Basics of Game Engines

Task 3A:

Graphical Assets

Player

Player

The player was the first asset created on illustrator, which was inspired by a PS2 Game, Ratchet and Clank, Clank being the inspiration. When added in the scene, it was changed to a sprite, purposely to be applicable for a 2D Game. It was set as a prefab and positioned at -7 on the Y-axis, the other 2 axes remained at 0. A RigidBody2D, being changed to Dynamic, and BoxCollider2D were set to the player in order for movement to take place and for physics to take place

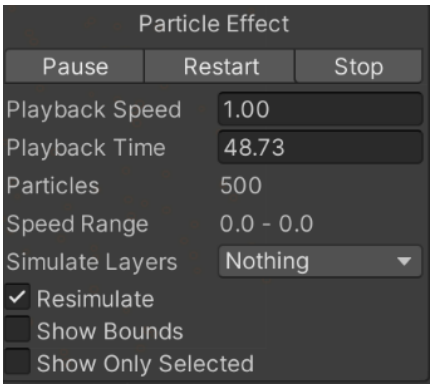
Clank (Reference for Player)

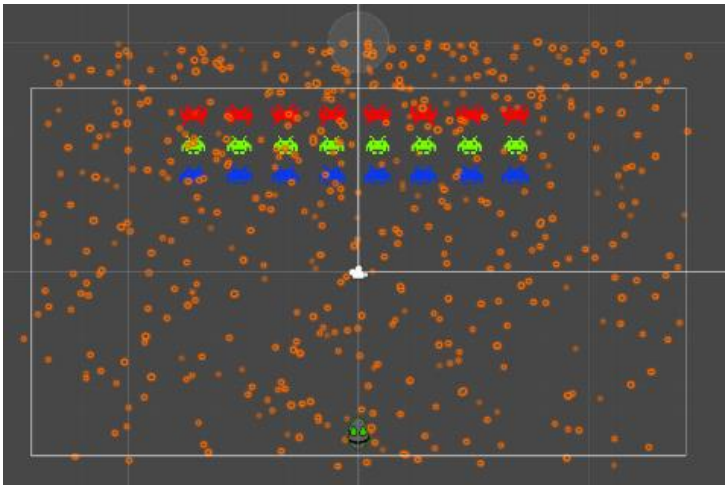
Enemies

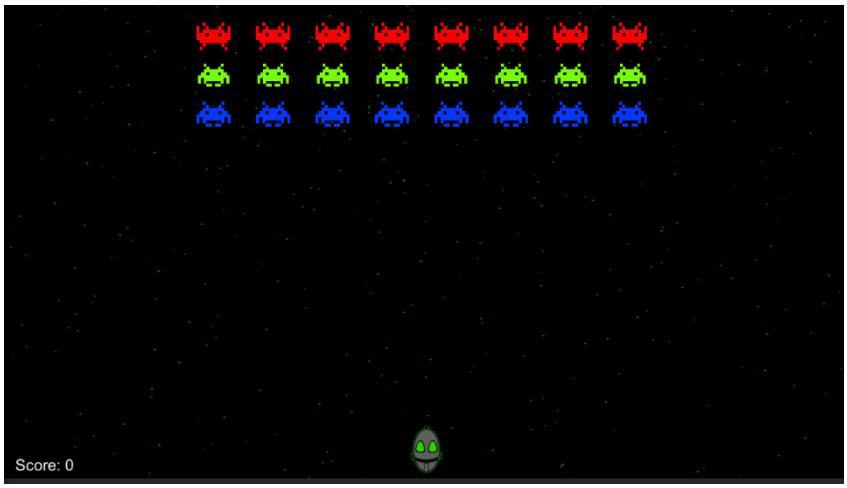
The enemies remained as the orginal enemies of Space Invaders, as it fit the look of the game better. As of the importation of these, 1 enemy was imported, then duplicated to make a single row of enemies, followed by the same process to make the other enemies, only changing the colour and positioning. Once satisfied with the placement, all enemies were put into an Enemy Holder and set as a prefab. These were aswell set as sprites for 2D and UI. The Enemy Holder itself was set with a RigidBody2D and the Y Axis was ticked in order for that axis to be frozen, giving no access for the enemies to move in that diretion. The enemies were also given a RigidBody2D, aswell as a BoxCollider2D, each one being set as a trigger for when being shot by the player.

Background

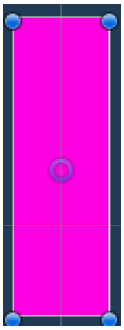
For the creation of the Background, a particle system was used to make the background move and look like you're actually in space, using the StarEffects. The positioning was set at 10 on the Y axis, the other two axis remaining at 0, followed by scaling dow the size as needed. The controls as shown on the left were fixed up, from a colour being set to green, the gravity being set to slow, duration of the effect and so on, in order to get the look of sparkly stars and movement. This was also set as a prefab.

