented by the amount necessary and so when the player loses all 3 of their lives, they are presented with a game over screen which displays the score they attained throughout their play session.

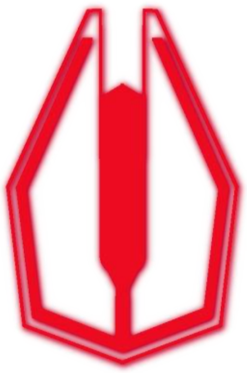
Enemies

The development and Conceptualisation of Enemy Types was a key element in the success of the game our desire was to create a game which could balance along a thin line of both challenging and enjoyable. The need for more than one was apparent from the offset as we needed to create some kind of diversity for the player as facing a singular enemy wouldn't give much of a feel of variety to the gameplay and could cause a lack of challenge as well as replay ability.

Due to some time constraints the creation of two different enemy types were carried out as outlined below they both serve a very important purpose as they offer a unique means of keeping the player on their toes during game time. Of the two we wanted to create one which would keep the player moving constantly and feel pressure; as though they cannot sit comfy otherwise they will likely die in the game and lose a life. The other was to fulfil the purpose of having a plethora of projectiles that the player needed to avoid and traverse through when avoiding the first enemy type.

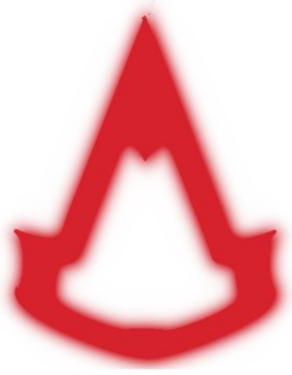
Enemy Type 1 :- Kamikaze

This was designed to keep a constant and consistent pressure on the player to stay mobile and avoid dying by keeping themselves on the move. The design when through a number of different visual concepts that evolved over time and were drawn up using gimp a photo editing software that can also be used for this kind of purpose. They were first drawn out and then after the colours and glow effects discussed between the two of us and then we opted for our preferred design with the preferred colour and glow that would be best suited to fit the overall aesthetic of the game with all of its surrounding details. That being we have focused on a sort of retro style neon theme and figured it would be best for the enemy types designs to follow suit.



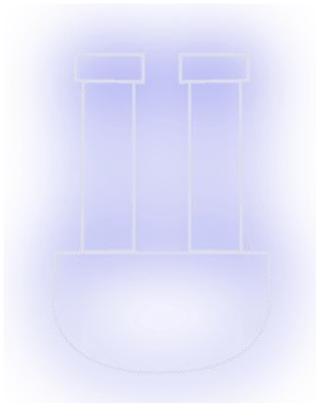
For the first model inspiration was taken from some existing media; some ideas were thrown about and it felt very similar to media such as star wars although in the end we found this was too blocky and felt as though it was too similar in design to the players ship model and so we opted for the latter design below which was a lot more curvy and is almost blade shaped which was fitting for the circumstances of it being a colliding enemy type that would try to hit and make contact with the player.

As shown to the right the design is quite unique and overall we were very happy with it and needless to say based on the gameplay it worked as expected in the in-game performance. The new model had two states the first being a grey or off-white colour until the enemy is within a certain range of the player it would then increase in speed and change to a sort or red glow in a sort of 'Kill Mode'.



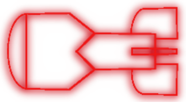
Enemy Type 2 :- Turret

This enemy type is designed to better fill the screen with projectiles to add a sort of ‘bullet hell’ type of dimension to the gameplay and so we alike with the first enemy type we began conceptualising the design in terms of both its functionality and its visual aspect and so a number of prototypes were drawn up again narrowing the developments down into a decision between two to find our best enemy design.

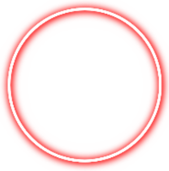


This first design is not the one we went with in the end but it took inspiration from a sort of double barrel shotgun type of turret that would rotate and scan around for a specific target until it was within a specific range and it would then fire a sort of spread of shots. With this design we found it to be very limiting in given the player speed was rather nippy we felt that this design did not pose enough of a threat.

This is why we found that a turret that shot at random intervals set within a certain range and it also instead of following the above double barrel design instead we opted for four barrels each 90 degrees apart from one another. This offered a great deal of projectiles in all directions as the turret then slowly rotated around and thus covered a greater area of effect. There would be a number of these spawned across the play area that would then cause further trouble for the player as they mauver around the many projectiles and the other enemy types for their life session.



Enemy Type 3 :- Mine



This is a stationary obstacle for the player. When the player hits the mine they get pushed in the opposite direction, with rigidbody.addforce, with the force mode set to impulse.

This enemy also spawns the powerups which are available to the player.

Power- Ups

Power up 1 :- Trishot

The first power up we added to the game is the trishot. This makes the player shoot 3 projectiles instead of just a singular one. This lasts several seconds

Power up 2 :- Shield

The second powerup we added was the shield powerup which shields the player from any damage. This lasts several seconds.

Art

All of the art in the game was created by us, in GIMP. This is an image manipulation program like photoshop. All sprites and menu elements were drawn up in this software.

Sounds

All sounds were taken from websites like freesound.org and youtube. The music was taken from geometry wars retro evolved. This was available free on youtube.

