Beach Ball Report
Unity Game
CA2 – Multimedia Programming

Submitted by:
Sylwia Calka

N00146095

30th March 2017

Introduction

This report discusses the design and implementation of the game made with Unity. This game is called Beach Ball (BB) and was developed as a part of Multimedia Programming Module. BB is a 3D environment game where the user is interacting with a ball. Unity is a cross-platform game engine used to develop PC games, mobile games and simulations.

Features of BB game:

1. Navigated by the user
2. Models are organised into prefabs
3. Textures and physical properties are added to models
4. Terrain is created with Terrain editor tools
5. Skybox
6. Collision detection between the player and objects
7. Lighten environment with baked lighting into the final textures
8. Sound
9. Particle effects
10. User interaction with pickup object

Mission of the game

The mission of the game is to collect all the pickups and gain as many points as possible. The ball needs to omit many obstacles in a form of a sea, ramps, trees and barrels. The game is finished when the user reaches the end of the level or falls into the sea.

Design and Implementation of the game environment

Game base

The basis for the game comes from a tutorial on the official Unity website [1]. This tutorial creates a simple rolling ball game in the 3D environment. It uses the shapes defined in Unity such as squares for pickups. The scene is made on a simple plane with walls around it. The player rolls a ball with key arrows.

Game interface

The environment for BB is set up on the platform which is a beach. The player starts from a portal where he is transported to from another level. On the platform, the terrain was built with the terrain tools. The material and shape imitate the real sand. On this terrain, palm trees are placed. The assets were downloaded from Unity Asset Store. Those assets include coconuts, treasure chest and bridge. The models for portals and pyramid were downloaded from SketchUp 3D Warehouse [2]. Those models had to be converted into 3D models with extension .dae in SketchUp software [3] since Unity does not recognise initial extension.

Light and Skybox

To lighten the scene Directional Light object was implemented. This object generates a beam of light. A skybox is a panoramic texture drawn behind all objects in the scene [4]. This box represents a sky and is made of six textures for each side of the sky (up, down, left, right, forward and backwards). In this game, the textures for skybox were downloaded from Unity Asset Store and they are images of a blue sky on a sunny day.

Sound effect

The sound was added to BB to improve the audial experience when playing the game. Every time the user collects a pick up the sound is played.

Prefabs

Assets which are reusable game objects are called prefabs. Those assets can be inserted into scenes multiple times. When prefab is added to the scene the instance of the object is created and all characteristics of the object are copied. If prefab is edited the changes are applied to all the instances.

Code

Scripts were written in C# language. There are four scripts in this game and they relate to objects in the game.

Player Controller

Player Controller is responsible for controlling the ball object. To initialise the game the Rigid body is attached to the ball and number of points is set to zero. To move the character the key arrows are used. When the arrows are pressed, the vectors are added to the Rigid body as a force and the ball object moves across the screen. Function OnTriggerEnter checks for collisions between ball and other objects. If the collision between the ball and an object with a tag “Pickup” occurs, points are added and displayed in the left top corner and pickup object is destroyed. Below is the screenshot of the Player Controller script.

```
public class PlayerController : MonoBehaviour {

    public float speed;
    public Text countText;
    private Rigidbody rb;
    private int count;
    public AudioSource audioSource;
    //initialization
    void Start () {
        rb = GetComponent<Rigidbody>();
        count = 0; // points
        SetCountText(); //setting text
    }
    void FixedUpdate() {
        //move charactes, speed can be adjusted in player controller
        //use this input to add forces to the rigidbody and move the player game object in the scene.
        float moveHorizontal = Input.GetAxis("Horizontal");
        float moveVertical = Input.GetAxis("Vertical");
        //public void AddForce(Vector3 force, ForceMode mode = ForceMode.Force);
        //The X, Y, Z values will determine the direction of the force added to ball
        //0.0f - we dont want to move up
        Vector3 movement = new Vector3(moveHorizontal, 0.0f, moveVertical);
        rb.AddForce(movement * speed);
    }
    // collision with other objects
    void OnTriggerEnter(Collider other) {
        //Destroy(other.gameObject);
        if (other.gameObject.CompareTag("Pickup"))
        {
            audioSource = GetComponent<AudioSource>();
            audioSource.Play();
            other.gameObject.SetActive(false);
            count = count + 1;
            SetCountText();
        }
    }
    void SetCountText() {
       countText.text = "Points: " + count.ToString();
    }

}
```

