FOUR-SIX WEEKS SUMMER TRAINING REPORT

ON

INTRODUCTION TO GAME DEVELOPMENT

Submitted by:
Prithve Kumar .P
Reg: 11601461
Computer Science(Hons)

Under the guidance of:
MR. Brain M. Winn
Associate Professor
Media and Information
Michigan State University.

School of Computer Science and Engineering
Lovely Professional University,Phagwara.
(June-July, 2018)

DECLARATION

I hereby declare that I have completed my four-six weeks summer training at **COURSERA** from **01-06-2018** (start date) to **01-07-2018** (end date) under the guidance of **MR. Brain M. Winn** . I have declare that I have worked with full dedication during these four-six weeks of training and my learning outcomes fulfill the requirements of training for the award of degree of **Introduction to Game Development** , Lovely Professional University, Phagwara.

PRITHVE KUMAR .P

11601461

Date: 01-07-2018

ACKNOWLEDGEMENT

Course Certificate:

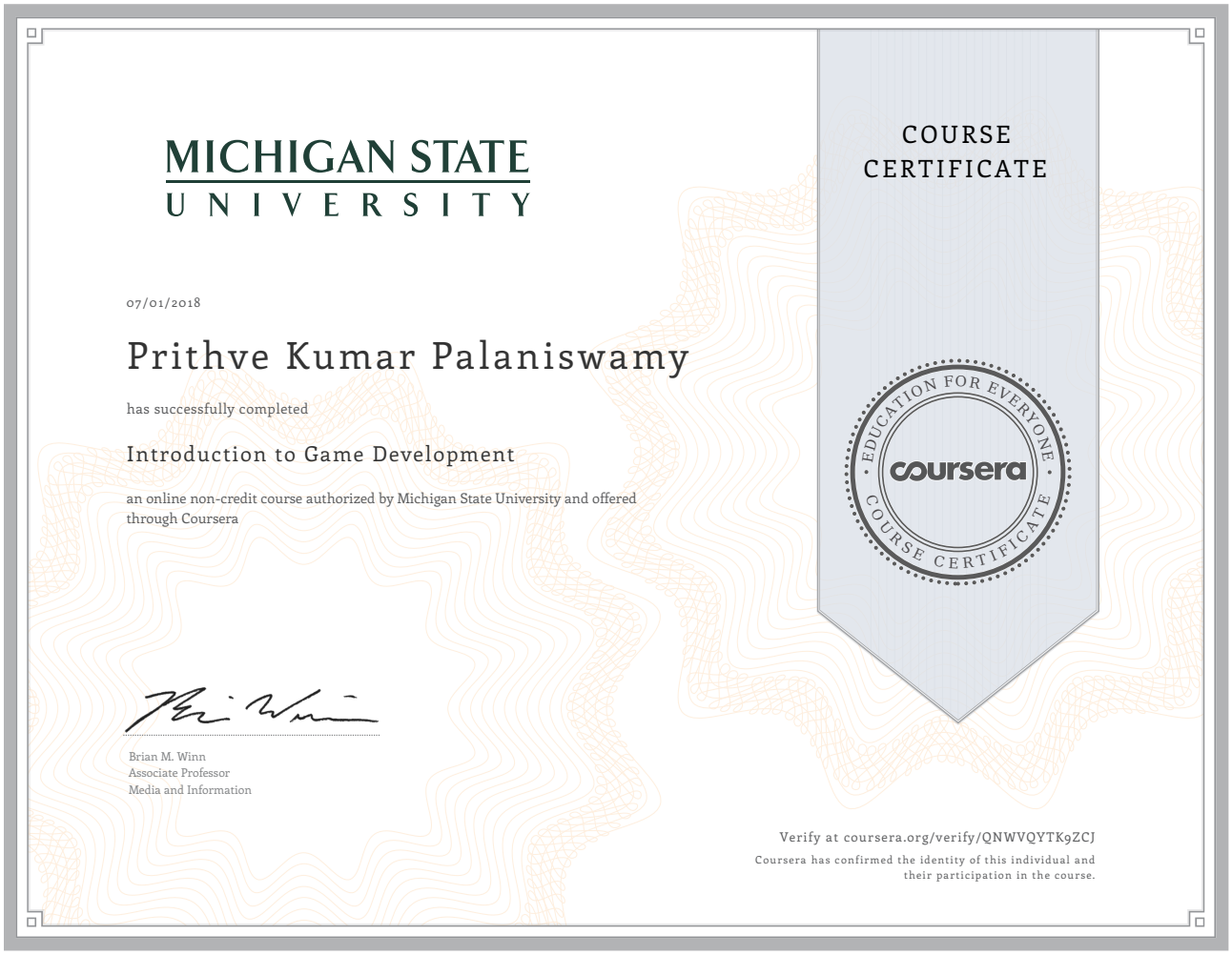


TABLE OF CONTENTS

INTRODUCTION

What is Game Development ?