CODEBASE

Enemy Manager

```
using System.Collections;

using System.Collections.Generic;

using UnityEngine;


public class EnemyManager : MonoBehaviour

{

    private int maxX = 70;//Max X position for enemy spawn

    private int maxY = 35;//Max Y position, both based on level bounds

    public int maxEnemy = 75; //Max enemies spawned in initial wave


    public List<GameObject> enemies = new List<GameObject>(); //List of all enemies

    public GameObject kamikazePrefab; //Kamikaze Enemy prefab

    public GameObject minePrefab; //Mine Prefab

    public GameObject turretPrefab; //Turret Prefab

    private IEnumerator coroutine; //Coroutine



     IEnumerator SpawnEnemies(){ //Spawn enemies every 5 seconds

        while(true){ //While true

            yield return new WaitForSeconds(5/1.5f); //Wait for seconds

            for(int i=0; i<6; i++){

                float r = Random.Range(0, 100); //Random number between 0 and 100

                GameObject prefab; //Prefab to spawn

                if(r >= 85){ //if r > 85 the prefab is the mine prefab

                   prefab = minePrefab;

                }else if(r >=55 && r < 85){ //if between 85 and 55, use turret

                    prefab = turretPrefab;
```

```
          }else{
             prefab=kamikazePrefab; //else use kamikaze
          }
          int x = (int)Random.Range(-70, 70); //Get Random X and Y
          int y = (int)Random.Range(-35, 35);
           Vector3 spawnPos = new Vector3(x, y, -.2f);//Put X and Y into Vector3
           GameObject enemy = Instantiate(prefab, spawnPos, Quaternion.identity);
           enemies.Add(enemy); //Add to our list of enemies
       }
    }

}
public void ResetWave(){   //Clears the wave of all enemies
    Debug.Log("Resetting Enemy Wave.");
    foreach(GameObject go in enemies){
       Destroy(go);
      }
    enemies.Clear();
}

public void StopSpawning(){ //Stops more enemies from spawning by ending coroutine
    Debug.Log("Spawning Stopped.");
    if(coroutine != null)
       StopCoroutine(coroutine);
}

public void SpawnWave(){
       StopSpawning();
       ResetWave();
```

```
Debug.Log("Spawning New Enemy Wave.");

int currentX = -maxX;
int currentY = maxY;
int currentRow = 0;

//Spawns enemies every X units in the world from Left=>Right and Top=>Bottom
for(int i=0; i < maxEnemy; i++){
    float r = Random.Range(0, 100);
    GameObject prefab;
    if(r >= 95){
      prefab = minePrefab;
    }else if(r >= 85 && r < 95){
        prefab = turretPrefab;
    }else{
        prefab = kamikazePrefab;
    }

    Vector3 spawnPos = new Vector3(currentX, currentY, -.2f);
    GameObject enemy = Instantiate(prefab, spawnPos, Quaternion.identity);

    currentX += 5;
    if(currentX >= maxX){ //If x goes above max, loop back to other side
        currentX = -maxX;
        currentRow += 1; //and add a row
        currentY += currentRow * -5;
    }
    if(currentX >= -30 && currentX <= 30){
        if(currentY >= -20 && currentY <= 20)
```

```
                currentX += 40; //enemies will not spawn in the spawn area
            }
            enemies.Add(enemy); //Adds to enemy list
        }
        Debug.Log("Spawned " + maxEnemy + " enemies.");
        coroutine = SpawnEnemies();
        StartCoroutine(coroutine);

    }

}
```

Game Manager

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class GameManager : MonoBehaviour
{
    //Helper Classes
    public UIManager ui; //UI manager
    public EnemyManager enemyManager; //Enemy manager
    public CameraController camController; //Camera Controller
    public SoundEffects SoundEffects; //Sound Effects

    //Score
    private double playerScore;
    private double highScore;
    //Player Health and lives
    private int playerHealth = 100;
    private int playerLives = 3;

    //Is player in scene
    private bool playerActive;
    public GameObject playerPrefab;
    private GameObject playerObject;

    private  IEnumerator coroutine;
    private IEnumerator invincibleCR;
    public bool playerInvincible = false;
    public double GetScore(){ //Get Score
        return playerScore;
    }

    void  Start(){

        highScore = double.Parse(PlayerPrefs.GetString("HS", "0")); //Load HS playerpref, parse
as double, set highscore to that value, default is "0"
        //using a string as we cannot save a double using player prefs

    }
    public void AddScore(double num){ //Add to our players score
        playerScore += num;

        if(highScore <= playerScore){

            SetHighscore(playerScore);
        }
    }
    public int GetLives(){
        return playerLives;
    }
    public double GetHighscore(){
        highScore = double.Parse(PlayerPrefs.GetString("HS", "0"));
        return highScore;
    }
    public void SetHighscore(double num){
        highScore = num;
        PlayerPrefs.SetString("HS", highScore.ToString());
        PlayerPrefs.Save();
    }
    public int GetHealth(){


        return playerHealth;
    }
    public void DamageHealth(int damage){ //Takes health away from player by amount
(damage)
        if(!playerInvincible)//check if player is invincible
            playerHealth -= damage;


        if(playerHealth <= 0){ //Kill player if health less than 0
            Debug.Log("Player Died.");
            Destroy(playerObject);


            playerActive = false;
            if(playerLives <= 1){
                ui.SetUI(3); //Load gameover screen if player lives are less than or equal to 1
            } else{ //Else reset player health and remove a live
                playerHealth = 100;
                playerLives -= 1;
                ui.SetUI(2);
            }

        }
        SoundEffects.PlayDamagePlayerSound(); //Play damage sound
    }

    public void AddHealth(int amount){ //Add amount to health
        playerHealth+= amount;
        if(playerHealth > 100) //if health greater than 100, health is 100
            playerHealth = 100;
    }

    public GameObject GetPlayer(){
        return playerObject;

    }
    public bool IsPlayer(){ //check for an active player
        if(playerActive){
            return true;
        }else{
            return false;
        }
    }

    void SpawnPlayer(){ //Spawn the player
```

```
          if(playerObject!= null){ //if player is not null
             playerActive = false; //set to false
             Debug.Log("Destroying old player.");
             Destroy(playerObject);//destroy player
          }
          Debug.Log("Spawning Player.");
          playerHealth=100; //Set health to 100
          playerObject = Instantiate(playerPrefab, new Vector3(0,0,-.2f), Quaternion.identity);
//spawn new player at origin
          camController.SetPlayer(playerObject); //set player object in camera script
          playerActive=true; //set to true
    }

    public void Restart(){


       playerHealth = 100; //set health to 0
       ui.SetUI(1); //open playing UI
       enemyManager.StopSpawning(); //Stop spawning
       enemyManager.ResetWave(); //reset the wave
       SpawnPlayer(); //spawn player
       enemyManager.SpawnWave(); //spawn wave
       GameObject[] pickups =  GameObject.FindGameObjectsWithTag("HealthPickup");
//Find all health pickups
       foreach(GameObject go in pickups){ //destroy all health pickups
          Destroy(go);
         }
    }

    public void ResetGame(){ //Reset the game
       Debug.Log("Setting Up Game.");
```

