CODE DOCUMENT

for

Virtual Museum

Theme : World War II

Version: 1.0
3th May 2017

Prepared by
Mayank Yadav (34), Harshit Bansal (26)
Nikunj Mittal (43)

IIT Guwahati

supervised by
Prof. Samit Bhattacharya

Contents

1 Introduction **1**
- 1.1 Purpose . . . 1
- 1.2 Document Conventions . . . 1
- 1.3 Project Scope . . . 2

2 Project Code **3**
- 2.1 Resources Used . . . 3
- 2.2 List Of Modules . . . 4
- 2.3 Code for each Module . . . 5
  - 2.3.1 Acceleration Input . . . 5
  - 2.3.2 Select-Object Input . . . 7
  - 2.3.3 Joystick Input . . . 9
  - 2.3.4 Pinch Zoom . . . 11
  - 2.3.5 Player Controller . . . 13
  - 2.3.6 Camera Controller . . . 14
  - 2.3.7 GUI List . . . 15
  - 2.3.8 Object Viewer . . . 16
  - 2.3.9 Object Rotate . . . 20
  - 2.3.10 Object Translate . . . 21
  - 2.3.11 Wish list Manager . . . 22
  - 2.3.12 Screen Resolution . . . 24
  - 2.3.13 Mini-Map . . . 26
  - 2.3.14 Scene Manager . . . 27
  - 2.3.15 Pause Module . . . 28

Revision History

Sno.	Date	Reason For Changes	Version
1	27/3/17	Original	1.0
2	5/3/17	Remarks from Prof. Bhattacharya	1.1

1 Introduction

1.1 Purpose

The purpose of Code Document for the Virtual Museum software is to provide well documented source code of the Project.

1.2 Document Conventions

Term	Definition
User	Person who shall be using the software to experience the Virtual Museum
Object	Exhibit for display
Player	User's avatar/character in the Virtual Museum
Device	An electronic device using which is used to run the software
UNITY	Unity is a cross-platform game engine developed by Unity Technologies.

1.3 Project Scope

This software will help users to visit the museums at their ease directly from their homes and see the artifacts of historical significance in a virtual 3-D interactive environment. The product will be very useful for enthusiasts who are interested about history but are unable to find time to visit the museums and also helpful for Archaeologists and professionals to have access to information from museums all the time. For people with no knowledge ,the easy to use app will act as an interesting way to learn about the museums and help them learn about the past.

2 Project Code

2.1 Resources Used

We have used the following resources in the process of making the Project.

Resource	Version	About
Unity	5.5.1	Platform used for Project Development
JDK Tools	1.8.0	Required for Building into Android
Android SDK	26.0	Command line tools integrated into Unity for Building Project into Android
C#		Programming Language Used
MonoDevelop-Unity	5.9.6	Development Environment used for programming in C#
Blender	2.78	Used for 3D Modeling

2.2 List Of Modules

Module Name	About
Acceleration Input	Gets the Accelerometer Values and processes them as Input for Camera Rotation and Camera Zoom
Select-Object Input	Uses Touch Input on screen to select object(exhibit) in world
Joystick Input	Virtual Joystick Functions
Pinch Zoom	Pinch Zoom Implementation - Zoom in/out is smoothe
Player Controller	Controls Player Movements including jump Movement
Camera Controller	Rotates Camera According to Acceleration Input Translates Camera According to Player Movement
GUI List	List Gameobject Functions [i.e Scrollable List in GUI]
Object Viewer	Manage all Object Viewer Functions
Object Rotate	Rotate the Object in Object Viewer
Object Translate	Move Object in Object Viewer in a bounded area
Wish list Manager	Manages All Wishlist and Catalogue Functions
Screen Resolution	Select Resolution of device
Mini-Map	Manage Mini Map#
Scene Manager	Manage Scene Loading
Pause Module	Pause the Tour - Time is frozen