Game development is the process of creating a video game. The effort is undertaken by a game developer, who may range from a single person to an international team dispersed across the globe.

The first video games were non-commercial, and were developed in the 1960s. They required mainframe computers to run and were not available to the general public.

Developments.

- Game AI
- Game design
- Interaction design
- Game programming
- Level design
- Game art
- Game design
- Game graphics
- Game music

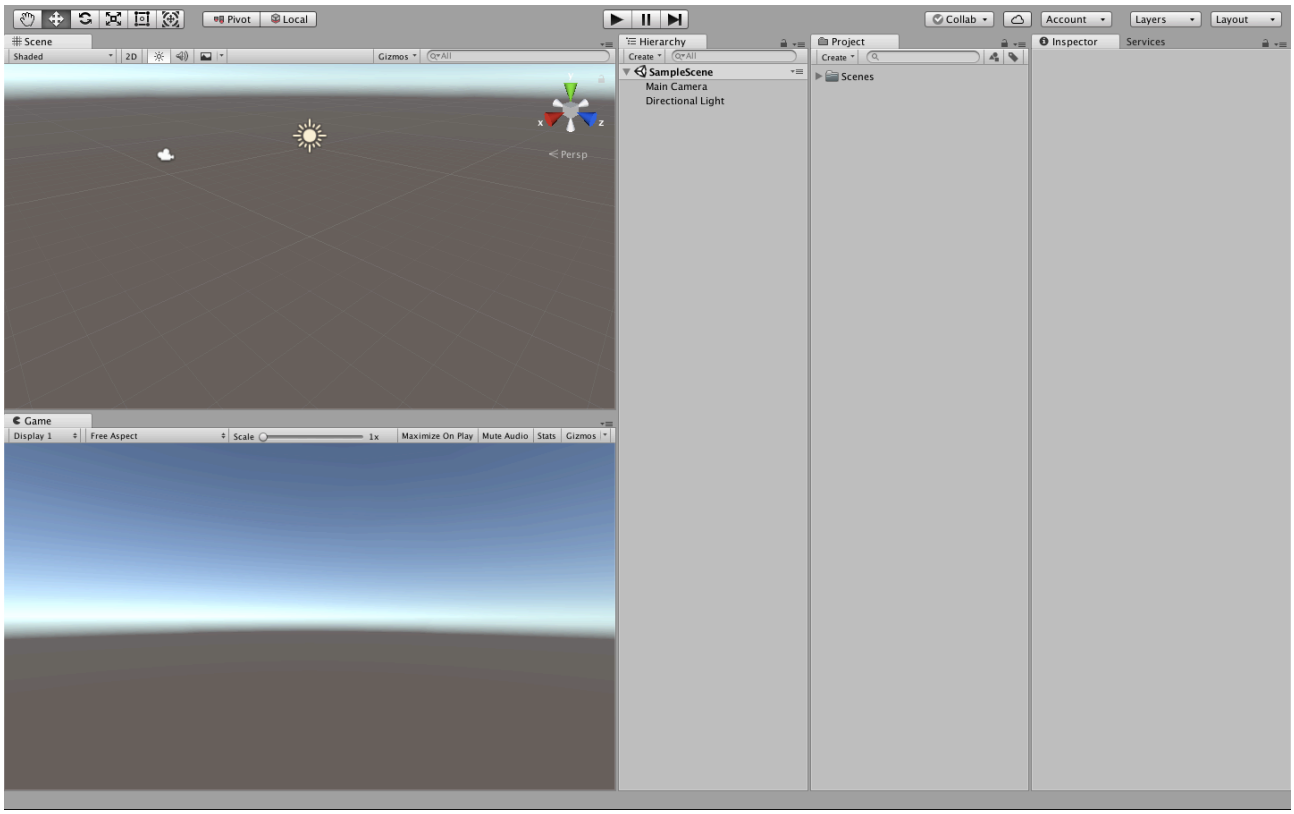
Types Of Game Developments

- Adult game
- Casual game
- Non-game
  - Interactive art
  - Interactive movie
- Nonviolent video game
- Serious game
  - Advergame
  - Art game
  - Edugame

TECHNOLOGY LEARNT

(Game Development)

I have done my training on Game Development using the software UNITY. I have developed four games(working) for MacOS.From this training I have learnt to use UNITY to create a wide range of games and I have also learnt C# which was used to code in game development.



