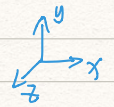
// Display text
// ViewController.swift




```
import UIKit
import SceneKit
import ARKit
```

// change the viewController attribute to the sample example class
// change ARWorldTrackingSessionConfiguration() to ARWorldTrackingConfiguration()

```
class ViewController: UIViewController, ARSCNViewDelegate {
    @IBOutlet var sceneView: ARSCNView!
    private let label: UILabel = UILabel()
    var planes = [OverlayPlane]()
    override func viewDidLoad(){
        super.viewDidLoad()
        sceneView.delegate = self
        sceneView.showsStatistics = true
        let scene = SCNScene()
        sceneView.scene = scene
```

test to see if actually needed

```
        let text = SCNText(string: "Hello ARKit!", extrusionDepth: 1.0)
        text.firstMaterial?.diffuse.contents = UIColor.blue
        let textNode = SCNNode(geometry: text)
```

// must have a node to display a scn

```
        textNode.position = SCNVector3(0, 0, -0.5)
        textNode.scale = SCNVector3(0.02, 0.02, 0.02)
        sceneView.scene.rootNode.addChildNode(textNode)

        let box = SCNBox(width: 0.2, height: 0.2, length: 0.2, chamferRadius: 0)
        let material = SCNMaterial()
        material.name = "Color"
        material.diffuse.contents = UIColor.red

        let node = SCNNode()
        node.geometry = box
        node.geometry?.materials = [material]
        node.position = SCNVector3(0, 0.1, -0.5)
        scene.rootNode.addChildNode(node)

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

    @objc func tapped(recognizer: UIGestureRecognizer){
        let sceneView = recognizer.view as! SCNView
        let touchLocation = recognizer.location(in: sceneView)
        let hitResult = sceneView.hitTest(touchLocation, options: [:])
```

not

```
        if !hitResults.isEmpty{
            let node = hitResult[0].node
            let material = node.geometry?.material(named: "Color")
            material?.diffuse.contents = UIColor.random()
        }
    }
```

⇒

// not connected

```
var sceneView: ARSCNView!
override func viewDidLoad(){
    self.sceneView = ARSCNView(frame: self.view.frame)
    self.view.addSubview(self.sceneView)
```

// self.sceneView.debugOptions = [ARSCNDebugOptions.showFeaturePoints, ARSCNDebugOptions.showWorldOrigin]

// first create the geometry
↓ assign material to the geometry
↓ assign geometry
↓ location and scale
↓ Add node to the root node

// Add a label

```
self.label.frame = CGRect(x:0, y:0, width:
    self.sceneView.frame.size.width, height:44)
self.label.center = self.sceneView.center
self.label.textAlignment = .center
self.label.textColor = UIColor.white
self.label.font = UIFont.preferredFont(forTextStyle: .headline)
self.label.alpha = 0
```

// Extensions.swift

```
import Foundation
import UIKit

extension CGFloat{
    static func random() -> CGFloat{
        return CGFloat(arc4random())/CGFloat(UInt32.max)
    }
}

extension UIColor{
    static func random() -> UIColor{
        return UIColor(red: .random(),
                       green: .random(),
                       blue: .random(),
                       alpha: 1.0)
    }
}
```

```
}

override func viewWillAppear(_ animated: Bool){
    super.viewWillAppear(animated)
    let configuration = ARWorldTrackingConfiguration()
    configuration.planeDetection = .horizontal    //planes

    sceneView.session.run(configuration)
}

                                  → called every time find an anchor
func renderer(_ renderer: SCNSceneRenderer, didAdd node: SCNNode, for anchor: ARAnchor){
        if !(anchor is ARPlaneAnchor){
            return
        }

        DispatchQueue.main.async{
            self.label.text = "Plane Detected"
            UIView.animate(withDuration: 3.0, animations: {
                self.label.alpha = 1.0
            }){ (completion: Bool) in
                self.label.alpha = 0.0
            }
        }
        let plane = OverlayPlane(anchor: anchor as! ARPlaneAnchor)
        self.planes.append(plane)

        node.addChildNode(plane)

}

                                  → call every frame
func renderer(_ renderer: SCNSceneRenderer, didUpdate node: SCNNode, for anchor: ARAnchor){
        let plane = self.planes.filter{ plane in
            return plane.anchor.identifier == anchor.identifier
            }.first

        if plane == nil{
            return
        }
        plane?.update(anchor: anchor as! ARPlaneAnchor)
}

override func viewWillDisappear(_ animated: Bool){
    super.viewWillDisappear(animated)
    sceneView.session.pause()
}
```

// OverlayPlane.swift

```
import ARKit

class OverlayPlane: SCNNode {
    var anchor: ARPlaneAnchor
    var planeGeometry: SCNPlane!

    init(anchor: ARPlaneAnchor) {
        self.anchor = anchor
        super.init()
        setup()
    }

    func update(anchor: ARPlaneAnchor) {
        self.planeGeometry.width = CGFloat(anchor.extent.x);
        self.planeGeometry.height = CGFloat(anchor.extent.y);
        self.position = SCNVector3Make(anchor.center.x, 0, anchor.center.z);
    }

    private func setup() {
        self.planeGeometry = SCNPlane(width: CGFloat(self.anchor.extent.x), height: CGFloat(self.anchor.extent.z))
        let material = SCNMaterial()
        material.diffuse.contents = UIImage(named: "overlay_grid.png")
        self.planeGeometry.materials = [material]
        let planeNode = SCNNode(geometry: self.planeGeometry)
        planeNode.position = SCNVector3Make(anchor.center.x, 0, anchor.center.z)
        planeNode.transform = SCNMatrix4MakeRotation(Float(-Double.pi/2.0), 1.0, 0.0, 0.0)

        self.addChildNode(planeNode)
    }

    required init?(coder aDecoder: NSCoder) {
        fatalError("init(coder:) has not been implemented")
    }
}
```