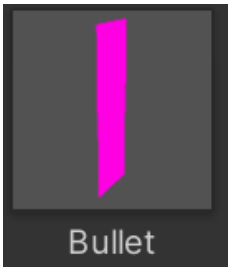




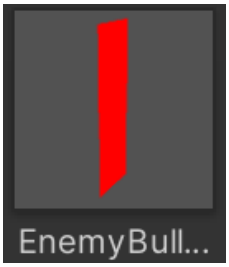


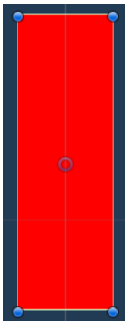
Bullets





Both of the bullets were basically made the same way, the diffence being each were associated with different characters, one being the player holding the pink bullet and the enemies holding the red bullet. They both had a RigidBody2D and BoxCollider2D added to them, along with being ticked as a trigger for the bullets to detsroy something. These were added as prefabs.





Scripts

#PlayerController

Three things were set for the players functionality, a private tranform command is set for the player in order to add the script in the player itself to function properly, a public float is set for the speed of the player as it moves across the screen, and another public float will be made for the maximum and minimum so the player wont move off screen.

```
5   public class PlayerController : MonoBehaviour
6   {
        4 references
7       private Transform player;
        1 reference
8       public float speed;
        1 reference | 1 reference
9       public float maxBound, minBound;
```

To begin this code, a function to choose the player, using the GetComponent command, was used to transform this specified player. In order to make it move only till secific bounds and to only move horizonatlly with the keys, the ''Horizontal'' axis was chosen and two commands were issued for this. An if statement was used for when the player position is less than the minimum bound and h is less than 0, it wouldn't be able to move any further to the left of the screen since h would be negative. The same thing was done for the right side of the screen, so an else if command was used for when the player position and h is greater than the maximum bound it would stop moving to the right. A Vector3 command was used to make the player actually move. Once ready, this code is dragged into the player on unity, the public component's elements are set accordingly.

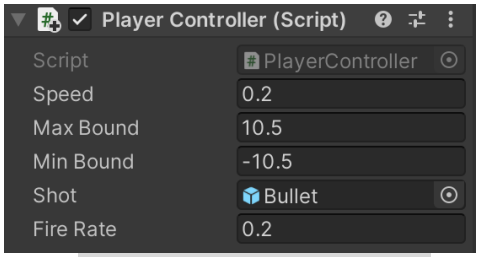
```
//Bullet creation
1 reference
public GameObject shot;
0 references
public float fireRate;

0 references
private float nextFire;
```

For the creation of the bullet, 2 public floats were created for the actual shot and the fire rate of the bullet being shot. A private float was to take control of the next shot to be fired (nextFire)

```
void Update(){
// if(Input.GetKeyDown(KeyCode.Space))
    if (Input.GetKeyDown(KeyCode.Space))
    {
        Instantiate (shot, transform.position,Quaternion.identity);
        Debug.Log("Space bar was pressed");
    }
```

Another void update was added for the player shot to be displayed when pressing the spacebar using a KeyCode for the if statement

```
void Start(){
    player = GetComponent<Transform> ();
}

0 references
void FixedUpdate()
// maximum and minimum bound of player when moving
{
    float h = Input.GetAxis("Horizontal");{
        if (player.position.x < minBound && h < 0)
    h = 0;
    else if (player.position.x > maxBound && h > 0)
    h = 0;
    }
        player.position += Vector3.right * h * speed;
}
```



Set numbers for rates of player movement and fired shots

#BulletController

A private transform was set for the bullet, followed by a public float called speed to control the speed of how the bullet moves.

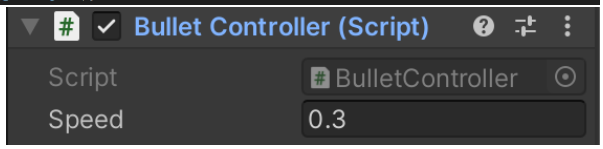
```
3 references
private Transform bullet;
1 reference
public float speed;
```

```
void Start()
{
    bullet = GetComponent<Transform>();
}

// Update is called once per frame
0 references
void FixedUpdate()
{
    bullet.position += Vector3.up * speed;

    if (bullet.position.y >=10)
    Destroy (gameObject);
}
```

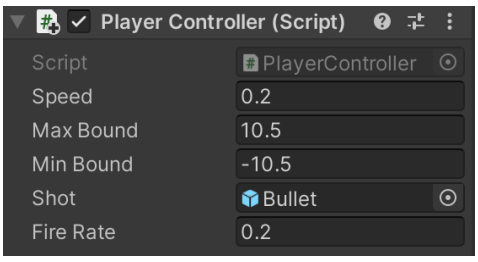
A void fixed update is used to make sure the bullet moves on a set period of time instead of frame rate. Vector3.up is used to make the bullet move upwards, multiplied by the speed to change how fast it moves. An if statement is set for when the bullet position is greater or equal to 10 it will be destroyed. This is done so there wouldn't be any issues with the game when a bullet is shot and it wont still be on screen.

```
void OnTriggerEnter2D(Collider2D other)
{
    if (other.tag == "RedEnemy")
    {
        PlayerScore playerScore = GameObject.Find("Score").GetComponent<PlayerScore>();
        if(playerScore != null)
        {
            playerScore.IncremenScoreRed();
        }
        Destroy(other.gameObject);
        Destroy (gameObject);

    }
    if (other.tag =="GreenEnemy")
    {
        PlayerScore playerScore = GameObject.Find("Score").GetComponent<PlayerScore>();
        if(playerScore != null)
        {
            playerScore.IncremenScoreGreen();
        }
        Destroy(other.gameObject);
        Destroy (gameObject);
    }
    if (other.tag == "BlueEnemy")
    {
        PlayerScore playerScore = GameObject.Find("Score").GetComponent<PlayerScore>();
        if(playerScore != null)
        {
            playerScore.IncremenScoreBlue();
        }
        Destroy(other.gameObject);
        Destroy (gameObject);
```

