



LAB FILE

DEPARTMENT: COMPUTER SCIENCE & ENGINEERING

SUBJECT: VIRTUAL AND AUGMENTED REALITY IN IOS

SUBJECT CODE: BTMACS601

SUBMITTED BY:

RAVNISH SINGH

19100BTCSEMA05495

SUBMITTED TO:

RAHUL CHOUDHARY SIR

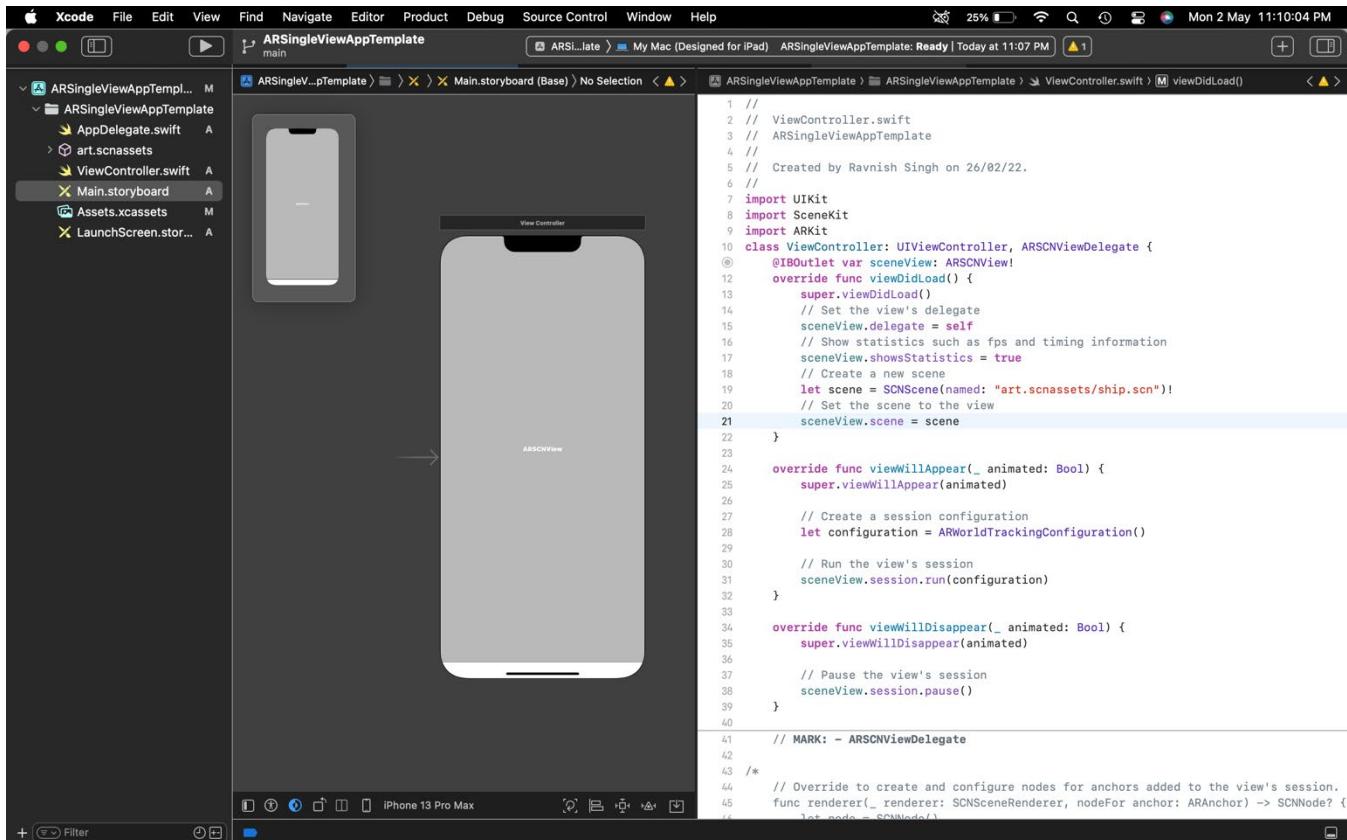
LIST OF PRACTICALS

S. No.	Practical	Page No.
1	Create an app using the Augmented Reality Template.	3
2	Create an app with the Single View App Template.	5
3	Create an AR app with Various Node.	7
4	Create an AR app to display and reset the World Origin.	9
5	Create an app to change position of object using Slider.	11
6	Create an app to add and remove multiple objects.	13
7	Create an app to display Text in AR.	15
8	Create an app to add textures to shape.	17
9	Create an app to use light, temperature, and intensity to modify the appearance of virtual objects.	19
10	Create an app that utilizes Positioning in AR.	21
11	Create an app to Rotate Objects in AR.	23
12	Create an app for Drawing in AR.	25
13	Create an app to recognize Pinch Gesture.	27
14	Create an app to recognize Rotation Gesture.	29

PRACTICAL – 1

Create an app using the Augmented Reality Template.

ViewController.swift:



```

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
ARSingleViewAppTemplate main ARSingleViewAppTemplate: Ready | Today at 11:07 PM
ARSingleViewAppTemplate AppDelegate.swift art.scnassets ViewController.swift Main.storyboard Assets.xcassets LaunchScreen.storyboard
ARSingleViewAppTemplate ARSingleViewAppTemplate ARSingleViewAppTemplate ViewController.swift viewDidLoad()
2 // ViewDidLoad
3 // ARSingleViewAppTemplate
4 //
5 // Created by Ravnish Singh on 26/02/22.
6 //
7 import UIKit
8 import SceneKit
9 import ARKit
10 class ViewController: UIViewController, ARSCNViewDelegate {
11     @IBOutlet var sceneView: ARSCNView!
12     override func viewDidLoad() {
13         super.viewDidLoad()
14         // Set the view's delegate
15         sceneView.delegate = self
16         // Show statistics such as fps and timing information
17         sceneView.showsStatistics = true
18         // Create a new scene
19         let scene = SCNScene(named: "art.scnassets/ship.scn")!
20         // Set the scene to the view
21         sceneView.scene = scene
22     }
23
24     override func viewWillAppear(_ animated: Bool) {
25         super.viewWillAppear(animated)
26
27         // Create a session configuration
28         let configuration = ARWorldTrackingConfiguration()
29
30         // Run the view's session
31         sceneView.session.run(configuration)
32     }
33
34     override func viewWillDisappear(_ animated: Bool) {
35         super.viewWillDisappear(animated)
36
37         // Pause the view's session
38         sceneView.session.pause()
39     }
40
41     // MARK: - ARSCNViewDelegate
42
43     /*
44     // Override to create and configure nodes for anchors added to the view's session.
45     func renderer(_ renderer: SCNSceneRenderer, nodeFor anchor: ARAnchor) -> SCNNode? {
46         let node = SCNNode()
47     }
48 */
49 }

```

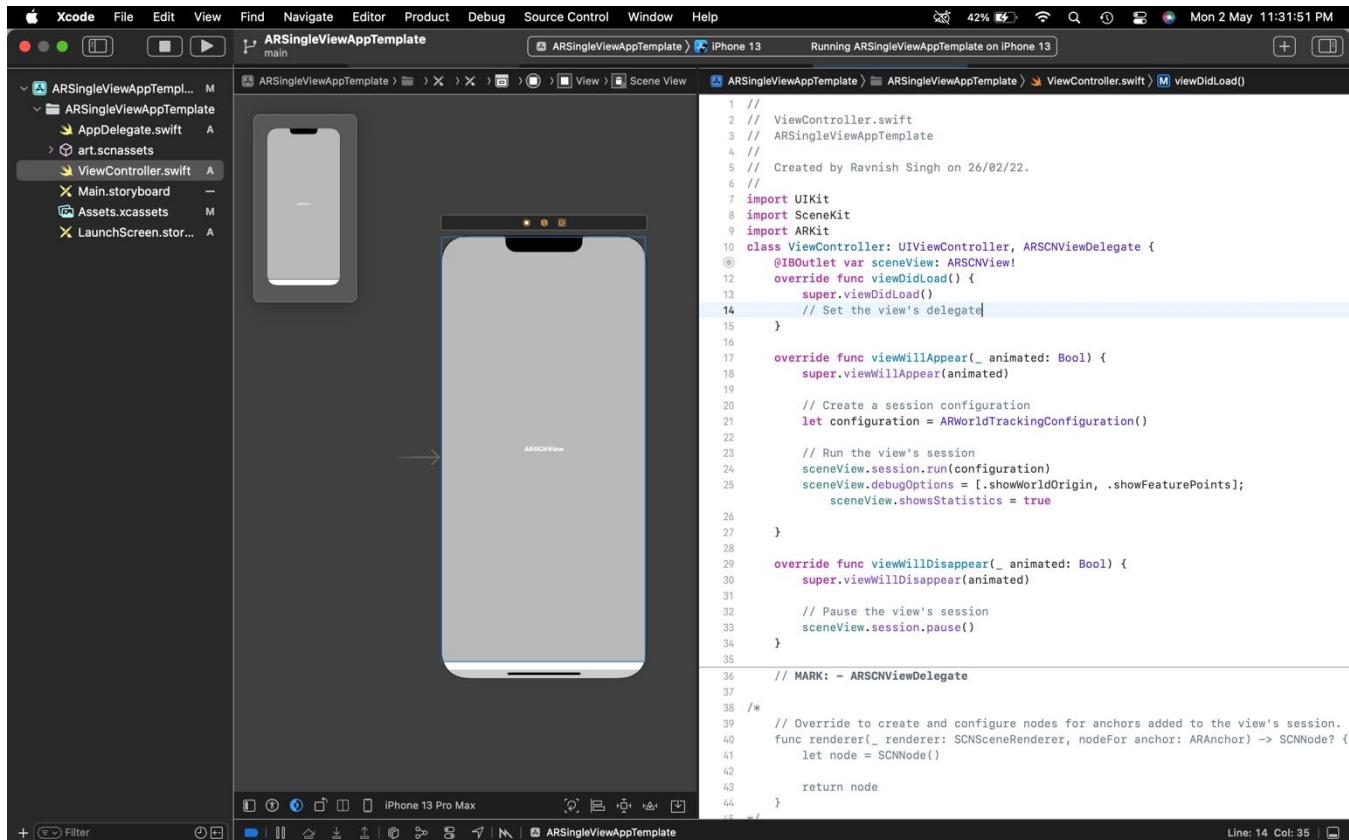
Simulator Output:



PRACTICAL – 2

Create an app with the Single View App Template.

ViewController.swift:



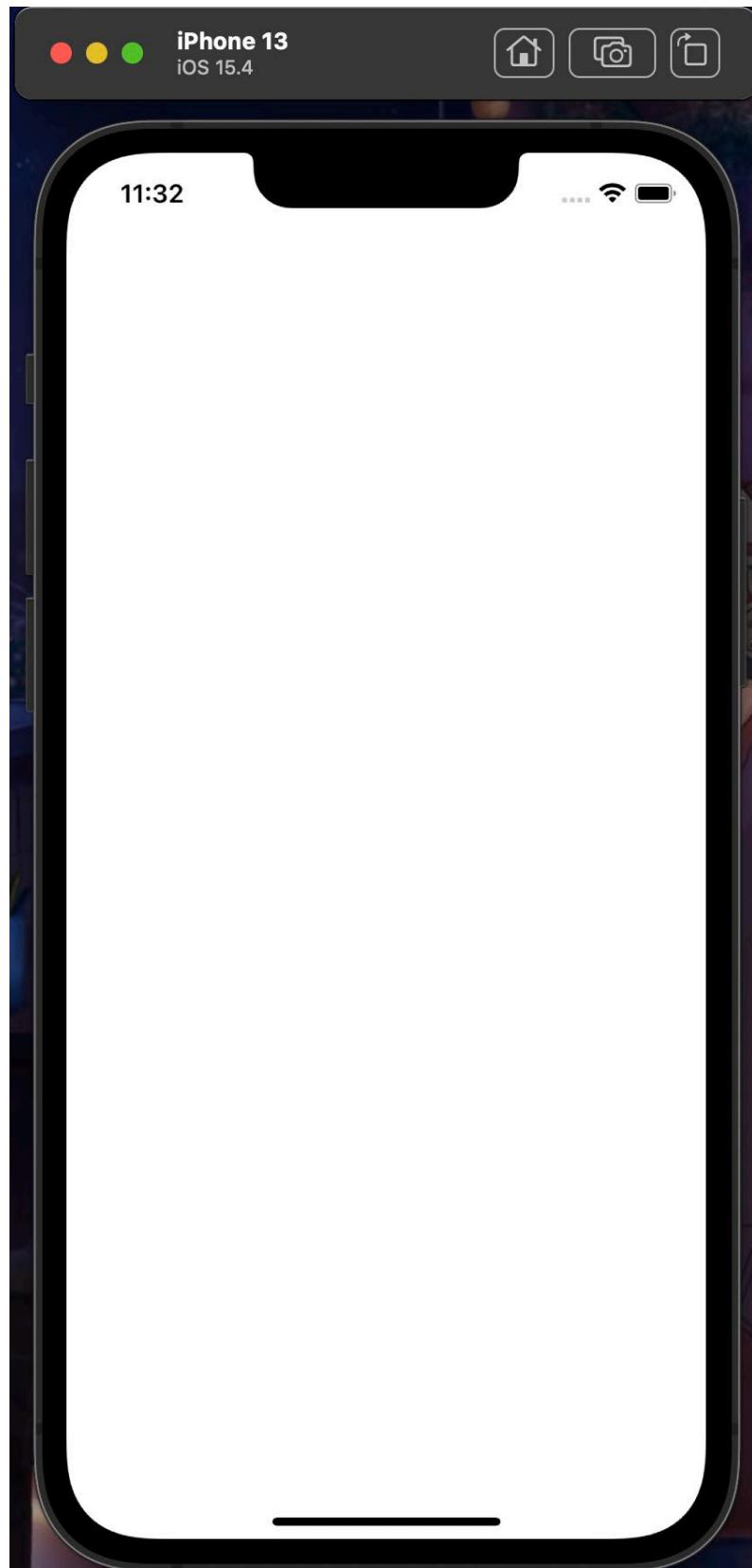
```

1 // ViewController.swift
2 // ARSingleViewAppTemplate
3 // ViewController.swift
4 // ARSingleViewAppTemplate
5 // Created by Ravnish Singh on 26/02/22.
6 //
7 import UIKit
8 import SceneKit
9 import ARKit
10 class ViewController: UIViewController, ARSCNViewDelegate {
11     @IBOutlet var sceneView: ARSCNView!
12     override func viewDidLoad() {
13         super.viewDidLoad()
14         // Set the view's delegate
15     }
16
17     override func viewWillAppear(_ animated: Bool) {
18         super.viewWillAppear(animated)
19
20         // Create a session configuration
21         let configuration = ARWorldTrackingConfiguration()
22
23         // Run the view's session
24         sceneView.session.run(configuration)
25         sceneView.debugOptions = [.showWorldOrigin, .showFeaturePoints];
26         sceneView.showsStatistics = true
27     }
28
29     override func viewWillDisappear(_ animated: Bool) {
30         super.viewWillDisappear(animated)
31
32         // Pause the view's session
33         sceneView.session.pause()
34     }
35
36     // MARK: - ARSCNViewDelegate
37
38 /*
39    // Override to create and configure nodes for anchors added to the view's session.
40    func renderer(_ renderer: SCNSceneRenderer, nodeFor anchor: ARAnchor) -> SCNNode? {
41        let node = SCNNode()
42
43        return node
44    }
45 */

