usdz format

```
<div id="container">
  <div>
    <a rel="ar" href="redchair.usdz">
      <img src="redChair.jpg" />
    </a>
  </div>

  <div>
    <a rel="ar" href="fish.usdz">
      <img src="fish.png" />
    </a>
  </div>
```

// surge.sh ∈ static web hosting

// converting custom models to the usdz format // poly.google.com
// PBR
In the command line // have to use physics based rendering to display color

↓

```
sudu xcrun usdz_converter model.obj fish.usdz
```

// Map Scene Kit

* cocoapods

cd (drag and drop the folder in terminal)

pod init

open Podfile

↓

```
# Pods for MapBoxARKit
  pod 'MapboxSceneKit', git ....
  pod 'MapboxMobileEvents'
end
```

↓

pod install

MapBoxARKit.xcworkspace // lat long.net

↓

Google search latitude and longitude → 4 different points

Lat, long,

lat_2 $long_2$

```
import UIKit
import SceneKit
import ARKit
import MapboxSceneKit

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()
    }

    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 touch = recognizer.location(in: sceneView)

        let hitTestResults = sceneView.hitTest(touch, types: .existingPlane)

        if !hitTestResults.isEmpty {

            let hitTestResult = hitTestResults.first!
            addTerrainNode(from: hitTestResult)
        }

    }
    private func addTerrainNode(from hitResult :ARHitTestResult) {

        // 38.874234,-104.875757
        //38.882720,-104.885666

        let terrainNode = TerrainNode(minLat: 38.874234, maxLat: 38.882720, minLon: -104.885666, maxLon: -104.875757)
        let scale = Float(0.333 * hitResult.distance) / terrainNode.boundingSphere.radius // scale based on the distance away
        terrainNode.transform = SCNMatrix4MakeScale(scale, scale, scale)
        terrainNode.position = SCNVector3(hitResult.worldTransform.columns.3.x, hitResult.worldTransform.columns.3.y, hitResult.worldTransform.columns.3.z)
        // set the material
        terrainNode.geometry?.materials = defaultMaterials()

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

        terrainNode.fetchTerrainHeights(minWallHeight: 50.0, enableDynamicShadows: true, progress: { _, _ in }, completion: {
            NSLog("Terrain load complete")
        })


        terrainNode.fetchTerrainTexture("mapbox/satellite-v9", progress: { _, _ in }, completion: { image in
            NSLog("Texture load complete")
            terrainNode.geometry?.materials[4].diffuse.contents = image
        })

    }

    private func defaultMaterials() -> [SCNMaterial] {
        let groundImage = SCNMaterial()
        groundImage.diffuse.contents = UIColor.darkGray
        groundImage.name = "Ground texture"

        let sideMaterial = SCNMaterial()
        sideMaterial.diffuse.contents = UIColor.darkGray
        //TODO: Some kind of bug with the normals for sides where not having them double-sided has them not show up
        sideMaterial.isDoubleSided = true
        sideMaterial.name = "Side"

        let bottomMaterial = SCNMaterial()
        bottomMaterial.diffuse.contents = UIColor.black
        bottomMaterial.name = "Bottom"

        return [sideMaterial, sideMaterial, sideMaterial, sideMaterial, groundImage, bottomMaterial]
    }

    override func viewWillAppear(_ animated: Bool) {
        super.viewWillAppear(animated)

        // Create a session configuration
        let configuration = ARWorldTrackingConfiguration()
        configuration.planeDetection = .horizontal

        // 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()
    }

}
```

// change the info.p list

MGLMapboxAccessToken string => register and get the token

=> // provided by the MapboxSceneKit docs

// Scan and detecting 3D objects

```
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
    }

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

        if anchor is ARObjectAnchor {

            let text = SCNText(string: "UFC Fighter", extrusionDepth: 0.0)
            text.firstMaterial?.diffuse.contents = UIColor.white

            let textNode = SCNNode(geometry: text)
            textNode.scale = SCNVector3(0.002,0.002,0.002)

            node.addChildNode(textNode)

        }

    }

    override func viewWillAppear(_ animated: Bool) {
        super.viewWillAppear(animated)

        guard let referenceObjects = ARReferenceObject.referenceObjects(inGroupNamed: "AR Resources", bundle: nil)
else {

            fatalError("Missing expected asset catalog")

        }

        // Create a session configuration
        let configuration = ARWorldTrackingConfiguration()
```

// use the app provided by apple to scan the object and save the file

Assets => new AR resources group => add .arobject

```
        configuration.detectionObjects = referenceObjects

        // 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()
    }

}
```