```
        enemyManager.ResetWave();//Reset wave
        GameObject[] pickups =  GameObject.FindGameObjectsWithTag("HealthPickup"); //find
health pickups
        foreach(GameObject go in pickups){ //destroy
            Destroy(go);
          }
        SpawnPlayer(); //Spawn

        playerHealth = 100; //set health to 100
        playerScore= 0; //set score to 0
        playerLives = 3;  //set lives to 3

        ui.SetUI(1); //Open playing UI
    }

    public void MakeInvincible(){
        playerInvincible = true; //player set to invincible
        invincibleCR = Invincible(); //set invincible co routine
        StartCoroutine(invincibleCR); //start invincible coroutine
    }

    public bool IsPlayerInvincible(){
        return playerInvincible;
    }

    IEnumerator Invincible(){
        int time = 8; //set time to 0
        while(time > 0){//every second, decrement time by 1
            yield return new WaitForSeconds(1);
            time-=1;
        }
        playerInvincible = false; //set invincible to false
        StopCoroutine(invincibleCR);//stop coroutine
    }




}
```

Keybind Manager

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class GameManager : MonoBehaviour
{
    //Helper Classes
    public UIManager ui; //UI manager
    public EnemyManager enemyManager; //Enemy manager
    public CameraController camController; //Camera Controller
    public SoundEffects SoundEffects; //Sound Effects

    //Score
    private double playerScore;
    private double highScore;

    //Player Health and lives
    private int playerHealth = 100;
    private int playerLives = 3;
    }


    //Is player in scene
    private bool playerActive;
    public GameObject playerPrefab;
    private GameObject playerObject;


    private  IEnumerator coroutine;
    private IEnumerator invincibleCR;
    public bool playerInvincible = false;
    public double GetScore(){ //Get Score
        return playerScore;
    }


    void  Start(){


        highScore = double.Parse(PlayerPrefs.GetString("HS", "0")); //Load HS playerpref, parse
as double, set highscore to that value, default is "0"
        //using a string as we cannot save a double using player prefs


    }
    public void AddScore(double num){ //Add to our players score
        playerScore += num;


        if(highScore <= playerScore){


            SetHighscore(playerScore);


        }
    }
    public int GetLives(){
        return playerLives;
    }
    public double GetHighscore(){
        highScore = double.Parse(PlayerPrefs.GetString("HS", "0"));
        return highScore;
    }
    public void SetHighscore(double num){
        highScore = num;
        PlayerPrefs.SetString("HS", highScore.ToString());
        PlayerPrefs.Save();
    }
    public int GetHealth(){


        return playerHealth;
    }
    public void DamageHealth(int damage){ //Takes health away from player by amount
(damage)
        if(!playerInvincible)//check if player is invincible
            playerHealth -= damage;


        if(playerHealth <= 0){ //Kill player if health less than 0
            Debug.Log("Player Died.");
            Destroy(playerObject);


            playerActive = false;
            if(playerLives <= 1){
                ui.SetUI(3); //Load gameover screen if player lives are less than or equal to 1
            } else{ //Else reset player health and remove a live
                playerHealth = 100;
                playerLives -= 1;
                ui.SetUI(2);
            }


        }
        SoundEffects.PlayDamagePlayerSound(); //Play damage sound
    }

    public void AddHealth(int amount){ //Add amount to health
        playerHealth+= amount;
        if(playerHealth > 100) //if health greater than 100, health is 100
            playerHealth = 100;
    }

    public GameObject GetPlayer(){
        return playerObject;


    }
    public bool IsPlayer(){ //check for an active player
        if(playerActive){
            return true;
        }else{
            return false;
        }
    }

    void SpawnPlayer(){ //Spawn the player
            if(playerObject!= null){ //if player is not null
                playerActive = false; //set to false
                Debug.Log("Destroying old player.");
            Destroy(playerObject);//destroy player
          }
          Debug.Log("Spawning Player.");
          playerHealth=100; //Set health to 100
          playerObject = Instantiate(playerPrefab, new Vector3(0,0,-.2f), Quaternion.identity);
//spawn new player at origin
          camController.SetPlayer(playerObject); //set player object in camera script
          playerActive=true; //set to true
    }

    public void Restart(){


        playerHealth = 100; //set health to 0
        ui.SetUI(1); //open playing UI
        enemyManager.StopSpawning(); //Stop spawning
        enemyManager.ResetWave(); //reset the wave
        SpawnPlayer(); //spawn player
        enemyManager.SpawnWave(); //spawn wave
        GameObject[] pickups =  GameObject.FindGameObjectsWithTag("HealthPickup");
//Find all health pickups
        foreach(GameObject go in pickups){ //destroy all health pickups
          Destroy(go);
         }
    }

    public void ResetGame(){ //Reset the game
        Debug.Log("Setting Up Game.");
        enemyManager.ResetWave();//Reset wave
        GameObject[] pickups =  GameObject.FindGameObjectsWithTag("HealthPickup"); //find
health pickups
```

```
    foreach(GameObject go in pickups){ //destroy
        Destroy(go);
      }
    SpawnPlayer(); //Spawn

    playerHealth = 100; //set health to 100
    playerScore= 0; //set score to 0
    playerLives = 3;  //set lives to 3

    ui.SetUI(1); //Open playing UI
}

public void MakeInvincible(){
    playerInvincible = true; //player set to invincible
    invincibleCR = Invincible(); //set invincible co routine
    StartCoroutine(invincibleCR); //start invincible coroutine
}

public bool IsPlayerInvincible(){
    return playerInvincible;
}

IEnumerator Invincible(){
    int time = 8; //set time to 0
    while(time > 0){//every second, decrement time by 1
        yield return new WaitForSeconds(1);
        time-=1;
    }
    playerInvincible = false; //set invincible to false
```

```
        StopCoroutine(invincibleCR);//stop coroutine
    }




}
```