```

Line: 14 Col: 35

Simulator Output:



PRACTICAL – 3

Create an AR app to Add Various Nodes.

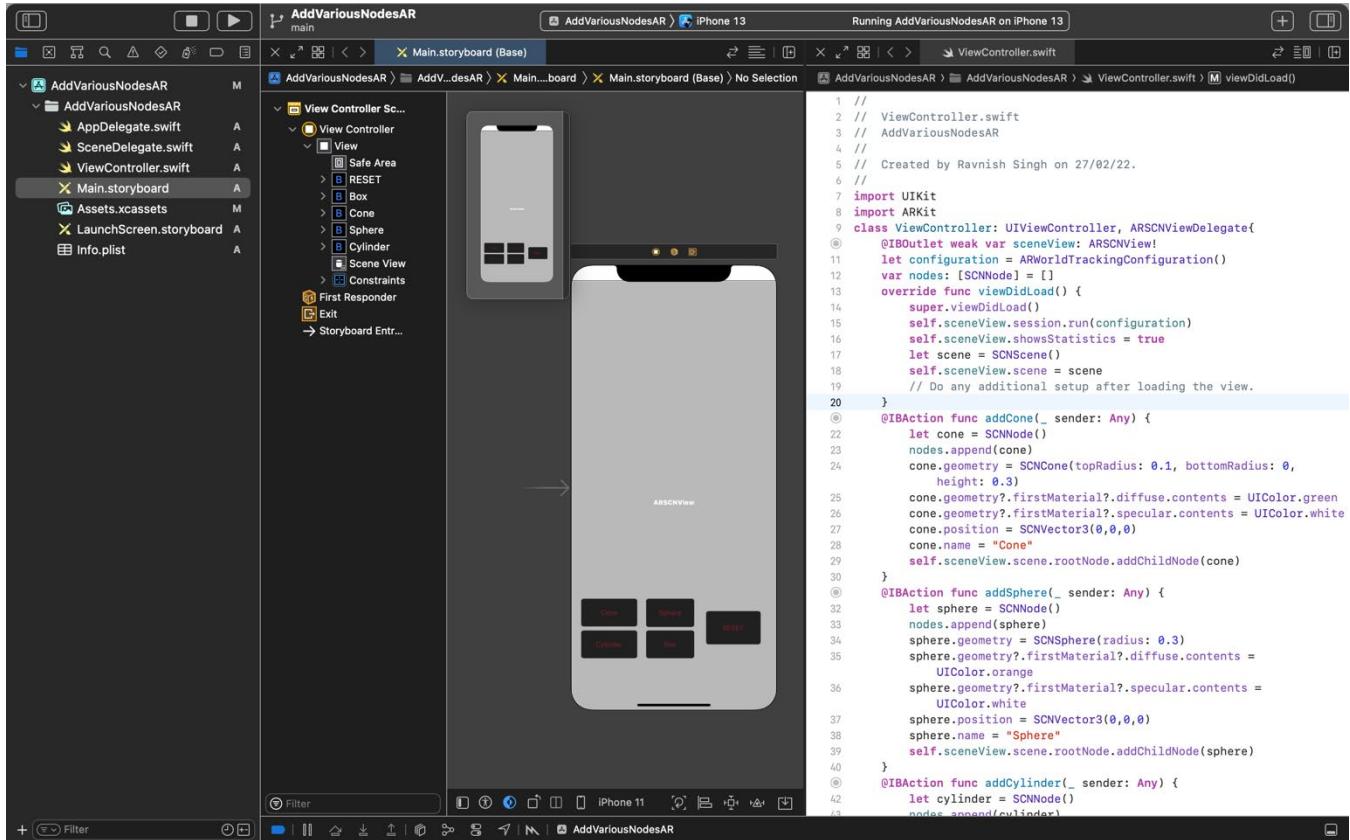
ViewController.swift:

```

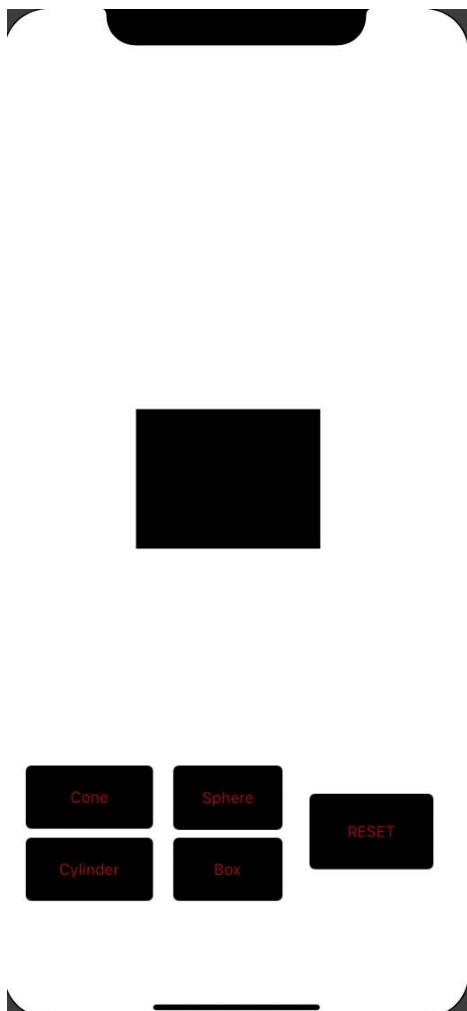
// ViewController.swift
import UIKit
import ARKit
class ViewController: UIViewController, ARSCNViewDelegate {
    @IBOutlet weak var sceneView: ARSCNView!
    let configuration = ARWorldTrackingConfiguration()
    var nodes: [SCNNode] = []
    override func viewDidLoad() {
        super.viewDidLoad()
        sceneView.session.run(configuration)
        sceneView.showsStatistics = true
        let scene = SCNScene()
        self.sceneView.scene = scene
        // Do any additional setup after loading the view.
    }
    @IBAction func addCone(_ sender: Any) {
        let cone = SCNNode()
        nodes.append(cone)
        cone.geometry = SCNConcavePolyhedron()
        cone.geometry?.firstMaterial?.diffuse.contents = UIColor.green
        cone.geometry?.firstMaterial?.specular.contents = UIColor.white
        cone.position = SCNVector3(0,0,0)
        cone.name = "Cone"
        self.sceneView.scene.rootNode.addChildNode(cone)
    }
    @IBAction func addSphere(_ sender: Any) {
        let sphere = SCNNode()
        nodes.append(sphere)
        sphere.geometry = SCNSphere(radius: 0.3)
        sphere.geometry?.firstMaterial?.diffuse.contents = UIColor.orange
        sphere.geometry?.firstMaterial?.specular.contents = UIColor.white
        sphere.position = SCNVector3(0,0,0)
        sphere.name = "Sphere"
        self.sceneView.scene.rootNode.addChildNode(sphere)
    }
    @IBAction func addCylinder(_ sender: Any) {
        let cylinder = SCNNode()
        nodes.append(cylinder)
        cylinder.geometry = SCNCylinder(radius: 0.2, height: 0.3)
        cylinder.geometry?.firstMaterial?.specular.contents = UIColor.white
        cylinder.geometry?.firstMaterial?.diffuse.contents = UIColor.cyan
        cylinder.position = SCNVector3(0,0,0)
        cylinder.name = "Cylinder"
        self.sceneView.scene.rootNode.addChildNode(cylinder)
    }
    @IBAction func addBox(_ sender: Any) {
        let box = SCNNode()
        nodes.append(box)
        box.geometry = SCNNBox(width: 0.1, height: 0.1, length: 0.2, chamferRadius: 0.05)
        box.geometry?.firstMaterial?.diffuse.contents = UIColor.magenta
        box.geometry?.firstMaterial?.specular.contents = UIColor.white
        box.position = SCNVector3(0,0,0)
        box.name = "Box"
        self.sceneView.scene.rootNode.addChildNode(box)
    }
    @IBAction func resetView(_ sender: Any) {
        self.sceneView.session.pause()
        self.sceneView.scene.rootNode.enumerateChildNodes{(node, _) in
            for node in nodes {
                node.removeFromParentNode()
            }
        }
        self.sceneView.session.run(configuration, options: [.removeExistingAnchors,.resetTracking])
    }
}

```

Line: 44 Col: 65 |



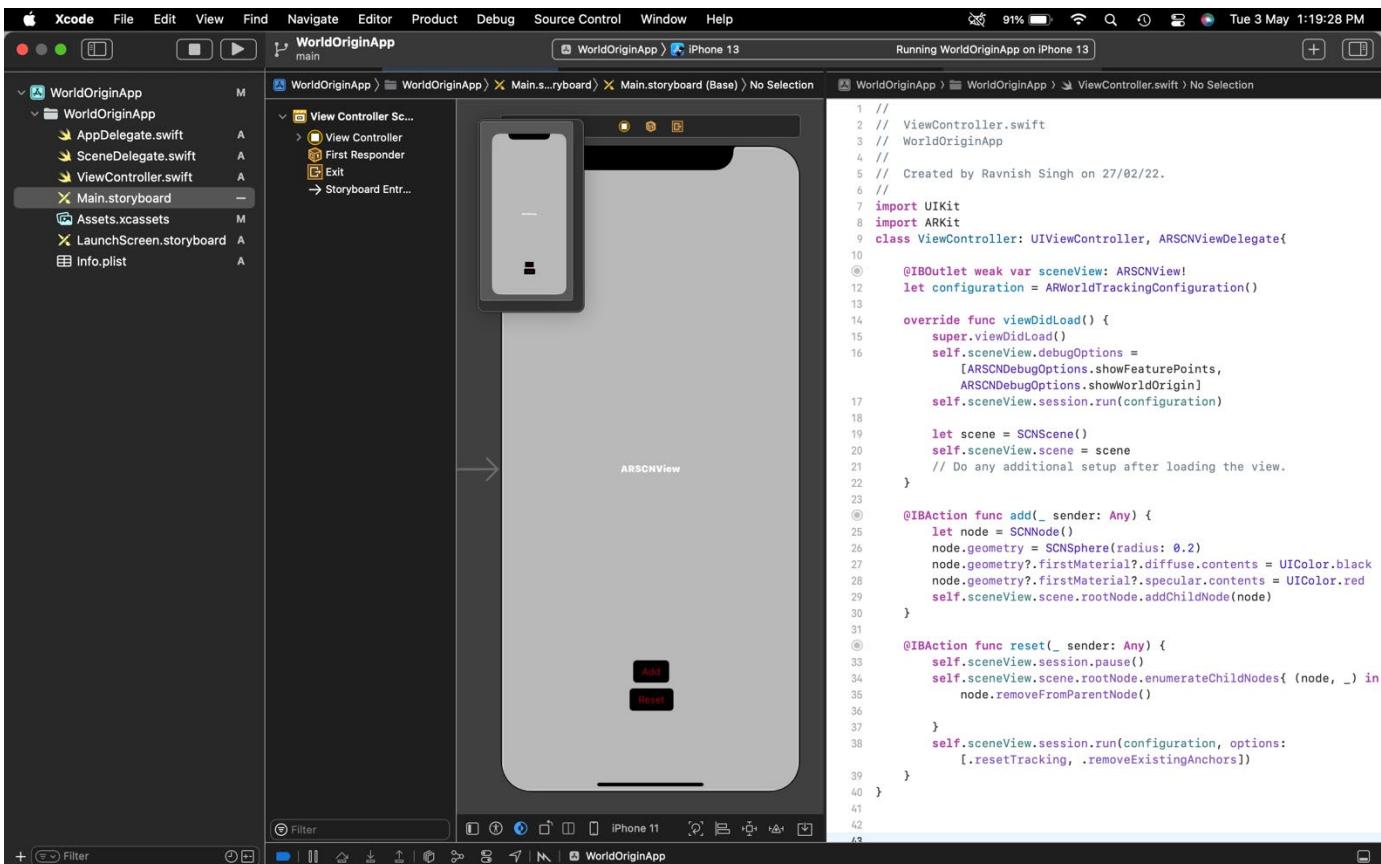
OUTPUT-



PRACTICAL – 4

Create an AR app to display and reset the World Origin.

