Project Design Document

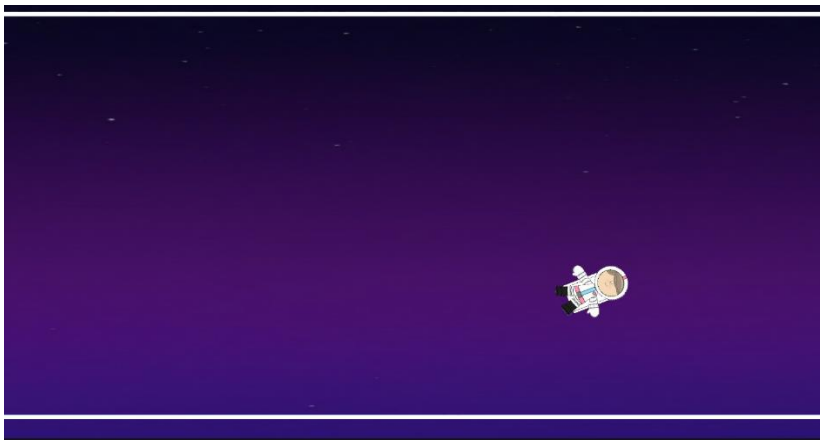
21/12/2022
S. SRIRAM

Game Name: Fly Away

Genre: Hyper Casual

Game Summary

In this side-scrolling game set in open space, player can control a Space through the mouse/touch. The objective is to maneuver the character within the endless imaginary boundary lines that appear at the top and bottom of the screen. This 2D game Is built In Unity.



Game Preview

Target Platform

- PC, Android Tablet and phone.
- This game will be In Landscape Mode.

Language

- C#

Gamplay video

- **For Pc**
- **For Mobile**

Project Concept

1. **Player Control**
   - In this game the player can control the character by touch through **PlayerController.cs** script
   - Navigating to the scripts folder in the project window and open up the **PlayerController.cs** file. To set up the Player object with the mouse position add this following variable in the **Update** method.

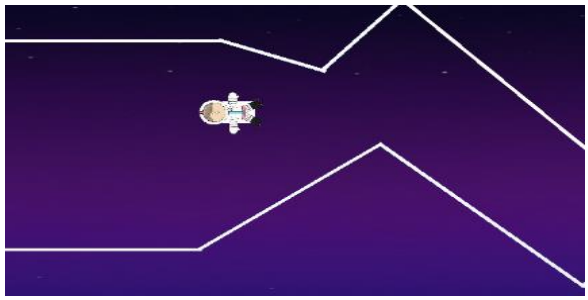
```
Vector3 mousePos = Camera.main.ScreenToWorldPoint(Input.mousePosition);
mousePos.z = -10f;
transform.position = mousePos;
```

   - **ScreenToWorld** will convert the Screen position of the mouse to a world position. The reason to use 10.0f on the z axis, because the camera has the z position of 10.0f.
   - An Invisible boundary was set up to control the **Player Object** within the **Camera Orthographic** projection. In the void Border method, the **transform.position** in the **Transform** class Is used to create an invisible wall and control the **Player Object** within the **Camera** screen. The position was set based on the **Camera** screen Position.

2. Basic Gameplay

- An Imaginary boundary line was created on the top and bottom of the game screen as per the player difficulty.

Game Pic-1



Game Pic-2

- During the game, random asteroids and UFOs will appear as enemies in the top and in front of the player. Using **BaseEnemy** Script, these enemy objects move towards the player at a set speed. In **BaseEnemy.cs**, Translate it to the right according to the speed variable. Add 2D collider on the enemy make sure it surrounded properly. Drag **Enemy** into the **Prefabs** folder to create a new **Prefab**, then delete **Enemy** from scene
- This function will require a Game Manager to position the enemy object and where the enemy will appear randomly.