Main Menu Controller

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
public class MainMenuController : MonoBehaviour
{

    public GameObject mainCanvas; //Main canvas, displays 2 text objects which guide player in
interactive menu
    public GameObject settingsCanvas; //Settings canvas, has sliders for volume and a button to
change to keybind menu
    public GameObject keybindsCanvas; //Keybinding canvas

    public Slider musicSlider; //slider for music volume
    public Slider sfxSlider; //slider for sfx volume
    public AudioSource menuMusic; //menu music audio source

    public CustomSettings settingsObj; //settings object which holds values and persists accross scene
    // Start is called before the first frame update
    void Start()
    {
        musicSlider.value = PlayerPrefs.GetFloat("musicVolume", .6f);  //load volume player pref if
exists, if not use default
```

```
        musicSlider.onValueChanged.AddListener(delegate {settingsObj.SetMusicVol(musicSlider.value);
}); //adds listener to slider for changed value, sets value in settings obj, saves value

        sfxSlider.value = PlayerPrefs.GetFloat("sfxVolume", .3f);//load volume player pref if exists, if not
use default

        sfxSlider.onValueChanged.AddListener(delegate {settingsObj.SetSFXVol(sfxSlider.value);
});//adds listener to slider for changed value, sets value in settings obj, saves value

    }


    void Update(){

        menuMusic.volume = musicSlider.value; //sets menu music volume

    }




    public void SwitchCanvas(int i){

        //i decides which to canvas switch to, using int;

        //0 settings

        //1 keybinds

        //2 out

        if(i==0){

            settingsCanvas.SetActive(true);

            keybindsCanvas.SetActive(false);

            mainCanvas.SetActive(false);

        }

        if(i==1){

            settingsCanvas.SetActive(false);

            keybindsCanvas.SetActive(true);

            mainCanvas.SetActive(false);

        }

        if(i==2){

           settingsCanvas.SetActive(false);

            keybindsCanvas.SetActive(false);

             mainCanvas.SetActive(true);
        }
    }

}
```

Menu Controller

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
public class MenuController : MonoBehaviour
{
    public MainMenuController menuCon;
    //player moves to a trigger in the scene to start the game or open settings
    void OnTriggerEnter2D(Collider2D other){ //when player enters trigger

        if(other.gameObject.tag == "playTrigger"){ //if tag is playTrigger
            SceneManager.LoadScene("SampleScene"); //load sample scene
        }
        if(other.gameObject.tag == "settingsTrigger"){ //if settings trigger
            menuCon.SwitchCanvas(0); //open settings menu
        }

    }
}
```

UI Manager

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;
```

```
public class UIManager : MonoBehaviour
{   //this manages ui in game scene
    public GameManager gm; //Game manager


    public Canvas[] canvasArray; //Array of canvases
    //Life images in Playing Canvas
    public Image lifeOne;
    public Image lifeTwo;
    public Image lifeThree;


    //Sprites for life on and life off, shows if player has one of the 3 lives
    public Sprite lifeOn;
    public Sprite lifeOff;


    public GameObject Crosshair; //the crosshair


    //Score and highscore text objects
    public Text scoreText;
    public Text highscoreText;


    //Game over screen text objects
    public Text gameOverScore;
    public Text gameOverHS;






    private int currentUI; //the current ui


    void Start()
    {
        canvasArray[0].gameObject.SetActive(true); //set initial ui to active


    }


    void Update()

    {

        if(currentUI == 1){ //if currentUI is == 1,  then call PlayingUI(), set cursor to invisible and set the
crosshair sprite to active

             PlayingUI();

               Cursor.visible = false;

                 Crosshair.SetActive(true);

        }else{

            Cursor.visible = true; // enable cursor disable crosshair

            Crosshair.SetActive(false);

        }




    }

    void PlayingUI(){ //Set up playing UI

        switch(gm.GetLives()){ //if lives == 0, set lives to lifeOff spire

            case 0:

                lifeOne.sprite = lifeOff;

                lifeTwo.sprite = lifeOff;

                lifeThree.sprite = lifeOff;

                break;

            case 1: //set lifeOne to lifeOn

                lifeOne.sprite = lifeOn;

                lifeTwo.sprite = lifeOff;

                lifeThree.sprite = lifeOff;

                break;

            case 2: //set lifeONe + lifeTwo to lifeOn
        lifeOne.sprite = lifeOn;
        lifeTwo.sprite = lifeOn;
        lifeThree.sprite = lifeOff;
        break;
      case 3: //set lifeONe + lifeTwo + lifeThree to lifeOn
        lifeOne.sprite = lifeOn;
        lifeTwo.sprite = lifeOn;
        lifeThree.sprite = lifeOn;
        break;
    }
    scoreText.text = gm.GetScore().ToString(); //sets score text to the score
}

public void SetUI(int number){ //sets the ui to number
    for(int i = 0; i<canvasArray.Length; i++){
      canvasArray[i].gameObject.SetActive(false); //disable all UI
    }
    canvasArray[number].gameObject.SetActive(true); //enable ui in canavasArray[number]
    currentUI = number; //current ui = number
    if(number == 3){ //if number is 3

      gameOverScore.text = gm.GetScore().ToString(); //sets gameover texts
      gameOverHS.text = gm.GetHighscore().ToString();
    }
}

public void PlayButton(){ //Play button method, restarts game loop.
    Debug.Log("Play Button Pressed");
    gm.Restart();
}
```

