CS223

Code Review

PROJECT-2

Landslide simulation

GROUP 8

B.Manoj Reddy --->160101020
M.Bhargav --->160101040
B.Tushara Langulya --->160101019

1.Introduction:-

This document contains the code review of our Landslide Simulator.The code review is performed after the module is compiled successfully and all syntax errors have been eliminated.

In general two types of code reviews can be carried out:

1.Code inspection

2. Code Walk through.

Here we are performing Code Inspection.The aim of code inspection is to discover some common types of errors caused due to oversight and improper programming.Adherence to coding standards is also checked during code inspection.

In this code inspection each member goes through code to discover some common types of errors caused due to oversight and improper programming.The errors which are majorly concentrated by the reviewers are:-

- Use of uninitialized variables.
- Incompatible assignments.
- Array indices out of bounds.
- Whether loops used are terminating or not
- Correct data types are being passed or not

2.Code

2.1Module:-Head tracker

```
/*10
Name of the module: Head tracker
Date on which the module is created: 10/4/18
Author of the module:Manoj Reddy
Modification History: By Bhargav Mallala 13/4/18
                      By Balabolu Tushara Langulya 15/4/18
Synopsis of the module : This module is executed when the user orients his head to look around
Name of function: 1)Update : output: rotation based on the input angles
                             input: Axis of rotation,rotationY angle and rotationX angle
                  2)Start: output:system gets ready to start rotation
                           input:No input
Global Variables:No global variables used in this module
 */

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

public class MouseLook : MonoBehaviour {
    // we have 3 possible RotationAxes with both x and y axes or only x axis or only y axis
    public enum RotationAxes { MouseXAndY = 0, MouseX = 1, MouseY = 2 }
    public RotationAxes axes = RotationAxes.MouseXAndY;
    public float sensitivityX = 15F;
    public float sensitivityY = 15F;
    // the limits of angle with X axis min and max value respectively
    public float minimumX = -360F;
    public float maximumX = 360F;
    // the limits of angle with Y axis min and max value respectively
    public float minimumY = -60F;
```

```
    public float minimumY = -60F;
    public float maximumY = 60F;
    float rotationY = 0F;
    void Update()
    {
        if (axes == RotationAxes.MouseXAndY)
        {
            //Input.GetAxis will give the value of virtual Xaxis .By multiplying with sensitivity X we get its real angle
            // transform.localEulerAngles.y gives us the angle with respect to y axis in degrees
            float rotationX = transform.localEulerAngles.y + Input.GetAxis("Mouse X") * sensitivityX;

            rotationY += Input.GetAxis("Mouse Y") * sensitivityY;
            // clamps the rotationY between minimumY and maximumY
            rotationY = Mathf.Clamp(rotationY, minimumY, maximumY);
            // this gives us new angle  at which we view
            transform.localEulerAngles = new Vector3(-rotationY, rotationX, 0);
        }
        else if (axes == RotationAxes.MouseX)
        {

            transform.Rotate(0, Input.GetAxis("Mouse X") * sensitivityX, 0);
        }
        else
        {
            rotationY += Input.GetAxis("Mouse Y") * sensitivityY;
            rotationY = Mathf.Clamp(rotationY, minimumY, maximumY);
            // this gives us new angle  at which we view
            transform.localEulerAngles = new Vector3(-rotationY, transform.localEulerAngles.y, 0);
        }
```

```
        }
    }

    void Start()
    {
        // Make the rigid body not change rotation
        if (GetComponent<Rigidbody>())
            GetComponent<Rigidbody>().freezeRotation = true;
    }
}

