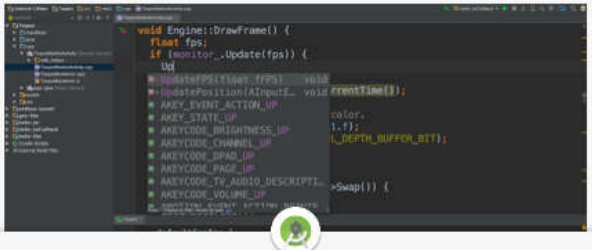
Google ARCORE

Nisfu Asrul Sani
@ Lab Langit 9 - PENS
20 - 22 November 2017

environtment

Dive right in

Choose a development environment



Android Studio

GET STARTED

Unity

GET STARTED

Unreal

GET STARTED

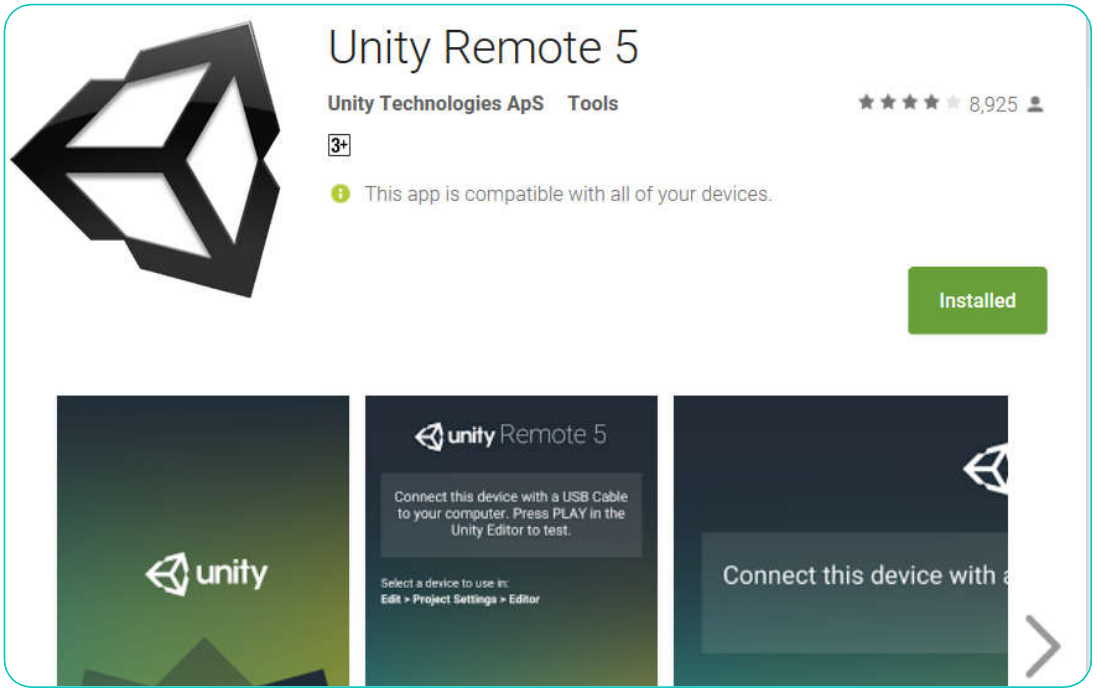
Setting up your development environment

- Install the Android SDK version 7.0 (API Level 24) or higher.
  - To install the Android SDK, install Android Studio.
  - To update the Android SDK, use the Android SDK Manager tool in Android Studio.
  - Install Unity 2017.2 Beta 11 or higher, with the Android Build Support component. For more info, see Downloading and Installing Unity.
- You will need to get the ARCore SDK for Unity. You can either:
  - Download the SDK Preview for Unity and extract it.
  - -or-
  - Clone the repository with the following command:
  - git clone https://github.com/google-ar/arcore-unity-sdk.git

Prepare your device

You must use a supported, physical device. ARCore does not support virtual devices such as the Android Emulator. To prepare your device:

- Enable developer options
- Enable USB debugging
- Install the ARCore Service on the device:
  - Download the ARCore Service
  - Connect your Android device to the development machine with a USB cable
  - Install the service by running the following adb command:
  - adb install -r -d arcore-preview.apk