```
    public void ToMenu(){ //back to menu
        Destroy(GameObject.FindWithTag("Settings"));
        SceneManager.LoadScene("mainMenu");
    }



}
```

Camera Controller

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class CameraController : MonoBehaviour
{
    public GameObject player; //player object
    public float smoothSpeed = 0.1f;
    public Vector3 offset;
    public GameManager gm; //game manager

    public void SetPlayer(GameObject obj){ //function called in game manager
        player = obj; //sets the player
    }

    void FixedUpdate() {
         if(gm.IsPlayer()){ //if there is a player, follow player
            Vector3 desiredPosition = player.transform.position + offset;
            Vector3 newPosition = new Vector3(desiredPosition.x, desiredPosition.y,
transform.position.z);
            Vector3 smoothedPosition = Vector3.Lerp(transform.position, newPosition, smoothSpeed);
            transform.position = smoothedPosition;
        }
    }
}
```

Player Controller

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PlayerController : MonoBehaviour
{
    //Player controller
    public SoundEffects SoundEffects; //SoundEFfects object
    CustomSettings settings; //custom settings object for keybinds
    public bool inMenu = false; //if in the menu
    GameManager gm; //game manager
    //Player sprites
    public Sprite highHP;
    public Sprite midHP;
    public Sprite lowHP;
    public Sprite shield;
    //sprite renderer
    SpriteRenderer spriteRenderer;

    public int lives = 3; //3 lives

    public GameObject playerGO; //Player GO
    private Rigidbody2D rb; // our rigidbody
    public float playerSpeed = 30f; //players speed
    public float currentSpeed = 30f; //the current speed
    private Vector3 playerRotation;  //players rotation
    //keycodes for movement
    public KeyCode up;
    public KeyCode down;
    public KeyCode left;
    public KeyCode right;

    //booleans for moving
    bool movingUp;
    bool movingLeft;
    bool movingRight;
    bool movingDown;
    // Start is called before the first frame update
    void Start()
    {
        settings=GameObject.FindWithTag("Settings").GetComponent<CustomSettings>(); //find the
custom settings object, this holds our keycodes
        //gets keycodes and assigns ot up, left , down, right
        up=settings.GetMUP();
        left = settings.GetMLEFT();
        down = settings.GetMDOWN();
        right = settings.GetMRIGHT();
        if(!inMenu)//if not in menu scene
            SoundEffects = GameObject.FindWithTag("GameManager").GetComponent<SoundEffects>();
//finds soundeffecst object

        gm = FindObjectOfType<GameManager>(); //find game manager
        playerGO = this.gameObject; //player gameobject = this gameobject
        rb = GetComponent<Rigidbody2D>(); //sets the rigidbody
        spriteRenderer = gameObject.transform.GetChild(0).GetComponent<SpriteRenderer>(); //gets
the spriterenderer
    }

    // Update is called once per frame
    void Update()
```

```
   {


       Vector3 direction = Input.mousePosition -
Camera.main.WorldToScreenPoint(transform.position); //gets the direction of the mouse from the
main camera

       float angle = Mathf.Atan2(direction.y,direction.x) * Mathf.Rad2Deg;   //gets tan of diection.y,
direction.x, , then turns the radians returned to degrees

       transform.rotation = Quaternion.AngleAxis(angle, Vector3.forward); //sets the rotation to the
angle got above

       PlayerInput();


       if(inMenu){ //if in menu, get keycodes

          up = (KeyCode) PlayerPrefs.GetInt("MUPKey", (int)KeyCode.W);

          down = (KeyCode) PlayerPrefs.GetInt("MDOWNKey", (int)KeyCode.S);

          left = (KeyCode) PlayerPrefs.GetInt("MLEFTKey", (int)KeyCode.A);

          right = (KeyCode) PlayerPrefs.GetInt("MRIGHTKey", (int)KeyCode.D);

       }


      if(!inMenu){ ///if not in menu

       if(gm.IsPlayerInvincible()){ //check if player is invincible

          spriteRenderer.sprite = shield; //set sprite renderer to shield

          }else{


             if(gm.GetHealth() <= 33.33f){ //if below a third health, use low health sprite

                spriteRenderer.sprite = lowHP;

             }else if(gm.GetHealth() > 33.33f && gm.GetHealth() < 66.66f){ //if between a third health
and two third health, use mid health sprite

                spriteRenderer.sprite = midHP;

             }else if(gm.GetHealth() > 66.66f){ //if above 2 thirds health, use high health sprite

                spriteRenderer.sprite = highHP;

             }

          }

    }
```

```
}


private void FixedUpdate() {
    Move();
}

private void PlayerInput(){ //Checks input of player
    if(Input.GetKeyDown(up)){
        movingUp = true;
    }else if(Input.GetKeyDown(down)){
        movingDown = true;
    }else if(Input.GetKeyDown(left)){
        movingLeft = true;
    }else if(Input.GetKeyDown(right)){
        movingRight = true;
    }
    if(Input.GetKeyUp(up)){
        movingUp = false;
    }else if(Input.GetKeyUp(down)){
        movingDown = false;
    }else if(Input.GetKeyUp(left)){
        movingLeft = false;
    }else if(Input.GetKeyUp(right)){
        movingRight = false;
    }
}

private void Move(){
    if(movingUp){
        if(movingRight || movingLeft){
            currentSpeed = playerSpeed*.7f;
            //sets speed to a lower value if moving up and to the side,
            // as when applying force in both directions the player moves a bit faster
        }
        else{
            currentSpeed = playerSpeed; //else speed sets it player player speed
        }
        rb.AddForce(new Vector4(0, currentSpeed, 0)); //applies force
    }

    if(movingDown){
     //sets speed to a lower value if moving down and to the side,
    // as when applying force in both directions the player moves a bit faster
         if(movingRight || movingLeft){
            currentSpeed = playerSpeed*.7f;
        }
        else{
            currentSpeed = playerSpeed;
        }
        rb.AddForce(new Vector3(0, -currentSpeed, 0)); //adds force
    }

    if(movingLeft)
    rb.AddForce(new Vector3(-currentSpeed, 0, 0)); //add force left

    if(movingRight)
    rb.AddForce(new Vector3( currentSpeed, 0,0)); //addforce right
}
```

