Space Shooter

I. Setting up the Project.

1.In this assignment first we will create a new project.

FileNew ProjectGive the name for the project and give a location where you want to store your project.

2.The next step is to import assets from the Asset Store.

Goto to Asset storeDownload Space shooter Tutorial.You will see a window

The window contains all the files we needed for the project.Click on all and then import .

3.Unity will open a default scene. We need to save the scene. Click on FileSave SceneSave the scene in asset directory of your project and name it as main.

4.To set the Build Target.

FileBuild settingsA build setting window will appear.

Change the build target to WebGLNow click on Switch platform.

5.Now click on Player Settings the player setting will get open in inspectorSet the resolution as

Width=600

Height=900

The Player game Object.

1.Add the player's ship model. Goto asset folderModels directoryDrag the vehicle_playerShip

From the Models directory to Hierarchy view.

To get a better view of model in our scene .Goto EditFrame Selected.

2.Rename the player ship.In the hierarchy click on the game object and name the object as Player.

3.In the upper right of the Transform Component select Reset. This will reset the Transform's position.

4.Click on Add ComponentPhysicsRigidBody.

Deselect the use gravity option in RigidBody.

5.Click on Add ComponentPhysicsCapsule Collider.

Change the Direction as Z-axis.

Radius=0.51

Height=1.95

For top-down view let's click on the Scene View Gizmo and click on the Y arm.

6.Click on Add ComponentPhysicsMesh Collider.

Uncheck the mesh Renderer.

Go to Models Click on Vehicle player (The first one).

Click on the arrow button player_ship_collider will appearDrag it in to the Mesh slot of the Mesh collider.

Again Check the mesh renderer.

In Mesh Collider select Is Trigger.

7.Goto PrefabVFXengines. Select engines_player prefab. Add this to Player By dragging it on to the Player ship in the hierarchy as a child game object.

If the gizmos in the scene are too large we can reduce them by selecting the gizmos button in the scene view toolbar and adjusting the slider.

3. Camera and Lighting:

- In Scene, select Main Camera -> position -> 0,10,5, Rotation -> 90,0,0, select orthographic in projection, change size to 10.
- In Game view, change the Background by setting the Clear Flags to Solid. In the color picker change each value to 0,0,0,5 respectively.
- Click on Create in the Hierarchy view -> Directional Light -> Rename it to Main Light.
- Reset the transform, set Rotation = 20, -115, 0, increase Intensity to 0.75.

- Select Main Light go to Edit -> Duplicate -> Rename as Fill Light, change Rotation to 5, 125, 0, color= 128, 192, 192, 255 and intensity to 0.5
- Duplicate the Fill Light and rename it as Rim Light, uncheck the Fill Light.
- Select Rim Light and reset the transform, set color as 255, 255, 255, 255.
- Set Rotation values to -15,65, 0 and intensity to 0.25.
- Organize the Scene by using empty game objects.
- In Scene View, Create Empty -> Game Object -> Rename as Lighting, reset the game object transform. Set position as 0, 100, 0.
- Drag all the light objects in the Lighting folder to make them its child.

4. Adding a Background:

- Deactivate the player game object.
- Click on create add a Quad and rename it as Background.
- Reset the transform.
- Set the rotation of X-axis as 90.
- Remove the Mesh collider because it has no use.
- Then we have to add texture to the background.
- In assets, there is a directory called Textures, click on it.
- Then select the tile_nebula_green_diff image and drag it to the background quad in the scene.
- Now change the scale value in the transform of x, y and z axis as 15, 30 and 1 respectively.
- Now in the Mesh renderer, go to shader, and select Unlit -> Texture.
- Now select the Player, and activate it.
- Again select the background from the hierarchy and change the position of y-axis to -10 in transform.

5. Moving the player:

- Select assets and in project view use create menu and select Folder, this will create a new folder in assets. Change its name to "Scripts".
- In player click on "Add component" new scriptset name as- "PlayerController"create and add.
- In Assets, open scripts PlayerController for editing.
- Type following code and return to unity.
- In PlayerController change the speed to 10. Save and play.
- Since our ship can leave the game area we need to make certain changes in our script code. Then we'll see the boundary section in the PlayerController. Set Xmin= -6, Xmax= 6, Zmin= -4, Zmax= 8.
- To add some tilt to our ship change the code and return to unity and set the tilt value to 4. Save and play.