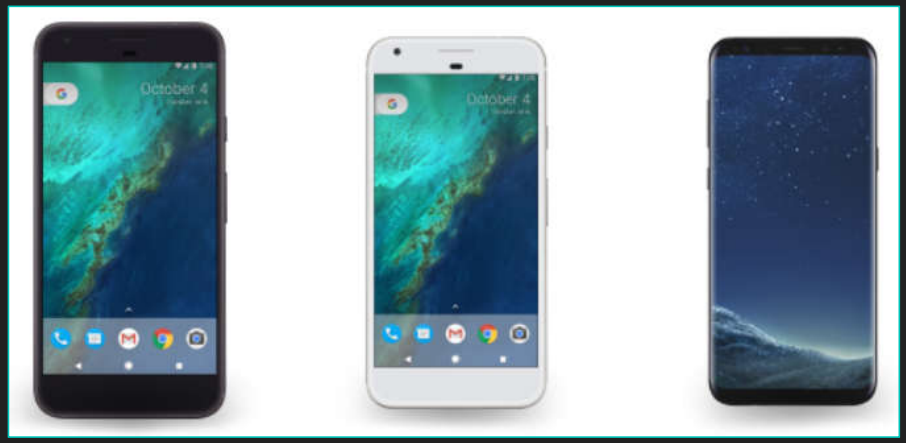


Additional

- https://play.google.com/store/apps/details?id=com.unity3d.genericremote

Supported Devices

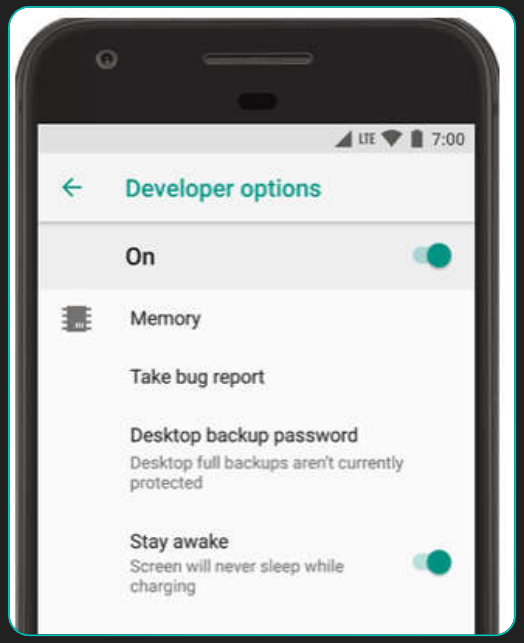
- ARCore is designed to work on a wide variety of qualified Android phones running N and later. During the SDK preview, ARCore supports the following devices:
  - Google Pixel, Pixel XL, Pixel 2, Pixel 2 XL
  - Samsung Galaxy S8 (SM-G950U, SM-G950N, SM-G950F, SM-G950FD, SM-G950W, SM-G950U1)
- Initially, ARCore will only work with Samsung's S8 and S8+ and Google's Pixel phone, but by the end of the year, Google promised to have support for 100 million Android phones, including devices from LG, Huawei and Asus, among others.



Enable developer options and debugging

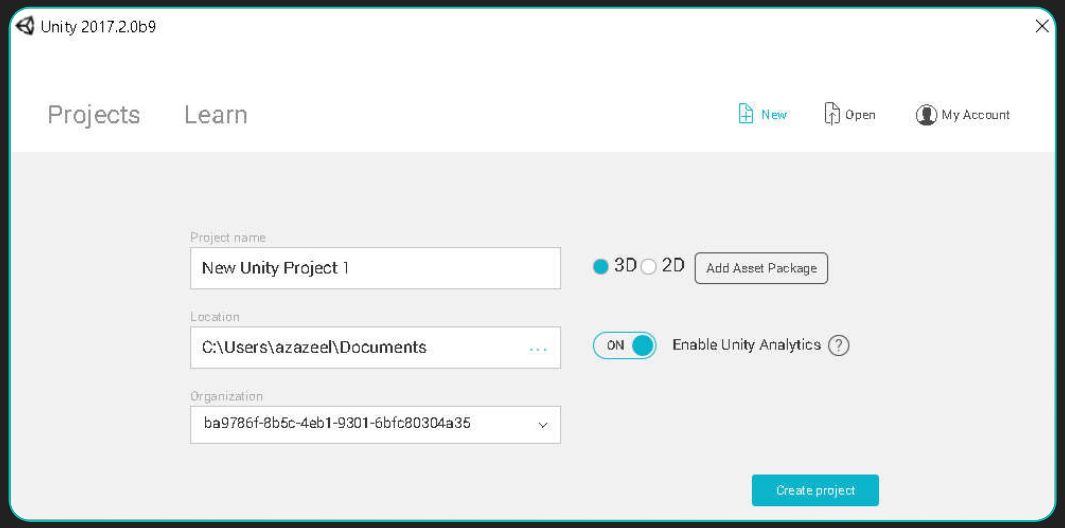
On Android 4.1 and lower, the Developer options screen is available by default. On Android 4.2 and higher, you must enable this screen as follows:

1. Open the Settings app.
2. (Only on Android 8.0 or higher) Select System.
3. Scroll to the bottom and select About phone.
4. Scroll to the bottom and tap Build number 7 times.
5. Return to the previous screen to find Developer options near the bottom.



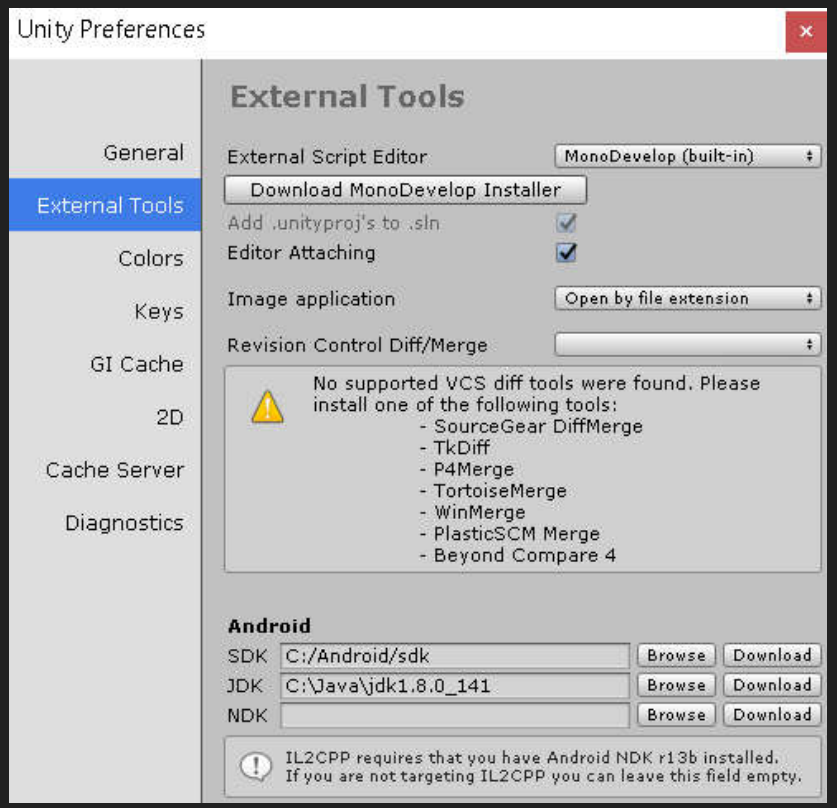
Create a new project

- In Unity, File > New Project.
- Name your project HelloAR, leave 3D mode selected, and click Create Project.