```

2.2Module:-Virtual Movement

```
/*
Name of the module: Virtual movement
Date on which the module is created: 18/4/18
Author of the module:Manoj Reddy
Modification History: By Bhargav Mallala 19/4/18
                      By Balabolu Tushara Langulya 19/4/18
Synopsis of the module : This module is executed when the user orients his bhead at appropriate angles to move
Functions: 1.Start :void Start()
           2.void Update()
Global Variables: No global variables
 */

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

public class VRLookWalk : MonoBehaviour
{

    public Transform vrCamera;
    // this is the minimum angle reqired for the user to move
    public float toggleAngle = 30.0f;
    // speed with which user moves
    public float speed = 3.0f;
    public bool moveForward;

    private CharacterController cc;

    // Use this for initialization
    void Start()
    {
        cc = GetComponent<CharacterController>();
    }

    // Update is called once per frame
    void Update()
    {
        // if these conditions are not satisfied then there wont be any movement of user
        if (vrCamera.eulerAngles.x >=toggleAngle && vrCamera.eulerAngles.x < 90.0f)
        {
            moveForward = true;
        }
        else
        {
            moveForward = false;
        }

        if (moveForward )
        {
            // forward is used to store the direction in which the user moves which will depend on the direction he is facing
            Vector3 forward = vrCamera.TransformDirection(Vector3.forward);
            // move in the forward direction at the rate of speed
            cc.SimpleMove(forward * speed);
        }
    }
}
```

```
58
59  function Start ()
60  {
61    textMesh = gameObject.GetComponent(MeshRenderer);
62  }
63
64  //-------------------------------------------------------------------------------------
65  function Update ()
66  {
67    // if the danger zone then printing message to go away from the are
68    if (Vector3.Distance(transform.position, player.position) < showOnDistance)
69      textMesh.enabled = true;
70    //else not enabling the text message
71      else
72       textMesh.enabled = false;
73
74  }
```

