// create a room in .SCN File

// Google poly.google.com

⓪ → Box

↓ perspective ⇒ front

scale ⇒ 8, 5, 6

click and drag y axis lift up

red arrow x
green arrow y
blue arrow z

⓪ → new material for every side ⇒ check Double sidded

Opacity ⇒ 0 (in the Diffuse menu) to chek if double sidded work

import both mtl and obj ⇒ check the target is correct

⓪ ⇒ Plane ← for the front door  right click duplicate node

// creat another Box to contain the house

⓪ ⇒ outerBox rendering order ⇒ 100

rendering order for everything in room and room itself ⇒ 200 (remove child node out to change the rendering order)

outerBox ⇒ Opacity ⇒ 1% white color

↓

0% will see the inner wall

// viewController.swift

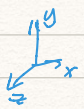
```
import UIKit
import SceneKit
import ARKit

class RemoteCarViewController: UIViewController, ARSCNViewDelegate{
    var sceneView: ARSCNView!
    var planes = [OverlayPlane]()

    override func viewDidLoad(){
        super.viewDidLoad()
        self.sceneView = ARSCNView(frame: self.view.frame)
        self.view.addSubview(self.sceneView)
        self.sceneView.autoenablesDefaultLighting = true
        sceneView.delegate = self
        let scene = SCNScene()
        sceneView.scene = scene
        setupControlPad()
        registerGestureRecognizers()
    }

    @objc func tapped(recognizer: UIGestureRecognizer){
        let sceneView = recognizer.view as! ARSCNView
        let touchLocation = recognizer.location(in: sceneView)

        let hitTestResult = sceneView.hitTest(touchLocation, types: .existingPlaneUsingExtent)
        if !hitTestResult.isEmpty{
```

```
        guard let hitResult = hitTestResult.first else {
            return
        }
        addPortal(ht: hitTestResult)
    }
}

private func addPortal(ht: ARHitTestResult){
    let portalScene = SCNScene(named: "art.scnassets/ship.scn")!
    let portalNode = (portalNode.rootNode.childNode(withName: "portalNode", recursively: true))!
    portalNode.position = SCNVector3(hitResult.worldTransform.columns.3.x,
                                     hitResult.worldTransform.columns.3.y,
                                     hitResult.worldTransform.columns.3.z)
    self.sceneView.scene.rootNode.addChildNode(portalNode)
}

private func registerGestureRecognizers()
    let tapGestureRecognizer = UITapGestureRecognizer(target: self, action: #selector(tapped))
    self.sceneView.addGestureRecognizer(tapGestureRecognizer)
}
                                        → 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 viewWillAppear(_ animated: Bool){
    super.viewWillAppear(animated)
    let configuration = ARWorldTrackingConfiguration()
    configuration.planeDetection = .horizontal    // planes

    sceneView.session.run(configuration)
}

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")
    }
}
```



# ARTV

newFile -> AR app

// ViewController.swift

```
import UIKit
import SceneKit
import ARKit

class ViewController: UIViewController, ARSCNViewDelegate {
    var sceneView: ARSCNView!


    override func viewDidLoad(){
        super.viewDidLoad()
        self.sceneView = ARSCNView(frame: self.view.frame)
        self.view.addSubview(self.sceneView)
        self.sceneView.autoenablesDefaultLighting = true
        sceneView.delegate = self
        let scene = SCNScene()
        sceneView.scene = scene

        registerGestureRecognizers()
    }

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

//@objc func tapped(recognizer: UIGestureRecognizer){
        let sceneView = recognizer.view as! ARSCNView
        let touchLocation = recognizer.location(in: sceneView)

        let hitTestResult = sceneView.hitTest(touchLocation, types: .existingPlaneUsingExtent)
        if !hitTestResult.isEmpty {
            guard let hitResult = hitTestResult.first else {
                return
            }
            addPortal(ht: hitTestResult)
        }
    }

@objc func tapped(recognizer: UIGestureRecognizer){
    guard let currentFrame = self.sceneView.session.currentFrame else {
        return
    }
                                    -> can use ck node in SCNKit
    let videoNode = SKVideoNode(fileName: "big_buck_bunny.mp4")  // check target membership
    videoNode.play()
```

```
let skScene = SKScene(size: CGSize(width: 640, height: 480))
skScene.addChild(video.node)
videoNode.position = CGPoint(x: skScene.size.width/2, y: skScene.size.height/2)
videoNode.size = skScene.size

let tvPlane = SCNPlane(width: 1.0, height: 0.75)
tvPlane.firstMaterial?.diffuse.contents = skScene → assign the scene to the geometry of a SCNNode
tvplane.firstMaterial?.isDoubleSided = true
let tvPlaneNode = SCNNode(geometry: tvPlane)
var translation = matrix_identity_float4x4
translation.columns.3.z = -1.0
tvPlaneNode.simdTransform = matrix_multiply(currentFrame.camera.transform, translation)

self.sceneView.scene.rootNode.addChildNode(tvPlaneNode)
tvPlaneNode.eulerAngles = SCNVector3(Double.pi, 0, 0)
}
```