Figure 1 Player Controller Script

Rotator

Rotator is a short script attached to the pickup object. It is responsible for rotating an object constantly. It transforms a position of the object at Vector3(15, 30, 45) once per frame. Below is the screenshot of the Rotator script.

```
using UnityEngine;
using System.Collections;

public class Rotator : MonoBehaviour {
    // Update is called once per frame
    void Update () {
        transform.Rotate(new Vector3(15, 30, 45) * Time.deltaTime);
    }
}
```

Figure 2 Rotator Script

Camera Controller

Camera Controller is a script which is responsible for the camera to follow the player. Once per frame, when the Update function is called the position of the camera is updated and it is equal to the position of the player plus offset. Below is the screenshot of the Camera Controller script.

```
using UnityEngine;
using System.Collections;

public class CameraController : MonoBehaviour {

    public GameObject player;
    private Vector3 offset;

    // Use this for initialization
    void Start () {
        offset = transform.position - player.transform.position;
    }

    // Update is called once per frame after all items were processed
    void LateUpdate () {
        transform.position = player.transform.position + offset;
    }
}
```

Figure 3 Camera Controller Script

Game Over

Game Over script uses OnCollisionEnter method which detects when the ball collides with the water. This script is attached to the sea object. When a collision is detected, Restart function is called. In this function, the scene is reloaded which means that the game is over and the game starts all over again. Script below shows Game Over script.

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

public class GameOver : MonoBehaviour {
    public Text gameOver;
    public int count;