```
   void OnCollisionEnter2D(Collision2D col){ //when player enters a collision


      if(col.gameObject.tag == "Mine"){  //if a mine

         rb.AddForce(-rb.velocity * 15, ForceMode2D.Impulse); //add force oppositie to rigidbodys
velocity, in impulse mode

         gm.DamageHealth(20); //damage player health by 25

      }

      if(col.gameObject.tag == "Bomb"){ //if bomb

         gm.DamageHealth(50); //damage health

         rb.AddForce(col.gameObject.GetComponent<Rigidbody2D>().velocity,
ForceMode2D.Impulse); //addforce in bombs direction, in impulse mod

         SoundEffects.PlayBombSound(); //play bomb sound


      }


   }

   void OnTriggerEnter2D(Collider2D col){ //if the player enters a trigger

      if(col.gameObject.tag == "HealthPickup"){ //if health pickup, add health destroy pickup

         gm.AddHealth(25);

         SoundEffects.PlayPickupHealthSound(); //play pickup sound

         Destroy(col.gameObject);

      }

      if(col.gameObject.tag == "ShieldPickup"){

         gm.AddHealth(100); //add 100 health

         gm.MakeInvincible(); //make player invincible

         Destroy(col.gameObject); //destroy pickup

         SoundEffects.PlayPickupShieldSound(); //play shield pickup sound

      }

      if(col.gameObject.tag == "TrishotPickup"){

         GetComponent<WeaponManager>().EnableTrishot();

         SoundEffects.PlayPickupTrishotSound(); //play trishot pickup sound

         Destroy(col.gameObject);
```

```
        }
    }




}
```

Weapon Manager

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class WeaponManager : MonoBehaviour
{

    //projectile speed
    public float shotSpeed;
    //trishot on or off
    public bool trishotEnabled = false;

    //spawn points for projectiles
    public GameObject spawnPointL;
    public GameObject spawnPointR;
    public GameObject spawnObject;
    //projectile prefab
    public GameObject[] projectilePrefab;
    //index for array, each prefab is a different color
    private int currentColor;
    //sound effects and settings objects
    public SoundEffects SoundEffects;
    public CustomSettings settings;
    //keycode for shooting
    public KeyCode shootButton;


    IEnumerator trishotCR; //coroutine of trishot
    // Start is called before the first frame update
    void Start(){
         SoundEffects = GameObject.FindWithTag("GameManager").GetComponent<SoundEffects>();
//gets game manager
         settings=GameObject.FindWithTag("Settings").GetComponent<CustomSettings>(); //gets
settings
         shootButton = settings.GetShoot(); //gets shoot button
    }


    // Update is called once per frame
    void Update()
    {
        Vector3 screenPoint = Camera.main.WorldToScreenPoint(transform.position); //point on screen
of player
        Vector3 direction = (Vector3)(Input.mousePosition-screenPoint); //direction from player to
mouse point
        direction.Normalize(); //normalize
        if(Input.GetKeyDown(shootButton)){ //if shootbutton is pressed
            if(!trishotEnabled){ //if trishot not enabled spawn projectile, increment color
                GameObject projectile = Instantiate(projectilePrefab[currentColor],
spawnObject.transform.position, transform.rotation);
                projectile.GetComponent<Rigidbody2D>().AddForce(direction * shotSpeed,
ForceMode2D.Impulse);


                currentColor += 1;
                if(currentColor == projectilePrefab.Length)
                //if color is out of bounds for array, reset to 0
                    currentColor = 0;
            }else{ //if trishot is not enabled, do same as above, but instantiate 3 projectiles
```

```
                GameObject projectile1 = Instantiate(projectilePrefab[currentColor], 
spawnObject.transform.position, spawnPointL.transform.rotation);

                GameObject projectile2 = Instantiate(projectilePrefab[currentColor], 
spawnPointL.transform.position, transform.rotation);

                GameObject projectile3 = Instantiate(projectilePrefab[currentColor], 
spawnPointR.transform.position, spawnPointR.transform.rotation);


                projectile1.GetComponent<Rigidbody2D>().AddForce(direction * shotSpeed, 
ForceMode2D.Impulse);

                projectile2.GetComponent<Rigidbody2D>().AddForce(direction * shotSpeed, 
ForceMode2D.Impulse);

                projectile3.GetComponent<Rigidbody2D>().AddForce(direction * shotSpeed, 
ForceMode2D.Impulse);


                currentColor += 1;

                if(currentColor == projectilePrefab.Length)

                    currentColor = 0;

            }

            SoundEffects.PlayGunSound(); //play gun sound

        }


    }


    public void EnableTrishot(){ //enable trishot

        trishotCR = Trishot();

        StartCoroutine(trishotCR); //start the trisot coroutine

    }



    IEnumerator Trishot(){

        trishotEnabled= true;

        //enable trishot

        int time = 10;
        //sets time to 10
        while(time > 0){
           //every second until time > 0
           //decrement time
           yield return new WaitForSeconds(1);
           time-=1;
        }
        trishotEnabled = false;
        //trishot enabled set to false
        StopCoroutine(trishotCR);
        //stop coroutine
    }
}
```

Bomb

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Bomb : MonoBehaviour
{
    void OnCollisionEnter2D(Collision2D col){ //when colliding, destroy this gameobject
        Destroy(this.gameObject);
    }
}
```

Enemy

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Enemy : MonoBehaviour
{
    public float health; //health


    public GameObject healthPickup; //health pickup


    public float GetHealth(){ //get the players health

        return health; //return the health

    }


    public void DamageHealth(float damage){


        health = health - damage; //lowers the players health

        if(health <= 0){ //if health is less than 0

            FindObjectOfType<GameManager>().AddScore(100); //add score

            float r = Random.Range(0,100); //random number between 0 and 100

            if(r >90){ //if r greater than 90

                Instantiate(healthPickup, transform.position, Quaternion.identity);

                //instantiates health pickup

            }

GameObject.FindWithTag("GameManager").GetComponent<SoundEffects>().PlayEnemyDeathSoun
d(); //play enemy death sound

            //play the enemy death sound

            Destroy(this.gameObject); //destroy this gameobject

        }

    }

}
```