2.3 Code for each Module

2.3.1 Acceleration Input

```
/*
 ACCELERATION INPUT MODULE.
 3/4/2017
 Mayank Yadav
 Synopsis - Gets the Accelerometer Values and processes them as Input for Camera Rotation and Camera Zoom

 Public Functions
  Horizontal() - Returns Acceleration Value along X - Axis
  Vertical() - Returns Acceleration Value along Z - Axis
  SetBaseAcc() - Change Value of Base Acceleration
  ResetBaseAcc() - Reset Value of Base Acceleration to Default
 Global variables accessed/modified by the module.
  zoomScript - CameraZoomScript
*/

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Lean.Touch;

public class AccInput : MonoBehaviour {

//
    ************************************************************************************************************

     // Class Variable Declarations

// Script handling the Pinch Zoom functionality
     public CameraZoomSmooth zoomScript;

// Time taken to adjust the Camera along -
     public float minVZoomLPKWidthInSecX = 0.5f;       // VERTICAL axis when Zoom is MINIMUM
     public float maxVZoomLPKWidthInSecX = 2.5f;       // VERTICAL axis when Zoom is MAXIMUM
     public float minHZoomLPKWidthInSecX = 0.0f;       // HORIZONTAL axis when Zoom is MINIMUM
     public float maxHZoomLPKWidthInSecX = 1.0f;       // HORIZONTAL axis when Zoom is MAXIMUM

// Base Acceleration Values of the Device i.e Default ORIENTATION of the Device
     private Vector3 baseAcceleration = Vector3.zero;

// Live Input Values From the ACCELEROMETER - Current Orientation of the device
     private Vector3 currentAcc;                                    // Current Acceleration Vector (Along All Axis - X,Y,Z)
     private Vector3 currentAccVerticalOnly = Vector3.zero;         // Vertical Component Only ( Along Z Axis )
```

```
41      private Vector3 currentAccHorizontalOnly = Vector3.zero;    // Horizontal Component
            Only ( Along X Axis )
42
43  // Time interval between UPDATION of Acceleration Values
44      private float accUpdateIntervalX = 1.0f / 60.0f;
45
46  // Linearly Interpolated value of Acceleration - Using Lerp Function
47      private Vector3 lowPassValueV = Vector3.zero;     // Vertical
48      private Vector3 lowPassValueH = Vector3.zero;     // Horizontal
49
50  // Final Acceleration Values
51      private Vector3 finalAcc = Vector3.zero;
52
53  // Lowpass Factors - Factor that decides the rate of linear Interpolation between Previous
        Value and Current Value
54      private float maxVZoomLPFFactor;        // For Maximum Zoom - Vertical
55      private float minVZoomLPFFactor;        // For Minimum Zoom - Vertical
56      private float maxHZoomLPFFactor;        // For Maximum Zoom - Horizontal
57      private float minHZoomLPFFactor;        // For Minimum Zoom - Horizontal
58
59  // Current Lowpass Factor
60      private float currentVZoomLPFFactor;    // Vertical
61      private float currentHZoomLPFFactor;    // Horizontal
62
63  //
    **********************************************************************************************
64      // Class Function Definitions
65
66  // Start is called only once ** UNITY FUNCTION
67  // Calculate Values For the Lowpass Factors for  -
68      void Start () {
69          minVZoomLPFFactor = accUpdateIntervalX / minVZoomLPKWidthInSecX; // Minimum Zoom
                - Vertical
70          maxVZoomLPFFactor = accUpdateIntervalX / maxVZoomLPKWidthInSecX; // Maximum Zoom
                - Vertical
71          minHZoomLPFFactor = accUpdateIntervalX / minHZoomLPKWidthInSecX; // Minimum Zoom
                - Horizontal
72          maxHZoomLPFFactor = accUpdateIntervalX / maxHZoomLPKWidthInSecX; // Maximum Zoom
                - Horizontal
73      }
74
75  // Update is called once per frame ** UNITY FUNCTION
76  // Calculate The Final Acceleration Values from the Input Acceleration Values each Frame
77      void Update () {
78
79      // Current Lowpass Factor is Scaled between it's Maximum and Mininum Values
80          currentVZoomLPFFactor = ScaleValue(minVZoomLPFFactor,maxVZoomLPFFactor,
                zoomScript.Maximum,zoomScript.Minimum,zoomScript.GetCurrent());
81          currentHZoomLPFFactor = ScaleValue(minHZoomLPFFactor,maxHZoomLPFFactor,
                zoomScript.Maximum,zoomScript.Minimum,zoomScript.GetCurrent());
82
83      // If not set, Base Acceleration is zero initially . So Current Acceleration = Input
            Acceleration
84          currentAcc = Input.acceleration - baseAcceleration;
85          currentAccVerticalOnly.z = currentAcc.z;
86          currentAccHorizontalOnly.x = currentAcc.x;
87
88      // Lerp - Linear Interpolation
89          lowPassValueV = Vector3.Lerp(lowPassValueV,currentAccVerticalOnly,
                currentVZoomLPFFactor); // Vertical
90          lowPassValueH = Vector3.Lerp(lowPassValueH,currentAccHorizontalOnly,
                currentHZoomLPFFactor);    // Horizontal
91
92      // Final Accleration Values - After Interpolation
```

```
93            finalAcc.z = -lowPassValueV.z;                    // Values Along Z-axis Are
                  inverted - For Orientaion Purposes
94            finalAcc.x = lowPassValueH.x;
95
96      }
97
98  // Scale Conversion
99  // Scale a valueA (between minA and maxB) is SCALED to valueB (between minB and maxB)
100     private float ScaleValue(float fromV1,float toV1,float fromV2, float toV2, float
            value){
101           if (value <= fromV2)
102                 return fromV1;
103           else if (value >= toV2)
104                 return toV1;
105           else
106                 return (toV1 - fromV1) * ((value - fromV2) / (toV2 - fromV2)) + fromV1;
107     }
108
109 // Return The Horizontal Component of Final Acceleration
110     public float Horizontal(){
111           return finalAcc.x;
112     }
113
114 // Return The Horizontal Component of Final Acceleration
115     public float Vertical(){
116           return finalAcc.z;
117     }
118
119 // Set Current Acceleration ( i.e Current Device Orientation) as
120 // Base Acceleration (i.e DEFAULT Device Orientation)
121     public void SetBaseAcc(){
122           baseAcceleration = Input.acceleration;
123     }
124
125 // RESET Current Acceleration ( i.e Current Device Orientation)
126     public void ResetBaseAcc(){
127           baseAcceleration = Vector3.zero;
128     }
129 }
130 // ***** Class Definition Ends
    ************************************************************//
```

Listing 2.1: Code for Acceleration Input Module

2.3.2 Select-Object Input

```
1   /*
2    MODULE FOR SELECTING OBJECT
3    1/4/2017
4    Mayank Yadav
5    Synopsis - Uses Touch Input on screen to select object(exhibit) in world
6
7    Public Functions
8     OnFingerTap()
9    Global variables accessed/modified by the module.
10    fpsCam
11    textObjectSelected
12    layerMask
13    viewObjectScript
14 */
15
16 using System.Collections;
17 using System.Collections.Generic;
18 using UnityEngine;
```

```
19 using UnityEngine.UI;
20 using UnityEngine.EventSystems;
21 using Lean.Touch;
22
23
24 namespace Lean.Touch{
25      public class LolActualHandler : MonoBehaviour {
26
27      // Name of the Selected Object
28           public Text textObjectSelected;
29      // Player Camera
30           public Camera fpsCam;
31      // Layers to be used for Raycasting for object selection
32           public LayerMask layerMask = Physics.DefaultRaycastLayers;
33      // ViewObject Class Script
34           public ViewObject viewObjectScript;
35      // Selected Object
36           private GameObject selectedObject;
37
38      // Add Function OnFingerTap() as a listener to OnFingerTap Event
39           protected virtual void OnEnable()
40           {
41                LeanTouch.OnFingerTap   += OnFingerTap;
42           }
43      // Remove Function OnFingerTap() as a listener to OnFingerTap Event
44           protected virtual void OnDisable()
45           {
46                LeanTouch.OnFingerTap   -= OnFingerTap;
47           }
48      // Function OnFingerTap
49           public void OnFingerTap(LeanFinger finger)
50           {
51                SelectObject (finger);
52           }
53      // Select the object in world space , according to touch input on the screen
54           private void SelectObject(LeanFinger finger)
55           {
56           // Select Finger as input ONLY if the touch did not start over a GUI object
57                if (!finger.StartedOverGui == true)
58                {
59                // Ray that is cast from the point of touch on screen into the world
60                     Ray ray;
61                // First Object the Raycast hits
62                     RaycastHit hit;
63
64                // Cast Ray
65                     ray = fpsCam.ScreenPointToRay (finger.ScreenPosition);
66                // If Ray hits any object specified on the layerMask
67                     if (Physics.Raycast (ray, out hit, Mathf.Infinity, layerMask))
68                     {
69                     // Check if the hit object is an Exhibit
70                          if (hit.collider.CompareTag ("Exhibit")) {
71                               selectedObject = hit.collider.gameObject;
72                               viewObjectScript.GetObjectToBeViewed (selectedObject);
73                               textObjectSelected.text = selectedObject.transform.name;
74                          }
75                     // Else Find such an object in its Parent Recurisvely
76                          else {
77                               selectedObject = GetParentWithTag (hit.collider.gameObject,
                                      "Exhibit");
78                               if (selectedObject != null) {
79                               // Display Selected Object's Name
80                                    textObjectSelected.text = selectedObject.transform.name
                                           ;
81                               // Get the Object Ready for View
```

```
82                                        viewObjectScript.GetObjectToBeViewed (selectedObject);
83                                   // Get the Object's Description Ready in Object Viewer
84                                        viewObjectScript.GetDescription (selectedObject);
85                                   }
86                              }
87                         }
88                    }
89               }
90          // Return Parent Object with given Tag
91               private GameObject GetParentWithTag(GameObject child,string tag){
92                    Transform t = child.transform;
93               // Search Recursively in Parent for Gameobject with given tag
94                    while (t.parent != null) {
95                         if (t.parent.CompareTag (tag))
96                              return t.parent.gameObject;
97
98                         t = t.parent.transform;
99
100                   }
101                   return null;
102              }
103         }
104   }
105   // ************************ Class Definition Ends ******************************** //
```