The three games are:

1. First is based on Solar System
2. Second two games are based on RollerMadness.
3. The third is based on Box Shooter.

The course, Introduction to Game Development was done in Coursera. This course was was made by MICHIGAN STATE UNIVERSITY. This course was thought by Mr. Brain M. Winn. This was a four-six weeks course on game development and UNITY(Software) was used for the game development

PROJECT 1: **Solar System Interactive Art.**

PROJECT 2: **Roller Madness Casual game.**

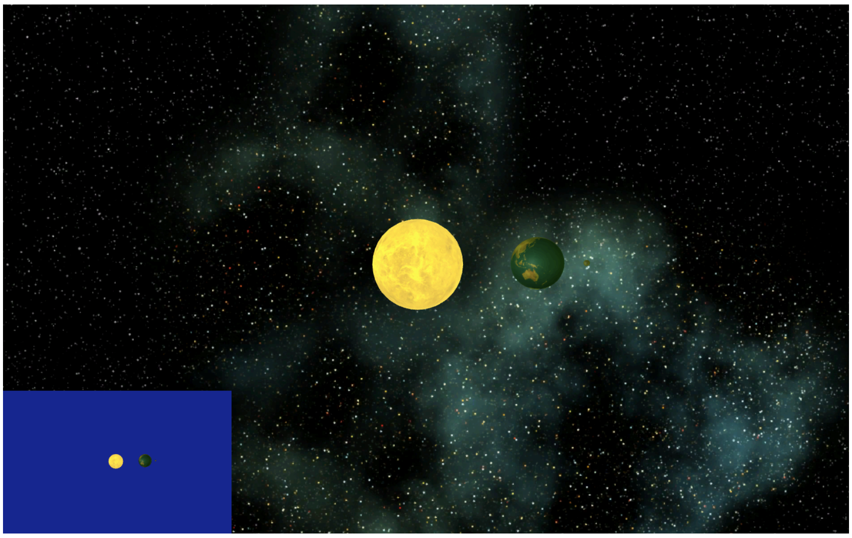
PROJECT 3: **Box Shooter Advergame**

All these games are made with UNITY software in various weeks of training. From project 1 to 4 there are various concepts thought by the trainer to develop these projects.

- ---

PROJECT 1

(Solar System)



Description:

In this project I have created a solar system with **SUN,EARTH and MOON.T**he Sun rotates itself and does not revolve around anything. The Earth revolves around the Sun and rotate around itself. The Moon revolves around the Earth and rotates around itself.

Concepts needed:

1.Creating Game Objects.
2. Adding materials to Game Objects.
3.Creating Skybox material (world).
4.Adding BEHAVIOURS to the Game Object.(scripts)
5. Adding Change-look-at-target script.
5.Adding Ambient Light
6.Creating a point Light to sun
7.Adding Audio source to the Camera.
8.Creating Mini-map Camera.
9.Build Settings.

Procedure:

1.**Open UNITY** and create new Project.
2. **Create GameObjects** like SUN,MOON,EARTH.(3D Sphere).
3.**Import Assets** to the UNITY(Scripts,Sound).

*****(ADD SPEED TO ALL ROTATE-AROUND SCRIPT)*****

4.Add **RotateAround** script to Sun and set the **target=Sun game object. (**rotation of sun)

5.Add **RotateAround** script to Earth and set the **target=Sun game object. (**revolution of Earth around Sun**).**

6.**Again add RotateAround script** to Earth and set the **target=Earth game object. (**rotation of Earth**).**

***7.Make the Moon child of Earth** so that Moon moves around the Sun along with the Earth.

8. Add **RotateAround script** to Moon and set the **target=Moon game object. (**rotation of Moon**).**

9.Again add RotateAround Script to Moon and set **target=Earth game object. (**Revolution of Moon around Earth).

10. **Create and add a skybox** for the world.
(Right click in project window- - -Material- - -Shader- - -Skybox)- - - - > creation.
(Window- - -Lighting- - -Settings- - -Scene- - -Environment- - -Skybox Material- - - - >add.

11.**Add materials to the Game Objects** from the Assets.
12.Add **Ambient Light** to the world (window- - -Lighting- - -settings- - -scene).
13.Add **Point Light** to Sun.(Component- - -Light- - -Point Light).
14.Add **Change-Look-At-Target script** to Sun.Earth,Moon.
15.Add **Audio Source to GameObjects** and **Audio Listener to main Camera.**
16.Create Mini-map Camera(GameObject- - -Camera- - -set it above the Main camera- - - change the viewport rect and Size).