Kamikaze Enemy

```
using System.Collections;

using System.Collections.Generic;

using UnityEngine;
```

```
public class KamikazeEnemy : MonoBehaviour
{

   GameManager gm; //gamemanager
   public Transform player; //players transform

   public float tailingSpeed; //speed when following player
   public float chargingSpeed; //speed when charging player
   private float moveSpeed; //movement speed
   private SpriteRenderer spriteRenderer; //sprite render
   public Sprite tailingSprite; //sprite when tailing
   public Sprite chargingSprite; //sprite when charging
   public float chargeRange; //charge range
   // Start is called before the first frame update
   void Start()
   {
      spriteRenderer = gameObject.transform.GetChild(0).GetComponent<SpriteRenderer>(); //gets
sprite rendered
      gm = FindObjectOfType<GameManager>(); //finds game manager
   }
   // Update is called once per frame
   void Update()
   {

       FollowPlayer();

   }

   void FollowPlayer(){
    if(gm.IsPlayer()){
      player = gm.GetPlayer().transform; //gets players transform
      Vector2 direction = player.position - transform.position; // gets direction of player

      float angle = Mathf.Atan2(direction.y, direction.x) * Mathf.Rad2Deg; //tan y/x to get angle in
radians, then convert to degrees
      Quaternion target = Quaternion.Euler(0, 0 , angle);
      transform.rotation = target; //sets rotation to above

      //if distance is whithin charge range, use charging sprite + speed
      if(Vector2.Distance(player.position, transform.position) < chargeRange) {
        moveSpeed = chargingSpeed;
        spriteRenderer.sprite = chargingSprite;
      } else{
        //else use tailing speed
        moveSpeed = tailingSpeed;
        spriteRenderer.sprite = tailingSprite;
      }

      if(Vector2.Distance(player.position, transform.position) <= 1f){
        //if distance is below  .5
        gm.DamageHealth(20); //damage player
        Destroy(this.gameObject); //destroy this

      }
      //move player
      transform.Translate(Vector3.right * moveSpeed * Time.deltaTime);
      }

  }
```

```
}
```

Laser

```
using System.Collections;

using System.Collections.Generic;

using UnityEngine;


public class Laser : MonoBehaviour

{


   public float damage = 20; //damage



   void OnCollisionEnter2D(Collision2D collider){ //on collision enter



      //damages health of whatever enemy it hits

      if(collider.gameObject.tag == "Enemy"){

         collider.gameObject.GetComponent<Enemy>().DamageHealth(20);

      }

      if(collider.gameObject.tag == "Mine"){

         collider.gameObject.GetComponent<Enemy>().DamageHealth(20);

      }

      if(collider.gameObject.tag == "Turret"){

         collider.gameObject.GetComponent<Enemy>().DamageHealth(20);

      }

      Destroy(this.gameObject); //destroy this object

   }

}
```

Mine

```
using System.Collections;

using System.Collections.Generic;

using UnityEngine;


public class Mine : MonoBehaviour

{


    public GameObject trishotPickup;

    public GameObject shieldPickup;

    // Start is called before the first frame update

    void Start()

    {

        float r = Random.Range(0,100); //random number between 0 and 100

        if(r >= 70){ //if above 70

            //generate new random number

            r = Random.Range(0,100);

            if(r>30)//if above 30 spawn trishot

                Instantiate(trishotPickup, new Vector3(transform.position.x - .5f, transform.position.y,
transform.position.z), Quaternion.identity);

            if(r<=30) //if below 30 spawn shield

                Instantiate(shieldPickup, new Vector3(transform.position.x - .5f, transform.position.y,
transform.position.z), Quaternion.identity);

        }

    }


}
```

Rotate Image

```
using System.Collections;

using System.Collections.Generic;

using UnityEngine;


public class RotateImage : MonoBehaviour

{

  //rotates an image in the initial menu when in the game scene

  public float speed;

  void Update()

  {

    transform.Rotate(0,0, speed*Time.deltaTime); //rotate by speed

  }

}
```

Turret

```
using System.Collections;

using System.Collections.Generic;

using UnityEngine;


public class Turret : MonoBehaviour

{

  //the 4 firepoints

  public Transform firePoint1, firePoint2, firePoint3, firePoint4;

  // prefab

  public GameObject bulletPrefab;

  //force applied to projectile

  public float bulletForce = 80f;

  //rotation speed of the turret

  private float rot = 50;


  void Start(){
    StartCoroutine(Shoot()); //start the coroutine
}

void Update(){
   transform.Rotate(0,0, rot* Time.deltaTime); //rotate by rot
}
IEnumerator Shoot(){

   while(true){
   //while true
   float waiting = Random.Range(3,12);
   //random time to wait
   //instantiate a projectile at the spawn point, addforce
   GameObject bullet1 = Instantiate(bulletPrefab, firePoint1.position, firePoint1.rotation);
   Rigidbody2D rb1 = bullet1.GetComponent<Rigidbody2D>();
   rb1.AddForce(firePoint1.up * bulletForce, ForceMode2D.Impulse);

   GameObject bullet2 = Instantiate(bulletPrefab, firePoint2.position, firePoint2.rotation);
   Rigidbody2D rb2 = bullet2.GetComponent<Rigidbody2D>();
   rb2.AddForce(firePoint2.up * bulletForce, ForceMode2D.Impulse);

   GameObject bullet3 = Instantiate(bulletPrefab, firePoint3.position, firePoint3.rotation);
   Rigidbody2D rb3 = bullet3.GetComponent<Rigidbody2D>();
   rb3.AddForce(firePoint3.up * bulletForce, ForceMode2D.Impulse);

   GameObject bullet4 = Instantiate(bulletPrefab, firePoint4.position, firePoint4.rotation);
   Rigidbody2D rb4 = bullet4.GetComponent<Rigidbody2D>();
   rb4.AddForce(firePoint4.up * bulletForce, ForceMode2D.Impulse);

   yield return new WaitForSeconds(4); //wait 4 seconds
```

```
        }
    }



}
```