ViewController.swift:



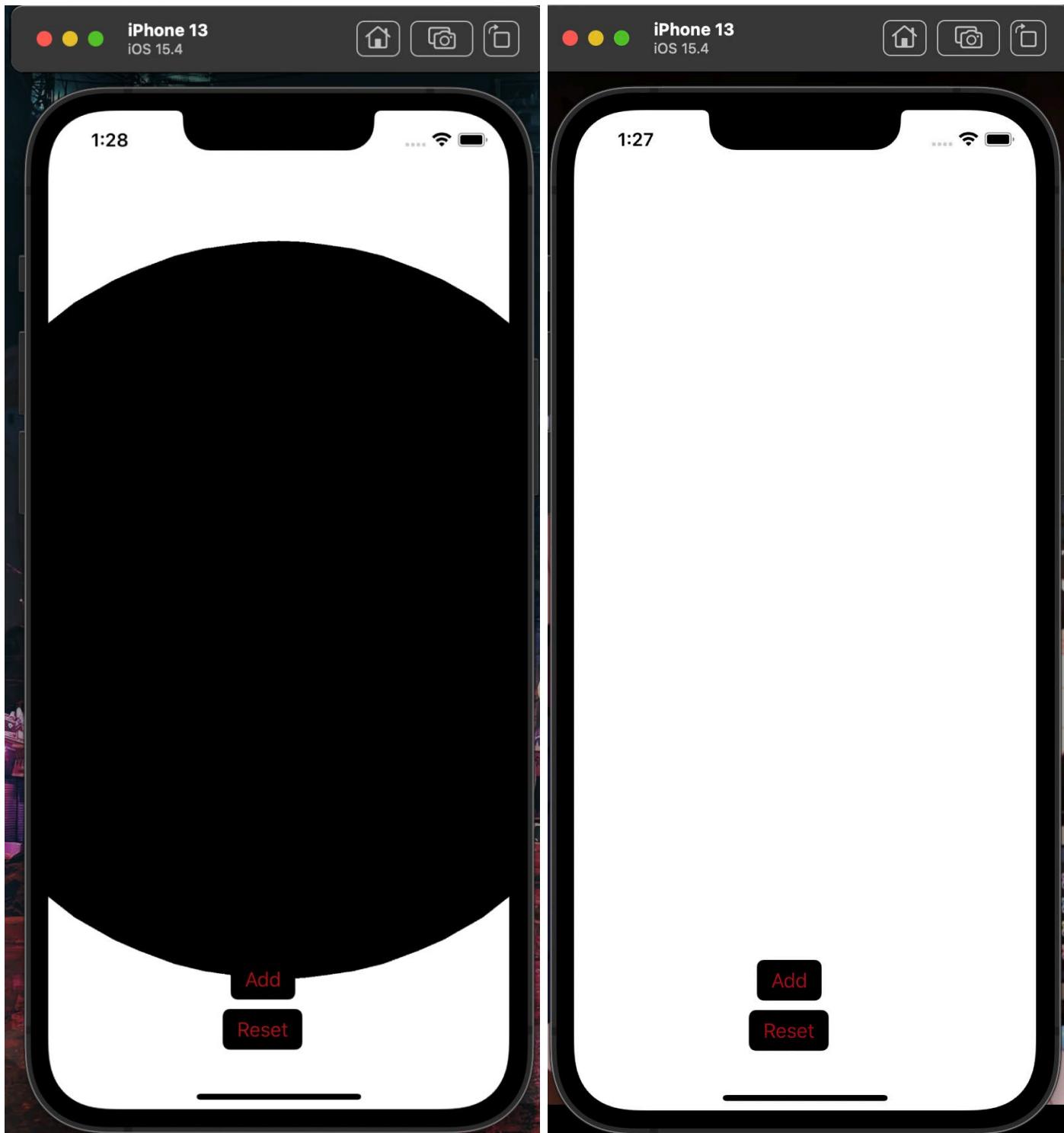
The screenshot shows the Xcode interface with the project 'WorldOriginApp' open. The left sidebar shows files like AppDelegate.swift, SceneDelegate.swift, ViewController.swift, Main.storyboard, Assets.xcassets, LaunchScreen.storyboard, and Info.plist. The storyboard preview on the right shows a smartphone screen with an ARSCNView containing two buttons labeled 'Add' and 'Reset'. The code editor on the right displays the ViewController.swift file, which contains code for initializing ARKit, adding a SCNNode to the scene, and handling 'Add' and 'Reset' actions.

```

1 // ViewController.swift
2 // WorldOriginApp
3 // Created by Ravnish Singh on 27/02/22.
4 //
5 import UIKit
6 import ARKit
7 class ViewController: UIViewController, ARSCNViewDelegate {
8     @IBOutlet weak var sceneView: ARSCNView!
9     let configuration = ARWorldTrackingConfiguration()
10
11    override func viewDidLoad() {
12        super.viewDidLoad()
13        self.sceneView.debugOptions =
14            [ARSCNDebugOptions.showFeaturePoints,
15             ARSCNDebugOptions.showWorldOrigin]
16        self.sceneView.session.run(configuration)
17
18        let scene = SCNScene()
19        self.sceneView.scene = scene
20        // Do any additional setup after loading the view.
21    }
22
23    @IBAction func add(_ sender: Any) {
24        let node = SCNNode()
25        node.geometry = SCNSphere(radius: 0.2)
26        node.geometry?.firstMaterial?.diffuse.contents = UIColor.black
27        node.geometry?.firstMaterial?.specular.contents = UIColor.red
28        self.sceneView.scene.rootNode.addChildNode(node)
29    }
30
31    @IBAction func reset(_ sender: Any) {
32        self.sceneView.session.pause()
33        self.sceneView.scene.rootNode.enumerateChildNodes { (node, _) in
34            node.removeFromParentNode()
35        }
36
37        self.sceneView.session.run(configuration, options:
38            [.resetTracking, .removeExistingAnchors])
39    }
40
41
42
43

```

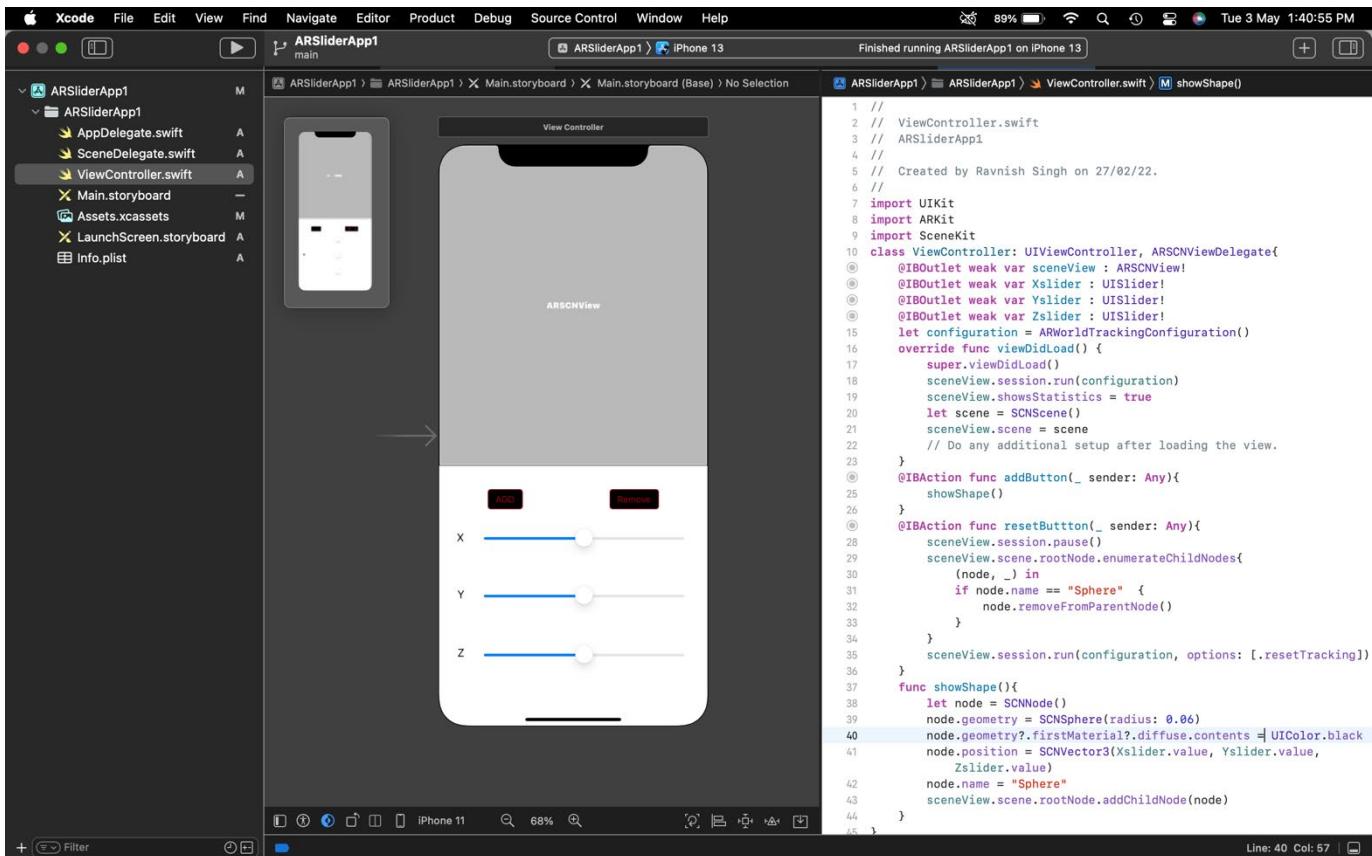
Simulator Output:



PRACTICAL – 5

Create an app to change position of object using Slider.