6.Creating Shots

1) Deactivate Player Gameobject.(i.e. Uncheck the Player)

2) Create a New Empty Gameobject.(shift+ctrl+N)

3) Rename Empty Gameobject as Bolt.(This will be Parent Gameobject for Shots.)

4) Reset Gameobject Transforms.

5) Create a Quad to hold visual effect.(Navigate to Create/3DObject/Quad)

6) Rename the quad as VFX and reset the quad's Transform.

7) Add VFX Gameobject as Child of Bolt.(i.e. Drag VFX to Bolt.)

8) Now Change the Transform of VFX as (Rotation : x=90).

9) Open Textures Folder and find 'fx_laser_orange'.(---)

10) Select Materials folder in the root level of Assets. With the Materials folder selected click on create menu in the

Project View and select Material.(Creates a new Material in Materials Folder.)

11) Now Rename the material 'fx_bolt_orange'.

12) Click on the Materials header to expand the Materials Inspector Panel if its closed.(Can see all of the fields for this Material.)

13) Now to associate texture with Material, we have 2 ways :

a) In the Materials main texture field, click on select and this will bring up an asset picker. Click on 'fx_laser_orange'.

b) Drag and Drop a texture into this field. i.e. Select textures in the root of assets and drag 'fx_laser_orange' onto our 'fx_bolt_orange' material and drop it into the main texture field.

14) Now to associate the material with the VFX quad simply drag the Material onto the quad.

15) Change the Shader of the VFX choose 'Particles -> Additive' or 'Mobile -> Particles -> Additive'.(Click on VFX under Bolt, You will see an Option as Shader).

16) Select Bolt and Click on Add Component and select 'Physics -> Rigidbody' and Uncheck 'Use Gravity'(Located under Rigidbody).

17) Click on VFX (under Bolt) and Remove Mesh Collider Component(Use Context Gear Menu and Click on Remove Component.)

18) Click on Bolt and Click on Add Component Select 'Physics -> Capsule Collider'

19) Change : Direction of Capsule Collider to Z-Axis, Radius to 0.03 and Height to 0.5 and Check 'Is Trigger'.

20) Select Bolt and Click on Add Component and Choose New Script and name it as 'Mover' and Click on Create and Add.

21) Place Mover into the Scripts Folder and Open it for Editing and write the following code :

```
using UnityEngine;
using System.Collections;

public class Mover : MonoBehaviour
```

```
{
    public float speed;

    void Start ()
    {
        rigidbody.velocity = transform.forward * speed;
    }
}
```

22) Now Save the Script and return to Unity.

23) Save Bolt Gameobject as Prefab(i.e. Drag the Bolt Gameobject from hierarchy view into the Prefab's Folder in Assets. Click on Prefab to see the Bolt Prefab.)

24) Now Set the Speed Value for Bolt.(Click on Bolt from hierarchy view, under Mover Script we can see 'Speed' as part of Mover Component and set Speed Value as 20 or Whatever you want).

25) Delete Working Instance of Bolt from Scene(Select Bolt from Hierarchy View and Press 'Del' Button) because we only want one instance of Bolt Gameobject in the scene.

26) Check the Player Gameobject and Save the Scene.

7. Shooting shots:

1. To enable Shooting, select the Player from Hierarchy and select the Player from Inspector.

2. Open the Player Controller script for updating, to perform the shooting action and type the following code. Save it and return to Unity.

3. Create a new Game Object as Shot Spawn. Drag it on the player to make it as a child. Set the position of Z-axis as 1.25 .

4. Open the Prefab folder, drag the Bolt onto the Shot in Player Controller script in Player Hierarchy.

5. Now, drag the Shot Spawn game object from the Hierarchy in the Shot Spawn panel in Player Controller script.

6. Set the Fire Rate as 0.25 .

II. BOUNDARY, HAZARDS AND ENEMIES:

1. Boundary:

- Click on Create -> Cube, rename as Boundary and reset its transform.
- Select Boundary, and in its box collider, select the isTrigger option as checked.
- Set the Position of z-axis to 5. Change the scale of x-axis to 15 and z-axis to 20.
- Untick the Mesh Renderer.
- Select Boundary -> Add component -> New script ->rename as DestroyByBoundary -> Create and Add.
- Now click on Assets, you will see the DestroyByBoundary, place it in the Scripts folder.
- Open it in Vs 2017, and write the code required and save it.
- In boundary, remove the mesh renderer and mesh filter.
- Save the scene.