Crosshair

```
using System.Collections;

using System.Collections.Generic;

using UnityEngine;


public class Crosshair : MonoBehaviour

{

    // Start is called before the first frame update

    void Start()

    {

         Cursor.visible = false;

         //disable cursor

    }


    // Update is called once per frame

    void Update()

    {


        transform.position = new
Vector3(Camera.main.ScreenToWorldPoint(Input.mousePosition).x,Camera.main.ScreenToWorldPoi
nt(Input.mousePosition).y, -8);

        //set position to mouse position

    }

}
```

Custom Settings

```
using System.Collections;

using System.Collections.Generic;
using UnityEngine;

public class CustomSettings : MonoBehaviour
{

    //this script holds the settings
    //which are laid out in the
    //main menu

    public KeyCode up;
    public KeyCode down;
    public KeyCode left;
    public KeyCode right;
    public KeyCode shoot;

    public float sfxVolume;
    public float musicVolume;



    public void Start(){
        //load the player prefs for all settings
        up = (KeyCode) PlayerPrefs.GetInt("MUPKey", (int)KeyCode.W);
        down = (KeyCode) PlayerPrefs.GetInt("MDOWNKey", (int)KeyCode.S);
        left = (KeyCode) PlayerPrefs.GetInt("MLEFTKey", (int)KeyCode.A);
        right = (KeyCode) PlayerPrefs.GetInt("MRIGHTKey", (int)KeyCode.D);
        shoot = (KeyCode) PlayerPrefs.GetInt("SHOOTKey", (int)KeyCode.Space);

        musicVolume = PlayerPrefs.GetFloat("musicVolume", .6f);
        sfxVolume = PlayerPrefs.GetFloat("sfxVolume", .3f);
        DontDestroyOnLoad(this.gameObject);
    }

    //Getter methods
    public KeyCode GetMUP(){
        return up;
    }
     public KeyCode GetMDOWN(){
        return down;
    }
     public KeyCode GetMLEFT(){
        return left;
    }
     public KeyCode GetMRIGHT(){
        return right;
    }
     public KeyCode GetShoot(){
        return shoot;
    }

    public float GetSFXVolume(){
        return sfxVolume;
    }
    public float GetMusicVolume(){
        return musicVolume;
    }
    //setter methods
    public void SetUp(KeyCode k){
        up = k;
        PlayerPrefs.SetInt("MUPKey", (int)k);
        PlayerPrefs.Save();
    }
```

```
public void SetDown(KeyCode k){
   down = k;
   PlayerPrefs.SetInt("MDOWNKey", (int)k);
   PlayerPrefs.Save();
}
public void SetLeft(KeyCode k){
   left = k;
   PlayerPrefs.SetInt("MLEFTKey", (int)k);
   PlayerPrefs.Save();
}
public void SetRight(KeyCode k){
   right = k;
   PlayerPrefs.SetInt("MRIGHTKey", (int)k);
   PlayerPrefs.Save();
}
public void SetShoot(KeyCode k){
   shoot = k;
   PlayerPrefs.SetInt("SHOOTKey", (int)k);
   PlayerPrefs.Save();
}
public void SetMusicVol(float f){
   musicVolume = f;
   PlayerPrefs.SetFloat("musicVolume", f);
   PlayerPrefs.Save();
   Debug.Log("Music changed to " + f + " and saved.");
}
public void SetSFXVol(float f){
   sfxVolume = f;
   PlayerPrefs.SetFloat("sfxVolume", f);
   PlayerPrefs.Save();
   Debug.Log("SFX changed to " + f + " and saved.");
```

```
    }


}
```

Sound Effects

```
using System.Collections;

using System.Collections.Generic;

using UnityEngine;


public class SoundEffects : MonoBehaviour

{

    //Audio sources

   public AudioSource bombSource;

   public AudioSource gunSource;

   public AudioSource enemydeathSource;

   public AudioSource musicSource;

   public AudioSource pickupShield;

   public AudioSource pickupHealth;

   public AudioSource pickupTrishot;

   public AudioSource damagePlayer;

   //bool to check if in main menu

   public bool inMenu = false;

   //custom settings object

   public CustomSettings settings;

   void Start(){

    if(!inMenu){

        //gets the volume for each audio source;

        settings = GameObject.FindWithTag("Settings").GetComponent<CustomSettings>();

         bombSource.volume = settings.GetSFXVolume();

         gunSource.volume = settings.GetSFXVolume();

         musicSource.volume = settings.GetMusicVolume();

         pickupShield.volume = settings.GetSFXVolume();
```

```
      pickupHealth.volume= settings.GetSFXVolume();
      pickupTrishot.volume = settings.GetSFXVolume();
      damagePlayer.volume = settings.GetSFXVolume();
      enemydeathSource.volume = settings.GetSFXVolume();
 }
}

//Methods to play each of the respective
public void PlayBombSound(){
     bombSource.Play();
}


 public void PlayGunSound(){
     gunSource.Play();
}
 public void PlayPickupShieldSound(){
     pickupShield.Play();
}
 public void PlayPickupTrishotSound(){
     pickupTrishot.Play();
}
 public void PlayPickupHealthSound(){
     pickupHealth.Play();
}
 public void PlayDamagePlayerSound(){
     damagePlayer.Play();
}
public void PlayEnemyDeathSound(){
     enemydeathSource.Play();
}
```

```
}
```