Next a new void OnTriggerEnter2D and Collider2D other followed along is set for when the bullet hits an enemy. If (other.tag == ''RedEnemy'') refers to the red enemies which is tagged under RedEnemy. A piece of code is set for when the player hits an enemy, the score goes up according to the enemy being hit. Using IncremenScore associates with which enemy the score will rise with. Destroy(other.gameObject) destroys the actual enemy, followed by the destruction of the actual bullet once hitting an enemy, using Destroy(gameObject). This process is done for the other two enemies, which are under their specified tags.



This script is then dragged onto the Bullet and speed is set to 0.3. The bullet is removed from the scene and the prefab is dragged into the shot of the player.



#PlayerScore

After creating the score text by right clicking on the Hierarchy and selecting UI>Text, the script was then created. using UnityEngine.UI was used to access the user interface from this script for the score. A public static float was added in order to update the score from this script to start from 0.

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

6 references
public class PlayerScore : MonoBehaviour
{
    7 references
    private float playerScore = 0;
    2 references
    private Text scoreText;
```

```
void Start()
{
    scoreText = GetComponent<Text> ();
}

// Update is called once per frame
0 references
void Update()
{
    scoreText.text = "Score:" + playerScore.ToString();
}
1 reference
public void IncremenScoreRed()
{
    playerScore += 6;
    Debug.Log("Increased Score by 6" + playerScore);
}
1 reference
public void IncremenScoreBlue()
{
    playerScore += 4;
    Debug.Log("Increase Score by 4" + playerScore);
}
1 reference
public void IncremenScoreGreen()
{
    playerScore += 2;
    Debug.Log("Increase Score by 2" + playerScore);
}
}
```

In the update, the score text would be set accordingly to change when destroying specific enemies. playerScore += is relevant to add up the current player score with the added points of the enemy being killed.

#EnemyController

using UnityEngine.Ui was once again relevant for this code. The enemy holder is referred to the empty object created which holds every enemy on screen. A Win text, Restart text and Game Over text were also added to the folder holding the player score, which are relevant with this code for the Enemy Controller. A fire rate was set to 0.997f for the enemy bullet.

On void Start, winText.enabled = false was set so when the game starts the text wont show up, only when the player wins. InvokeRepeating was added for the enemy movement to start moving at 0.1 seconds and repeat the movement every 0.3 seconds, frames in this case (f).

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
0 references
public class EnemyController : MonoBehaviour{
    6 references
    private Transform enemyHolder;
    3 references
    public float speed;

    1 reference
    public GameObject shot;
    2 references
    public Text winText;
    1 reference
    public Text restartText;
    1 reference
    public float fireRate = 0.997f;
```

```
void Start(){
    winText.enabled = false;
    InvokeRepeating ("MoveEnemy", 0.1f, 0.3f);
    enemyHolder = GetComponent<Transform> ();
}
```

```
void MoveEnemy()
{
    enemyHolder.position += Vector3.right * speed;

    foreach (Transform enemy in enemyHolder) {
        if (enemy.position.x < -10.5 || enemy.position.x > 10.5){
        speed = -speed;
        enemyHolder.position += Vector3.down * 0.5f;
        return;
    }

    if(Random.value > fireRate)
    {
        Instantiate (shot, enemy.position, enemy.rotation);
    }
    if (enemy.position.y <= -4)
    {
        GameOver.isPlayerDead = true;
        Time.timeScale = 0;
    }
}
    if (enemyHolder.childCount == 1){
    CancelInvoke ();
    InvokeRepeating ("MoveEnemy", 0.1f, 0.25f);
}
if (enemyHolder.childCount == 0){
    winText.enabled = true;
    restartText.enabled = true;
}
}
```

To make the enemies move in a specific direction, another piece of code was added. The enemies position begins moving to the right as += Vector3.right is used with the speed being multiplied with it. Foreach is used to pick every enemy in the enemyholder. If the enemy's position is less than -10.5 or greater than 10.5, the direction will be reverted accordingly to remain on screen. enemyHolder.position += Vector3.down means the enemies will move downwards at a rate of 0.5f, when hitting the maximum and minimum bounds that have been set. To make the enemies shoot random bullets, an if statement was created to set random values, Instantiate being used to spawn these bullets. If (enemy.position.y <= -4) is referred to when the enemy position is less than or equal to -4, the player dies (GameOver).

The game over text will be displayed through the following code, GameOver.isPlayerDead = True. The time is then reset to 0.

The next piece of code holding the child count of the enemies indicates that when 1 enemy is left from the enemy holder, that last remaining enemy will move at a faster speed of 0.25f. Once the child count of the enemy holder hits 0, the player wins, then displaying the win text and restart text, as they would be equal to true as shown in the last bit of code.