ViewController.swift:



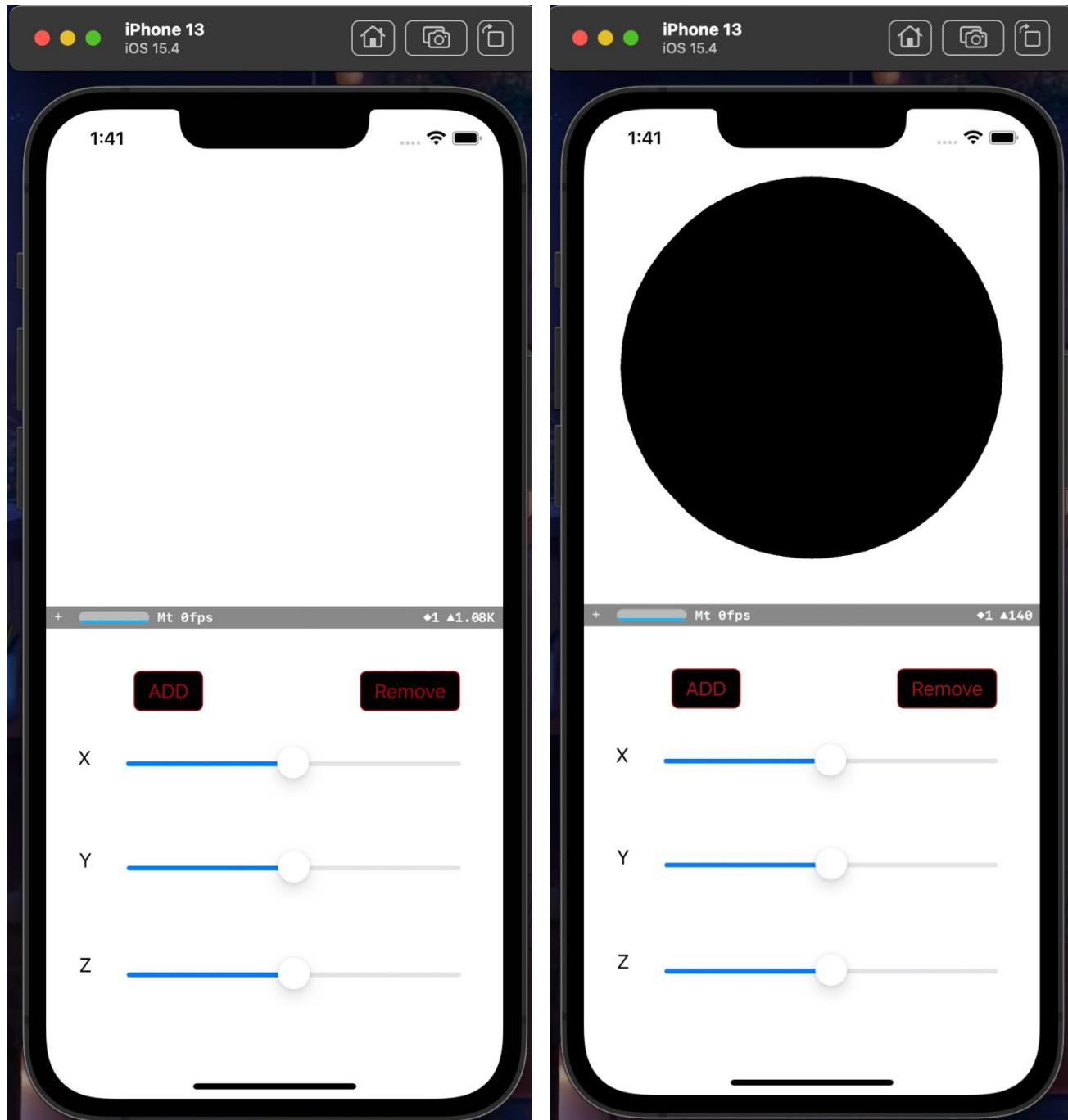
```

1 // ViewController.swift
2 // ARSliderApp1
3 // ARSliderApp1
4 //
5 // Created by Ravnish Singh on 27/02/22.
6 //
7 import UIKit
8 import ARKit
9 import SceneKit
10 class ViewController: UIViewController, ARSCNViewDelegate{
11     @IBOutlet weak var sceneView : ARSCNView!
12     @IBOutlet weak var Xslider : UISlider!
13     @IBOutlet weak var Yslider : UISlider!
14     @IBOutlet weak var Zslider : UISlider!
15     let configuration = ARWorldTrackingConfiguration()
16     override func viewDidLoad() {
17         super.viewDidLoad()
18         sceneView.session.run(configuration)
19         sceneView.showsStatistics = true
20         let scene = SCNScene()
21         sceneView.scene = scene
22         // Do any additional setup after loading the view.
23     }
24     @IBAction func addButton(_ sender: Any){
25         showShape()
26     }
27     @IBAction func resetButton(_ sender: Any){
28         sceneView.session.pause()
29         sceneView.scene.rootNode.enumerateChildNodes{
30             (node, _) in
31             if node.name == "Sphere" {
32                 node.removeFromParentNode()
33             }
34         }
35         sceneView.session.run(configuration, options: [.resetTracking])
36     }
37     func showShape(){
38         let node = SCNNode()
39         node.geometry = SCNSphere(radius: 0.06)
40         node.geometry?.firstMaterial?.diffuse.contents = UIColor.black
41         node.position = SCNVector3(Xslider.value, Yslider.value,
42                                   Zslider.value)
43         node.name = "Sphere"
44         sceneView.scene.rootNode.addChildNode(node)
45     }
}

```

Line: 40 Col: 57 |

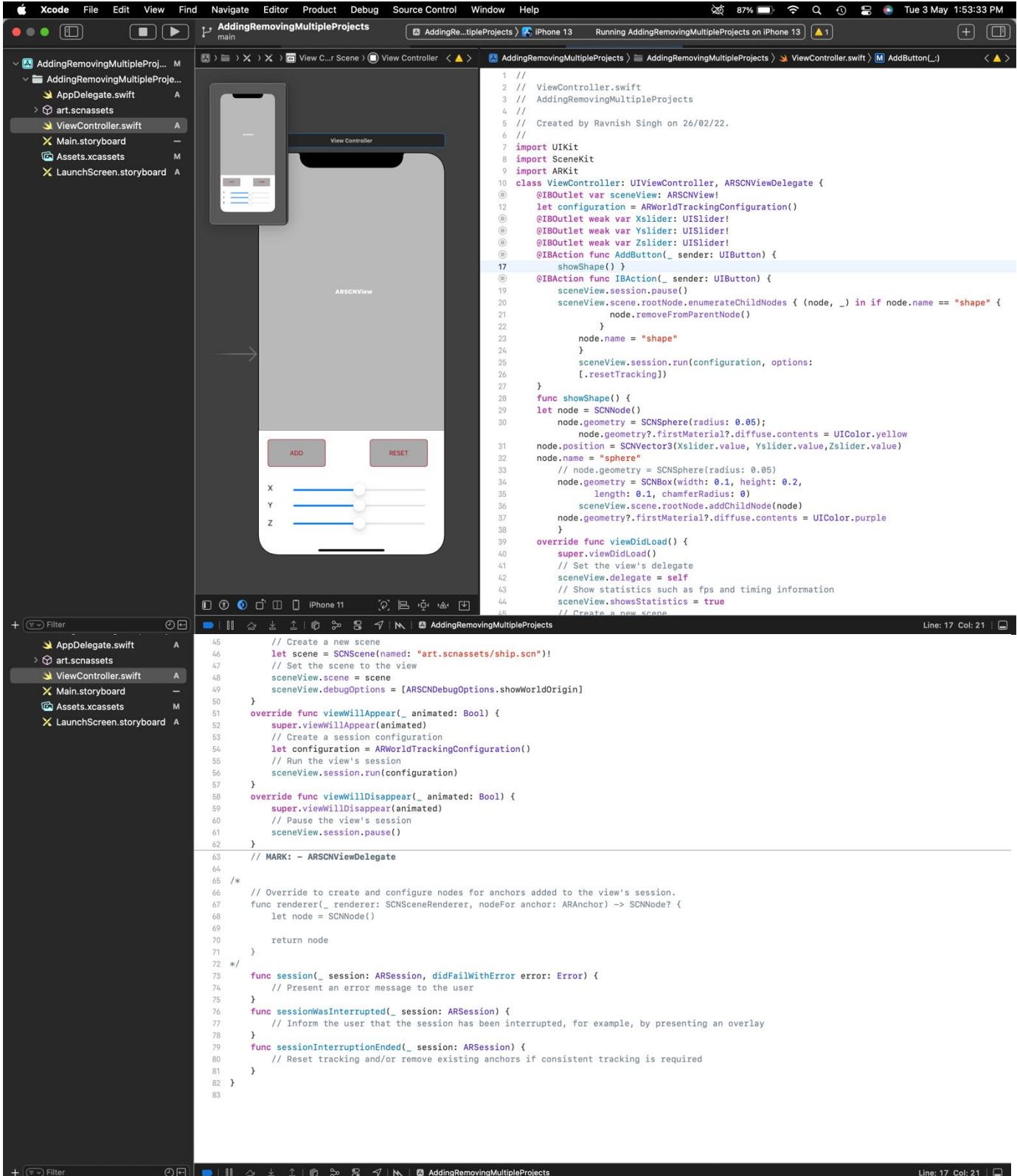
Simulator Output:



PRACTICAL – 6

Create an app to add and remove multiple objects.

ViewController.swift:



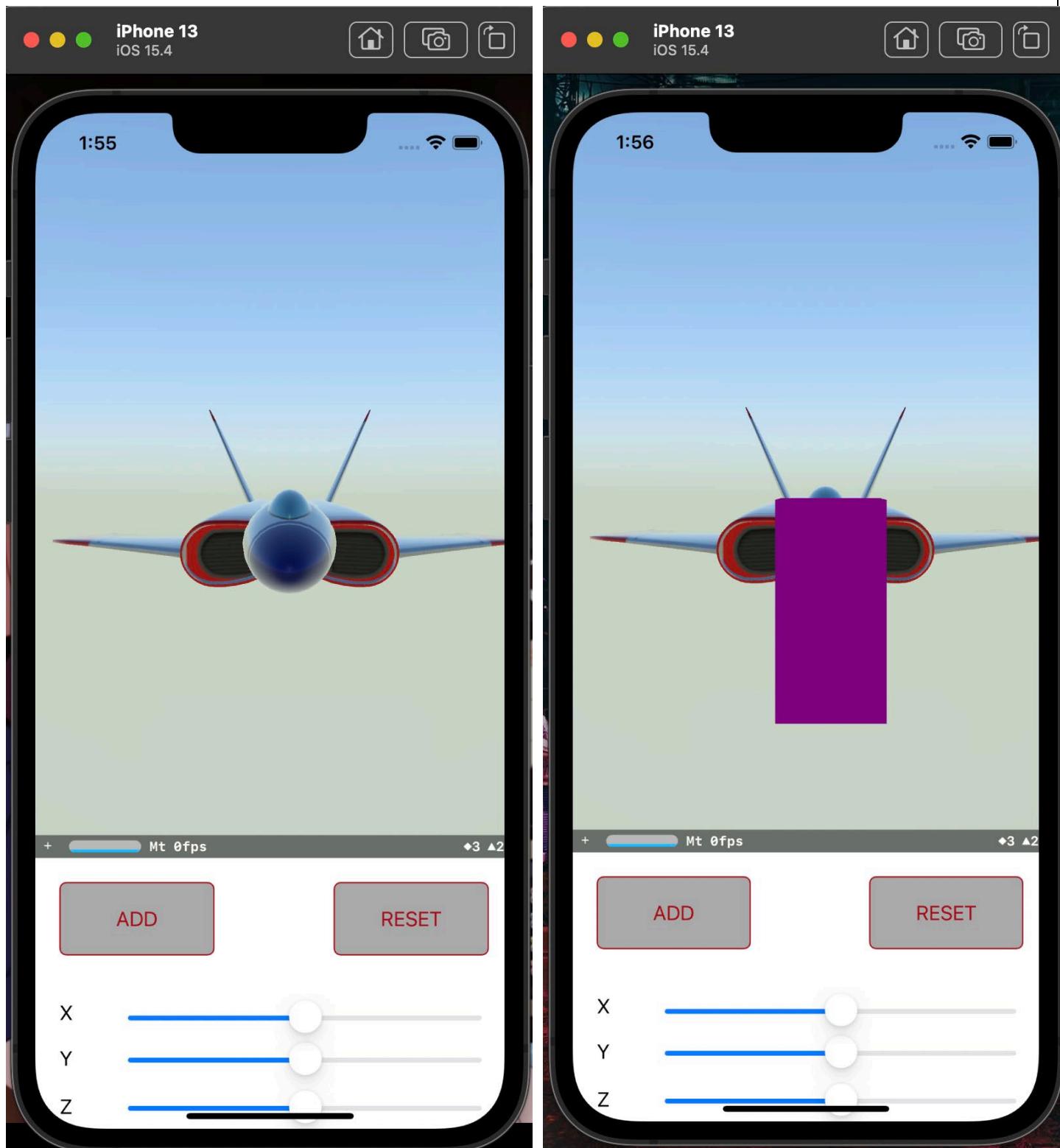
The screenshot shows the Xcode interface with the project "AddingRemovingMultipleProjects" open. The storyboard is displayed on the left, showing a single view controller with a title bar "View Controller". The main area shows the storyboard interface with two buttons ("ADD" and "RESET") and three sliders (X, Y, Z) for controlling object position. The code editor on the right contains the ViewController.swift file, which defines a class that interacts with an ARSCNViewDelegate to handle scene changes and object addition/removal based on user input from the storyboard controls.

```

1 // ViewController.swift
2 // AddingRemovingMultipleProjects
3 // Created by Ravnish Singh on 26/02/22.
4 //
5 import UIKit
6 import SceneKit
7 import ARKit
8
9 class ViewController: UIViewController, ARSCNViewDelegate {
10     @IBOutlet var sceneView: ARSCNView!
11     let configuration = ARWorldTrackingConfiguration()
12     @IBOutlet weak var Xslider: UISlider!
13     @IBOutlet weak var Yslider: UISlider!
14     @IBOutlet weak var Zslider: UISlider!
15     @IBAction func AddButton(_ sender: UIButton) {
16         showShape()
17     }
18     @IBAction func IBAction(_ sender: UIButton) {
19         sceneView.session.pause()
20         sceneView.scene.rootNode.enumerateChildNodes { (node, _) in if node.name == "shape" {
21             node.removeFromParentNode()
22         }
23         node.name = "shape"
24     }
25     sceneView.session.run(configuration, options: [.resetTracking])
26 }
27
28 func showShape() {
29     let node = SCNNode()
30     node.geometry = SCNSphere(radius: 0.05);
31     node.geometry?.firstMaterial?.diffuse.contents = UIColor.yellow
32     node.position = SCNVector3(Xslider.value, Yslider.value, Zslider.value)
33     node.name = "sphere"
34     // node.geometry = SCNSphere(radius: 0.05)
35     // node.geometry = SCNBox(width: 0.1, height: 0.2,
36     // length: 0.1, chamferRadius: 0)
37     // sceneView.scene.rootNode.addChildNode(node)
38     node.geometry?.firstMaterial?.diffuse.contents = UIColor.purple
39
40     override func viewDidLoad() {
41         super.viewDidLoad()
42         // Set the view's delegate
43         sceneView.delegate = self
44         // Show statistics such as fps and timing information
45         sceneView.showsStatistics = true
46         // Create a new scene
47         let scene = SCNScene(named: "art.scnassets/ship.scn")!
48         // Set the scene to the view
49         sceneView.scene = scene
50         sceneView.debugOptions = [ARSCNDebugOptions.showWorldOrigin]
51     }
52     override func viewDidAppear(_ animated: Bool) {
53         super.viewDidAppear(animated)
54         // Create a session configuration
55         let configuration = ARWorldTrackingConfiguration()
56         // Run the view's session
57         sceneView.session.run(configuration)
58     }
59     override func viewWillDisappear(_ animated: Bool) {
60         super.viewWillDisappear(animated)
61         // Pause the view's session
62         sceneView.session.pause()
63     }
64     // MARK: - ARSCNviewDelegate
65
66     /*
67     // Override to create and configure nodes for anchors added to the view's session.
68     func renderer(_ renderer: SCNSceneRenderer, nodeFor anchor: ARAnchor) -> SCNNode? {
69         let node = SCNNode()
70
71         return node
72     }
73
74     func session(_ session: ARSession, didFailWithError error: Error) {
75         // Present an error message to the user
76     }
77     func sessionWasInterrupted(_ session: ARSession) {
78         // Inform the user that the session has been interrupted, for example, by presenting an overlay
79     }
80     func sessionInterruptionEnded(_ session: ARSession) {
81         // Reset tracking and/or remove existing anchors if consistent tracking is required
82     }
83
84 */
85 }