Listing 2.2: Code for Select-Object Input Module

2.3.3 Joystick Input

```
1     /*
2      JOYSTICK MODULE
3      3/4/2017
4      Author - Mayank Yadav
5      Synopsis - Virtual Joystick Functions - Moving the Joystick only if the touch started
       on Joystick Background Image.
6                                                  - Joystick Image cannot go completely
                                                     outside the Joystick Background
                                                     Image
7                                                  - Player Moves based on difference
                                                     between Joystick Image's current and
                                                      original position
8      Public Functions
9        OnFingerDown()
10       OnFingerSet()
11       OnFingerUp()
12       Horizontal()
13       Vertical()
14     Include Definition for Exhibit Class
15     Global variables accessed/modified by the module.
16       layerMask
17  */
18
19  using System.Collections;
20  using System.Collections.Generic;
21  using UnityEngine;
22  using UnityEngine.UI;
23  using UnityEngine.EventSystems;
24  using Lean.Touch;
25
26
27  namespace Lean.Touch{
28       public class VirtualJoystick : MonoBehaviour {
29       // Public Variables - Accessed by the module from outside the Module
30            public LayerMask layerMask = 1 << 8;
```

```
31
32      // Private Variables
33           private Image bgImg;
34           private Image joystickImg;
35           private Vector3 InputVector = Vector3.zero;
36           private Vector2 pos;
37           private bool startedOverJoystick = false;
38
39           protected virtual void OnEnable()
40           {
41           // Hook into the events we need
42                LeanTouch.OnFingerDown  += OnFingerDown;
43                LeanTouch.OnFingerSet   += OnFingerSet;
44                LeanTouch.OnFingerUp    += OnFingerUp;
45           }
46           protected virtual void OnDisable()
47           {
48           // Unhook the events
49                LeanTouch.OnFingerDown  -= OnFingerDown;
50                LeanTouch.OnFingerSet   -= OnFingerSet;
51                LeanTouch.OnFingerUp    -= OnFingerUp;
52           }
53      // Get Image Components for Joystick and Joystick Background
54           private void Start(){
55                bgImg = GetComponent<Image> ();
56                joystickImg = transform.GetChild (0).GetComponent<Image> ();
57           }
58      // When Starts the touch, Check if it started on the Joystick Background. If yes the
          Move Joystick
59           private void OnFingerDown(LeanFinger finger)
60           {
61                if (finger.StartedOverGui == true)
62                {
63                     RaycastResult guiObj;
64                     guiObj = LeanTouch.RaycastGui (finger.ScreenPosition);
65                     if (guiObj.gameObject.CompareTag ("Joystick")) {
66                          startedOverJoystick = true;
67                          MoveJoystick (finger);
68                     }
69                }
70
71           }
72      // If Touch [Finger] moves on screen joystick also moves IF the touch started from
          Joystick Background
73           private void OnFingerSet(LeanFinger finger)
74           {
75                if (startedOverJoystick == true) {
76                     if (LeanTouch.PointOverLayer (finger.ScreenPosition, layerMask))
77                          MoveJoystick (finger);
78                }
79
80           }
81      // Touch Ends - Reset Joystick Position to centre
82           private void OnFingerUp(LeanFinger finger)
83           {
84                startedOverJoystick = false;
85                InputVector = Vector3.zero;
86                joystickImg.rectTransform.anchoredPosition = Vector3.zero;
87           }
88      // Move Joystick along with touch [Bounded by Joystick Background] - AND Assign Input
          Vector
89           private void MoveJoystick(LeanFinger finger)
90           {
91                if (RectTransformUtility.ScreenPointToLocalPointInRectangle (bgImg.
                     rectTransform, finger.ScreenPosition, null, out pos))
```

```
92                {
93                     pos.x = (pos.x / bgImg.rectTransform.sizeDelta.x);
94                     pos.y = (pos.y / bgImg.rectTransform.sizeDelta.y);
95
96                     InputVector = new Vector3 (pos.x * 2 + 1, 0, pos.y * 2 - 1);
97                     InputVector = (InputVector.magnitude > 1.0f) ? InputVector.normalized
                           : InputVector;
98
99                     joystickImg.rectTransform.anchoredPosition = new Vector2 (InputVector.
                           x * (bgImg.rectTransform.sizeDelta.x / 2), InputVector.z * (bgImg.
                           rectTransform.sizeDelta.y / 2));
100               }
101
102          }
103     // Return Horizontal Input Value
104          public float Horizontal(){
105               if(InputVector.x != 0)
106                    return InputVector.x;
107               else
108                    return Input.GetAxis ("Horizontal");
109
110          }
111     // Return Vertical Input Value
112          public float Vertical(){
113               if (InputVector.z != 0)
114                    return InputVector.z;
115               else
116                    return Input.GetAxis ("Vertical");
117          }
118     }
119 }
120 // *************** Class Definition Ends
    ***************************************************//
```