RotateAround Script:

```
using UnityEngine;
using System.Collections;

public class RotateAround : MonoBehaviour {

    public Transform target; // the object to rotate around
    public int speed; // the speed of rotation

    void Start() {
        if (target == null)
        {
            target = this.gameObject.transform;
            Debug.Log ("RotateAround target not specified. Defaulting to parent GameObject");
        }
    }

    // Update is called once per frame
    void Update () {
        // RotateAround takes three arguments, first is the Vector to rotate around
        // second is a vector that axis to rotate around
        // third is the degrees to rotate, in this case the speed per second
        transform.RotateAround(target.transform.position,target.transform.up,speed * Time.deltaTime);
    }
}
```

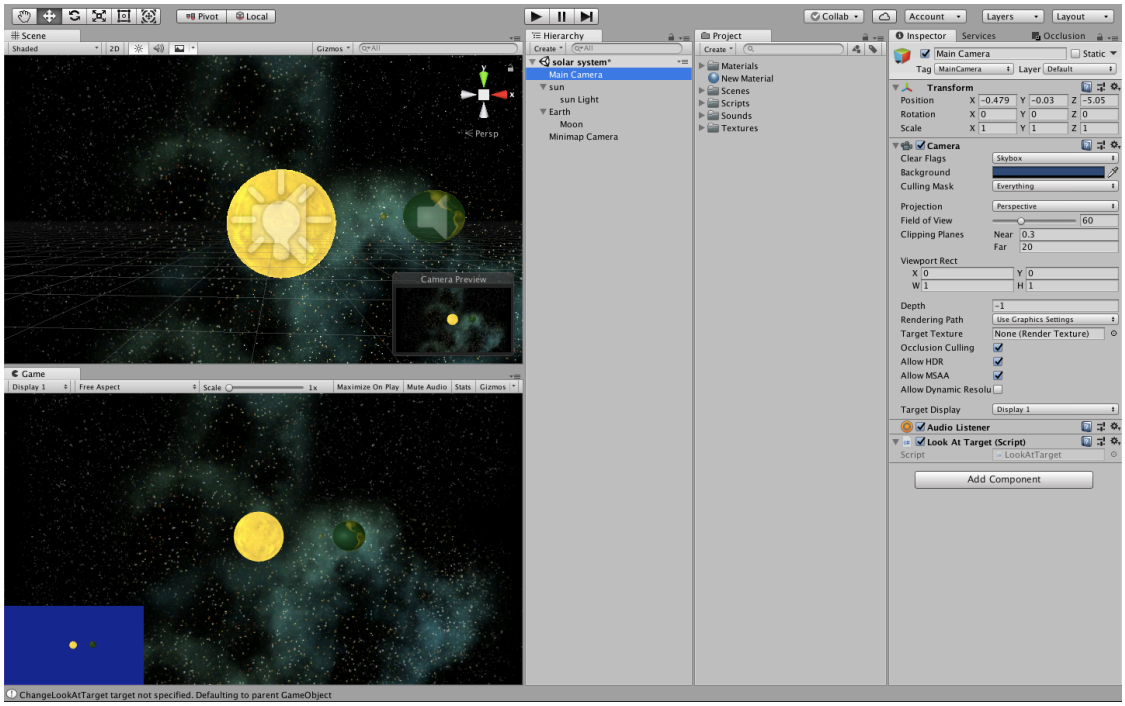
Change-Look-At-Target Script:

```
using UnityEngine;
using System.Collections;

public class ChangeLookAtTarget : MonoBehaviour {

    public GameObject target; // the target that the camera should look at

    void Start() {
        if (target == null)
        {
            target = this.gameObject;
            Debug.Log ("ChangeLookAtTarget target not specified. Defaulting to parent GameObject");
        }
    }

    // Called when MouseDown on this gameObject
    void OnMouseDown () {
        // change the target of the LookAtTarget script to be this gameobject.
        LookAtTarget.target = target;
        // change the field of view on the perspective camera based on the distance from center of world,
        clamp it to a reasonable field of view
        Camera.main.fieldOfView = Mathf.Clamp(60 * target.transform.localScale.x, 1, 100);
    }
}
```