```

Simulator Output:



PRACTICAL – 7

Create an app to display Text in AR.

ViewController.swift:

The screenshot shows the Xcode interface with the following details:

- Top Bar:** Xcode, File, Edit, View, Find, Navigate, Editor, Product, Debug, Source Control, Window, Help.
- Project Navigator:** TextARApp PID 14821, CPU 0%, Memory 15.3 MB, Disk 3 MB/s, Network Zero KB/s, FPS.
- Editor:** TextARApp > TextARApp > ViewController.swift. The code is as follows:

```
1 //  
2 // ViewController.swift  
3 // TextARApp  
4 //  
5 // Created by Ravnish Singh on 26/02/22.  
6 //  
7  
8 import UIKit  
9 import ARKit  
10  
11 class ViewController: UIViewController {  
12  
13     @IBOutlet weak var sceneView: ARSCNView!  
14     let configuration = ARWorldTrackingConfiguration()  
15     override func viewDidLoad() {  
16         super.viewDidLoad()  
17         // Do any additional setup after loading the view.  
18         let scene = SCNScene()  
19         sceneView.scene = scene  
20         sceneView.session  
21             .run(configuration)  
22         sceneView.showsStatistics = true  
23  
24         let text = SCNNode()  
25         text.geometry = SCNText(string: "SINGH_RAVNISH",  
26             extrusionDepth: 0.7)  
27         text.geometry?.firstMaterial?.diffuse.contents = UIColor.red  
28         text.position = SCNVector3(0,0,0)  
29         sceneView.scene.rootNode.addChildNode(text)  
30     }  
31 }  
32
```

Preview: An iPhone 11 simulator is shown displaying an AR scene with a red 3D text node reading "SINGH_RAVNISH".

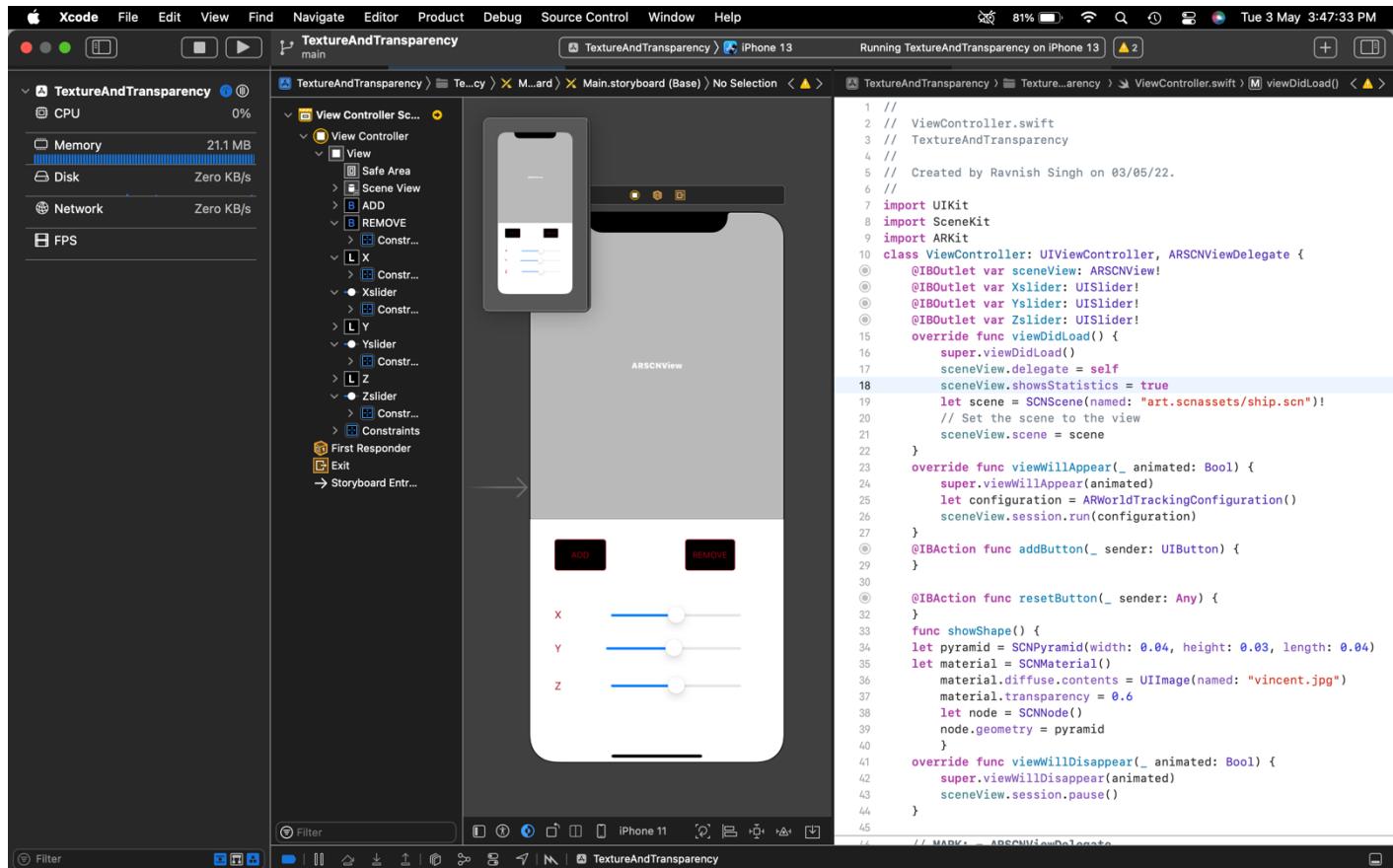
Simulator Output:



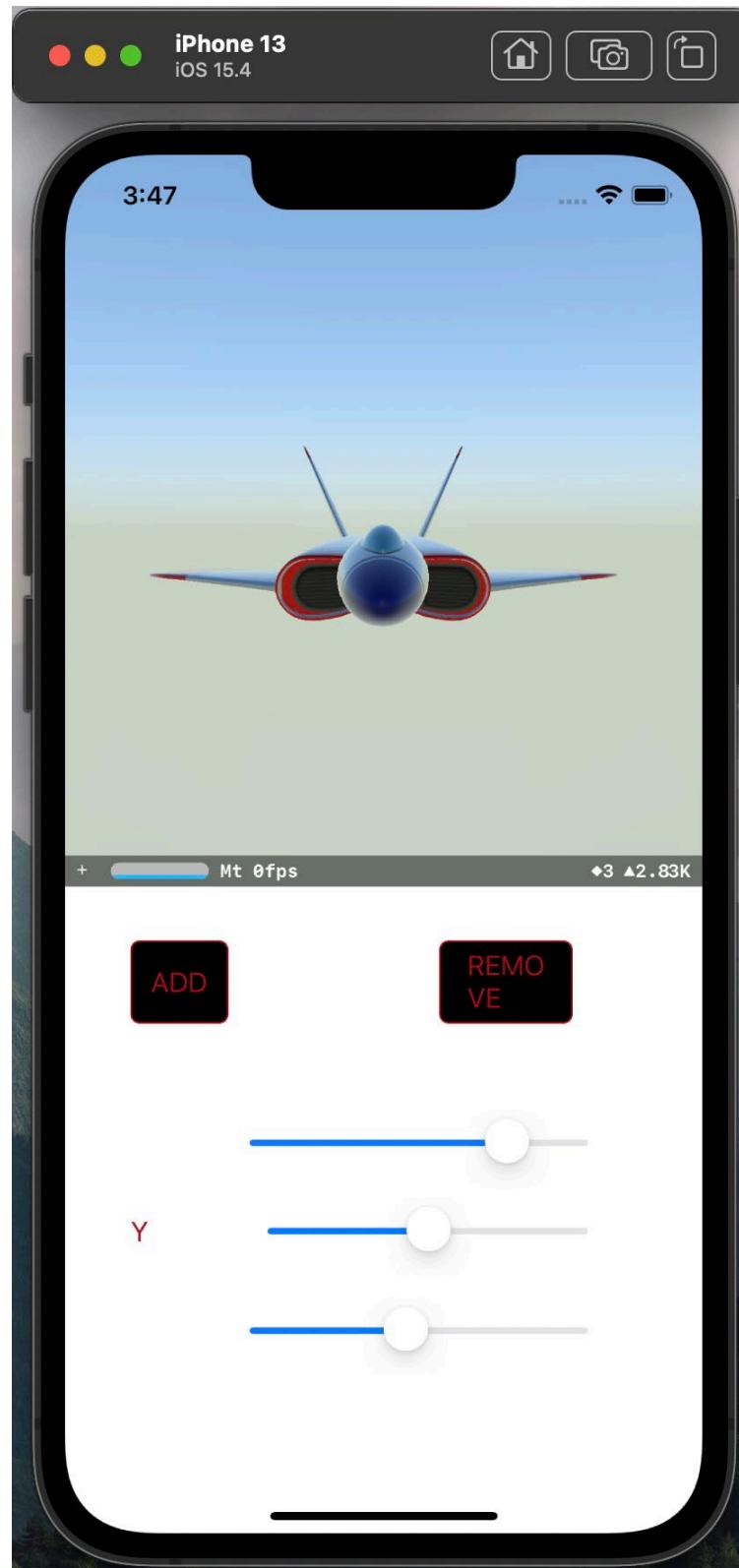
PRACTICAL – 8

Create an app to add textures to shape.