Listing 2.3: Code for Joystick Input Module

2.3.4 Pinch Zoom

```
1    /*
2
3     Pinch Zoom Module
4     7/4/2017
5     Author - Mayank Yadav
6     Synopsis - Pinch Zoom - Zoom in/out is smoothe
7
8  */
9
10 using UnityEngine;
11
12 namespace Lean.Touch
13 {
14      // This script allows you to zoom a camera in and out based on the pinch gesture
15      // This supports both perspective and orthographic cameras
16      public class CameraZoomSmooth : MonoBehaviour
17      {
18      //"Ignore fingers with StartedOverGui?"
19           public bool IgnoreGuiFingers = true;
20
21      //"Allows you to force rotation with a specific amount of fingers (0 = any)")
22           public int RequiredFingerCount;
23
24      //"If you want the mouse wheel to simulate pinching then set the strength of it here.
            Range(-1.0f, 1.0f)
25           public float WheelSensitivity;
```

```
26
27      //"The camera we will be moving")
28           public Camera Camera;
29
30      //"The target FOV/Size")
31           public float Target = 10.0f;
32
33      //"The minimum FOV/Size we want to zoom to")
34           public float Minimum = 10.0f;
35
36      //"The maximum FOV/Size we want to zoom to")
37           public float Maximum = 60.0f;
38
39      //"How quickly the zoom reaches the target value")
40           public float Dampening = 10.0f;
41
42      // Start - Runs at the start of Game - Once
43           protected virtual void Start()
44           {
45                if (LeanTouch.GetCamera(ref Camera) == true)
46                {
47                     Target = GetCurrent();
48                }
49           }
50      // LateUpdate Runs after Every Frame
51           protected virtual void LateUpdate()
52           {
53                // If camera is null, try and get the main camera, return true if a camera
                       was found
54                if (LeanTouch.GetCamera(ref Camera) == true)
55                {
56                     // Get the fingers we want to use
57                     var fingers = LeanTouch.GetFingers(IgnoreGuiFingers,
                           RequiredFingerCount);
58
59                     // Scale the current value based on the pinch ratio
60                     Target *= LeanGesture.GetPinchRatio(fingers, WheelSensitivity);
61
62                     // Clamp the current value to min/max values
63                     Target = Mathf.Clamp(Target, Minimum, Maximum);
64
65                     // The framerate independent damping factor
66                     var factor = 1.0f - Mathf.Exp(-Dampening * Time.deltaTime);
67
68                     // Store the current size/fov in a temp variable
69                     var current = GetCurrent();
70
71                     current = Mathf.Lerp(current, Target, factor);
72
73                     SetCurrent(current);
74                }
75           }
76      // Get Current Camera Field of View
77           public float GetCurrent()
78           {
79                if (Camera.orthographic == true)
80                {
81                     return Camera.orthographicSize;
82                }
83                else
84                {
85                     return Camera.fieldOfView;
86                }
87           }
88      // Set Camera Field of View
```

```
89            private void SetCurrent(float current)
90            {
91                  if (Camera.orthographic == true)
92                  {
93                        Camera.orthographicSize = current;
94                  }
95                  else
96                  {
97                        Camera.fieldOfView = current;
98                  }
99            }
100     }
101 }
```

Listing 2.4: Code for Pinch Zoom Module

2.3.5 Player Controller

```
1     /*
2      PLAYER CONTROLLER
3      5/4/2017
4      Author - Nikunj Mittal
5      Synopsis - Controls Player Movements , Jump Movement
6
7      Public Functions
8       JumpInput()
9      Global variables accessed/modified by the module.
10      vjMov
11      accCam
12  */
13
14  using System.Collections;
15  using System.Collections.Generic;
16  using UnityEngine;
17  using UnityEngine.UI;
18
19  public class PlayerController : MonoBehaviour {
20
21  // Public Variables - Accessed by the Module from outside
22        public Lean.Touch.VirtualJoystick vjMov;     // Virtual Joystick - For Player
              Movement Input
23        public AccInput accCam;                      // Acceleration Input Script
24
25  // Public Variables - Can be Accessed by outside the Module
26        public float jumpSpeed = 8.0f;               // Jump Speed -_-
27        public float speed = 6.0f;                   // Player Movement Speed
28        public float gravity = 20.0f;                // Gravity Value
29        public float speedH = 2.0f;                  // Player Horizontal Rotation Speed
30
31  // Private Variables
32        private bool jumpValue = false;                   // Should I Jump? , says the
              Player
33        private float yaw = 0.0f;                         // All For Horizontal Rotation
34        private Vector3 moveDirection = Vector3.zero;     // Direction in which Player is to
              Move
35        private CharacterController controller;           // Character Controller Component
              of Player
36
37
38        void Start()
39        {
40        // Get the Character Controller Component from Player GameObject
41              controller = GetComponent<CharacterController> ();
42        }
```

```
43      // Update - Runs before every frame refresh - UNITY
44      void Update()
45      {
46           // Rotate the Player as the Camera Rotates [Horizontal Only]
47           yaw += speedH * accCam.Horizontal();
48           transform.eulerAngles = new Vector3(0.0f, yaw, 0.0f);
49           // If Player is on Ground - Calculate Move Direction, Check for Jump Input
50           if (controller.isGrounded) {
51                moveDirection = new Vector3 (vjMov.Horizontal () , 0 , vjMov.Vertical ());
52                moveDirection = transform.TransformDirection(moveDirection);
53                moveDirection *= speed;
54                if (JumpInput () == true) {
55                     JumpInput (false);
56                     moveDirection.y = jumpSpeed;
57                }
58           }
59           // Effect of Gravity
60           moveDirection.y -= gravity * Time.deltaTime;
61           // Move Player in Calculated Direction
62           controller.Move(moveDirection * Time.deltaTime);
63      }
64      // Check Status of Jump Input
65      private bool JumpInput(){
66           return jumpValue;
67      }
68      // Set Jump Input - Set to 1 for Jump
69      public void JumpInput(bool newValue){
70           jumpValue = newValue;
71      }
72 }
73 // ***** Class Definition Ends
    *************************************************************//
```

Listing 2.5: Code for Player Controller Module

2.3.6 Camera Controller

```
1    /*
2     CAMERA CONTROLLER.
3     4/4/2017
4     Author - Nikunj Mittal
5     Synopsis - Rotates Camera According to Acceleration Input
6               Translates Camera According to Player Movement
7
8     Global variables accessed/modified by the module.
9      player - Player Gameobject
10     accCam - Acceleration Input Script
11 */
12
13
14 using System.Collections;
15 using System.Collections.Generic;
16 using UnityEngine;
17 using UnityEngine.UI;
18
19 public class CameraController : MonoBehaviour {
20
21 // Player Gameobject - To which the Camera is attached
22      public GameObject player;
23 // Acceleration Input Script - Provides Values for Camera Along X - Y Axis
24      public AccInput accCam;
25
26 // Vertical Camera Rotation is CLAMPED between a Minimum and Maximum
27      public float vMin = -10.0f;
```

```
28      public float vMax = 10.0f;
29
30  // Speed of Movement of Camera
31      public float speedH = 2.0f;
32      public float speedV = 2.0f;
33
34      private float yaw = 0.0f;
35      private float pitch = 0.0f;
36       private Vector3 offset;
37
38       void Start ()
39       {
40            offset = transform.position - player.transform.position;
41       }
42      void Update()
43      {
44  // yaw - Rotation along the Horizontal
45            yaw += speedH * accCam.Horizontal ();
46  // pitch - Rotation along the Vertical
47            pitch = speedV * accCam.Vertical();
48
49  // Clamp the Vertical Rotation Values - To Avoid full rotation along Vertical
50            pitch = Mathf.Clamp(pitch,vMin,vMax);
51  // Assign the Values to the Camera
52            transform.eulerAngles = new Vector3(pitch, yaw, 0.0f);
53      }
54       void LateUpdate ()
55       {
56  // Camera moves along with the player
57            transform.position = player.transform.position + offset;
58      }
59  }
60  // ******* Class Definition Ends *****************************************************//
```