#EnemyBulletController

For the enemy bullet, a public float was used for the speed of the bullet. In the fixed update, bullet position was set as Vector3.down multiplied by the speed to move downwards towards the player, opposite of the player bullet basically. This time, to destroy the bullet in order to help the performance of the game, an if command was set to the position of the enemy bullet as being less than or equal to -10.

For the player to be hit with the enemy bullet, a VoidOnTrigger2D(Collider2d other) was added to the script. If(other.tag..) refers to if the enemy bullet is hitting the player. The same process with the player bullet, the last bit of code is used to destroy the player and the enemy bullet itself.

GameOver.isPlayerDead = true means that GameOver would be displayed since the enemy bullet would hit the player, as it is marked as true.

```
public class EnemyBulletController : MonoBehaviour
{

    4 references
    private Transform bullet;
    1 reference
    public float speed;
```

```
    void Start()
    {
        bullet = GetComponent<Transform> ();
    }

    // Update is called once per frame
    0 references
    void FixedUpdate()
    {
        bullet.position += Vector3.down * speed;
            if (bullet.position.y <= -10)
            Destroy (bullet.gameObject);
    }

    0 references
    void OnTriggerEnter2D(Collider2D other)
    {
        if (other.tag == "Player") {
            Destroy (other.gameObject);
            Destroy (gameObject);
            GameOver.isPlayerDead = true;
        }
```

#GameOver

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

4 references
public class GameOver : MonoBehaviour
{
    5 references
    public static bool isPlayerDead = false;
    3 references
    private Text gameOver;

    // Start is called before the first frame update
    0 references
    void Start()
    {
        gameOver = GetComponent<Text> ();
        gameOver.enabled = false;
    }

    // Update is called once per frame
    0 references
    void Update()
    {
        if (isPlayerDead) {
            Time.timeScale = 0;
            gameOver.enabled = true;
        }
    }
}
```

Adding using Unity.UI, to use the UI to display something on the screen, a public static bool was used to figure out when the player dies, in this code isPlayerDead = false, since the default is that the player is alive. A private text was set for GameOver, this is why using UnityEngine.UI was added.

gameOver.enabled = false was set, meaning the text wont be enabled at the start, but when the player dies.

In the void Update, the if (isPlayerDead) command is set for when this states it as true, the game will automatically pause and the gameover text is displayed, including the restart text.

#RestartLevel

using UnityEngine.SceneManagement was added to the script, in order to reload the scene when pressing the 'R' key.

A void Update was included for inputting an if statement for when pressing the 'R' key, if (Input.GetDownKey(KEyCode.R). This will then mean the GameOver.isPlayerDead = false, automatically restarting the game along with resetting the time scale to 1.

Adding the SceneMAanger.LoadScene(''SpaceInvaders'') makes the loaded scene appear when hitting 'R'.

The same code was going to be set for the MainMenu option when pressing 'M', but unfortunately did not work, so was left out of the game.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
0 references
public class RestartLevel : MonoBehaviour
{
    // Start is called before the first frame update
    0 references
    void Update(){
        if(Input.GetKeyDown(KeyCode.R)){
        //PlayerScore.playerScore = 0;
        GameOver.isPlayerDead = false;
        Time.timeScale = 1;

        SceneManager.LoadScene("SpaceInvaders");

        if(Input.GetKeyDown(KeyCode.M)){
        GameOver.isPlayerDead = false;

        SceneManager.LoadScene("MainMenu");
    }
    }
    }
}
```