2. Creating hazards:

- Click on Player and press Ctrl+Shift+N, rename it as Asteroid, and reset its transform.
- Click on Asteroid, change the position of z-axis to 8.
- Click on Models folder in assets, choose the first asteroid, drag it and drop it in the Asteroid folder in the hierarchy view.
- Now click on the dragged object whose name is prop_asteroid, and reset its transform to origin.
- click on Asteroid, add new component -> physics -> Rigid body.

3. Select the aste
Player on to the a

- Deselect the use gravity of the asteroid.
- Again click on add component -> Capsule Collider and change the height to 1.34.
- Add component -> new script -> rename it as RandomRotator. add code.
- Select asteroid and set the tumble to 5.
- Change the angular drag of rigid body to 0.
- again add component -> new script -> rename it as DestroyByContact. Add code.
- Select the boundary and in the right side you will see tagged button, click on it and add tag.
- By default it is Untagged, Add a new tag boundary(Click on + sign).
- Now again go back, and select the Tag as Boundary(you will see the Boundary option now in the drop down list).
- Save scene.

8.Explosions

- Keeping explosions selected in hierarchy

In prefabs folder go to explosions and drag the explosions to explosion slot to create a reference on DistroyByContract script. Save the scene

2. Select the Player game object and in Player inspector give the tag name Player.

3. Select the asteroid game object and drag the explosion player on to the explosion slot to the

DistroyByContract component. Save the scene.

4. Keeping the asteroid selected in hierarchy,

Open the scripts folder drag the mover script on to the asteroid game object in the inspector.

5. Set the speed value to -5(to bring the asteroids down).

6. Drag the asteroids game object from hierarchy and onto prefabs folder in assets (save the asteroid as prefabs)

7. Delete the instance of a asteroid from the scene.

9.Game Controller

1. Create a game object and name it as Game Controller and then reset the transform.

2. Select the tag dropdown and give game controller.

3. Keep the game controller selected and add a component to create a new script called

Game controller.

4. Select the assets and move the game controller script to scripts folder.

5. Open the script and write the code and save the script.

On unity,

6. Drag the asteroid prefab to the game controller component and drop it on to the hazard slots to create a reference.

7. Set the Spawn values (x=6, y=0 , z=16).

8. Save the scene .

10.Spawning Waves.

- Select the game controller change the hazard count to 10.

- Keeping the gamecontoller selected
- Spawn wait to 0.5 (spawn 2 asteroid)
- Start wait to 1 (giving a player 1 second to wait)
- Set the wave wait property to 4.
- Select the scripts folder and use the create menu in the project view and choose the C# script.
- Rename the script as DestroyByTime.
- Write the code and save it.

11. ***AUDIO:***

- Audio clips hold our audio data or sound files, audio sources play our audio clips in the scene.
- In assets there is a audio folder, if we select any one of the audio clips we will see its import settings in the inspector.
- Deselect 3d sound for all audio clips.
- Change existing project view from 2 column layout to using a single column layout.
- Drag the explosion_asteroid audio clip on to the game object in the inspector and drop it on to the asteroid explosion prefab asset.
- Play on awake should be on by default.
- Drag the explosion_player audio clip on to the game object in the inspector and drop it on to the explosion_player prefab asset.
- Select weapon_player audio clip and drag it into the scene view and drop it on to the player game object.
- Deselect play on awake.
- Select player controller script and open it for editing.

- Type code,save and return to unity.
- Select music background audio clip and drag it into the scene view and drop it on to the game controller object.
- Select play on awake and loop as well.
- Playeraudio source componentselect volume property to 0.5.
- Game controlleraudio source componentselect volume property to 0.5.

12. *COUNTING POINTS AND DISPLAYING THE SCORE:*

- Using create menu in the heierarchy view select GUI text.
- Change pixel offset to 10 and -10.
- Select game controller and open it for editing, type code, save and return to unity.
- Select asteroid prefab, in the inspector open destroy by contact script. Edit the code, save and return to unity.
- Edit the code again to remove error .
- Select asteroid prefab in game object and type score value to 10.
- Open destroy by contact script. Edit the code, save and return to unity.
- Open game controller script. Edit the code, save and return to unity.
- Keep the game controller selected drag score text game object on to the score text property in the game controller. Save and play.

13. *ENDING THE GAME:*

- Rename the game object to display text and reset the game object transform.