Listing 2.6: Code for Camera Controller Module

2.3.7 GUI List

```
1     /*
2      LIST CONTROLLER
3      1/4/2017
4      Author - Mayank Yadav
5      Synopsis - List Gameobject Functions [i.e Scrollable List in GUI]
6
7      Public Functions
8       AddToList()
9       DestroyAllChildren()
10     Global variables accessed/modified by the module.
11      ContentPanel - Panel Containing the List
12      ListItemPrefab -  Template for List item
13  */
14
15  using System.Collections;
16  using System.Collections.Generic;
17  using UnityEngine;
18  using UnityEngine.UI;
19
20  public class ListController : MonoBehaviour {
21
22  // Panel Containing the List
23       public GameObject ContentPanel;
24  // Template for List item
25       public GameObject ListItemPrefab;
26
```

```
// Add an Item to the List
     public void AddToList(string itemName,string shortDesc,bool status, int index){
          GameObject newItem = Instantiate(ListItemPrefab) as GameObject;
          ListItemController controller = newItem.GetComponent<ListItemController>();
          controller.itemName = itemName;
          controller.status = status;
          controller.index = index;

          newItem.SetActive (true);
          newItem.transform.GetChild (0).GetComponent<Text> ().text = itemName;
          newItem.transform.GetChild (2).GetComponent<Text> ().text = shortDesc;
          newItem.transform.SetParent(ContentPanel.transform,false);
     }

// Empty List - Destroy All List Item Gameobjects
     public void DestroyAllChildren(){
          Transform Content = ContentPanel.transform;
          int childs = Content.childCount;
          for (int i = childs-1; i >= 0; i--) {
               GameObject.Destroy (Content.GetChild (i).gameObject);
          }
     }

}
// ***************** Class Definition Ends ************************************//
```

Listing 2.7: Code for GUI List Module

2.3.8 Object Viewer

```
   /*
    OBJECT VIEWER MODULE
    3/4/2017
    Author - Mayank Yadav
    Synopsis - Manages All Wishlist and Catalogue Functions
    Public Functions
      ViewSelectedObject()
      GetObjectToBeViewed()
      GetDescription()
      SwitchMode()
      enableSandbox()
      disableSandbox()
      DestroyAllChildren()
      ResetObjectViewer()
      StartObjectViewer()
      ResetObject()
    Include Definition for Exhibit Class
    Global variables accessed/modified by the module.
      Description
      objectPostion
      objectDistance
      rotateScript
      translateScript
      zoomScript
      sandboxD,sandboxE,toggleMode
      resetPos
*/

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Lean.Touch;
using UnityEngine.UI;
using UnityEngine.SceneManagement;
```

```
35
36
37  public class ViewObject : MonoBehaviour {
38
39  // Public Variables - Accessed by the module from outside the Module [ i.e they are
        initializes in the UNITY Editer ]
40        public Text Description;
41        public ObjectRotate rotateScript;
42        public ObjectTranslate translateScript;
43        public CameraZoomSmooth zoomScript;
44        public Button sandboxD,sandboxE,toggleMode;
45        public Vector3 resetPos;
46
47  // Private Variables
48
49        private Vector3 objectPostion;
50        private float objectDistance;
51        private GameObject objectToBeViewed;
52        private Transform objCollider;
53        private Quaternion objectRotation;
54        private Vector3 centerOffset = Vector3.zero;
55        private int viewMode = 1;  // 1 -> Rotate, 0 -> Translate
56
57  // Instantiate Selected Object in the Object Viewer
58        public void ViewSelectedObject(){
59              if (objectToBeViewed == null)
60                    return;
61              else {
62              // Minimum Distance the Object should be from the Camera to fully inside the
                    camera's field of view
63                    float viewDistance = objectDistance * 0.5f / Mathf.Tan (zoomScript.Maximum
                          * 0.5f * Mathf.Deg2Rad);
64                    transform.position = resetPos + (Max(viewDistance,objectDistance,1.0f) *
                          transform.forward) ;
65                    objectRotation = objectToBeViewed.transform.rotation;
66                    objectPostion = transform.position - centerOffset;
67                    Instantiate (objectToBeViewed,objectPostion,objectRotation,transform);
68              // If Auto Tour , DeActivate the original reference Object
69                    if(SceneManager.GetActiveScene().buildIndex == 2)
70                          objectToBeViewed.SetActive (false);
71
72              }
73        }
74  // Gets selected Object and Stores Its 'Exhibit' Object[Complete Exhibit] and 'Collider'
        Object [Child with Collider]
75  // Both May be Same
76        public void GetObjectToBeViewed(GameObject objectReference){
77              Transform temp;
78              if (objectReference.CompareTag ("Exhibit"))
79                    objectToBeViewed = objectReference;
80              else {
81                    temp = GetChildObjectWithTag (objectReference.transform, "Exhibit", true);
82                    if (temp != null)
83                          objectToBeViewed = temp.gameObject;
84                    else
85                          objectToBeViewed = objectReference;
86              }
87              objectToBeViewed.SetActive (true);
88              if (objectToBeViewed.GetComponent<Collider> () == true) {
89                    objCollider = objectToBeViewed.transform;
90              }
91              else {
92                    objCollider = GetChildObjectWithTag (objectToBeViewed.transform,"Collider")
                          ;
93
```

```
94            }
95            Vector3 size = objCollider.GetComponent<Collider> ().bounds.size;
96            Vector3 center = objCollider.GetComponent<Collider> ().bounds.center;
97            centerOffset = center - objectToBeViewed.transform.position;
98            objectDistance = Max (size.x,size.y,size.z);
99            translateScript.SetObjectSize (objectDistance/2);
100
101     }
102 // Get Description of Selected Object from File
103     public void GetDescription(GameObject objectReference){
104           string filename = objectReference.name;
105           string contents;
106           TextAsset txtAssets = (TextAsset)Resources.Load (filename);
107           contents = txtAssets.text;
108
109           Description.text = contents;
110     }
111 // Toggle between Translate Mode and Rotate Mode
112     public void SwitchMode(){
113           if (viewMode == 1) {
114                 rotateScript.enabled = false;
115                 translateScript.enabled = true;
116                 viewMode = 2;
117           }
118           else if (viewMode == 2) {
119                 rotateScript.enabled = true;
120                 translateScript.enabled = false;
121                 viewMode = 1;
122           }
123     }
124 // Switch to Sandbox Mode -> Rotation along All 3 Axis is allowed
125     public void enableSandbox(){
126           rotateScript.sandboxMode = true;
127           rotateScript.enabled = true;
128           translateScript.enabled = false;
129           viewMode = 1;
130     }
131 // Switch to Normal Mode -> Rotation along Horizontal Axis Only
132     public void disableSandbox(){
133           rotateScript.sandboxMode = false;
134           rotateScript.enabled = true;
135           translateScript.enabled = false;
136           viewMode = 1;
137
138     }
139 // Destroy All Objects in View [View -> in Object Viewer]
140     public void DestroyAllChildren(){
141           int childs = transform.childCount;
142           for (int i = childs-1; i >= 0; i--) {
143                 GameObject.Destroy (transform.GetChild (i).gameObject);
144           }
145
146     }
147 // Reset Object Viewer
148     public void ResetObjectViewer(){
149           transform.position = new Vector3 (0,-80,20);
150           transform.rotation = Quaternion.identity;
151           rotateScript.enabled = false;
152           translateScript.enabled = false;
153     }
154 // Initialize Object Viewer
155     public void StartObjectViewer(){
156           rotateScript.enabled = true;
157           translateScript.enabled = false;
158           transform.position = new Vector3 (0,-80,20);
```

```
159             transform.rotation = Quaternion.identity;
160             sandboxE.gameObject.SetActive (true);
161             sandboxD.gameObject.SetActive (false);
162             disableSandbox ();
163         }
164     // Reset Object's Position
165         public void ResetObject(){
166             transform.position = objectPostion + centerOffset;
167             transform.rotation = Quaternion.identity;
168         }
169     // Auxilary Functions
170     // Find Child with given tag OR 'Exhibit' tag in children recursively
171         private Transform GetChildObjectWithTag(Transform parent, string tag){
172             for (int i = 0; i < parent.childCount; i++) {
173                 Transform child = parent.GetChild (i);
174                 Transform c2;
175                 if (child.CompareTag (tag) || child.CompareTag ("Exhibit")) {
176                     if (child.CompareTag ("Exhibit")) {
177                         if (child.GetComponent<Collider> () == true)
178                             return child;
179                     }
180                     else
181                         return child;
182                 }
183                 if (child.childCount > 0) {
184                     c2 = GetChildObjectWithTag (child, tag);
185                     if (c2 != null)
186                         return c2;
187                 }
188             }
189             return null;
190         }
191 
192         // Find Child with given tag ONLY in children recursively
193         private Transform GetChildObjectWithTag(Transform parent, string tag, bool
                notforCollider){
194             for (int i = 0; i < parent.childCount; i++) {
195                 Transform child = parent.GetChild (i);
196                 Transform c2;
197                 if (child.CompareTag (tag)) {
198                         return child;
199                     }
200                 if (child.childCount > 0) {
201                     c2 = GetChildObjectWithTag (child, tag);
202                     if (c2 != null)
203                         return c2;
204                 }
205             }
206             return null;
207         }
208     // Simply returns the Max value between A and B
209         private float Max(float A,float B){
210             if (A >= B)
211                 return A;
212             else
213                 return B;
214         }
215     // Simply returns the Max value between A, B and C
216         private float Max(float A,float B, float C){
217             return Max (Max(A,B),C);
218         }
219 
220     }
221     // *************** Class Definition Ends
            ***********************************************//
```