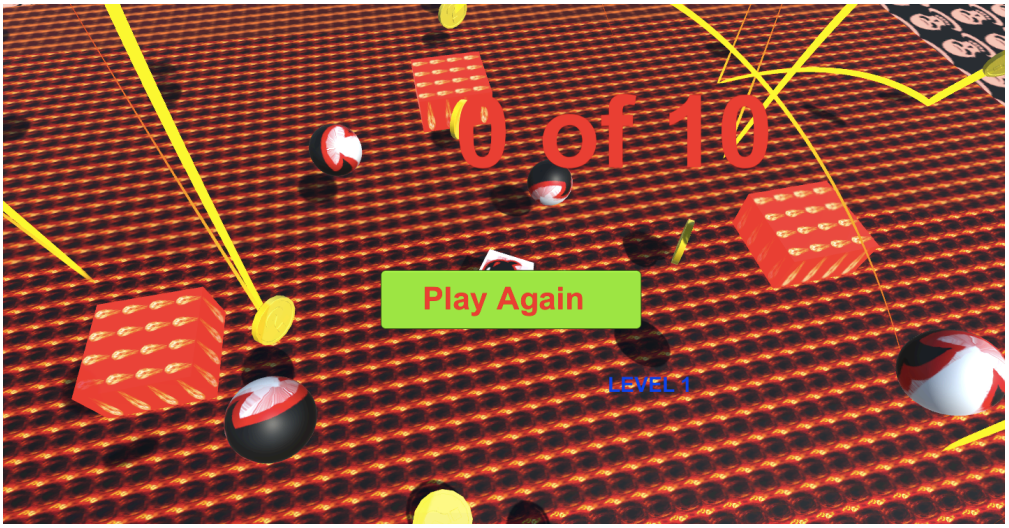
Save it as a Scene.

At last build the project (File- -Build Settings).

We can build it in any form like MacOS,Windows,Linux etc.



Project 2 and 3
(Roller Madness\RollandRoll)



Description:

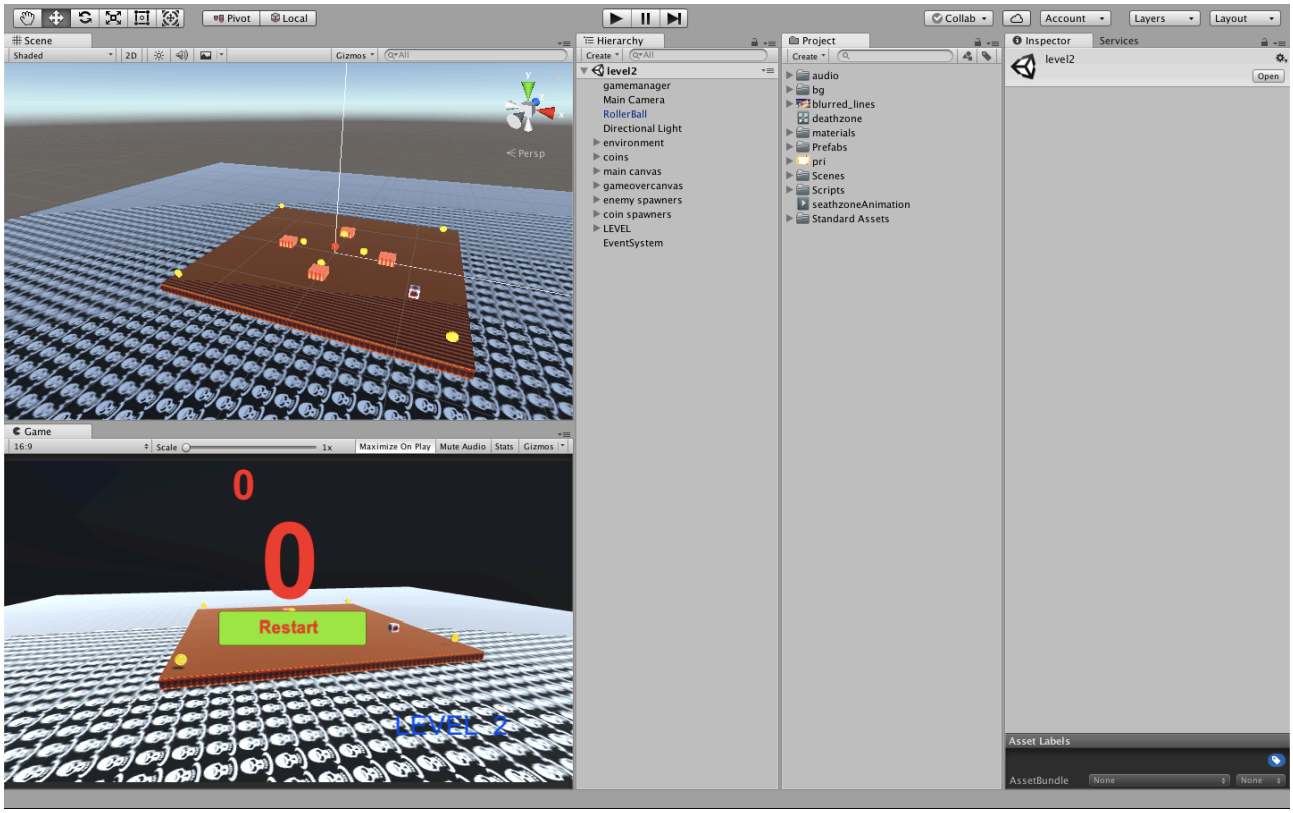
In this game, the theme is RollerMadness(lot of coins and enemies).I have created two levels. The first level is a bit easier(score needed to cross the level is less).the second level becomes tough as the score needed is more. The game has bumpers all around which makes the ball to deflect.We have the enemy spawners and coin spawners which creates a lot of enemies and coins. The player has to collect the minimum number of coins to move to the next level.