- Add score text to display text. Create a new GUI text, rename it to restart text and change its name property to restart text.
- Change the transform position to 1,1, change anchor property to upper right , alignment to right and pixel offset to -10 and -10.
- Add restart text to display text.
- Create a new GUI text, rename it to game over text and change its name property to restart text.
- Change anchor property to middle center , alignment to center and change the transform position of y to 0.6.
- Add game over text to display text.
- Open game controller script and update its code, save and return to unity.
- Drag restart text game object onto the restart text slot on to the game controller component.
- Drag game over text game object onto game over text slot on to the game controller component.
- Open destroy by contact script, edit code, save and return to unity.
- Select both restart text and score text change its font size to 20.
- Select both game over text change its font size to 25.

coding.

PlayerController.cs

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

```
using UnityEngine;

public class PlayerController : MonoBehaviour
{
    public float fireRate = 0.5F;
    public float speed;
    public float tilt;
    public Boundary boundary;
    public GameObject shot;
    public Transform shotSpawn;

    private float myTime = 0.0F;
    private float nextFire = 0.5F;
    private Rigidbody rb;
    private AudioSource audioSource;

    void Start()
    {
        rb = GetComponent<Rigidbody>();
        audioSource = GetComponent<AudioSource>();
    }

    void Update()
    {
        myTime = myTime + Time.deltaTime;

        if (Input.GetButton("Fire1") && myTime > nextFire) {
            nextFire = myTime + fireRate;
            Instantiate(shot, shotSpawn.position,
shotSpawn.rotation);

            audioSource.Play();

            nextFire = nextFire - myTime;
            myTime = 0.0F;
        }
    }

    void FixedUpdate()
    {
        float moveHorizontal = Input.GetAxis("Horizontal");
        float moveVertical = Input.GetAxis("Vertical");

        Vector3 movement = new Vector3(moveHorizontal, 0.0f,
moveVertical);
        rb.velocity = movement * speed;

        rb.position = new Vector3(
        Mathf.Clamp(rb.position.x, boundary.xMin,
boundary.xMax), 0.0f,
            Mathf.Clamp(rb.position.z, boundary.zMin,
boundary.zMax)
```

```
        );

        rb.rotation = Quaternion.Euler(
            0.0f,
            0.0f,
            rb.velocity.x * -tilt
        );
    }
}
```

BackgroundScroller.cs

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

public class BackgroundScroller : MonoBehaviour
{
    public float scrollSpeed;
    private float tileSizeZ;

    private Vector3 startPosition;

    void Start()
    {
        startPosition = transform.position;
        tileSizeZ = transform.localScale.y;
    }

    void Update()
    {
        float newPosition = Mathf.Repeat(Time.time * scrollSpeed,
tileSizeZ);
        transform.position = startPosition + Vector3.forward *
newPosition;
    }
}
```

Boundary.cs

```
using System;

[System.Serializable]
public class Boundary
{
    public float xMin, xMax, zMin, zMax;
}
```

DestroyByBoundary.cs

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

public class DestroyByBoundary : MonoBehaviour
{
    void OnTriggerExit(Collider other)
    {
        Destroy(other.gameObject);
    }
}
```

DestroyByContact.cs

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

public class DestroyByContact : MonoBehaviour
{
    public GameObject explosion;
    public GameObject playerExplosion;
    public int scoreValue;

    private GameController gameController;

    void Start()
    {
        GameObject gameControllerObject =
GameObject.FindWithTag("GameController");

        if (gameControllerObject != null) {
            gameController =
gameControllerObject.GetComponent<GameController>();
        }

        if (gameControllerObject == null) {
            Debug.Log("Cannot find 'GameController' script");
        }
    }

    void OnTriggerEnter(Collider other)
    {
```

```
        if (other.CompareTag("Boundary") ||
other.CompareTag("Enemy")) {
            // Ignore Boundary
            return;
        }

        if (explosion != null) {
            Instantiate(explosion, transform.position,
 transform.rotation);
        }

        if (other.CompareTag("Player")) {
            Instantiate(playerExplosion, other.transform.position,
 other.transform.rotation);

            gameController.GameOver();
        } else {
            gameController.AddScore(scoreValue); // Only add to
 the score when not hitting the Player!
        }

        Destroy(other.gameObject); // Bolt or Player
        Destroy(gameObject); // Asteroid
    }
}
```

DestroyByTime.cs

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

public class DestroyByTime : MonoBehaviour
{
    public float lifetime;

    void Start()
    {
        Destroy(gameObject, lifetime);
    }
}
```