ViewController.swift:



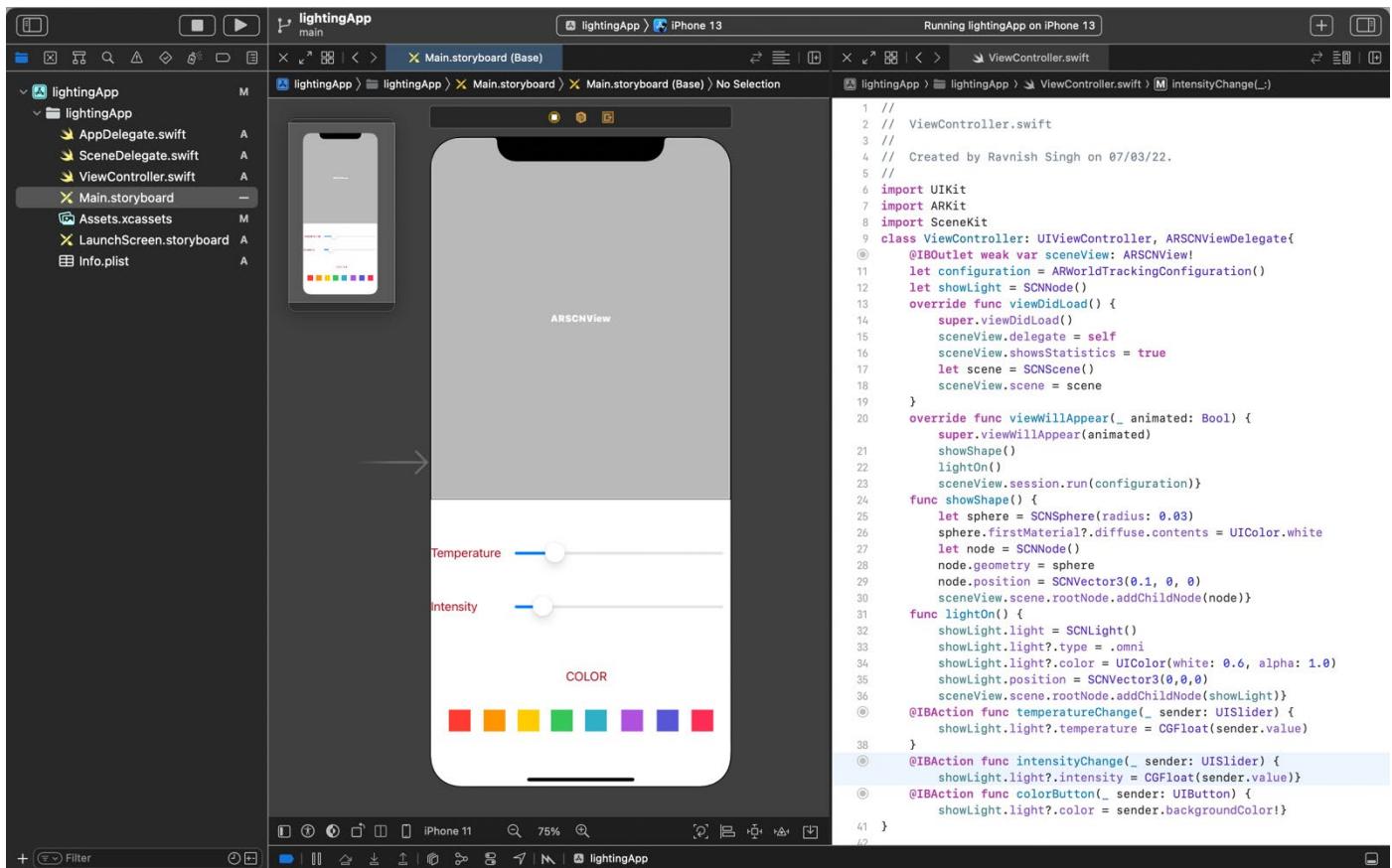
Simulator Output:



PRACTICAL – 9

Create an app to use light, temperature, and intensity to modify the appearance of virtual objects.

ViewController.swift:



The screenshot shows the Xcode interface with the following details:

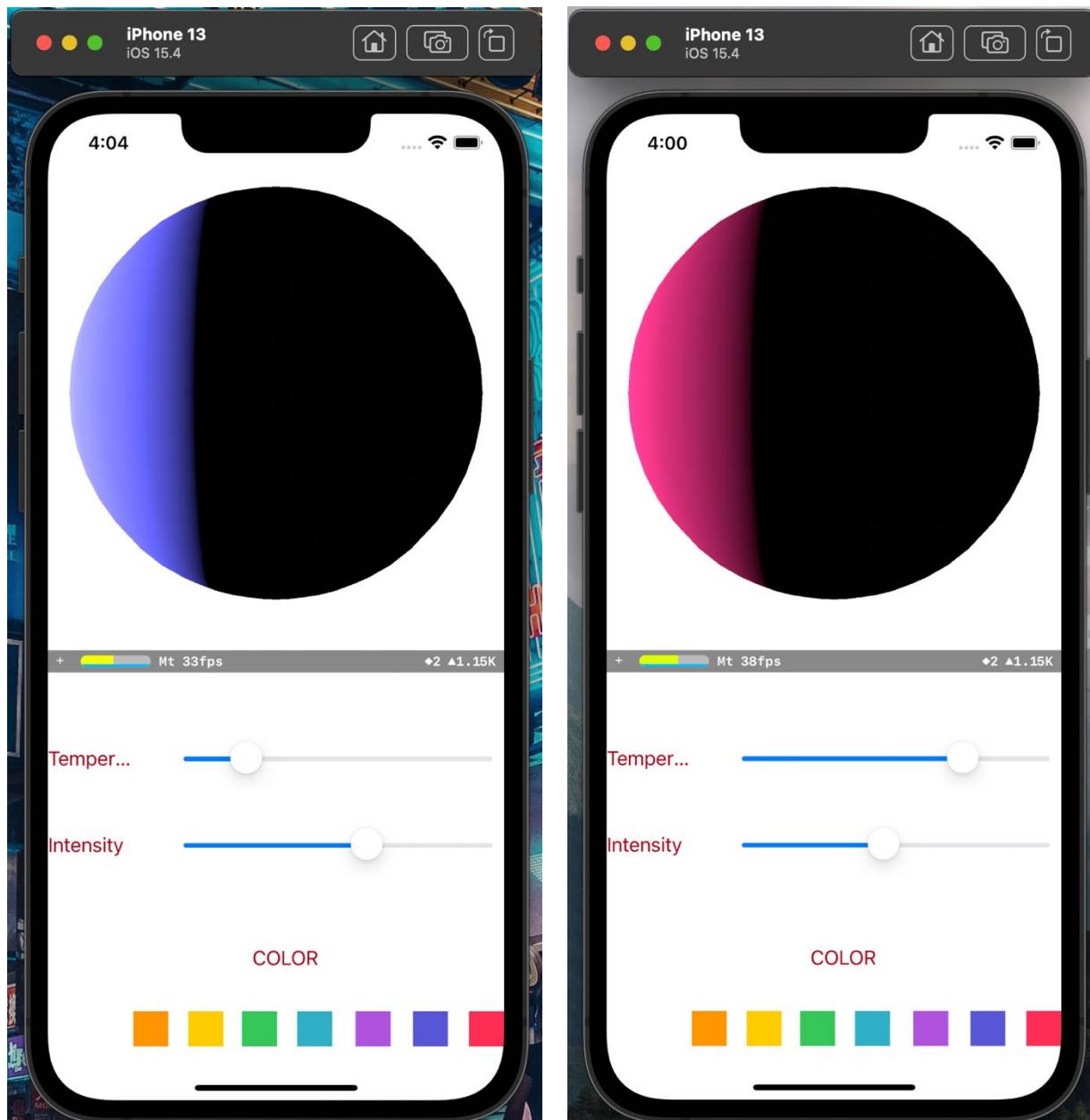
- Project Navigator:** Shows files for the "lightingApp" project, including Main.storyboard, AppDelegate.swift, SceneDelegate.swift, ViewController.swift, Assets.xcassets, LaunchScreen.storyboard, and Info.plist.
- Main.storyboard Preview:** Displays a smartphone screen with an ARSCNView placeholder. Below it, a color picker interface includes two sliders labeled "Temperature" and "Intensity", and a "COLOR" button below a color swatch.
- ViewController.swift Content:** The code implements a UIViewController subclass that conforms to ARSCNViewDelegate. It creates a SCNNode named "showLight" and adds it to the scene. It also handles UI interactions for temperature and intensity changes, as well as a color selection button.

```

1 // 
2 // ViewController.swift
3 // 
4 // Created by Ravnish Singh on 07/03/22.
5 // 
6 import UIKit
7 import ARKit
8 import SceneKit
9 class ViewController: UIViewController, ARSCNViewDelegate{
10     @IBOutlet weak var sceneView: ARSCNView!
11     let configuration = ARWorldTrackingConfiguration()
12     let showLight = SCNNode()
13     override func viewDidLoad() {
14         super.viewDidLoad()
15         sceneView.delegate = self
16         sceneView.showsStatistics = true
17         let scene = SCNScene()
18         sceneView.scene = scene
19     }
20     override func viewWillAppear(_ animated: Bool) {
21         super.viewWillAppear(animated)
22         showShape()
23         lightOn()
24         sceneView.session.run(configuration)
25     }
26     func showShape() {
27         let sphere = SCNSphere(radius: 0.03)
28         sphere.firstMaterial?.diffuse.contents = UIColor.white
29         let node = SCNNode()
30         node.geometry = sphere
31         node.position = SCNVector3(0.1, 0, 0)
32         sceneView.scene.rootNode.addChildNode(node)
33     }
34     func lightOn() {
35         showLight.light = SCNLight()
36         showLight.light?.type = .omni
37         showLight.light?.color = UIColor(white: 0.6, alpha: 1.0)
38         showLight.position = SCNVector3(0,0,0)
39         sceneView.scene.rootNode.addChildNode(showLight)
40     }
41     @IBAction func temperatureChange(_ sender: UISlider) {
42         showLight.light?.temperature = CGFloat(sender.value)
43     }
44     @IBAction func intensityChange(_ sender: UISlider) {
45         showLight.light?.intensity = CGFloat(sender.value)
46     }
47     @IBAction func colorButton(_ sender: UIButton) {
48         showLight.light?.color = sender.backgroundColor!
49     }
50 }

```

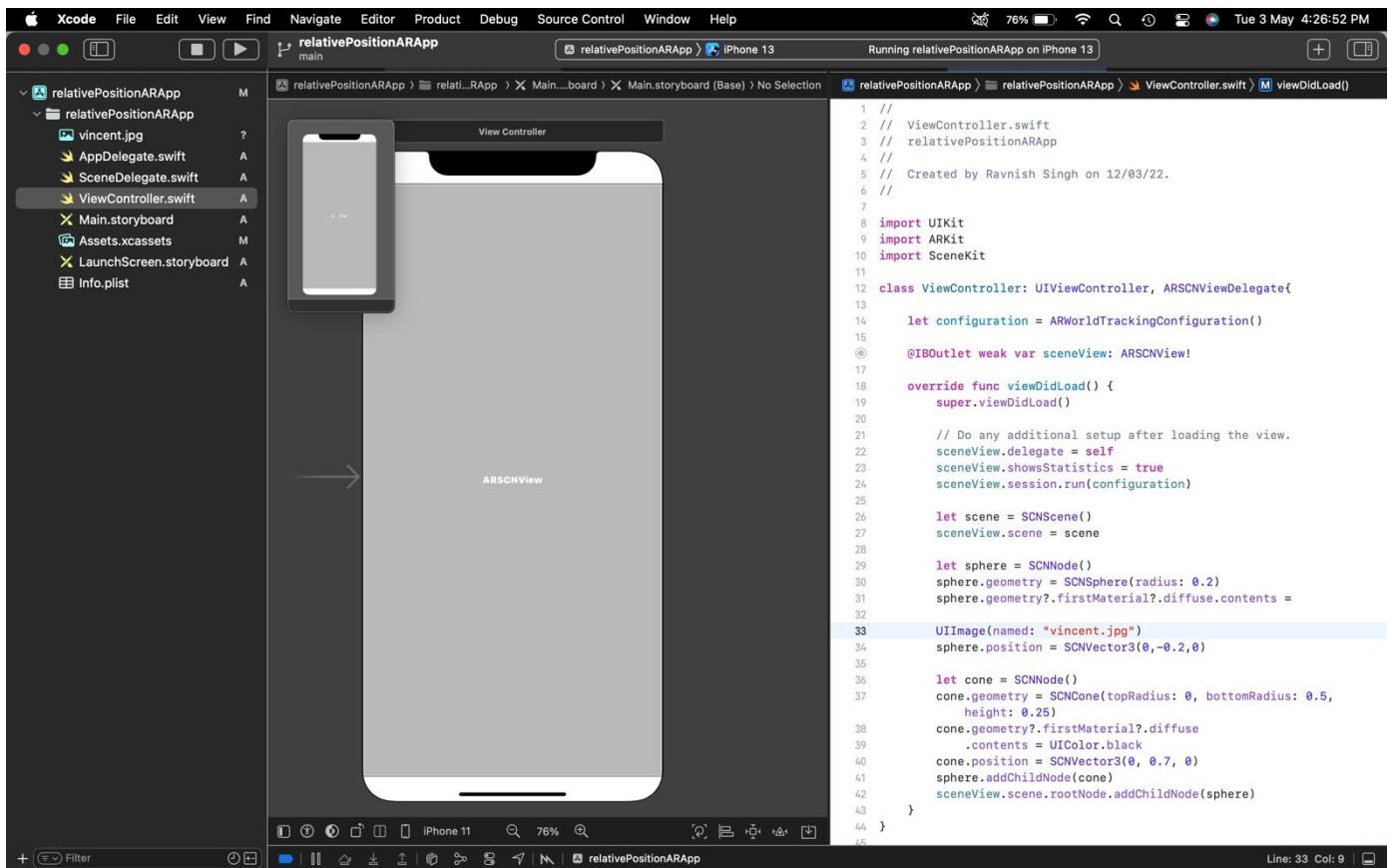
Simulator Output:



PRACTICAL – 10

Create an app that utilizes Positioning in AR.