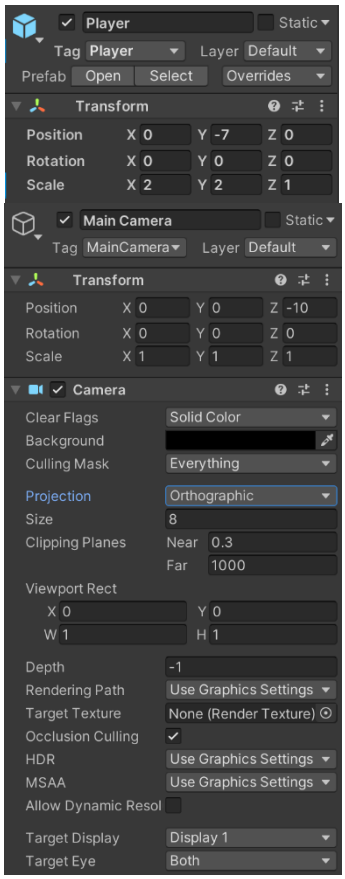
#MainMenu

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

0 references
public class MainMenu : MonoBehaviour
{
    0 references
    public void PlayGame ()
    {
        SceneManager.LoadScene(1);
    }
}
```

This was the simplest piece of code. using UnityEngine.SceneManagement was added in the script to load the actual scene.

Public void was added to the PlayGame button

Once pressng the button it will load the first scene, SceneManager.LoadScene(1).



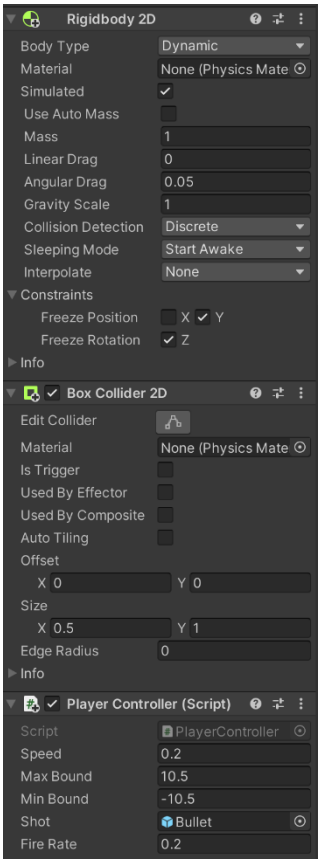
Process

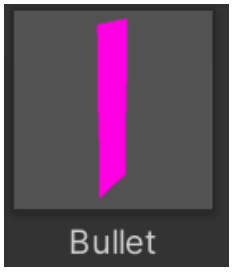
After sketching out the characters and knowing what the game would look like, a new project was opened on Unity. Once setting the screen size and so on, the main camera was fixed up to being 2D, so the projection was changed to orthographic, followed by the colour to be black for the stars to be added in later.

Once this was done, the artwork of the character was added in, changed to a sprite and set as a prefab. The tag was named under Player for the script work, the positioning was set at -7 on the Y axis, leaving X and Z at 0 in order to keep the movement going horizontally. A RigidBody2D, being changed to Dynamic, and BoxCollider2D were set to the player in order for movement to take place and for physics to take place.

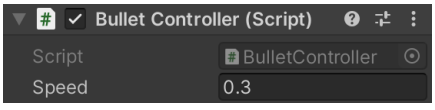
The player script was then added to the player, including the bullets the player shoots.

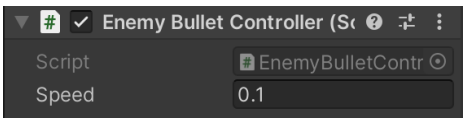
The players bullet was then created using a shape and changing its colour to purple. A RidigBody 2D and BoxCollider 2Dwere added to it, setting the RigidBody to Kinematic so the collisions wouldn't affect the Rigid Body, and the Box Collider was set to a trigger. The bullet Controller script is then added to the bullet prefab, setting the speed to 0.3 when being shot. This prefab was tagged under Bullet.

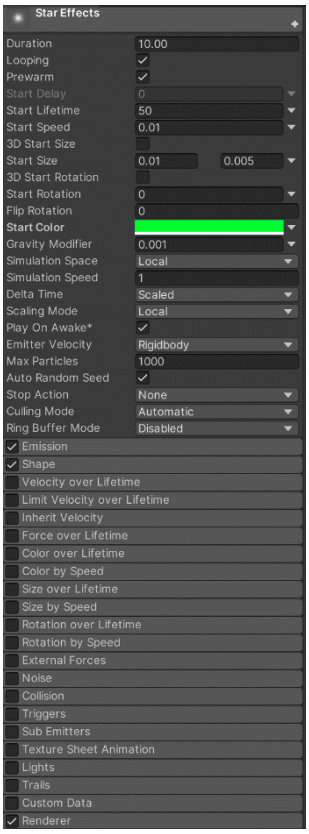


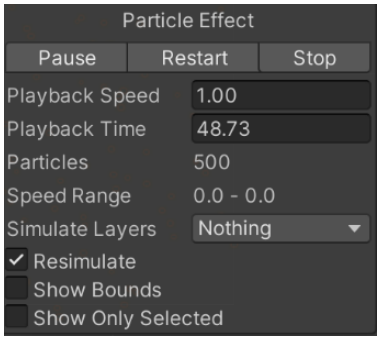


The enemy bullet was made the same way the player bullet was made, being a trigger, just the colour of this is red and a different script was added to this, with the bullet speed at 0.1.



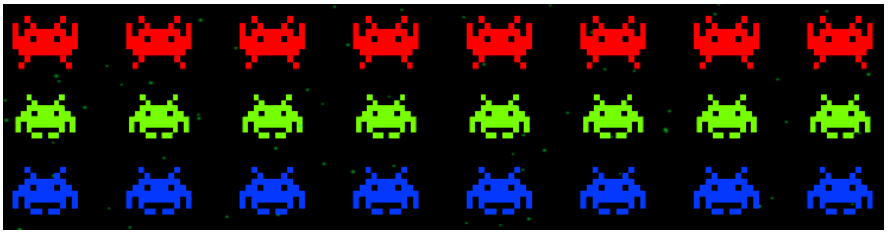




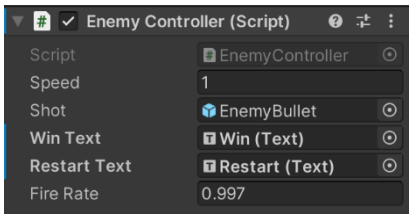


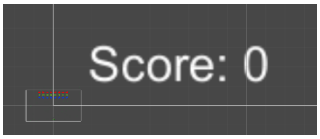
For the background of the game itself, a particle system was used to make the background move and look like you're actually in space, using the StarEffects. The positioning was set at 10 on the Y axis, the other two axis remaining at 0, followed by scaling dow the size as needed. The controls as shown on the left were fixed up, from a colour being set to green, the gravity being set to slow, duration of the effect and so on, in order to get the look of sparkly stars and movement. This was also set as a prefab.

Next was the creation of the enemies. The enemies remained as the orginal enemies of Space Invaders, as it fit the look of the game better. One enemy was imported, then duplicated to make a single row of enemies, followed by the same process to make the other enemies, only changing the colour and positioning. Once satisfied with the placement, all enemies were put into an Enemy Holder, which is an empty object, and set as a prefab. These were aswell set as sprites for 2D and UI. The Enemy Holder itself was set with a RigidBody2D and the Y Axis was ticked in order for that axis to be frozen, giving no access for the enemies to move in that diretion. The enemies were also given a RigidBody2D, aswell as a BoxCollider2D, each one being set as a trigger for when being shot by the player.



The enemy holder itself was left untagged, and set with a RigidBody 2D and the EnemyController script, while each enemy in the enemy holder was tagged under the specific colour of the enemy, each also including a RigidBody 2D and BoxCollider2D as a trigger. The enemy bullet prefab was then dragged onto the shot of the enemy.

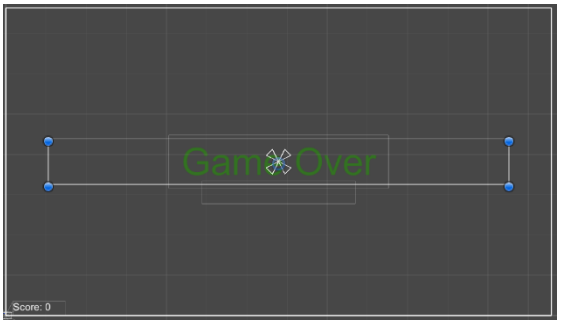


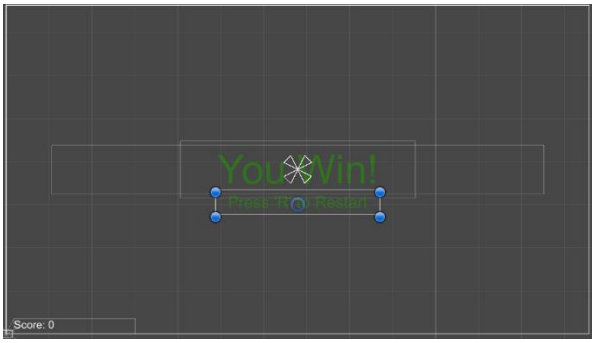