Concepts Needed:

1.Setting up the Scene.
2.Setup the Camera in the Scene.
3.Adding Physics.
4.Using trail Render to player.
5.DeathZones and Health
6.Add Pickups in game.
7.Create basic UI.
8.Game manager.
9.Adding Enemies.
10.Particle Systems(explosion).
11.Spawners in game(coin and enemies).

Procedure:

Please refer the procedure separately. Thank you.



Scripts:

Chaser
Damage
Health
Spawn-Game-Objects
Treasure
Game Manager

Game Manager Script:

```
using UnityEngine;
using System.Collections;
using UnityEngine.UI;
public class GameManager : MonoBehaviour {
	public static GameManager gm;
	[Tooltip("If not set, the player will default to the gameObject tagged as Player.")]
	public GameObject player;
	public enum gameStates {Playing, Death, GameOver, BeatLevel};
	public gameStates gameState = gameStates.Playing;
	public int score=0;
	public bool canBeatLevel = false;
	public int beatLevelScore=0;
	public GameObject mainCanvas;
	public Text mainScoreDisplay;
	public GameObject gameOverCanvas;
	public Text gameOverScoreDisplay;
	[Tooltip("Only need to set if canBeatLevel is set to true.")]
	public GameObject beatLevelCanvas;
	public AudioSource backgroundMusic;
	public AudioClip gameOverSFX;
	[Tooltip("Only need to set if canBeatLevel is set to true.")]
	public AudioClip beatLevelSFX;
	private Health playerHealth;
	void Start () {
		if (gm == null)
			gm = gameObject.GetComponent<GameManager>();
		if (player == null) {
			player = GameObject.FindWithTag("Player");
		}
		playerHealth = player.GetComponent<Health>();
		// setup score display
		Collect (0);
		// make other UI inactive
		gameOverCanvas.SetActive (false);
		if (canBeatLevel)
			beatLevelCanvas.SetActive (false);
	}
	void Update () {
		switch (gameState)
		{
			case gameStates.Playing:
```

```
                        if (playerHealth.isAlive == false)
                        {
                                // update gameState
                            gameState = gameStates.Death;
                                // set the end game score
                            gameOverScoreDisplay.text = mainScoreDisplay.text;
                                // switch which GUI is showing
                            mainCanvas.SetActive (false);
                            gameOverCanvas.SetActive (true);
                        } else if (canBeatLevel && score>=beatLevelScore) {
                                // update gameState
                            gameState = gameStates.BeatLevel;
                                // hide the player so game doesn't continue playing
                            player.SetActive(false);
                                // switch which GUI is showing
                            mainCanvas.SetActive (false);
                            beatLevelCanvas.SetActive (true);
                        }
                        break;
                    case gameStates.Death:
                        backgroundMusic.volume -= 0.01f;
                        if (backgroundMusic.volume<=0.0f) {
                            AudioSource.PlayClipAtPoint
                                (gameOverSFX,gameObject.transform.position);
                            gameState = gameStates.GameOver;
                        }
                        break;
                    case gameStates.BeatLevel:
                        backgroundMusic.volume -= 0.01f;
                        if (backgroundMusic.volume<=0.0f) {
                            AudioSource.PlayClipAtPoint
                                (beatLevelSFX,gameObject.transform.position);
                            gameState = gameStates.GameOver;
                        }
                        break;
                    case gameStates.GameOver:
                        // nothing
                        break;
                }
        }
        public void Collect(int amount) {
                score += amount;
                if (canBeatLevel) {
                 mainScoreDisplay.text = score.ToString () + " of "+beatLevelScore.ToString();
                }
                else
                 {
                    mainScoreDisplay.text = score.ToString ();
                }

        }
}
-  ------------------------------------------------------------------
```

