```
// LightController.swift                    // ⌘ ⇒ check is initial ViewController
import UIKit                                // copied from the plan detect add Object file
import SceneKit                             // check the sample code to see how to turn the Grid off
import ARKit

enum BodyType : Int {
    case box = 1
    case plane = 2
}

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

    override func viewDidLoad(){
        super.viewDidLoad()
        self.sceneView = ARSCNView(frame: self.view.frame)
        self.sceneView.debugOptions = [ARSCNDebugOptions.showFeaturePoints, ARSCNDebugOptions.showWorldOrigin]
        self.view.addSubview(self.sceneView)
        // self.sceneView.autoenablesDefaultLighting = true // enable auto lighting, good for well lighted environment
        sceneView.delegate = self
        sceneView.showsStatistics = true
        let scene = SCNScene()
        sceneView.scene = scene

        registerGestureRecognizers()
        setupPlaneToggleSwitch()
        insertSpotLight(position: SCNVector3(0, 1.0, 0))

    }
    private func insertSpotLight(position: SCNVector3){
        let spotLight = SCNLight()
        spotLight.type = .spot
        spotlight.spotInnerAngle = 45
        spotlight.spotOuterAngle = 45

        let spotNode = SCNNode()
        spotNode.name = "SpotNode"
        spotNode.light = spotLight
        spotNode.position = position
        spotNode.eulerAngles = SCNVector3(-Double.pi/2.0, 0, -0.2)
        self.sceneView.scene.rootNode.addChildNode(spotNode)



}

private func setupPlaneToggleSwitch(){                    x →
                                                          y ↓
    let planeToggleSwitch = UISwitch(frame: CGRect(x: 10, y: self.sceneView.frame.height - 44, width: 100, height: 33))
    planeToggleSwitch.addTarget(self, action: #selector(planeSwitchToggled), for: .valueChanged)
    self.sceneView.addSubview(planeToggleSwitch)
}
```

```
// turn off plane detection and remove the grid from the plane
@objc func planeSwitchToggled(planeSwitch: UISwitch){
    let configuration = self.sceneView.session.configuration as! ARWorldTrackingConfiguration
    configuration.planeDetection = []
    self.sceneView.session.run(configuration, options: [])

    for plane in self.planes {
        plane.planeGeometry.materials.forEach { material in
            material.diffuse.contents = UIColor.clear
        }
    }
}

func renderer(_ renderer: SCNSceneRenderer, updateAtTime time: TimeInterval){
    let estimate = self.sceneView.session.currentFrame?.lightEstimate   // light intensity of the room
    if estimate == nil {
        return
    }                                                                   // scene -> rootNode -> childNode with names
    let spotNode = self.sceneView.scene.rootNode.childNode(withName: "SpotNode", recursively: true)
    spotNode?.light?.intensity = (estimate?.ambientIntensity)!

}

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

    sceneView.session.run(configuration)
}
private func registerGestureRecognizers(){
    let tapGestureRecognizer = UITapGestureRecognizer(target: self, action: #selector(tapped))
    tapGestureRecognizer.numberOfTapsRequired = 1
    let doubleTappedGestureRecognizer = UITapGestureRecognizer(target: self, action: #selector(doubleTapped))
    doubleTappedGestureRecognizer.numberOfTapsRequired = 2
    tapGestureRecognizer.require(toFail: doubleTappedGestureRecognizer) // single tap will fail if double tap performed

    self.sceneView.addGestureRecognizer(tapGestureRecognizer)
    self.sceneView.addGestureRecognizer(doubleTappedGestureRecognizer)
}
@objc func doubleTapped(recognizer: UIGestureRecognizer){
    let sceneView = recognizer.view as! ARSCNView
    let touchLocation = recognizer.location(in: sceneView)
    let hitTestResult = sceneView.hitTest(touchLocation,
    if !hitTestResult.isEmpty {
        guard let hitResult = hitTestResult.first else {
            return
        }
```

```
        let node = hitResult.node
        node.physicsBody?.applyForce(SCNVector3(hitResult.worldCoordinates.x, 2.0
                ,hitResult.worldCoordinates.z), asImpulse: true)
    }
}

@objc func tappe(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
        }
        addBox(hitResult: hitResult)

    }
}

private func addBox(hitResult: ARHitTestResult){
    let box = SCNBox(width: 0.2, height: 0.2, length: 0.2, chamferRadius: 0)
    let material = SCNMaterial()
    material.diffuse.contents = UIColor.red
    box.materials = [material]
    let boxNode = SCNNode(geometry: box)
    boxNode.physicsBody = SCNPhysicsBody(type: .dynamic, shape: nil)
    boxNode.physicsBody?.categoryBitMask = BodyType.box.rawValue

    boxNode.position = SCNVector3(hitResult.worldTransform.columns.3.x, hitResult.worldTransform.columns.3.y +
                Float(box.height/2), hitResult.worldTransform.column.3.z)

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

}

private func addTable(hitResult:ARHitTestResult){
    let tableScene = SCNScene(named: "art.scnassets/bench.dae")
    let tableNode = tableScene?.rootNode.childNode(withName: "Sketch Up", recursively: true)

    tableNode.position = SCNVector3(hitResult.worldTransform.columns.3.x, hitResult.worldTransform.columns.3.y
                ,hitResult.worldTransform.column.3.z)

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

}

}
```

large force

// dynamic ⇒ can be moving, can be affected by forces
kinematic ⇒ unaffected by forces but can cause collisions when moved
static ⇒ can not moving ⇒ like a wall, can not move

⇒ automatic find the best shape

// sketch up ⇒ 3D warehouse
// turbosquid.com

download ⇒ 3DS or DAE
export as ⇒ collada file .dae
Bounding Box ⇒ in meters

// 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);
        let planeNode = self.childNodes.first
        planeNode.physicsBody = SCNPhysicsBody(type: .static, shape: SCNPhysicsShape(geometry: self.planeGeometry, options: nil))
    }

    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.physicsBody = SCNPhysicsBody(type: .static, shape: SCNPhysicsShape(geometry: self.planeGeometry, options: nil))
        planeNode.physicsBody?.categoryBitMask = BodyType.plane.rawValue
        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")
    }
}
```