Listing 2.8: Code for Object Viewer Module

2.3.9 Object Rotate

```
/*
 OBJECT VIEWER - OBJECT ROTATE.
 3/4/2017
 Mayank Yadav
 Synopsis - Rotate the Object in Object Viewer

 Public Functions - None
 Global variables accessed/modified by the module.
  IgnoreGuiFingers
  RequiredFingerCount
  screenScale
  sandboxMode
*/

using UnityEngine;

namespace Lean.Touch
{
     // This script allows you to transform the current GameObject
     public class ObjectRotate : MonoBehaviour
     {
     //"Ignore fingers with StartedOverGui?"
          public bool IgnoreGuiFingers = false;

     //"Ignore fingers if the finger count doesn't match? (0 = any)"
          public int RequiredFingerCount;

     //"The camera the translation will be calculated using (default = MainCamera)"
          public float screenScale;

     //"Sandbox Mode Allows Free rotation along all 3 Axis [ Normal Mode Allows only
         Horizontal Rotation ]
          public bool sandboxMode = false;

     // Update is Called Every Frame
          protected virtual void Update()
          {
               // Get the fingers we want to use
               var fingers = LeanTouch.GetFingers(IgnoreGuiFingers, RequiredFingerCount);

               // Calculate the screenDelta value based on these fingers
               var screenDelta = LeanGesture.GetScreenDelta(fingers);
               var degrees = LeanGesture.GetTwistDegrees(fingers);

               // Perform the translation
               Rotate(screenDelta,degrees);
          }
     // Performs Rotation on the Object
          private void Rotate(Vector2 screenDelta,float degreeZ)
          {
          // Sandbox Mode - Allow Rotation Along All 3 Axis (X,Y,Z)
               if (sandboxMode == true) {
                    float degreeX, degreeY;
               // Calculate Rotation Values from screenDelta [Screen Delta = Change in
                   touch position on screen for a finger]
                    degreeY = -1 * screenScale * screenDelta.x;
                    degreeX = screenScale * screenDelta.y;
               // Rotate According to World Axis
```

```
57                      transform.Rotate (degreeX,degreeY,degreeZ,Space.World );
58                  }
59                  else {
60                      float degreeY = -1 * screenScale * screenDelta.x;
61                  // Rotate According to World Axis
62                      transform.Rotate (0,degreeY,0,Space.World );
63                  }
64              }
65      }
66  }
67  // ****************** Class Definition Ends ************************************ //
```

Listing 2.9: Code for Object Rotate Module

2.3.10 Object Translate

```
1     /*
2      OBJECT VIEWER – OBJECT TRANSLATE
3      3/4/2017
4      Mayank Yadav
5      Synopsis – Move Object in View in a bounded
6
7      Public Functions
8
9      Global variables accessed/modified by the module.
10      IgnoreGuiFingers
11      RequiredFingerCount
12      screenScale
13  */
14
15  using UnityEngine;
16
17  namespace Lean.Touch
18  {
19  // This script allows you to transform the current GameObject
20      public class ObjectTranslate : MonoBehaviour
21      {
22      // Ignore fingers with StartedOverGui?"
23          public bool IgnoreGuiFingers = false;
24
25      // Ignore fingers if the finger count doesn't match? (0 = any)"
26          public int RequiredFingerCount;
27
28      // The camera the translation will be calculated using (default = MainCamera)"
29          public float screenScale;
30
31      // Size of the Object – max[width,height,length]
32          private float objectSize = 10.0f;
33
34          protected virtual void Update()
35          {
36          // Get the fingers we want to use
37              var fingers = LeanTouch.GetFingers(IgnoreGuiFingers, RequiredFingerCount);
38
39          // Calculate the screenDelta value based on these fingers
40              var screenDelta = LeanGesture.GetScreenDelta(fingers);
41
42          // Perform the translation
43              Translate(screenDelta);
44          }
45      // Move The Object in View
46          private void Translate(Vector2 screenDelta)
47          {
48              Vector3 newPosition = transform.localPosition;
```

```
49                Vector2 tempDelta;
50            // tempDelta = 2D Vector of Movement of Object
51                tempDelta = screenDelta * ((screenScale * objectSize) / 100.0f);
52                newPosition += (Vector3)tempDelta;
53            // Object in View's position w.r.t parent Container object
54                transform.localPosition = new Vector3(Mathf.Clamp(newPosition.x,-objectSize
                      ,objectSize),Mathf.Clamp(newPosition.y,-objectSize,objectSize),
                      newPosition.z);
55
56            }
57        // Set Object Size - Used to decide the boundary of area of allowed movement of the
              object
58            public void SetObjectSize(float size){
59                objectSize = size;
60            }
61        }
62    }
```