Project 4
(Box Shooter)

Description:

In this game, the theme is Box Shooter(shooting the moving targets) . The game has limited time for each level(varies with level). The time varies as the target,(time increases/decreases based on the target) The player has to shoot the targets within the given time and have to collect the minimum needed score to cross the level. The level 2 becomes difficult as the time given to the player is less ,have to collect more score and the target varies(the score might increase or decrease depends on the target).

Concepts Needed:

1.Create moveable targets.
2.Create player.
3.Create Projectile for shooting.
4.Create UI and Game Manager

Procedure:

Please refer the procedure separately. Thank you.

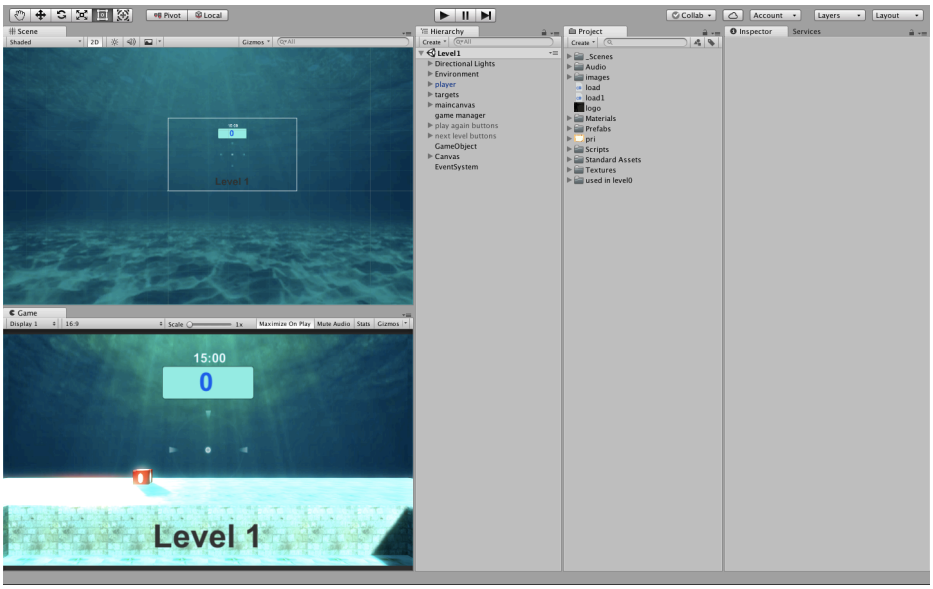
TargetMover script.

```
using UnityEngine;
using System.Collections;
public class TargetMover : MonoBehaviour {
	// define the possible states through an enumeration
	public enum motionDirections {Spin, Horizontal, Vertical};
	// store the state
	public motionDirections motionState = motionDirections.Horizontal;
	// motion parameters
	public float spinSpeed = 180.0f;
	public float motionMagnitude = 0.1f;
	// Update is called once per frame
	void Update () {
		// do the appropriate motion based on the motionState
		switch(motionState) {
			case motionDirections.Spin:
				// rotate around the up axix of the gameObject
			    gameObject.transform.Rotate (Vector3.up * spinSpeed *Time.deltaTime);
				break;
			case motionDirections.Vertical:
				// move up and down over time
				gameObject.transform.Translate(Vector3.up *
					Mathf.Cos(Time.timeSinceLevelLoad) * motionMagnitude);
				break;
			 case motionDirections.Horizontal:
				  // move up and down over time
				gameObject.transform.Translate(Vector3.right *
					Mathf.Cos(Time.timeSinceLevelLoad) * motionMagnitude);
				 break;
		}
	}
}
```