Asteroids Spawn

UFOs Spawn

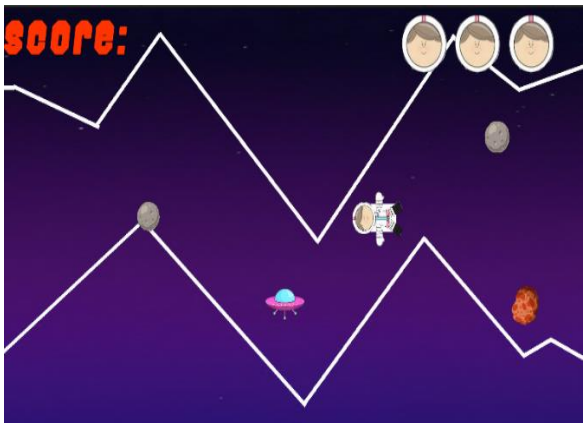
- Create **GameManager** game object then add the **GameManager** script that handle the enemy object spawning position. Create a sprite in the script folder and name the script **GameManager.**
- Declare new array **public GameObject[ ] enemyPrefabs**. In the Inspector, change the Array size to match the enemy count, then assign enemy object by dragging them from the Project window into the empty slots.

```
public GameObject[] enemyPrefab;
InvokeRepeating("Spawning", 3, 3);
```

- Create new **void Spawning** method, then move the **Instantiate** call inside it and create new float variables for **startDelay** and **repeatRate**. Obstacles spawn on intervals using the **InvokeRepeating()** method. A function **Vector3 SpawnPos()** is created and to return the method for the position of enemies In the Instantiate call method.

```
int index = Random.Range(0, enemyPrefab.Length);
Instantiate(enemyPrefab[index], SpawnPos(), enemyPrefab[index].transform.rotation);
```

- In the **Player** object Add **RigidBody 2D** and **2D Collider** components for the collision, Colliders will surround objects properly and freeze the constraints for the rotation in the **rigidbody 2D** component during collision. If the player collides with **gameobject**(enemy) the game will end.
- The goal of the game Is set up the Player to avoid boundary line and upcoming enemies.



Gameplay Screen

Game Over Screen

3. Gameplay Mechanics

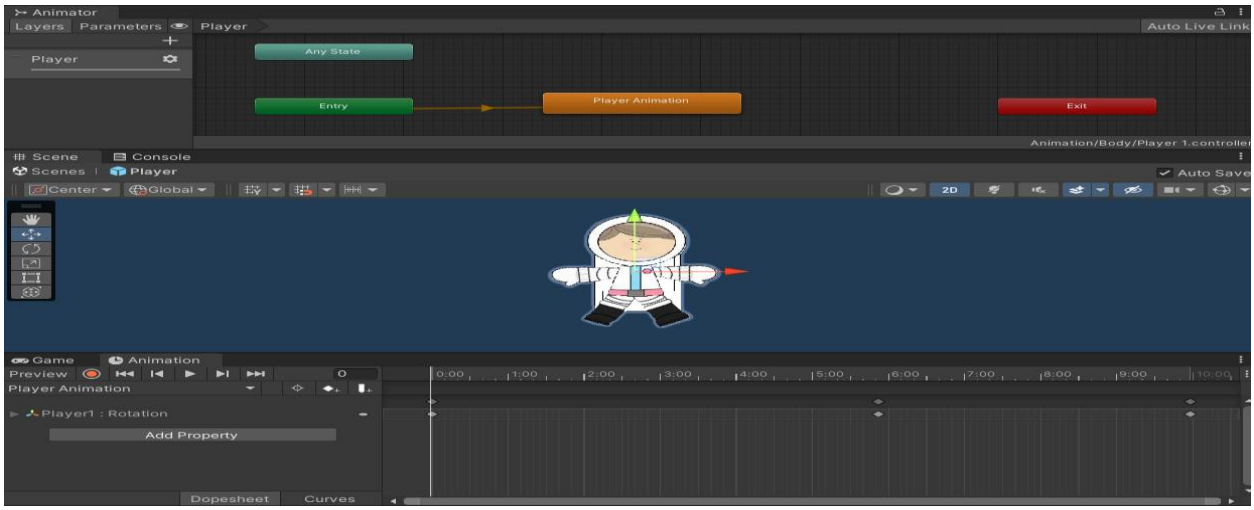
- For the background repeating on the camera view declare private float **repeatWidth** variable in **Awake()** and incorporate the **repeatWidth** variable into the repeat function.

```
if (transform.position.x > -repeatwidth)
{
    transform.position = startPos;
}
```

- In void **Game Over**, add **Boolean** declaring **game Over = true** for the game over function**.** Declare new private **GameManager gameManagerScript** for the reference between the scripts. In **Start()** or **Awake()**, initialize it by finding the **GameManager** and getting the **GameManager** component then wrap the translate method in an if-statement checking if game is not over and add a condition for the game action between scripts if **gameOver == false** or **!gameOver**.
- Background sprite has been used for the game backgound. The background repeat by declare a new private float **repeatWidth** variable In the Repeat Background script. Incorporate the **repeatWidth** variable into the repeat function
- As the game progresses, the game speed will increase by changing the value of **Time.timeScale** in **void Speed()** in **BaseEnemy.cs** and **RepeatBackground.cs** make the player difficult to avoid the obstacles.

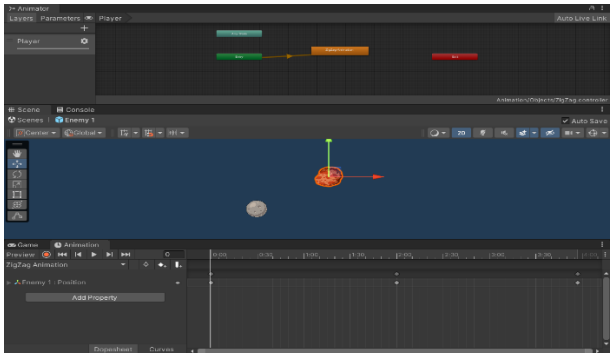
4. Animations

- **Player animation**
  - In order to get this character moving their body, arms and legs different animation were created in each part of the body and added in the **Animator** on the Player game Object to make the player look like they're really struggling in the open space. In **Animation Window** record the animation by moving the game object, the **Dropsheet** and **Curves** used to adjust the duration and movement of the animation.

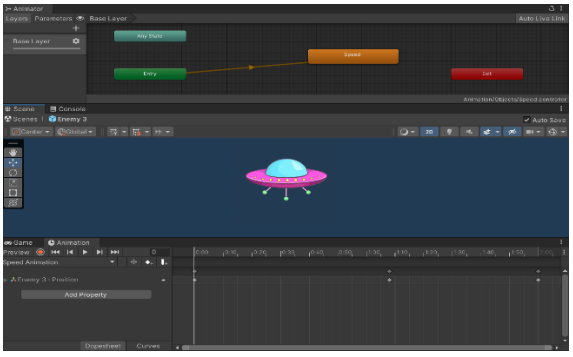


Player animation

- **Enemies' animation**
  - To simulate the motions of asteroids and UFOs, multiple animations were created in Animator, with the asteroids moving randomly and the UFOs traveling continuously in outer space.



Asteroids Animation



UFOs Animation

- These animation for the game is completely done in unity. Use Time to stop the animation after the game ends by **Time.timeScale = 0** In the **GameOver()** on the **gameManager.cs** Script and add **Time.timeScale=1** In **Start()** to resume the animation when the game restarts.

5. User Interface

- At the start of the game the title will appear. For the game title window create a new **Scene** and add sprite for the background. Name the tittle and the other option **button** added for the game in the **Canvas** using **TextMeshPro**. Create a new **MainMenu** script and add the using **Unity.SceneManagement** and declare the **Playgame()** to start the game Scene by **SceneManager** method in the appropriate name and arrange both the Scenes in the **Build Index** option. To exit the game, add the **Application.Quit()** in the **Quit().** Add the recommended function for these option buttons add the scripts to the **On click()** function in the button.