ViewController.swift:



The screenshot shows the Xcode IDE with the following details:

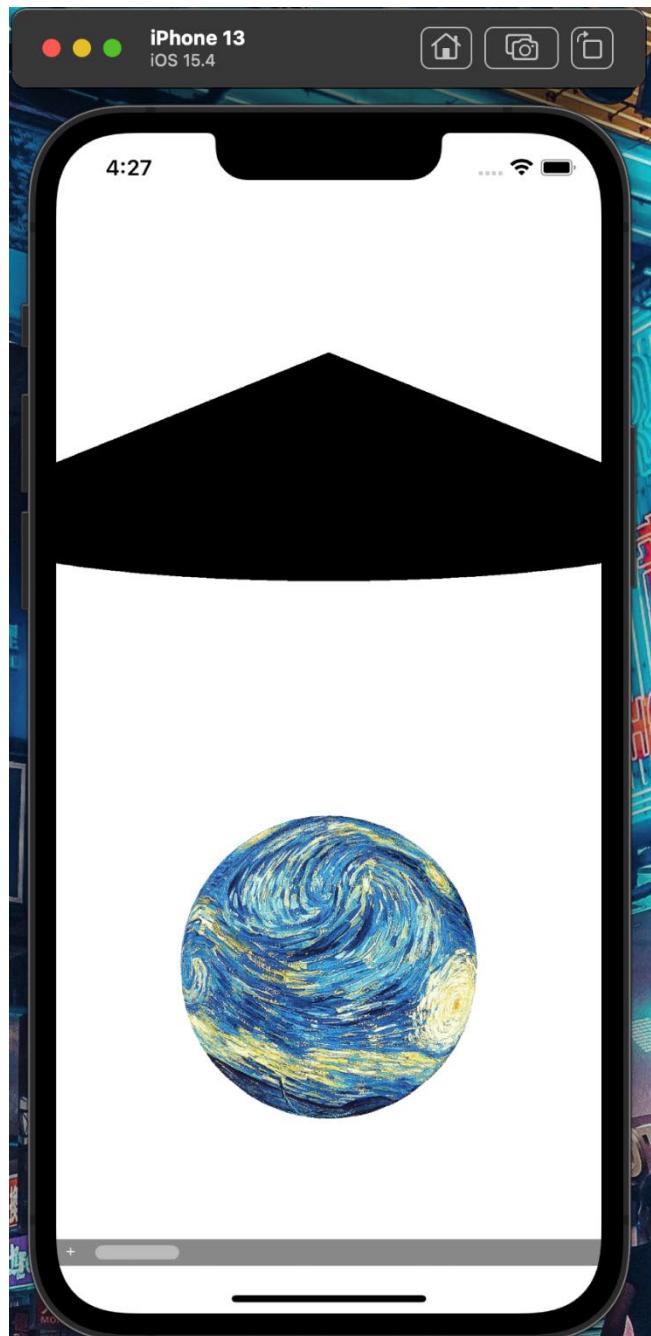
- Project Structure:** The left sidebar shows the project structure for "relativePositionARApp" with files like "viewController.swift" selected.
- Main.storyboard:** The central storyboard editor displays a view controller containing an "ARSCNView".
- ViewController.swift:** The right pane shows the Swift code for the view controller, specifically the `viewDidLoad()` method which initializes an AR scene with a sphere and a cone.

```

1 // 
2 //  ViewController.swift
3 //  relativePositionARApp
4 // 
5 //  Created by Ravnish Singh on 12/03/22.
6 // 
7
8 import UIKit
9 import ARKit
10 import SceneKit
11
12 class ViewController: UIViewController, ARSCNViewDelegate{
13
14     let configuration = ARWorldTrackingConfiguration()
15
16     @IBOutlet weak var sceneView: ARSCNView!
17
18     override func viewDidLoad() {
19         super.viewDidLoad()
20
21         // Do any additional setup after loading the view.
22         sceneView.delegate = self
23         sceneView.showsStatistics = true
24         sceneView.session.run(configuration)
25
26         let scene = SCNScene()
27         sceneView.scene = scene
28
29         let sphere = SCNNode()
30         sphere.geometry = SCNSphere(radius: 0.2)
31         sphere.geometry?.firstMaterial?.diffuse.contents =
32
33         UIImage(named: "vincent.jpg")
34         sphere.position = SCNVector3(0, -0.2, 0)
35
36         let cone = SCNNode()
37         cone.geometry = SCNConcavePolyhedron()
38         cone.geometry?.firstMaterial?.diffuse
39             .contents = UIColor.black
40         cone.position = SCNVector3(0, 0.7, 0)
41         sphere.addChildNode(cone)
42
43     }
44 }

```

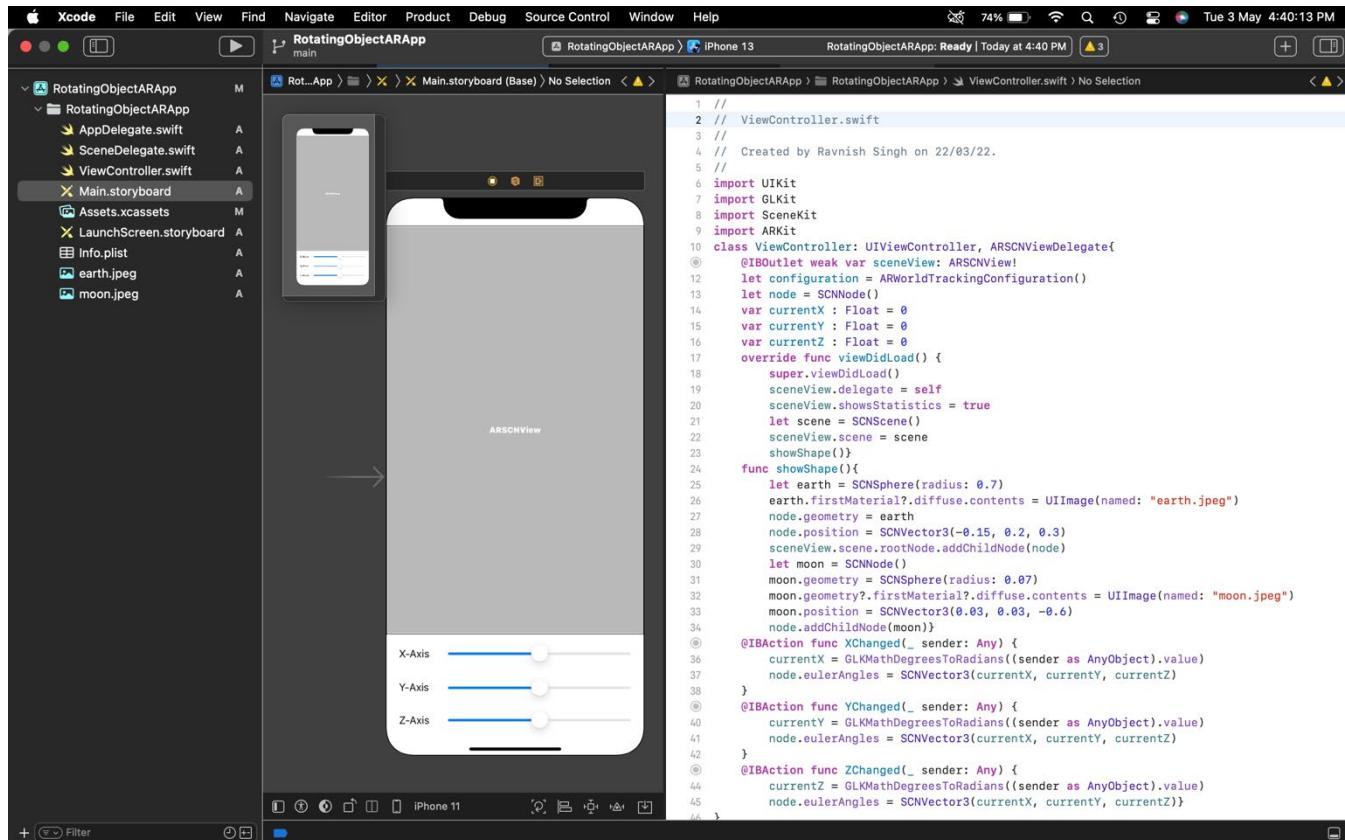
Simulator Output:



PRACTICAL – 11

Create an app to Rotate Objects in AR.

ViewController.swift:



The screenshot shows the Xcode interface with the following details:

- Project Structure:** RotatingObjectARApp contains Main.storyboard, AppDelegate.swift, SceneDelegate.swift, ViewController.swift, Assets.xcassets, and LaunchScreen.storyboard.
- Storyboard:** Main.storyboard is open, showing a single view controller with an ARSCNView. Below the view, there are three sliders labeled X-Axis, Y-Axis, and Z-Axis.
- Code Editor:** ViewController.swift is open, displaying the following Swift code:

```

1 // ViewController: UIViewController, ARSCNViewDelegate
2 // Viewcontroller.swift
3 //
4 // Created by Ravnish Singh on 22/03/22.
5 //
6 import UIKit
7 import GLKit
8 import SceneKit
9 import ARKit
10 class ViewController: UIViewController, ARSCNViewDelegate{
11     @IBOutlet weak var sceneView: ARSCNView!
12     let configuration = ARWorldTrackingConfiguration()
13     let node = SCNNode()
14     var currentX : Float = 0
15     var currentY : Float = 0
16     var currentZ : Float = 0
17     override func viewDidLoad() {
18         super.viewDidLoad()
19         sceneView.delegate = self
20         sceneView.showsStatistics = true
21         let scene = SCNScene()
22         sceneView.scene = scene
23         showShape()
24     }
25     func showShape(){
26         let earth = SCNSphere(radius: 0.7)
27         earth.firstMaterial?.diffuse.contents = UIImage(named: "earth.jpeg")
28         node.geometry = earth
29         node.position = SCNVector3(-0.15, 0.2, 0.3)
30         sceneView.scene.rootNode.addChildNode(node)
31         let moon = SCNNode()
32         moon.geometry = SCNSphere(radius: 0.07)
33         moon.geometry?.firstMaterial?.diffuse.contents = UIImage(named: "moon.jpeg")
34         moon.position = SCNVector3(0.03, 0.03, -0.6)
35         node.addChildNode(moon)
36     }
37     @IBAction func XChanged(_ sender: Any) {
38         currentX = GLKMathDegreesToRadians((sender as AnyObject).value)
39         node.eulerAngles = SCNVector3(currentX, currentY, currentZ)
40     }
41     @IBAction func YChanged(_ sender: Any) {
42         currentY = GLKMathDegreesToRadians((sender as AnyObject).value)
43         node.eulerAngles = SCNVector3(currentX, currentY, currentZ)
44     }
45     @IBAction func ZChanged(_ sender: Any) {
46         currentZ = GLKMathDegreesToRadians((sender as AnyObject).value)
47         node.eulerAngles = SCNVector3(currentX, currentY, currentZ)
48     }
49 }

```

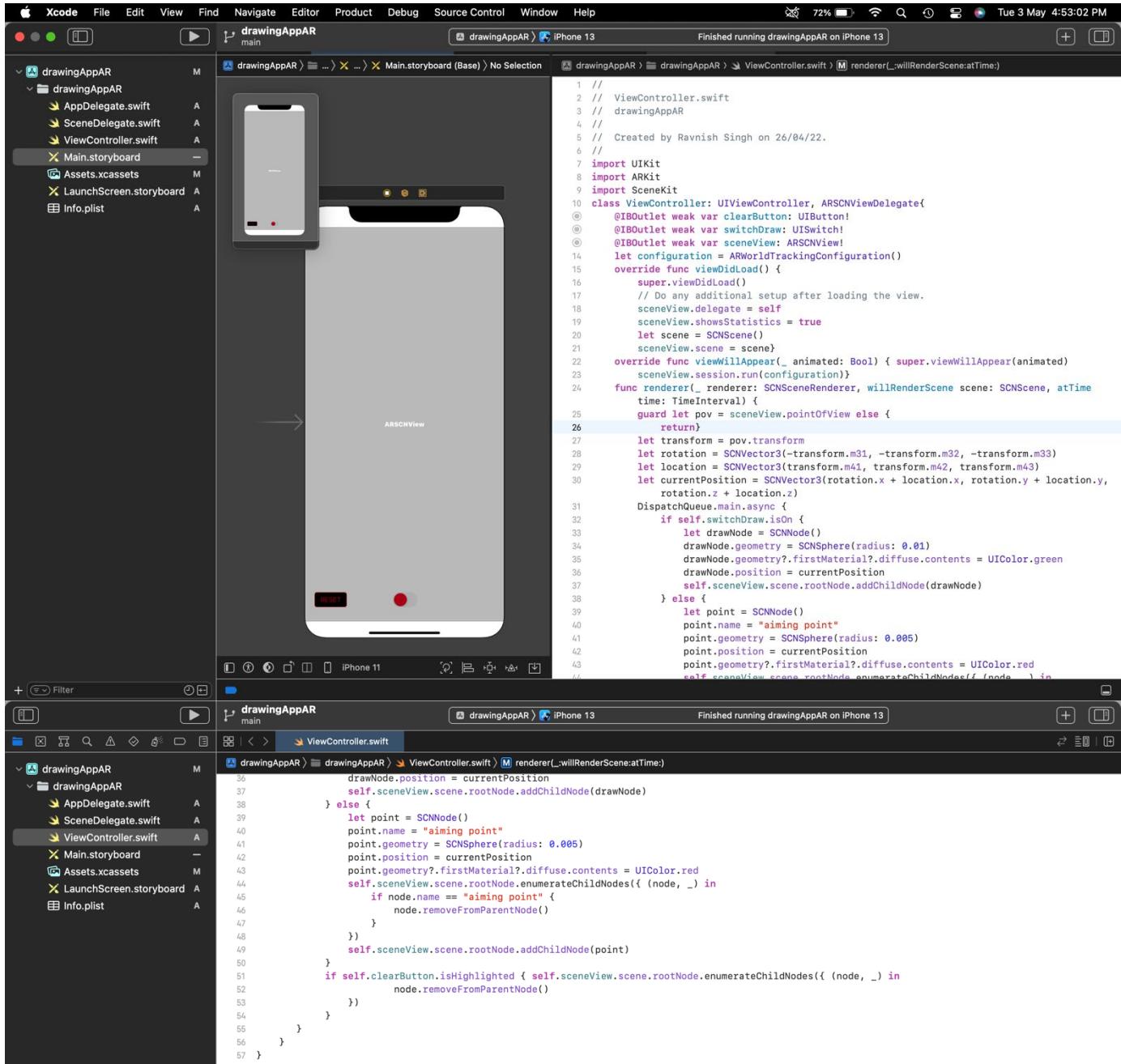
Simulator Output:



PRACTICAL – 12

Create an app for Drawing in AR.