Controller script.

```
using UnityEngine;
using System.Collections;
public class Controller : MonoBehaviour {
	// public variables
	public float moveSpeed = 3.0f;
	public float gravity = 9.81f;
	private CharacterController myController;
	// Use this for initialization
```

```
        void Start () {
                // store a reference to the CharacterController component on this gameObject
                // it is much more efficient to use GetComponent() once in Start and store
                // the result rather than continually use etComponent() in the Update function
                myController = gameObject.GetComponent<CharacterController>();
        }
        // Update is called once per frame
        void Update () {
                        // Determine how much should move in the z-direction
                Vector3 movementZ = Input.GetAxis("Vertical") * Vector3.forward * moveSpeed *
                                                                                Time.deltaTime;
                        // Determine how much should move in the x-direction
                Vector3 movementX = Input.GetAxis("Horizontal") * Vector3.right * moveSpeed *
                                                                                Time.deltaTime;
                  // Convert combined Vector3 from local space to world space based on the.
                                                        position of the current gameobject (player)
                Vector3 movement = transform.TransformDirection(movementZ+movementX);
                        // Apply gravity (so the object will fall if not grounded)
                movement.y -= gravity * Time.deltaTime;
                        //Debug.Log ("Movement Vector = " + movement);
                        // Actually move the character controller in the movement direction
                myController.Move(movement);
        }
}
```



Reason for choosing this Technology:

1. **Create 2D Games Easily**

Unity 3D is known for 3D liveliness, but then it is also very effective in 2D game development for PCs, mobiles and even for gaming consoles. With Unity 4.3, an implicit 2D motor helps game developers to develop effective and efficient 2D games. One can easily do sprite activity, implement the physical science of a 2D world, and do much more.

2. **Create Multiplayer Games**

Some of the best multiplayer games are made on the Unity engine. The fact is that the platform provides versatile features that aid developers to create mind-boggling games for players all across the web. One can check out the coolest multiplayer games like the Solstice Arena that have earned a large following and popularity. The game development process is a monstrous under-taking and Unity gives the backing of a huge group to deliver popular multiplayer amusement as the developers provide it.

3. **Online Tutorials**

The best part about Unity is that there are lots of tutorials and training videos available on the web for anybody who is keen to learn. One can even get small and simple games done with training although it will soon prove to be simple for those who have been in the development industry since long.

4.**The Asset Store**

The Unity Asset Store provides developers with all the requirements for creating your game without letting you get stuck while making characters, buildings, backdrops and so forth. The store also provides the opportunity for craftsmen, musical artists, or modelers to earn more money.

5.**It supports multiple platforms**

Unity 3D supports iOS, Android, Macs, PCs, Steam, and even the consoles too. You can literally create several games for all platforms and design several stages of the games easily. With Unity, porting to the next stage is less difficult than the others. Regardless, you can put every stage's interesting elements into the next one and make it more interesting than before.

6.**The Ease of Use**

Although some might presume that Unity game development is incredibly difficult, one should understand that the platform is not complex in itself. It is easy to use without the help of a supervisor. The richness of ideas and gameplay can be realized easily after one is aware of the platform features.

7.**Scripting Languages**

You can script for Unity 3D game development with the help of Javascript or C#, the scripting languages that are not very difficult to use.

8.**It's free to use**

Unity 3D includes a free version and also a Proform, with several features embedded in both installment choices. One can opt for the free form so that one can have major gaming feature highlights. Additionally, one can utilize the distinct focal points with the Proform once you develop games with high-end features like sound channel, feature playback, 3D composition booster etc. Unity permits all sorts of developers to make full version games without having to rev up costs of any sort.

Learning Outcome:

1. Design gaming environment and levels.
2. Can develop games in multiple platforms (MacOS,Android).
3. Conceptualize and create 2D and 3D artwork for use in games.
4. Able to understand the Scripting using C#.
5. Develop, debug code to meet design specifications for games.
6. Choose game strategies and patterns based on an analysis of past and present trends.
7. Learnt many concepts include Skybox,Trail render,Game Manager.
8. Learnt a variety of scripts that are used in the game development.
9. Able to develop game from beginning to the publishing of game.
10. Recognise and Evaluate the problems in Game development.

Bibliography

The project Description in the report are purely written by myself and are not copied from any website . The Images that are used in the report are the screen shots of my project and the scripts are used in the project only. Few contents like “Reason for choosing this technology” is referred from google and no other content is got from google.

Work Division

DESCRIPTION	WEEK1	WEEK2	WEEK3	WEEK4
Project 1	30	100	0	0
Project 2 and 3	0	40	100	0
Project 4	0	0	40	100

Table 1-1

Description	Week1	Week 2	Week 3	Week 4
Project 1	Basic knowledge	Solar System Project	——	——-
Project 2	Basic knowledge	——-	Roller Madness Project	——-
Project 3	Basic knowledge	——-	——	Box Shooter Project

Table 2

Description	Week 1	Week 2	Week 3	Week 4
Project 1				
Project 2				
Project 3				

- - - - -END OF REPORT - - - - - - - -