Game Menu Scene

- The score will be added whenever the player avoids the enemy objects. For the score updates, declare a new public **TextMeshProUGUI scoreText** in the **gameManager.cs** script, then assign that variable in the inspector add the appropriate font for the text. Create a private int score variable and initialize it in Start() as score = 0 and add **scoreText.text = "Score: " + score** into the **UpdateScore()**, increment the score by adding **score+=scoreToAdd** and call **gameManager.UpdateScore(5)** on the **Update()** in the **BaseEnemy.cs** script.

Updating Score

- In **GameManager** scripts add two variable one for the lives as game object by creating an array in the script and other for the amount of lives available. After the **Update** method, **LivesCount** method Is added to adjust the amount of lives and display the result as sprite In Canvas that was created to check to see If the lives get reduce the game object get destroyed by **Destroy** method. This lives game object will display on the game start by update this method on the **Start().** And the game end when the life run out by calling the **GameOver** method.

```
public GameObject[] life;
public int lives;
public void LivesCount(int livesChange)
{
    lives += livesChange ;

    if (lives <= 0 )
    {
        GameOver();
    }
   else if (lives < 1)
   {

        Destroy(life[2].gameObject);

    } else if (lives < 2)
    {

        Destroy(life[1].gameObject);

    }
   else if (lives < 3)
   {

       Destroy(life[0].gameObject);

    }
}
```

```
public void GameOver()
{
    GameOverText.gameObject.SetActive(true);
    gameOver = true;
    Time.timeScale = 0;

}
```

Lives at it full

Lives when reduce

Lives run out

- In the **PlayerController** Script, declare the **private void OnCollision2D** method for which particular object that the player should collide by giving a specific **tag** for game objects for the require conditions to make the game object collide.

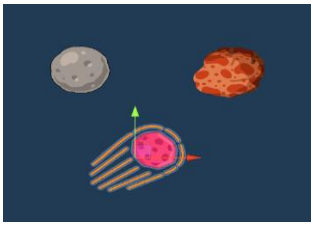
```
private void OnCollisionEnter2D(Collision2D collision)
{
    if (collision.gameObject.CompareTag("Enemy")||collision.gameObject.CompareTag("Border"))
    {
        gameManager.LivesCount(-1);

    }
}
```

- Game Over **Panel** created in the **Canvas** that will display when the game end using **TextMeshPro**, Add the restart and main menu button along with the panel. The restart button will restart the game and main menu button go to the Main Menu Scene; this function will take place using **SceneManager** In the **gameManager** script.

Game Over

6. **Assets**
   - **Game Sprites**
   - In this 2D game the assets are download nor created and making It as 2D sprites. Select the game object in the hierarchy, then in the **Sprite Renderer** component select the required sprites for the player and enemy game objects.

Player Sprite



Asteroids Sprites

UFO Sprite

- Other Sprites
  - For the game's background and other UI settings, utilize recommended sprites to make the game seem visually appealing.

Project Timeline

Milestone	Description	Due
#1	- *Project / Camera set up with primitive objects for all gameplay objects.*	*22/12*
#2	- *Player can move in all direction and cannot leave the gameplay.* - *Scrolling background.*	*22/12*
#3	- *Objects spawn randomly from the top of screen.* - *When player collides with other object, lives reduce.* - *When player avoid other object successfully get score.*	*22/12*
#4	- *Primitive object and background replaced 2D sprites*	*23/12*
#5	- *Lives / Gameover mechanics programmed.*	*23/12*
#6	- *Animation for the game object.*	*25/12*
Backlog	- *Powerup object for bonus points.* - *High Score board when the game end.* - *Sound effects.*	*28/12*

Project Sketch