ViewController.swift:



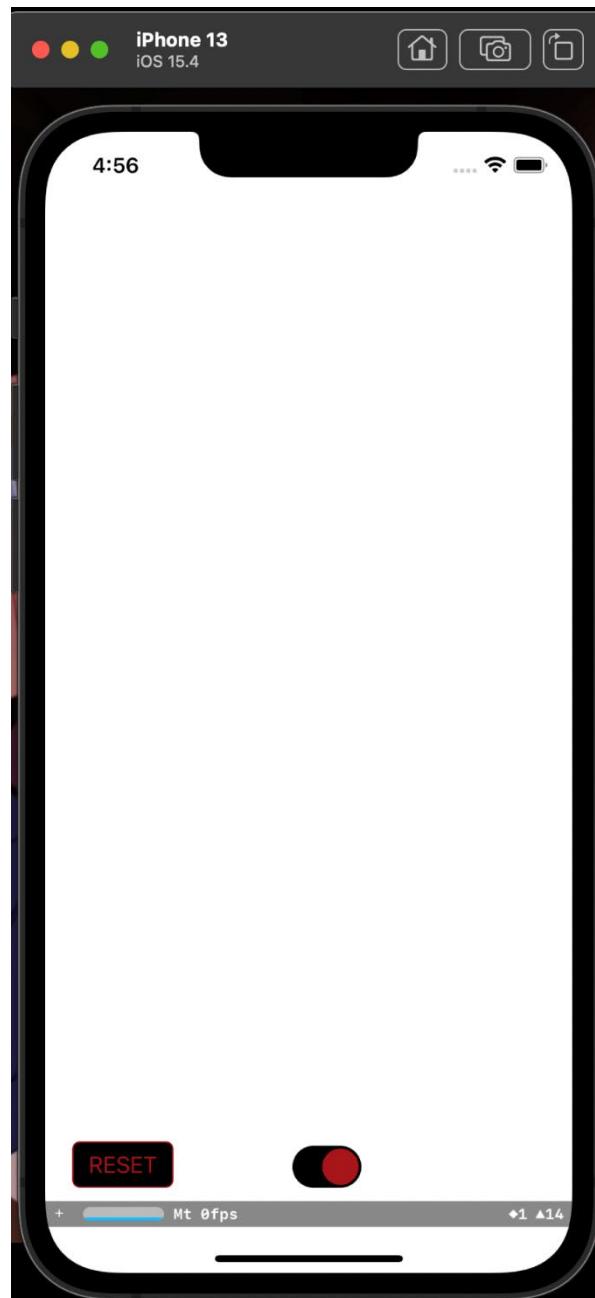
The screenshot shows the Xcode interface with two main windows. The top window displays the storyboard for 'Main.storyboard' with a preview of an iPhone 13 showing an ARSCNView. The bottom window shows the code for 'ViewController.swift'. Both windows show the same code, which is a Swift implementation for rendering AR scenes.

```

1 // ViewController.swift
2 // drawingAppAR
3 // Created by Ravnish Singh on 26/04/22.
4 //
5 import UIKit
6 import ARKit
7 import SceneKit
8
9 class ViewController: UIViewController, ARSCNViewDelegate {
10     @IBOutlet weak var clearButton: UIButton!
11     @IBOutlet weak var switchDraw: UISwitch!
12     @IBOutlet weak var sceneView: ARSCNView!
13     let configuration = ARWorldTrackingConfiguration()
14     override func viewDidLoad() {
15         super.viewDidLoad()
16         // Do any additional setup after loading the view.
17         sceneView.delegate = self
18         sceneView.showsStatistics = true
19         let scene = SCNScene()
20         sceneView.scene = scene
21
22     override func viewWillAppear(_ animated: Bool) { super.viewWillAppear(animated)
23         sceneView.session.run(configuration)
24     }
25     func renderer(_ renderer: SCNSceneRenderer, willRenderScene scene: SCNScene, atTime time: TimeInterval) {
26         guard let pov = sceneView.pointOfView else {
27             return
28         }
29         let transform = pov.transform
30         let rotation = SCNVector3(-transform.m31, -transform.m32, -transform.m33)
31         let location = SCNVector3(transform.m41, transform.m42, transform.m43)
32         let currentPosition = SCNVector3(rotation.x + location.x, rotation.y + location.y,
33                                         rotation.z + location.z)
34         DispatchQueue.main.async {
35             if self.switchDraw.isOn {
36                 let drawNode = SCNNode()
37                 drawNode.geometry = SCNSphere(radius: 0.01)
38                 drawNode.geometry?.firstMaterial?.diffuse.contents = UIColor.green
39                 drawNode.position = currentPosition
40                 self.sceneView.scene.rootNode.addChildNode(drawNode)
41             } else {
42                 let point = SCNNode()
43                 point.name = "aiming point"
44                 point.geometry = SCNSphere(radius: 0.005)
45                 point.position = currentPosition
46                 point.geometry?.firstMaterial?.diffuse.contents = UIColor.red
47             }
48         }
49         self.sceneView.scene.rootNode.addChildNode(point)
50     }
51     if self.clearButton.isHighlighted { self.sceneView.scene.rootNode.enumerateChildNodes({ (node, _) in
52         node.removeFromParentNode()
53     })
54 }
55 }
56 }
57 }

```

Simulator Output:



PRACTICAL – 13

Create an app to recognize Pinch Gesture.

ViewController.swift:

The screenshot shows the Xcode interface with the following details:

- Sidemenu:** Shows project structure: pinchGestureApp (selected), pinchGestureApp (target), AppDelegate.swift, art.scnassets, ViewController.swift (selected), Main.storyboard, Assets.xcassets, and LaunchScreen.storyboard.
- Main Area:** Displays Main.storyboard (Base) which contains two iPhone 11 simulators. The left simulator shows a white screen, and the right one shows a black ARSCNView placeholder.
- Code Editor:** Displays ViewController.swift code. The code imports UIKit, SceneKit, and ARKit, and defines a ViewController class that conforms to ARSCNViewDelegate. It initializes an ARSCNView, sets its delegate to self, and loads a SCNScene named "art.scnassets/ship.scn". It also handles viewDidLoad, viewWillAppear, and viewWillDisappear methods.
- Toolbar:** Shows various Xcode icons and the text "iPhone 11".
- Status Bar:** Shows "main" and "Finished running pinchScaleGestureApp on iPhone 13".

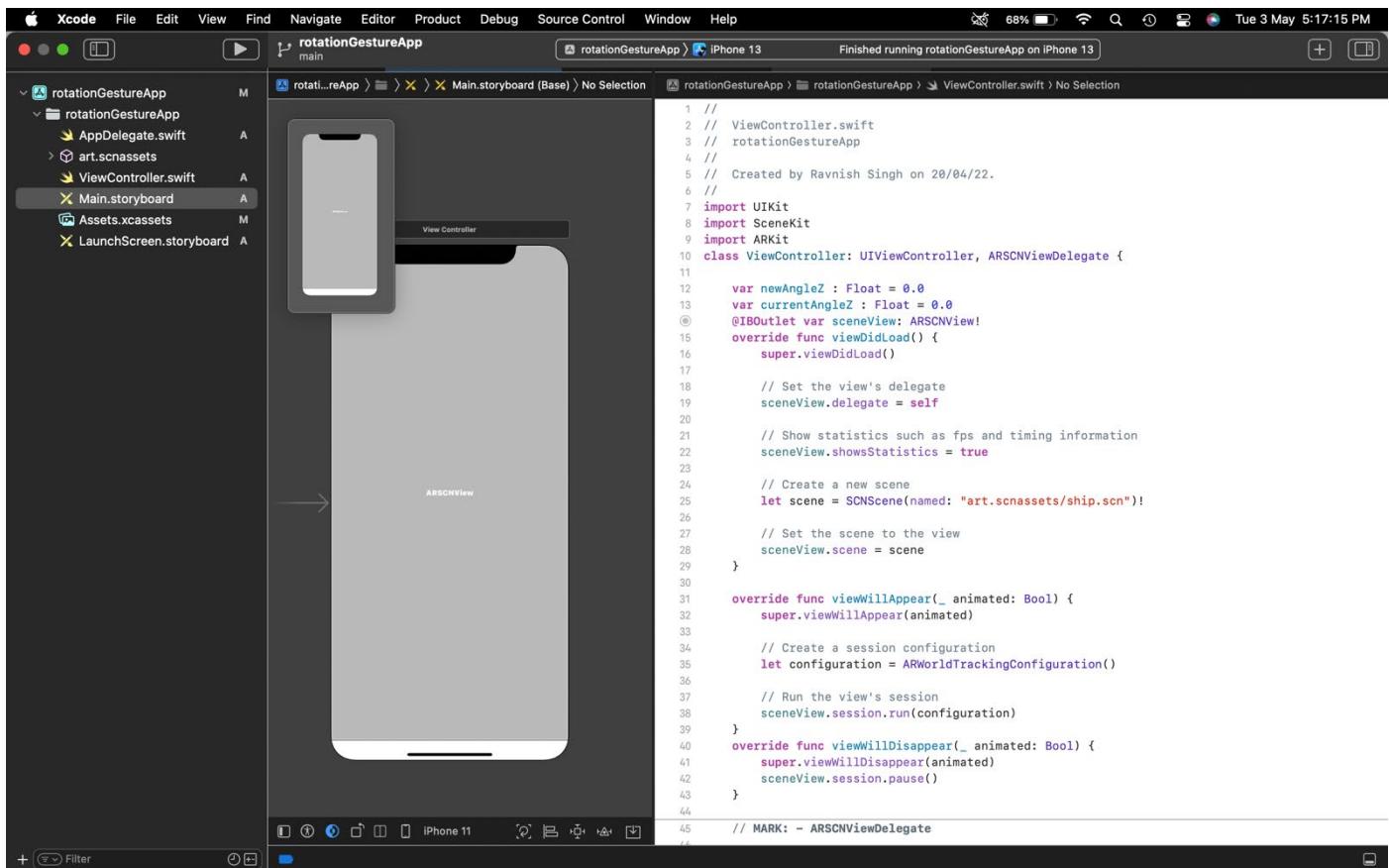
Simulator Output:



PRACTICAL – 14

Create an app to recognize Rotation Gesture.

ViewController.swift:



The screenshot shows the Xcode IDE with the project "rotationGestureApp" open. The main window displays the storyboard and the corresponding Swift code for the ViewController.

Main.storyboard: Shows a single view controller embedded in a navigation controller. The view controller's class is set to "ViewController". Inside the view, there is a label with the text "ARSCNView".

ViewController.swift: The code defines a ViewController class that conforms to ARSCNViewDelegate. It initializes an ARSCNView, sets its delegate to self, and configures the AR session. It also handles viewWillappear and viewWillDisappear events.

```

1 // ViewController.swift
2 // rotationGestureApp
3 // rotationGestureApp
4 //
5 // Created by Ravnish Singh on 20/04/22.
6 //
7 import UIKit
8 import SceneKit
9 import ARKit
10 class ViewController: UIViewController, ARSCNViewDelegate {
11
12     var newAngleZ : Float = 0.0
13     var currentAngleZ : Float = 0.0
14     @IBOutlet var sceneView: ARSCNView!
15     override func viewDidLoad() {
16         super.viewDidLoad()
17
18         // Set the view's delegate
19         sceneView.delegate = self
20
21         // Show statistics such as fps and timing information
22         sceneView.showsStatistics = true
23
24         // Create a new scene
25         let scene = SCNScene(named: "art.scnassets/ship.scn")!
26
27         // Set the scene to the view
28         sceneView.scene = scene
29     }
30
31     override func viewWillAppear(_ animated: Bool) {
32         super.viewWillAppear(animated)
33
34         // Create a session configuration
35         let configuration = ARWorldTrackingConfiguration()
36
37         // Run the view's session
38         sceneView.session.run(configuration)
39     }
40     override func viewWillDisappear(_ animated: Bool) {
41         super.viewWillDisappear(animated)
42         sceneView.session.pause()
43     }
44
45     // MARK: - ARSCNViewDelegate

```

Simulator Output:

