

// Google Blocks → Gallery → obj → blender → dae → scn

```
import UIKit  
import SceneKit  
import ARKit  
  
enum BodyType: Int {  
    case box = 1  
    case plane = 2  
    case car = 4  
}
```

```
3 class RemoteCarViewController: UIViewController, ARSCNViewDelegate {
```

```
    var sceneView: ARSCNView!
```

```
    var planes = [OverlayPlane]()
```

```
    private var car: Car!
```

```
    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 carScene = SCNScene(named: "car.dae")
```

```
        guard let node = carScene?.rootNode.childNode(withName: "car", recursively: true) else {
```

```
            return
```

```
        }
```

```
        self.car = Car(node: node)
```

```
        let scene = SCNScene()
```

```
        sceneView.scene = scene
```

```
        setupControlPad()
```

```
        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
```

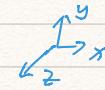
```
    let touchLocation = recognizer.location(in: sceneView)
```

```
    let hitTestResult = sceneView.hitTest(touchLocation, types: .existingPlaneUsingExtent)
```

```
    if !hitTestResult.isEmpty {
```

```
        guard let hitResult = hitTestResult.first else {
```

```
            return
```



```
    }  
    self.car.position = SCNVector3(hitResult.worldTransform.columns.3.x,  
                                hitResult.worldTransform.columns.3.y + 0.1  
                                hitResult.worldTransform.columns.3.z)
```

```
    self.sceneView.scene.rootNode.addChildNode(self.car)
```

```
}
```

```
}
```



```
private func setupControlPad() {
```

```
    let leftButton = GameButton(frame: CGRect(x: 0, y: self.sceneView.frame.height - 40, width: 50, height: 50)) {  
        self.car.turnLeft()  
    }
```

```
    leftButton.setTitle("left", for: .normal)
```

```
    let rightButton = GameButton(frame: CGRect(x: 60, y: self.sceneView.frame.height - 40, width: 50, height: 50)) {  
        self.car.turnRight()  
    }
```

```
    leftButton.setTitle("right", for: .normal)
```

```
    let acceleratorButton = UIButton(frame: CGRect(x: 120, y: self.sceneView.frame.height - 40, width: 60, height: 20)) {  
        self.car.accelerate()  
    }
```

```
    acceleratorButton.backgroundColor = UIColor.red
```

```
    acceleratorButton.layer.cornerRadius = 10.0
```

```
    acceleratorButton.layer.masksToBounds = true
```

```
    self.sceneView.addSubview(leftButton)
```

```
    self.sceneView.addSubview(rightButton)
```

```
    self.sceneView.addSubview(acceleratorButton)
```

```
}
```

11 GameButton.swift

```
import Foundation  
import UIKit
```

```
class GameButton : UIButton {
```

```
    var callback: () -> ()
```

```
    private var timer: Timer!
```

```
    init(frame: CGRect, callback: @escaping () -> ()) {
```

```
        self.callback = callback
```

```
        super.init(frame: frame)
```

```
}
```

```
    override func touchesBegan(_ touches: Set<UITouch>, with event: UIEvent?) {
```

```
        self.timer = Timer.scheduledTimer(withTimeInterval: 0.01, repeats: true,
```

```
            block: { [weak self] (timer: Timer) in
```

```
                self?.callback()
```

```
)
```

```
}
```

```
    override func touchesEnded(_ touches: Set<UITouch>, with event: UIEvent?) {
```

```
        self.timer.invalidate()
```

```
}
```

```
    required init?(coder aDecoder: NSCoder) {
```

```
        fatalError("init(coder:) has not been implemented")
```

```
}
```

```
}
```

11 Models.swift

```
import Foundation
```

```
import SceneKit
```

```
class Car : SCNNode {
```

```
    var carNode: SCNNode
```

```
    private var zVelocityOffset = 0.1
```

```
    init(node: SCNNode) {
```

```
        self.carNode = node
```

```
        super.init()
```

```
        setup()
```

```
}
```

```
    private func setup() {
```

```
        self.addChildNode(self.carNode)
```

```
        self.physicsBody = SCNPhysicsBody(type: .dynamic, shape: nil)
```

```
        self.physicsBody?.categoryBitMask = BodyType.car.rawValue
```

```
}
```

```
func accelerate() {  
    let force = SIMD_make_float4(0, 0, -10, 0) // depend on the node's direction in the scene  
    let rotatedForce = SIMD_mul(self.presentation.SIMDTransform, force)  
    let vectorForce = SCNVector3(rotatedForce.x, rotatedForce.y, rotatedForce.z)  
    self.physicsBody?.applyForce(vectorForce, asImpulse: false)
```

{

```
func turnLeft() {  
    self.physicsBody?.applyTorque(SCNVector4(0, 1, 0, 1.0), asImpulse: false)
```

}

```
func turnRight() {  
    self.physicsBody?.applyTorque(SCNVector4(0, 1, 0, -1.0), asImpulse: false)
```

}

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

}

3