Once placing each enemy on the correct axes, next was the creation of text on the screen. Starting off by creating an empty object and setting text UI, the score text was set at the bottom left corner of the screen, using the ALT button to get these options. The font size and colour were changed to fit the colour scheme of the game, and the PlayerScore script was added to this.

Another text UI was set inside the canvas folder, named GameOver to be displayed when the player dies. The colour was changed to green and font size was enlarged, then centred. The GameOver script was added to this text, therefore the visibility was turned off from the inspector since the script controls when the text would be displayed.

This same process was used to make the Restart text and Win text. The Restart text in the canvas had its GameOver script

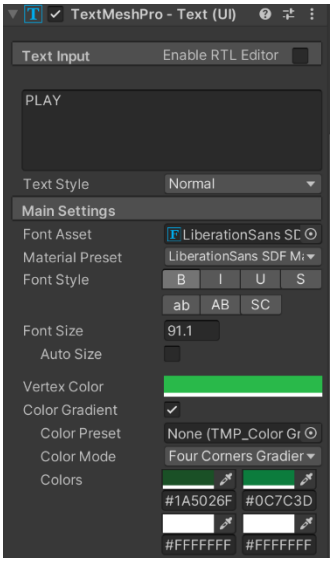




added, while another restart text was added in the hierarchy using an empty object, this time with its own script, RestartLevel.

This was basically it for the game play itself, the scripts worked properly and the game played properly, so onto the MainMenu Scene. A new scene was added in assets, being named MainMenu, this is the starting scene when opening the game, that leads to playing the game. The main menu would display the added points each specific enemy holds as well. The scene was set to 2D and Culling Mask changed to Everything, along with the background colour set to black.

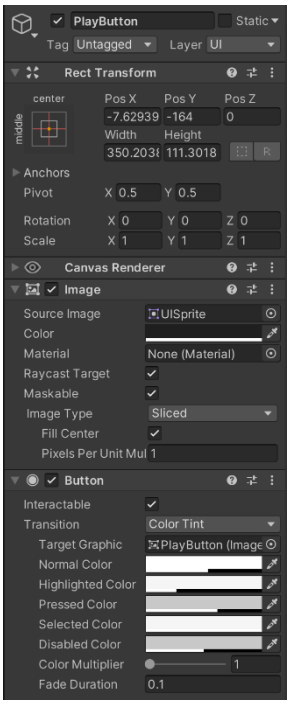
A UI Panel was created , automatically creating a canvas, where the panel was set to an image of a Green galaxy (imported image). This images texture type was set to a sprite (2D and UI). The colour was then set to a dark green to darken the background up a bit. This Panel was then renamed Background.



For the text to be added, when right clicking on the canvas, a TextMeshPro-Text was added as a UI. This makes the text show much more clear and more pleasing to the eye. The colours and size of the text was played around with and set In the centre both horizontally and vertically from the text box itself. The text was set to ‘ PLAY’. Extra setting were used to create shadows and more detail round the text. A colour gradient from the TextMeshPro was added to include several shades of green to get the shadow effect of the text.

Once ready, the text would be transformed into a button. When adding a button UI from the canvas, the size of the button was set to the exact same size of the PLAY text box, as well as the colour of the image being set to black then unticked. Under the button, an empty text object will be displayed, which will be replaced with the ‘PLAY’ text created previously. The text is named ‘Text’ while the button is renamed ‘PlayButton’ in the Hierarchy.

Then, when pressing the text in the PlayButton, while holding down ALT, the button closest to the bottom right corner is selected, which makes the text ‘PLAY’ fit inside the centre of the PlayButton.



The image of the PlayButton is then ticked again to be displayed, so the box turns black, then being fixed up to a slightly lighter shade of black. In the Button, the normal colour is set to white, while the rest of the colours are changed to lighter shades of white and grey, in order to achieve the button to change colour when being pressed and hovered over with the mouse pointer.

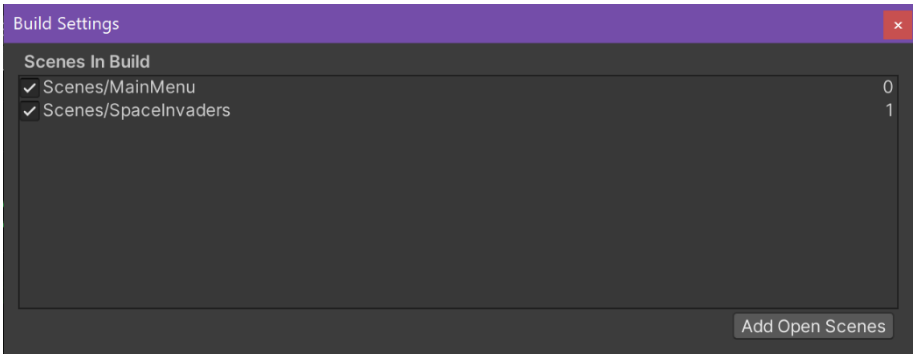
Three More text UI objects were added to the canvas which included of the amount of points added when hitting each specific enemy. The colour of each text was set according to the colour of the enemy, along with an added prefab (sprite) of each enemy.

The Logo was also added as a sprite in prefabs to be displayed at the top of the MainMenu scene.

Once ready a new Empty was created from the canvas and sized up to fit all the elements the main menu would consist of. Everything was then dragged under the MainMenu object. The MainMenu was then added its MainMenu script.