2.3Module:-Scenario generator

```
1  /*
2  Name of the module: Scenario generator
3  Date on which the module is created: 13/4/18
4  Author of the module:Manoj Reddy
5  Modification History: By Bhargav Mallala 15/4/18
6                        By Balabolu Tushara Langulya 16/4/18
7  Synapsis of the module : This module is executed to generate the scenario of landslide
8  Functions: in WaterBase class
9                 1.  public void UpdateShader()
10                2.  public void WaterTileBeingRendered(Transform tr, Camera currentCam)
11                3.  public void Update()
12             in RainScript class
13                1.  private void UpdateRain()
14                2.  protected override void Start()
15                3.  protected override void Update()
16 Global Variables: 1.Shader used in waterbase class
17                   2.RainFallParticleSystem in RainScript class
18                   3.RainMistParticleSystem in RainScript class
19  */
20 using System;
21 using UnityEngine;
22 using System.Collections;
23 namespace UnityStandardAssets.Water
24 {
25     //indicating the water quality
26     public enum WaterQuality
27     {
28         High = 2,
29         Medium = 1,
```

```
        Low = 0,
    }

    [ExecuteInEditMode]
    public class WaterBase : MonoBehaviour
    {
        //the shared material of this object
        public Material sharedMaterial;

        //the waterquality is set to high
        public WaterQuality waterQuality = WaterQuality.High;
        public bool edgeBlend = true;


        public void UpdateShader()
        {

            //setting shader level of detail(LOD) for this shared materialbased on water quality
            //higher the water quality higher the maximumLOD
            if (waterQuality > WaterQuality.Medium)
            {

                sharedMaterial.shader.maximumLOD = 501;
            }
            else if (waterQuality > WaterQuality.Low)
            {
                sharedMaterial.shader.maximumLOD = 301;
            }
            else
```

```
            else
            {
                sharedMaterial.shader.maximumLOD = 201;
            }

            // If the system does not support depth textures (ie. NaCl), turn off edge bleeding,
            // as the shader will render everything as transparent if the depth texture is not valid.
            if (!SystemInfo.SupportsRenderTextureFormat(RenderTextureFormat.Depth))
            {
                edgeBlend = false;
            }

            if (edgeBlend)
            {
                Shader.EnableKeyword("WATER_EDGEBLEND_ON");
                Shader.DisableKeyword("WATER_EDGEBLEND_OFF");
                // just to make sure (some peeps might forget to add a water tile to the patches)
                if (Camera.main)
                {
                    Camera.main.depthTextureMode |= DepthTextureMode.Depth;
                }
            }
            else
            {
                //setting the globally shared keyword to "WATER_EDGEBLEND_ON"
                Shader.EnableKeyword("WATER_EDGEBLEND_OFF");

                //unsetting the globally shared keyword "WATER_EDGEBLEND_ON"
                Shader.DisableKeyword("WATER_EDGEBLEND_ON");
            }
```

```
                Shader.DisableKeyword("WATER_EDGEBLEND_ON");
            }
        }


        public void WaterTileBeingRendered(Transform tr, Camera currentCam)
        {
            if (currentCam && edgeBlend)
            {
                currentCam.depthTextureMode |= DepthTextureMode.Depth;
            }
        }


        public void Update()
        {
            if (sharedMaterial)
            {
                //shader is updated based on sharedMaterial
                UpdateShader();
            }
        }
    }
}


namespace DigitalRuby.RainMaker
{
    public class RainScript : BaseRainScript
    {
```

```
{
    public class RainScript : BaseRainScript
    {
      //The height above the camera that the rain will start falling from
        public float RainHeight = 25.0f;

      //How far the rain particle system is ahead of the player
        public float RainForwardOffset = -7.0f;

        //The top y value of the mist particles
        public float RainMistHeight = 3.0f;

        private void UpdateRain()
        {

            if (RainFallParticleSystem != null)
            {
                if (FollowCamera)
                {
                    var s = RainFallParticleSystem.shape;

                    //setting the shape of rainfall particle to Conevolume
                    s.shapeType = ParticleSystemShapeType.ConeVolume;
                    RainFallParticleSystem.transform.position = Camera.transform.position;

                    //translates the rainparticle along x,y,z axis in the values mentioned
                    RainFallParticleSystem.transform.Translate(0.0f, RainHeight, RainForwardOffset);

                    //applies a rotation of eulerAngles.y around the y-axis
                    RainFallParticleSystem.transform.rotation = Quaternion.Euler(0.0f,
                     Camera.transform.rotation.eulerAngles.y, 0.0f);

                    if (RainMistParticleSystem != null)
                    {
                        var s2 = RainMistParticleSystem.shape;

                        //setting the rain particle shape to hemisphericall
                        s2.shapeType = ParticleSystemShapeType.HemisphereShell;
                        Vector3 pos = Camera.transform.position;

                        //increasing the y-cordinate by RainMistHeight
                        pos.y += RainMistHeight;

                        //transforming the position to pos
                        RainMistParticleSystem.transform.position = pos;
                    }
                }
                else
                {
                    var s = RainFallParticleSystem.shape;
                    //setting the shape of the rain particle to box
                    s.shapeType = ParticleSystemShapeType.Box;
                    if (RainMistParticleSystem != null)
                    {
                        var s2 = RainMistParticleSystem.shape;

                        //setting the shape of the rain particle to box
                        s2.shapeType = ParticleSystemShapeType.Box;
                        Vector3 pos = RainFallParticleSystem.transform.position;
```

```
                        pos.y += RainMistHeight;
                        pos.y -= RainHeight;
                        RainMistParticleSystem.transform.position = pos;
                    }
                }
            }
        }

        protected override void Start()
        {
          //start the rainfall
            base.Start();
        }

        protected override void Update()
        {
            base.Update();
            UpdateRain();
        }
    }
}

```

2.4Module:-Menu

```
/*
Name of the module: Menu
Date on which the module is created: 20/4/18
Author of the module:Manoj Reddy
Modification History: By Bhargav Mallala 21/4/18
                      By Balabolu Tushara Langulya 21/4/18
Synapsis of the module : This module is executed to generate menu options
Functions : in Mainmenu class
        1.  public void RunScenario ()
        2.  public void QuitApp()
      in InGameMenu class
        1.  public void RestartScenario ()
        2.  public void QuitApplication()
Global Variables:No global variables used
 */
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class Mainmenu : MonoBehaviour {

  public void RunScenario ()
  {
    SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);
  }
  public void QuitApp()
  {
    Application.Quit();
  }
```

```
    Application.Quit();
  }
}
public class InGameMenu : MonoBehaviour {

  public void RestartScenario ()
  {
    SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);
  }
  public void QuitApplication()
  {
    Application.Quit();
  }
}
```