    void Start()
    {
        gameOver.text = "";
    }

    void OnCollisionEnter()
    {
        Invoke("Reload", 1.59f);
        gameOver.text = "GAME OVER";
    }
    void Reload()
    {
        Application.LoadLevel(Application.loadedLevel);
    }
}
```

Figure 4 Game Over Script

Game Screenshots

Images below shows the screenshots of Beach Game.

Figure 5 BB Screenshot 1

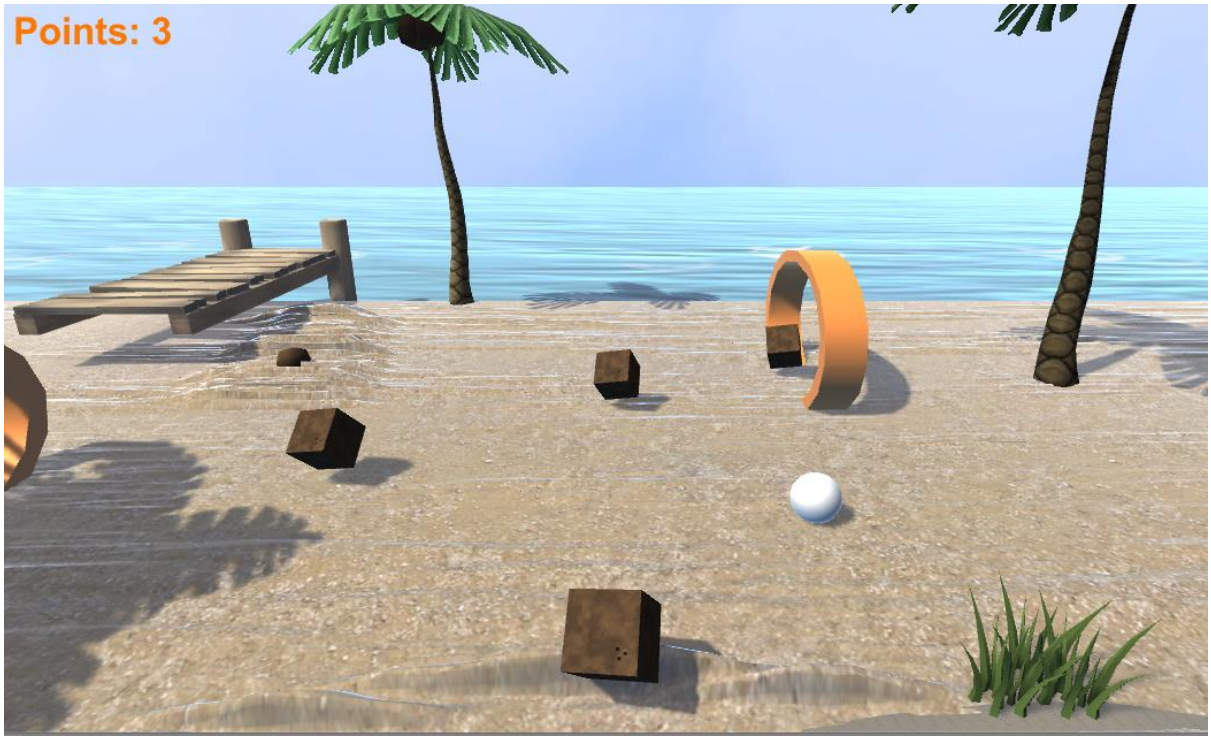

Figure 6 BB Screenshot 2

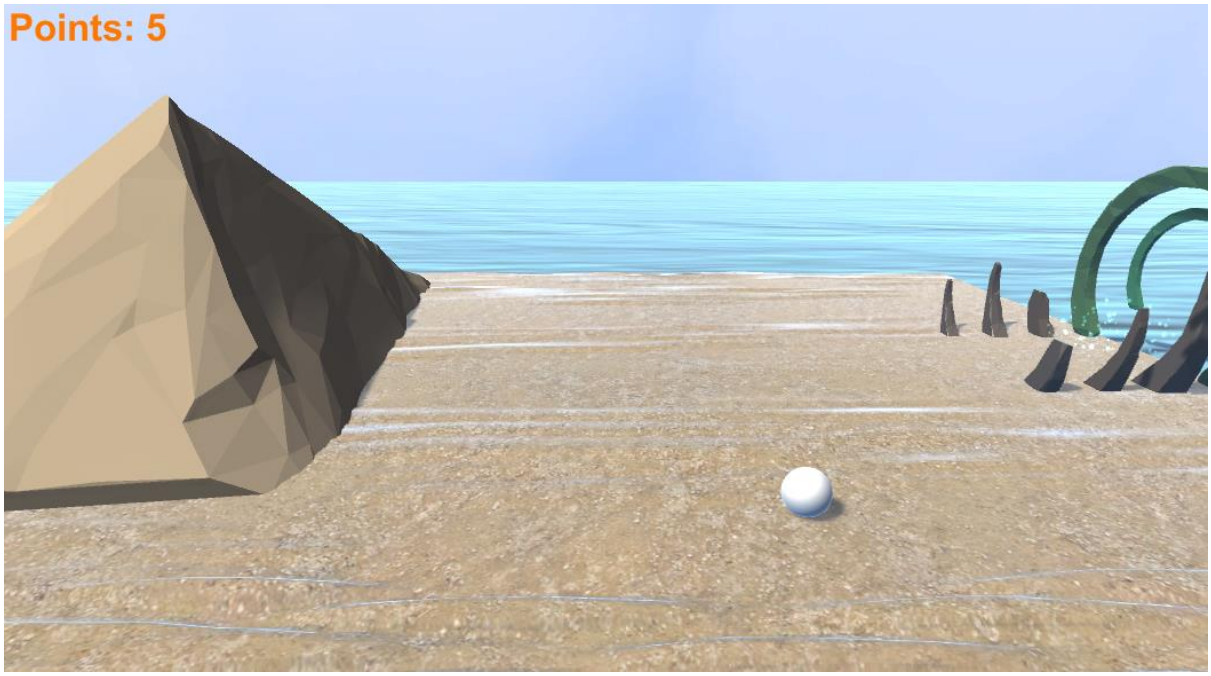

Figure 7 BB Screenshot 3

Bibliography

[1] Unity, “Roll-a-ball tutorial,” 2016. [Online]. Available: https://unity3d.com/learn/tutorials/projects/roll-ball-tutorial.

[2] “Models,” *SketchUp 3D Warehouse*. [Online]. Available: https://3dwarehouse.sketchup.com/.

[3] “SketchUp,” *SketchUp*. [Online]. Available: https://www.sketchup.com.

[4] “Using Skyboxes,” *Unity*. [Online]. Available: https://unity3d.com/learn/tutorials/topics/graphics/using-skyboxes.