Proceeding to make the PlayButton function, on the event, the On Click option was added with an action. The MainMenu was set as the object, followed by the MainMenu script being selected to the function, PlayGame() was chosen from the script, since this is the function created previously in the script.

Lastly, to add the scene to the queue, going on File>Build Settings, the MainMenu scene would be dragged onto the Queue, making sure the MainMenu would be on 0.



Final MainMenu

Testing

Test 1

Functionality	Expected Output	Actual Output	
Pressing the Space Bar makes the player shoot	Player Shoots	Player shoots	
Holding down the Right Arrow Key makes the Player move to the right	Player moves to the right	Player moves to the right	
Holding down the Left Arrow Key makes the Player move to the left	Player moves to the left	Player moves to the left	
Enemy is destroyed when being hit by the player bullet	Enemy Dies	Enemy Dies	
Player dies when hit by enemy bullet	Player Dies	Player Dies	
Enemies shoot at given time specified	Enemies shoot accordingly	Enemies shoot accordingly	
Pressing the Play button will lead you to the SpaceInvaders Scene	Pressing Play loads the Game Scene	Pressing Play loads the Game Scene	
Pressing 'R' restarts the Game	Game Restarts	Game Restarts	
The Particle effects on the background remain moving at the set speed	Functioning moving background	Functioning moving background	
Player score will increase when destroying a specific enemy	Player score rises	Player score rises	
Score is set back to 0 whe restarting the game	Score starts at 0	Score starts at 0	
6 points are added to the player score when hitting a red enemy	Score Increases by 6	Score Increases by 6	
4 points are adde to the player score when hitting a blue enemy	Score Increases by 4	Score Increases by 4	
2 Points are added to the player score when hitting a green enemy	Score Increases by 2	Score Increases by 2	
Last remaining enemy picks up speed when moving	Last enemy moves faster	Last enemy moves faster	

Test 2

Functionality	Expected Output	Actual Output	
Pressing the Space Bar makes the player shoot	Player Shoots	Player shoots	
Holding down the Right Arrow Key makes the Player move to the right	Player moves to the right	Player moves to the right	
Holding down the Left Arrow Key makes the Player move to the left	Player moves to the left	Player moves to the left	
Enemy is destroyed when being hit by the player bullet	Enemy Dies	Enemy Dies	
Player dies when hit by enemy bullet	Player Dies	Player Dies	
Enemies shoot at given time specified	Enemies shoot accordingly	Enemies shoot accordingly	
Pressing the Play button will lead you to the SpaceInvaders Scene	Pressing Play loads the Game Scene	Pressing Play loads the Game Scene	
Pressing 'R' restarts the Game	Game Restarts	Game Restarts	
The Particle effects on the background remain moving at the set speed	Functioning moving background	Functioning moving background	
Player score will increase when destroying a specific enemy	Player score rises	Player score rises	
Score is set back to 0 whe restarting the game	Score starts at 0	Score starts at 0	
6 points are added to the player score when hitting a red enemy	Score Increases by 6	Score Increases by 6	
4 points are adde to the player score when hitting a blue enemy	Score Increases by 4	Score Increases by 4	
2 Points are added to the player score when hitting a green enemy	Score Increases by 2	Score Increases by 2	
Last remaining enemy picks up speed when moving	Last enemy moves faster	Last enemy moves faster	

Test 3

Functionality	Expected Output	Actual Output	
Pressing the Space Bar makes the player shoot	Player Shoots	Player shoots	
Holding down the Right Arrow Key makes the Player move to the right	Player moves to the right	Player moves to the right	
Holding down the Left Arrow Key makes the Player move to the left	Player moves to the left	Player moves to the left	
Enemy is destroyed when being hit by the player bullet	Enemy Dies	Enemy Dies	
Player dies when hit by enemy bullet	Player Dies	Player Dies	
Enemies shoot at given time specified	Enemies shoot accordingly	Enemies shoot at specified time, while other game plays have small delay in the enemy fire	