2.5 Special effects

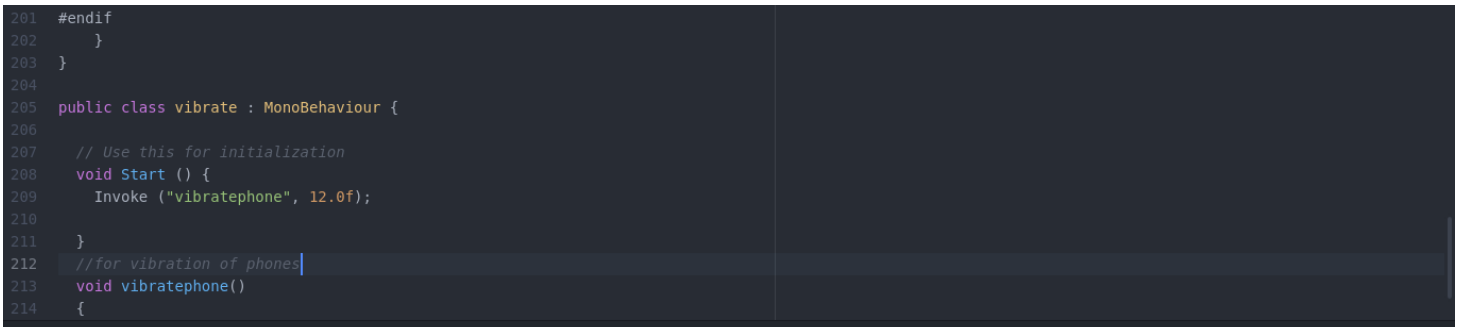
```
/*
Name of the module: Special effects
Date on which the module is created: 18/4/18
Author of the module:Manoj Reddy
Modification History: By Bhargav Mallala 19/4/18
                      By Balabolu Tushara Langulya 19/4/18
Synapsis of the module : This module is executed to generate sound and vibrations for the landslide
Functions: in TestAudioConfiguration class
               1.  void start()
               2.  void OnAudioConfigurationChanged(bool deviceWasChanged)
               3.  int GUIRow(string name, int[] valid, int value, ref bool modified)
               4.  void OnGUI()
            in Vibration class
               1.  public static void Vibrate()
               2.  public static void Vibrate(long milliseconds)
               3.  public static void Vibrate(long[] pattern, int repeat)
               4.  public static bool HasVibrator()
               5.  public static void Cancel()
               6.  private static bool isAndroid()
            in vibrate AndroidJavaClass
               1.void Start()
               2.void vibratephone()
Global Variables: 1.Shader used in waterbase class
                  2.RainFallParticleSystem in RainScript class
                  3.RainMistParticleSystem in RainScript class
 */
using UnityEngine;
using System.Collections;

public class TestAudioConfiguration : MonoBehaviour
```

```
using System.Collections;

public class TestAudioConfiguration : MonoBehaviour
{
    void Start()
    {
        AudioSettings.OnAudioConfigurationChanged += OnAudioConfigurationChanged;
    }

    void OnAudioConfigurationChanged(bool deviceWasChanged)
    {
        Debug.Log(deviceWasChanged ? "Device was changed" : "Reset was called");
        if (deviceWasChanged)
        {
            ///returns the current configuration of the audio device and System
            AudioConfiguration config = AudioSettings.GetConfiguration();
            config.dspBufferSize = 64;
            //resetting the dspbuffersize using config and then resetting
            AudioSettings.Reset(config);
        }
        //calls up the audio file
        GetComponent<AudioSource>().Play();
    }

    //defineing the various speaker mmodes
    static int[] validSpeakerModes =
    {

        (int)AudioSpeakerMode.Mono,
        (int)AudioSpeakerMode.Stereo,
        (int)AudioSpeakerMode.Quad,
        (int)AudioSpeakerMode.Surround,
        (int)AudioSpeakerMode.Mode5point1,
    };

      //the  buffersizes we intend to have are defined here
    static int[] validDSPBufferSizes =
    {
        32, 64, 128, 256, 340, 480, 512, 1024, 2048, 4096, 8192
    };

      //The sample rate setting used within the AudioImporter.
    static int[] validSampleRates =
    {
        11025, 22050, 44100, 48000, 88200, 96000,
    };

    static int[] validNumRealVoices =
    {
        1, 2, 4, 8, 16, 32, 50, 64, 100, 128, 256, 512,
    };

    static int[] validNumVirtualVoices =
    {
        1, 2, 4, 8, 16, 32, 50, 64, 100, 128, 256, 512,
    };

    int GUIRow(string name, int[] valid, int value, ref bool modified)
    {
```

```
        //begin a horizontal control group
        GUILayout.BeginHorizontal();
          //button with name=value
        GUILayout.Button(name + "=" + value);
        for (int i = 0; i < valid.Length; i++)
        {
            string s = valid[i].ToString();
            if (valid[i] == value)
                s = "[" + s + "]";
                 //if clicked on button GUILayout.Button(s) becomes true
            if (GUILayout.Button(s))