EvasiveManeuver.cs

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

```
public class EvasiveManeuver : MonoBehaviour
{
	public float dodge;
	public float smoothing;
	public float tilt;
	public Boundary boundary;
	public Vector2 startWait;
	public Vector2 maneuverTime;
	public Vector2 maneuverWait;

	private float currentSpeed;
	private float targetManeuver;
	private Rigidbody rb;

	void Start()
	{
		rb = GetComponent<Rigidbody>();
		currentSpeed = rb.velocity.z;

		StartCoroutine(Evade());
	}

	IEnumerator Evade()
	{
		yield return new WaitForSeconds(Random.Range(startWait.x,
startWait.y));

		while (true) {
			targetManeuver = Random.Range(1, dodge) * -
Mathf.Sign(transform.position.x);
			yield return new
WaitForSeconds(Random.Range(maneuverTime.x, maneuverTime.y));
			targetManeuver = 0;
			yield return new
WaitForSeconds(Random.Range(maneuverWait.x, maneuverWait.y));
		}
	}

	void FixedUpdate()
	{
		float newManeuver = Mathf.MoveTowards(rb.velocity.x,
targetManeuver, Time.deltaTime * smoothing);
		rb.velocity = new Vector3(newManeuver, 0.0f, currentSpeed);

		rb.position = new Vector3(
			Mathf.Clamp(rb.position.x, boundary.xMin,
boundary.xMax),
			0.0f,
			Mathf.Clamp(rb.position.z, boundary.zMin,
boundary.zMax)
		);
```

```
        rb.rotation = Quaternion.Euler(
            0.0f,
            0.0f,
            rb.velocity.x * -tilt
        );
    }
}
```

GameController.cs

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

public class GameController : MonoBehaviour
{
    public GameObject[] asteroids;
    public Vector3 spawnValues;
    public int asteroidCount;
    public float startWait;
    public float spawnWait;
    public float waveWait;
    public Text scoreText;
    public Text restartText;
    public Text gameOverText;

    private int score;
    private bool gameOver;
    private bool restart;

    void Start()
    {
        score = 0;
        UpdateScore();

        gameOver = false;
        gameOverText.text = "";

        restart = false;
        restartText.text = "";

        StartCoroutine(SpawnWaves());
    }
```

```
void Update()
{
        if (restart) {
            if (Input.GetKeyDown(KeyCode.R)) {
                Application.LoadLevel(Application.loadedLevel);
            }
        }
}

IEnumerator SpawnWaves()
{
        yield return new WaitForSeconds(startWait);

        while(true) {
            for (int i = 0; i < asteroidCount; i++) {
                GameObject asteroid = asteroids[Random.Range(0,
asteroids.Length)];

                Vector3 spawnPosition = new
Vector3(Random.Range(-spawnValues.x, spawnValues.x), spawnValues.y,
spawnValues.z);
                Quaternion spawnRotation = Quaternion.identity;

                Instantiate(asteroid, spawnPosition,
spawnRotation);

                yield return new WaitForSeconds(spawnWait);
            }

            yield return new WaitForSeconds(waveWait);

            if (gameOver) {
                restart = true;
                restartText.text = "Press 'R' to Restart";
                break;
            }
        }
}

void UpdateScore()
{
    scoreText.text = "Score: " + score.ToString();
}

public void AddScore(int newScoreValue)
{
    score += newScoreValue;
    UpdateScore();
}

public void GameOver()
```

```
        {
            gameOver = true;
            gameOverText.text = "GAME OVER";
        }
    }
```

Mover.cs

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

public class Mover : MonoBehaviour
{
    public float speed;

    private Rigidbody rb;

    void Start()
    {
        rb = GetComponent<Rigidbody>();

        rb.velocity = transform.forward * speed;
    }
}
```

RandomRotator.cs

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

public class RandomRotator : MonoBehaviour
{
    public float tumble;

    private Rigidbody rb;

    void Start()
    {
        rb = GetComponent<Rigidbody>();

        rb.angularVelocity = Random.insideUnitSphere * tumble;
    }
}
```

WeaponContoller.cs

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

public class WeaponController : MonoBehaviour
{
    public GameObject shot;
    public Transform shotSpawn;
    public float fireRate;
    public float delay;

    private AudioSource audioSource;

    void Start()
    {
        audioSource = GetComponent<AudioSource>();

        InvokeRepeating("Fire", delay, fireRate);
    }

    void Fire()
    {
        Instantiate(shot, shotSpawn.position, shotSpawn.rotation);

        audioSource.Play();
    }
}
```