```
import UIKit
import SceneKit
import ARKit

class ViewController: UIViewController, ARSCNViewDelegate, ARSessionDelegate {

    @IBOutlet var sceneView: ARSCNView!

    private var hud :MBProgressHUD!

    private lazy var worldMapStatusLabel :UILabel = {

        let label = UILabel()
        label.textAlignment = .center
        label.textColor = UIColor.white
        label.translatesAutoresizingMaskIntoConstraints = false
        return label
    }()

    private lazy var saveWorldMapButton :UIButton = {

        let button = UIButton(type: .custom)
        button.setTitle("Save", for: .normal)
        button.translatesAutoresizingMaskIntoConstraints = false
        button.tintColor = UIColor.white
        button.backgroundColor = UIColor(red: 53/255, green: 73/255, blue: 94/255, alpha: 1)
        button.addTarget(self, action: #selector(saveWorldMap), for: .touchUpInside)
        return button

    }()

    @objc func saveWorldMap() {

        self.sceneView.session.getCurrentWorldMap { worldMap, error in

            if error != nil {
                print(error?.localizedDescription)
                return
            }

            if let map = worldMap {

                let data = try! NSKeyedArchiver.archivedData(withRootObject: map, requiringSecureCoding: true)

                // save in user defaults
                let userDefaults = UserDefaults.standard
                userDefaults.set(data, forKey: "box")
                userDefaults.synchronize()

                self.hud = MBProgressHUD.showAdded(to: self.view, animated: true)
                self.hud.label.text = "World Map Saved!"
                self.hud.hide(animated: true, afterDelay: 2.0)
            }

        }

    }
```

```
    override func viewDidLoad() {
        super.viewDidLoad()

        self.sceneView.autoenablesDefaultLighting = true

        // Set the view's delegate
        sceneView.delegate = self
        self.sceneView.session.delegate = self


        // Create a new scene
        let scene = SCNScene()

        // Set the scene to the view
        sceneView.scene = scene

        registerGestureRecognizers()

        setupUI()

        // show the hud
        self.hud = MBProgressHUD.showAdded(to: self.view, animated: true)
        self.hud.label.text = "Detecting Surfaces..."

    }

    private func registerGestureRecognizers() {

        let tapGestureRecognizer = UITapGestureRecognizer(target: self, action: #selector(tapped))
        self.sceneView.addGestureRecognizer(tapGestureRecognizer)
    }

    @objc func tapped(recognizer :UITapGestureRecognizer) {

        guard let sceneView = recognizer.view as? ARSCNView else {
            return
        }

        let touch = recognizer.location(in: sceneView)

        let hitTestResults = sceneView.hitTest(touch, types: .existingPlane)

        if !hitTestResults.isEmpty {

            if let hitTestResult = hitTestResults.first {

                let boxAnchor = ARAnchor(name: "box-anchor", transform: hitTestResult.worldTransform)
                self.sceneView.session.add(anchor: boxAnchor)
            }

        }
    }
```

```
func session(_ session: ARSession, didUpdate frame: ARFrame) {

    switch frame.worldMappingStatus {
        case .notAvailable:
            self.worldMapStatusLabel.text = "NOT AVAILABLE"
        case .limited:
            self.worldMapStatusLabel.text = "LIMITED"
        case .extending:
            self.worldMapStatusLabel.text = "EXTENDING"
        case .mapped:
            self.worldMapStatusLabel.text = "MAPPED"
    }

}

func renderer(_ renderer: SCNSceneRenderer, didAdd node: SCNNode, for anchor: ARAnchor) {

    if anchor is ARPlaneAnchor {
        print("plane has been detected")

        DispatchQueue.main.async {
            self.hud.label.text = "Surface Detected!"