Listing 2.10: Code for Object Translate Module

2.3.11 Wish list Manager

```
1     /*
2      WISHLIST MODULE
3      3/4/2017
4      Author - Mayank Yadav
5      Synopsis - Manages All Wishlist and Catalogue Functions
6      Public Functions
7        AddToWishlist()
8        LoadExhibit()
9        NextWishlistItem()
10       PrevWishlistItem()
11       ResetWishlistIndex()
12       ResetWishlist()
13       GetDescription()
14       MakeCatalogue()
15       WriteCatalogue()
16     Include Definition for Exhibit Class
17     Global variables accessed/modified by the module.
18       CatalogueController
19       ViewObjectScript
20       ListItemActive
21       ListItemDisabled
22       Exhibits
23       Description
24  */
25
26  using System.Collections;
27  using System.Collections.Generic;
28  using UnityEngine;
29  using UnityEngine.UI;
30
31  public class WishlistManager : MonoBehaviour {
32
33  // Public Variables - Accessed by the module from outside the Module
34      public ListController CatalogueController;   // List Controller for Catalogue [List
            in GUI]
35      public ViewObject ViewObjectScript;              // ViewObject Script
36      public Sprite ListItemActive;                // Image/Icon for Item Added to Wishlist
37      public Sprite ListItemDisabled;                  // Image/Icon for Item Removed From
            Wishlist
38      public Transform Exhibits;                       // Parent GameObject Containing ALL
            Exhibits
39      public Text Description;                     // Description of Object
```

```
40
41 // Private Variables
42      private List<Exhibit> Catalogue = new List<Exhibit>(); // List of All Exhibits
43      private List<Exhibit> Wishlist = new List<Exhibit> (); // List of Exhibits User Wants
            to View
44      private int wishlistIndex = 0;                                    // Index of Current
            Object in View in Wishlist
45      private int catalogueCount = 0;                                   // Total no. of
            Exhibits
46
47 // When the Scene is Loaded - Extract List of Objects from 'Exhibits' , and Add them to
       Catalogue
48      void Start(){
49           MakeCatalogue ();
50           WriteCatalogue ();
51      }
52 // Add an Exhibit to the Wishlist
53      public void AddToWishlist(ListItemController item){
54           if (item.status == false) {
55                item.gameObject.transform.GetChild (1).GetComponent<Image> ().sprite =
                     ListItemActive;
56                Catalogue [item.index].inWishlist = true;
57                Wishlist.Add (Catalogue[item.index]);
58                item.status = true;                                  // status = 1 => Present
                     in Wishlist
59           }
60           else {
61
62                item.gameObject.transform.GetChild (1).GetComponent<Image> ().sprite =
                     ListItemDisabled;
63                Catalogue [item.index].inWishlist = false;
64                Wishlist.Remove (Catalogue[item.index]);
65                item.status = false;
66           }
67      }
68 // Load Exhibit at Current Wishlist Index to Object Viewer
69      public void LoadExhibit(){
70
71           if(Wishlist.Count>0){
72                if (wishlistIndex >= Wishlist.Count) {
73                     wishlistIndex = Wishlist.Count - 1;
74                }
75           // Get ObjectViewer Ready - Destroy Exisiting objects, Get Current Object, Load
                it in Viewer
76                ViewObjectScript.DestroyAllChildren ();
77                ViewObjectScript.StartObjectViewer ();
78                ViewObjectScript.GetObjectToBeViewed (Exhibits.GetChild (Wishlist [
                     wishlistIndex].index).gameObject);
79                ViewObjectScript.ViewSelectedObject ();
80           }
81
82      }
83 // Load Next Item in Wishlist
84      public void NextWishlistItem(){
85           if (wishlistIndex < Wishlist.Count - 1) {
86                wishlistIndex++;
87                LoadExhibit ();
88           }
89      }
90 // Load Previous Item in Wishlist
91      public void PrevWishlistItem(){
92           if (wishlistIndex > 0) {
93                wishlistIndex--;
94                LoadExhibit ();
95           }
```

```
96        }
97  // Reset WishlistIndex to 0
98        public void ResetWishlistIndex(){
99              wishlistIndex = 0;
100       }
101 // Clear Wishlist
102       public void ResetWishlist(){
103             wishlistIndex = 0;
104             Wishlist.Clear ();
105       }
106
107 // Load Description of Current Object from it's corresponding file
108       public void GetDescription(){
109             Exhibit currExhibit = Wishlist [wishlistIndex];
110             string filename = currExhibit.name;
111             string contents;
112             TextAsset txtAssets = (TextAsset)Resources.Load (filename);
113             contents = txtAssets.text;
114
115             Description.text = contents;
116       }
117 // Get Objects from 'Exhibits', and store them into Catalogue
118       private void MakeCatalogue(){
119             int i;
120             Transform obj;
121             catalogueCount = Exhibits.childCount;
122             for (i = 0; i < catalogueCount; i++) {
123                   obj = Exhibits.GetChild (i);
124                   Catalogue.Add(new Exhibit(obj.name,obj.GetComponent<Text>().text,false,i));
125             }
126       }
127 // Make a GUI List from Catalogue
128       private void WriteCatalogue(){
129             int i;
130             for (i = 0; i < catalogueCount; i++) {
131                   CatalogueController.AddToList(Catalogue[i].name,Catalogue[i].
                          shortDescription,false,i);
132             }
133       }
134 }
135 // *************** Class Definition Ends
      *************************************************//
136
137 // Exhibit Class
138 public class Exhibit{
139       public string name;
140       public string shortDescription;
141       public bool inWishlist;
142       public int index;
143 // Constructor for Exhibit Class
144       public Exhibit(string Name,string shortDesc,bool InWishlist,int Index){
145             this.name = Name;
146             this.inWishlist = InWishlist;
147             this.shortDescription = shortDesc;
148             this.index = Index;
149       }
150 }
151 // *************** Class Definition Ends
      *************************************************//
```

