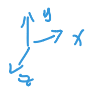
```
import UIKit
import SceneKit
import ARKit

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()

        addLight()
    }

    private func registerGestureRecognizers() {

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

    }

    @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 {

            if let hitTestResult = hitTestResults.first {

                let position = SCNVector3(hitTestResult.worldTransform.columns.3.x,hitTestResult.worldTransform.columns.3.y,hitTestResult.worldTransform.columns.3.z)


                addCake(at: position)

            }

        }

    }
```

```swift
    private func addCake(at position: SCNVector3) {

        let cakeScene = SCNScene(named: "art.scnassets/cake-assets/cake-model.dae")!
        let baseNode = cakeScene.rootNode.childNode(withName: "baseNode", recursively: true)!
        let cakeNode = baseNode.childNode(withName: "cake", recursively: true)!
        let plateNode = baseNode.childNode(withName: "plate", recursively: true)!

        cakeNode.geometry?.firstMaterial?.lightingModel = .physicallyBased
        cakeNode.geometry?.firstMaterial?.normal.contents = UIImage(named: "art.scnassets/cake-assets/CL_LR_01NormalsMap.jpg")
        cakeNode.geometry?.firstMaterial?.diffuse.contents = UIImage(named: "art.scnassets/cake-assets/CL_LR_01DiffuseMap.jpg")

        plateNode.geometry?.firstMaterial?.lightingModel = .physicallyBased
        plateNode.geometry?.firstMaterial?.metalness.contents = 0.8
        plateNode.geometry?.firstMaterial?.roughness.contents = 0.2

        baseNode.position = position

        addPlaneTo(baseNode)

        self.sceneView.scene.rootNode.addChildNode(baseNode)

    }

    private func addPlaneTo(_ node: SCNNode) {

        let plane = SCNPlane(width: 200, height: 200)
        plane.firstMaterial = SCNMaterial()
        plane.firstMaterial?.isDoubleSided = true
        plane.firstMaterial?.colorBufferWriteMask = .init(rawValue: 0)

        let planeNode = SCNNode(geometry: plane)
        planeNode.eulerAngles.x = .pi/2
        node.addChildNode(planeNode)
    }

    private func addLight() {

        let directionalLight = SCNLight()
        directionalLight.type = .directional
        directionalLight.intensity = 0
        directionalLight.castsShadow = true
        directionalLight.shadowMode = .deferred
        directionalLight.shadowColor = UIColor(displayP3Red: 0, green: 0, blue: 0, alpha: 0.5)
        directionalLight.shadowSampleCount = 10

        let directionalLightNode = SCNNode()
        directionalLightNode.light = directionalLight
        directionalLightNode.rotation = SCNVector4Make(1,0,0,-Float.pi/2)

        self.sceneView.scene.rootNode.addChildNode(directionalLightNode)

    }

    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()
    }

}
```

//sketchfab.com
turbo squid.com
google Poly

```
import UIKit
import SceneKit
import ARKit

class ViewController: UIViewController, ARSCNViewDelegate {

    @IBOutlet var sceneView: ARSCNView!

    let availableColors = [UIColor.red, UIColor.purple, UIColor.orange, UIColor.blue]

    private var offsetX: CGFloat = 20

    private var watchNode: SCNNode!

    override func viewDidLoad() {
        super.viewDidLoad()

        sceneView.autoenablesDefaultLighting = true

        // 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

        addColorSwatches()                                    //create color swatches
    }

    private func addColorSwatches() {

        for availableColor in self.availableColors {

            let swatchView = ColorSwatch(color: availableColor) { color in

                guard let bandNode = self.watchNode.childNode(withName: "band", recursively: true) else {
                    return
                }

                bandNode.geometry?.firstMaterial?.diffuse.contents = color

            }

            self.view.addSubview(swatchView)
            // configure constraints
            configureConstraints(for: swatchView)

        }

    }
```

```
    private func configureConstraints(for swatchView: UIView) {

        swatchView.translatesAutoresizingMaskIntoConstraints = false

        swatchView.widthAnchor.constraint(equalToConstant: swatchView.frame.size.width).isActive = true
        swatchView.heightAnchor.constraint(equalToConstant: swatchView.frame.size.height).isActive = true

        swatchView.bottomAnchor.constraint(equalTo: self.view.bottomAnchor, constant: -20).isActive = true
        swatchView.leftAnchor.constraint(equalTo: self.view.leftAnchor, constant: offsetX).isActive = true

        offsetX += self.view.frame.width / 4

    }

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

        if let anchor = anchor as? ARImageAnchor {

            let refImage = anchor.referenceImage
            addWatch(to: node, referenceImage: refImage)

        }

    }

    private func addWatch(to node: SCNNode, referenceImage: ARReferenceImage) {
                                   → happening in the background
        DispatchQueue.global().async {

            let watchScene = SCNScene(named: "watch-model.dae")!
            self.watchNode = watchScene.rootNode.childNode(withName: "watch", recursively: true)!

            let cylinder = SCNCylinder(radius: referenceImage.physicalSize.width/1.8, height: referenceImage.physicalSize.height)
            cylinder.firstMaterial?.diffuse.contents = UIColor.purple
            cylinder.firstMaterial?.colorBufferWriteMask = [] // hide the cylender

            let cylinderNode = SCNNode(geometry: cylinder)
            cylinderNode.eulerAngles.x = .pi/2
            cylinderNode.renderingOrder = -1 //

            let centerY = (self.watchNode.boundingBox.max.y + self.watchNode.boundingBox.min.y) / 2

            cylinderNode.position.y = centerY + 0.008


            node.addChildNode(self.watchNode)
            node.addChildNode(cylinderNode)
        }
    }
```



```
    override func viewWillAppear(_ animated: Bool) {
        super.viewWillAppear(animated)
        
        // Create a session configuration
        let configuration = ARImageTrackingConfiguration()
        
        guard let referenceImages = ARReferenceImage.referenceImages(inGroupNamed: "AR Resources", bundle: nil) else {
            fatalError("No reference images found...")
        }
        
        configuration.trackingImages = referenceImages

        // 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()
    }

}
```

```
import Foundation            // ColorSwatch.swift
import UIKit

class ColorSwatch: UIView {
    
    private var color: UIColor
    typealias ColorSelected = (UIColor) -> Void     // closure
    
    private var colorSelected: ColorSelected
    
    init(color: UIColor, frame: CGRect = CGRect(x: 0, y: 0, width: 50, height: 50), colorSelected: @escaping ColorSelected) {
        
        self.colorSelected = colorSelected
        self.color = color
        super.init(frame: frame)
        
        registerGestureRecognizers()
        
    }
    
    private func registerGestureRecognizers() {
        
        let tapGestureRecognizer = UITapGestureRecognizer(target: self, action: #selector(tapped))
        self.addGestureRecognizer(tapGestureRecognizer)
    }
    
    @objc func tapped(recognizer: UITapGestureRecognizer) {
        
        self.colorSelected(self.color)
    }
    
    override func draw(_ rect: CGRect) {
        
        let path = UIBezierPath(ovalIn: CGRect(x: 0, y: 0, width: 50, height: 50))
        let layer = CAShapeLayer()
        layer.path = path.cgPath
        layer.fillColor = self.color.cgColor
        self.layer.addSublayer(layer)
        
    }
    
    required init?(coder aDecoder: NSCoder) {
        fatalError("init(coder:) has not been implemented")
    }

}
```