Pressing the Play button will lead you to the SpaceInvaders Scene	Pressing Play loads the Game Scene	Pressing Play loads the Game Scene	
Pressing 'R' restarts the Game	Game Restarts	Game Restarts	
The Particle effects on the background remain moving at the set speed	Functioning moving background	Functioning moving background	
Player score will increase when destroying a specific enemy	Player score rises	Player score rises	
Score is set back to 0 whe restarting the game	Score starts at 0	Score starts at 0	
6 points are added to the player score when hitting a red enemy	Score Increases by 6	Score Increases by 6	
4 points are adde to the player score when hitting a blue enemy	Score Increases by 4	Score Increases by 4	
2 Points are added to the player score when hitting a green enemy	Score Increases by 2	Score Increases by 2	
Last remaining enemy picks up speed when moving	Last enemy moves faster	Last enemy moves faster	

Test 4

Functionality	Expected Output	Actual Output	
Pressing the Space Bar makes the player shoot	Player Shoots	Player shoots	
Holding down the Right Arrow Key makes the Player move to the right	Player moves to the right	Player moves to the right	
Holding down the Left Arrow Key makes the Player move to the left	Player moves to the left	Player moves to the left	
Enemy is destroyed when being hit by the player bullet	Enemy Dies	Enemy Dies	
Player dies when hit by enemy bullet	Player Dies	Player Dies	
Enemies shoot at given time specified	Enemies shoot accordingly	Enemies shoot at specified time, while other game plays have small delay in the enemy fire	
Pressing the Play button will lead you to the SpaceInvaders Scene	Pressing Play loads the Game Scene	Pressing Play loads the Game Scene	
Pressing 'R' restarts the Game	Game Restarts	Game Restarts	
The Particle effects on the background remain moving at the set speed	Functioning moving background	Functioning moving background	
Player score will increase when destroying a specific enemy	Player score rises	Player score rises	
Score is set back to 0 whe restarting the game	Score starts at 0	Score starts at 0	
6 points are added to the player score when hitting a red enemy	Score Increases by 6	Score Increases by 6	
4 points are adde to the player score when hitting a blue enemy	Score Increases by 4	Score Increases by 4	
2 Points are added to the player score when hitting a green enemy	Score Increases by 2	Score Increases by 2	
Last remaining enemy picks up speed when moving	Last enemy moves faster	Last enemy moves faster	

Test 5

Functionality	Expected Output	Actual Output	
Pressing the Space Bar makes the player shoot	Player Shoots	Player shoots	
Holding down the Right Arrow Key makes the Player move to the right	Player moves to the right	Player moves to the right	
Holding down the Left Arrow Key makes the Player move to the left	Player moves to the left	Player moves to the left	
Enemy is destroyed when being hit by the player bullet	Enemy Dies	Enemy Dies	
Player dies when hit by enemy bullet	Player Dies	Player Dies	
Enemies shoot at given time specified	Enemies shoot accordingly	Enemies shoot accordingly	
Pressing the Play button will lead you to the SpaceInvaders Scene	Pressing Play loads the Game Scene	Pressing Play loads the Game Scene	
Pressing 'R' restarts the Game	Game Restarts	Game Restarts	
The Particle effects on the background remain moving at the set speed	Functioning moving background	Functioning moving background	
Player score will increase when destroying a specific enemy	Player score rises	Player score rises	
Score is set back to 0 whe restarting the game	Score starts at 0	Score starts at 0	
6 points are added to the player score when hitting a red enemy	Score Increases by 6	Score Increases by 6	
4 points are adde to the player score when hitting a blue enemy	Score Increases by 4	Score Increases by 4	
2 Points are added to the player score when hitting a green enemy	Score Increases by 2	Score Increases by 2	
Last remaining enemy picks up speed when moving	Last enemy moves faster	Last enemy moves faster	