Listing 2.11: Code for Wish list Manager Module

2.3.12 Screen Resolution

```
1   /*
2    MODULE FOR SELECTING SCREEN RESOLUTION
3    1/4/2017
4    Author - Nikunj Mittal
5    Synopsis - Select Resolution
6
7    Public Functions
8     SetScreenRes()
9     SetScreenRes(int)
10   Global variables accessed/modified by the module.
11    ResList - Dropdown Menu listing available resolutions
12 */
13
14 using System.Collections;
15 using System.Collections.Generic;
16 using UnityEngine;
17 using UnityEngine.UI;
18
19 public class ScreenResolution : MonoBehaviour {
20
21 //Dropdown Menu listing available resolutions
22      public Dropdown ResList;
23
24 // Initial Resolution - 1280 x 720
25      void Start(){
26           SetScreenRes (2);
27      }
28 // Set Screen Resolution according to value selected in the Dropdown List
29      public void SetScreenRes(){
30      // Resolution index in Dropdown Menu
31           int index = ResList.value;
32           switch (index) {
33           case 0:
34                Screen.SetResolution (2560, 1440, true);
35                break;
36           case 1:
37                Screen.SetResolution (1920, 1080, true);
38                break;
39           case 2:
40                Screen.SetResolution (1280, 720, true);
41                break;
42           case 3:
43                Screen.SetResolution (1136, 640, true);
44                break;
45           case 4:
46                Screen.SetResolution (960, 540, true);
47                break;
48           case 5:
49                Screen.SetResolution (800, 480, true);
50                break;
51           }
52      }
53 // Select Resolution According to provided dropdown index
54      public void SetScreenRes(int index){
55
56           ResList.value = index;
57           switch (index) {
58           case 0:
59                Screen.SetResolution (2560, 1440, true);
60                break;
61           case 1:
62                Screen.SetResolution (1920, 1080, true);
63                break;
64           case 2:
65                Screen.SetResolution (1280, 720, true);
```

```
66                break;
67           case 3:
68                Screen.SetResolution (1136, 640, true);
69                break;
70           case 4:
71                Screen.SetResolution (960, 540, true);
72                break;
73           case 5:
74                Screen.SetResolution (800, 480, true);
75                break;
76           }
77      }
78 }
79 // ******************* Class Definition Ends ************************************** //
```

Listing 2.12: Code for Screen Resolution Module

2.3.13 Mini-Map

```
1    /*
2     MiniMap Module
3     7/4/2017
4     Author - HarShit Bansal
5     Synopsis - Manage Mini Map
6
7     Public Functions
8      SetScreenRes()
9      SetScreenRes(int)
10    Global variables accessed/modified by the module.
11     ResList - Dropdown Menu listing available resolutions
12 */
13
14 using System.Collections;
15 using System.Collections.Generic;
16 using UnityEngine;
17 using UnityEngine.UI;
18
19 public class MapView : MonoBehaviour {
20
21      public Transform player;
22      public Image mapptr;
23      public Transform viewmapbutton;
24      public Transform viewmapwindow;
25      public Transform input;
26      public Transform inputobj;
27      Vector3 temp;
28      Vector3 pos;
29      Vector3 dummy;
30      void Start()
31      {
32           dummy = player.transform.position;
33           pos = player.transform.position;
34      }
35      void Update()
36      {
37           pos = player.transform.position - dummy;
38           Vector3 ptr = (5f * pos);
39           mapptr.rectTransform.anchoredPosition = new Vector2 (ptr.x, ptr.z);
40      }
41      public void section()
42      {
43           temp =new Vector3(-4.5f,0f,4.5f);
44           if (viewmapbutton.gameObject.activeInHierarchy == false){
45                viewmapbutton.gameObject.SetActive(true);
```

```
46            }
47      }
48      public void go()
49      {
50            player.transform.position = temp;
51            viewmapbutton.gameObject.SetActive(false);
52            back ();
53      }
54      public void disp()
55      {
56            //Time.timeScale = 0;
57            viewmapwindow.gameObject.SetActive (true);
58      }
59      public void back()
60      {
61            //viewmapbutton.gameObject.SetActive(false);
62            viewmapwindow.gameObject.SetActive (false);
63            //Time.timeScale = 1;
64      }
65      public void dispnotes()
66      {
67
68            if (input.gameObject.activeInHierarchy == true)
69            {
70                  input.gameObject.SetActive(false);
71            }
72            else{input.gameObject.SetActive(true);}
73
74      }
75      public void hideobjectwindow()
76      {
77
78            if (inputobj.gameObject.activeInHierarchy == true)
79            {
80                  inputobj.gameObject.SetActive(false);
81            }
82
83      }
84
85
86
87
88
89  }
```

Listing 2.13: Code for Mini-Map Module

2.3.14 Scene Manager

```
1     /*
2      LIST CONTROLLER
3      1/4/2017
4      Author - HarShit Bansal
5      Synopsis - List Gameobject Functions
6
7      Public Functions
8       LoadLevel()
9
10 */
11
12 using System.Collections;
13 using System.Collections.Generic;
14 using UnityEngine;
15 using UnityEngine.SceneManagement;
```

```
using UnityEngine.UI;
using System;


public class MenuManager : MonoBehaviour {

// Load new Scene
     public void LoadLevel(int level){
          SceneManager.LoadScene (level,LoadSceneMode.Single);
     }
}
```

Listing 2.14: Code for Scene Manager Module

2.3.15 Pause Module

```
   /*
    PAUSE MODULE
    3/4/2017
    Author - Mayank Yadav
    Synopsis - Pause the Tour - Time is frozen

    Public Functions
     Pause() - Sets Timescale = 0 i.e TIME STOPS !
    Global variables accessed/modified by the module.
     canvas - Pause Menu Gameobject
*/

using UnityEngine;
using System.Collections;

public class PauseMenu : MonoBehaviour {

// Pause Menu Gameobject
     public Transform canvas;

// Sets Timescale = 0  for pause , And Timescale = 1 for unpause
     public void Pause()
     {
     // If Pause Menu is NOT Active , Activate Pause Menu and Set Timescale to zero
          if (canvas.gameObject.activeInHierarchy == false)
          {
               canvas.gameObject.SetActive(true);
               Time.timeScale = 0;
          }
     // If Pause Menu is Active , De-activate Pause Menu and Set Timescale to 1
          else
          {
               canvas.gameObject.SetActive(false);
               Time.timeScale = 1;
          }
     }

}
// *************** Class Definition Ends *************************************//
```

Listing 2.15: Code for Pause Module Module