            self.hud.hide(animated: true, afterDelay: 2.0)
        }

        return
    }

    // add a virtual object
    let box = SCNBox(width: 0.2, height: 0.2, length: 0.2, chamferRadius: 0)

    let material = SCNMaterial()
    material.diffuse.contents = UIColor.purple

    box.materials = [material]

    let boxNode = SCNNode(geometry: box)
    node.addChildNode(boxNode)
}

private func setupUI() {

    self.view.addSubview(self.worldMapStatusLabel)
    self.view.addSubview(self.saveWorldMapButton)

    //add constraints to label
    self.worldMapStatusLabel.topAnchor.constraint(equalTo: self.sceneView.topAnchor, constant: 20).isActive = true
    self.worldMapStatusLabel.rightAnchor.constraint(equalTo: self.sceneView.rightAnchor, constant: -20).isActive = true
    self.worldMapStatusLabel.heightAnchor.constraint(equalToConstant: 44).isActive = true

    // add constraints to save world map button
    self.saveWorldMapButton.centerXAnchor.constraint(equalTo: self.sceneView.centerXAnchor).isActive = true
    self.saveWorldMapButton.bottomAnchor.constraint(equalTo: self.sceneView.bottomAnchor, constant: -20).isActive = true
    self.saveWorldMapButton.widthAnchor.constraint(equalToConstant: 100).isActive = true
    self.saveWorldMapButton.heightAnchor.constraint(equalToConstant: 44).isActive = true

}
```

```
 private func restoreWorldMap() {

     let userDefaults = UserDefaults.standard

     if let data = userDefaults.data(forKey: "box") {

         if let unarchived = try? NSKeyedUnarchiver.unarchivedObject(of: ARWorldMap.classForKeyedUnarchiver(), from: data),
             let worldMap = unarchived as? ARWorldMap {

             let configuration = ARWorldTrackingConfiguration()
             configuration.initialWorldMap = worldMap
             configuration.planeDetection = .horizontal

             sceneView.session.run(configuration)

         }

     } else {
         let configuration = ARWorldTrackingConfiguration()
         configuration.planeDetection = .horizontal
         sceneView.session.run(configuration)
     }

 }

 override func viewWillAppear(_ animated: Bool) {
     super.viewWillAppear(animated)

     restoreWorldMap()
 }

 override func viewWillDisappear(_ animated: Bool) {
     super.viewWillDisappear(animated)

     // Pause the view's session
     sceneView.session.pause()
 }

}
```

```
import UIKit
import SceneKit
import ARKit

class ViewController: UIViewController, ARSCNViewDelegate {

    @IBOutlet var sceneView: ARSCNView!

    override func viewDidLoad() {
        super.viewDidLoad()

        // Set the view's delegate
        sceneView.delegate = self

        // Show statistics such as fps and timing information
        sceneView.showsStatistics = true

        // Create a new scene
        let scene = SCNScene()

        // Set the scene to the view
        sceneView.scene = scene

        registerGestureRecognizers()
    }

    private func registerGestureRecognizers() {

        let tapGestureRecognizer = UITapGestureRecognizer(target: self, action: #selector(tapped))
        self.sceneView.addGestureRecognizer(tapGestureRecognizer)
    }

    func renderer(_ renderer: SCNSceneRenderer, didAdd node: SCNNode, for anchor: ARAnchor) {

        if anchor is ARPlaneAnchor {
            print("Plane is detected")
        } else if anchor is AREnvironmentProbeAnchor {
            return
        }
        else {

            let box = SCNBox(width: 0.2, height: 0.2, length: 0.2, chamferRadius: 0)

            let material = SCNMaterial()
            material.lightingModel = .physicallyBased
            material.diffuse.contents = UIColor.purple
            material.metalness.contents = UIImage(named :"streakedmetal-metalness")
            material.roughness.contents = UIImage(named :"streakedmetal-roughness")

            box.materials = [material]

            let boxNode = SCNNode(geometry: box)
            boxNode.position.y = 0.2/2

            node.addChildNode(boxNode)

        }
    }

    @objc func tapped(recognizer :UITapGestureRecognizer) {

        let sceneView = recognizer.view as! ARSCNView
        let touch = recognizer.location(in: sceneView)

        let hitTestResults = sceneView.hitTest(touch, types: .existingPlane)

        if !hitTestResults.isEmpty {

            let hitTestResult = hitTestResults.first!

            let anchor = ARAnchor(name: "box", transform: hitTestResult.worldTransform)

            self.sceneView.session.add(anchor: anchor)

        }

    }

    override func viewWillAppear(_ animated: Bool) {
        super.viewWillAppear(animated)

        // Create a session configuration
        let configuration = ARWorldTrackingConfiguration()
        configuration.planeDetection = .horizontal
        configuration.environmentTexturing = .automatic

        // Run the view's session
        sceneView.session.run(configuration)
    }

    override func viewWillDisappear(_ animated: Bool) {
        super.viewWillDisappear(animated)

        // Pause the view's session
        sceneView.session.pause()
    }

}
```