            {
                value = valid[i];
                 //if value is modified the modified becomes true
                modified = true;
            }
        }
         //ending the horizontal control group
        GUILayout.EndHorizontal();
        return value;
    }

    void OnGUI()
    {
        //storing the audio source in source
        AudioSource source = GetComponent<AudioSource>();
        bool modified = false;

        AudioConfiguration config = AudioSettings.GetConfiguration();
```

```
        //reconfiguring the values and appropraitely changing the modified values
        config.speakerMode = (AudioSpeakerMode)GUIRow("speakerMode", validSpeakerModes,config.speakerMode, ref modified);
        config.dspBufferSize = GUIRow("dspBufferSize", validDSPBufferSizes, config.dspBufferSize, ref modified);
        config.sampleRate = GUIRow("sampleRate", validSampleRates, config.sampleRate, ref modified);
        config.numRealVoices = GUIRow("RealVoices", validNumRealVoices, config.numRealVoices, ref modified);
        config.numVirtualVoices = GUIRow("numVirtualVoices", validNumVirtualVoices, config.numVirtualVoices, ref modified);

          //we reset if the values have changed
        if (modified)
            AudioSettings.Reset(config);

            //when the button pressed the audio starts playing
        if (GUILayout.Button("Start"))
            source.Play();

           //audio stops playing on pressing the Stop
        if (GUILayout.Button("Stop"))
            source.Stop();
    }
}

public static class Vibration
{

#if UNITY_ANDROID && !UNITY_EDITOR
    public static AndroidJavaClass unityPlayer = new AndroidJavaClass("com.unity3d.player.UnityPlayer");
    public static AndroidJavaObject currentActivity = unityPlayer.GetStatic<AndroidJavaObject>("currentActivity");
    public static AndroidJavaObject vibrator = currentActivity.Call<AndroidJavaObject>("getSystemService", "vibrator");
#else
```

```
    public static AndroidJavaClass unityPlayer;
    public static AndroidJavaObject currentActivity;
    public static AndroidJavaObject vibrator;
#endif

    public static void Vibrate()
    {
        //if device is android calling vibrator
        if (isAndroid())
            vibrator.Call("vibrate");
        //triggering the vibration
        else
            Handheld.Vibrate();
    }


    public static void Vibrate(long milliseconds)
    {
        //calling vibrator for milliseconds mentioned
        if (isAndroid())
            vibrator.Call("vibrate", milliseconds);
          //triggering the vibration
        else
            Handheld.Vibrate();
    }

    public static void Vibrate(long[] pattern, int repeat)
    {
        //calling vibrator for the pattern and repeating it
        if (isAndroid())
            vibrator.Call("vibrate", pattern, repeat);
        //  //triggering the vibration
        else
            Handheld.Vibrate();
    }

    public static bool HasVibrator()

    {
      //if the device is android it has a vibrator
        return isAndroid();
    }

    public static void Cancel()
    {
        //calling off the vibrator
        if (isAndroid())
            vibrator.Call("cancel");
    }

    private static bool isAndroid()
    {
#if UNITY_ANDROID && !UNITY_EDITOR
  return true;
#else
        return false;
#endif
    }
```



```
#endif
    }
}

public class vibrate : MonoBehaviour {

  // Use this for initialization
  void Start () {
    Invoke ("vibratephone", 12.0f);

  }
  //for vibration of phones
  void vibratephone()
  {
```

3.Code Review Team

3.1 Team Profile

The code testing team comprises of the following members, all of whom are Undergraduates currently pursuing Bachelor of Technology at Indian Institute of Technology Guwahati, India in the Department of Computer Science & Engineering.

1. Mukul Verma
2. Varun Kedia
3. Ekta Dhan

All the members in coding team are proficient in C# and have good coding experience.

4. CODE INSPECTION REPORTS

4.1 MODULES Reviewed BY THE TEAM

The modules reviewed by the team are:-

- Scenario generator
- Head tracker
- Virtual movements
- Menu
- Special Effects

4.2 REPORT BY EKTA DHAN:-

The conclusions I have found after going through the code have been reported in this document.

- The headers of each module had all the details that are required from a good header, like - Name of the module, Date on which the module was created, Author's name, Modification history, Synopsis of the module, Different functions supported,along with their input parameters.
- Proper indentation and proper naming of variables is maintained in all modules .
- There were none **uninitialized variables** found in any module.
- No **JUMP (go to)** statements were found in the module.
- There is scope of improvement in commenting of the code as the functionality of some functions are not explained properly and hence the complete functionality of those functions could not be understood properly.
- No mismatches between the actual and formal parameters in the code. The bool modified is the only parameter in the function OnGui which is passed by reference to the GuiRow in the TestAudioConfiguration class.
- A few modules have too few lines of codes like Menu and few modules have a large number of lines of codes. A better division of functions would have been better.
- All variables have been properly assigned to their values.
- Different modules have same function names like Update is present in both Virtual Movement module and Head Tracker

4.2 REPORT BY Varun Kedia:-

After inspecting the code , the observations made by him have been recorded.

- The description of functions is poor and can be done in a better way. Improvement is needed in this matter
- All the array references used in the code were in the bound of the array.
- Poor naming of the modules which needs to be improved.
- Modules Special Effects and Scenario generator has around 190 lines of code each. Better division of modules would be better.
- All the loops terminated according to their condition. No Non-Terminating Loops were found.
- The headers of each module had all the details that are required from a good header, like - Name of the module, Date on which the module was created, Author's name, Modification history, Synopsis of the module, Different functions supported,along with their input/output parameters.
- In the OnGui function of SpecialEffects module config.speakerMode is not an int type data but the parameter to be sent to GuiRow is an integer. Hence correct data type is not being sent
- Uninitialized variables = NIL found in all modules.
- Coding Style is very good with proper indentation being followed.
- All variables have been properly assigned to their values

4.3 REPORT BY Mukul Verma:-

- All the variables are named properly and are in lower camel case
- Loop variables are not modified whenever the loops are used
- Special Effects module and Scenario generator module has a length of 190 lines. This is not desirable as it increases difficulty in understanding the module functionality.
- Indentation was followed in a proper way
- Float data type has been sent to the GuiRow function instead of int data type
- All modules have proper header files which explain the module functionality properly
- It would have been better if more commenting was done to explain the functionality of certain functions like OnGui
- Non Terminating loops present

5. CONCLUSION

The members of the code review team submitted the reports during their final meeting with the development team. From these submitted reports, we get to know about a few changes which should be made to our system for better understanding and readability.

- In OnGui we are sending config.speakerMode to GUIRow . This shouldn't be done as it is float and the parameter of GuiRow is integer. This should be rectified
- Functions of many modules and within a module also share common names which is causing a confusion. Thus naming should be changed in better way.
- Some modules are having a large number of lines of code while some are having very few lines of code. Thus division of functions into modules should be changed for better understandability

By fixing these a the application will become more bug free and more accurate