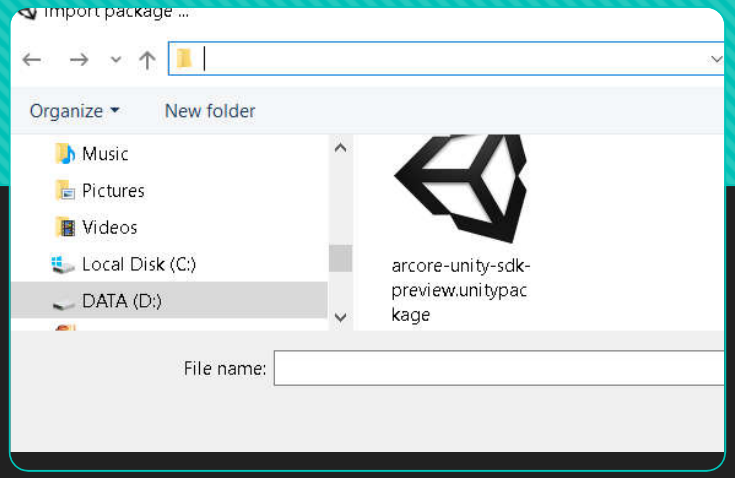
For the first Time, setting unity

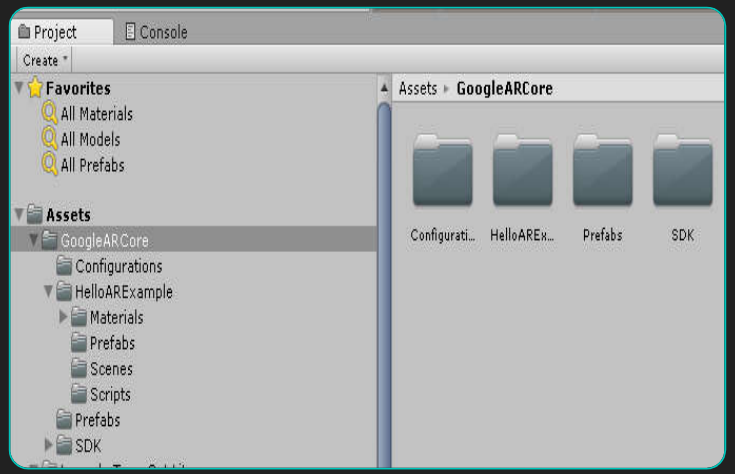
- Unity, Edit > Preferences
- External Tools Tab
  - SDK : on your android sdk path
  - Java : on your android java path

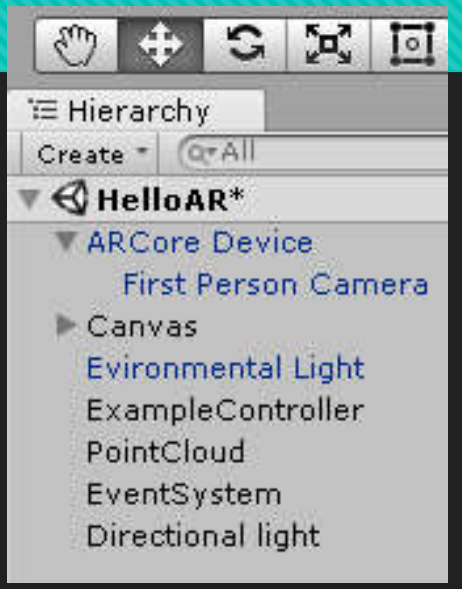


Import the SDK

- Import the arcore-unity-sdk-preview.unitypackage by clicking the menu Assets > Import package > Custom Package and import everything in the Unity package.
- In the Unity's Project window, you can see the both the Unity SDK files and the HelloAR sample app.

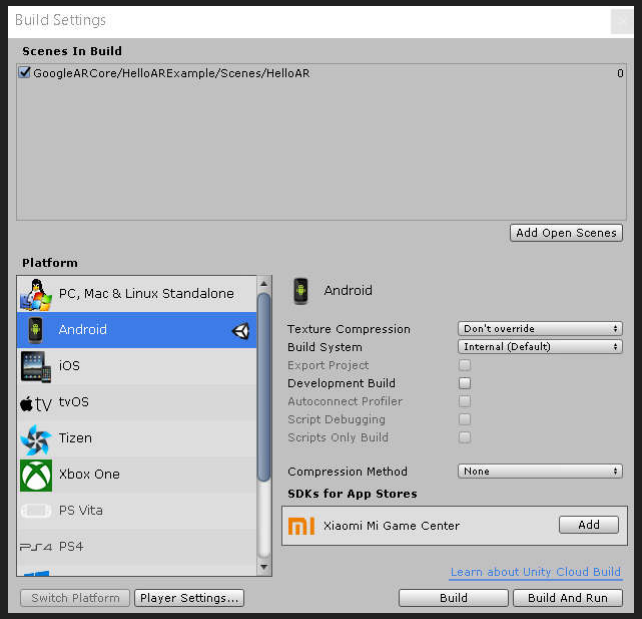


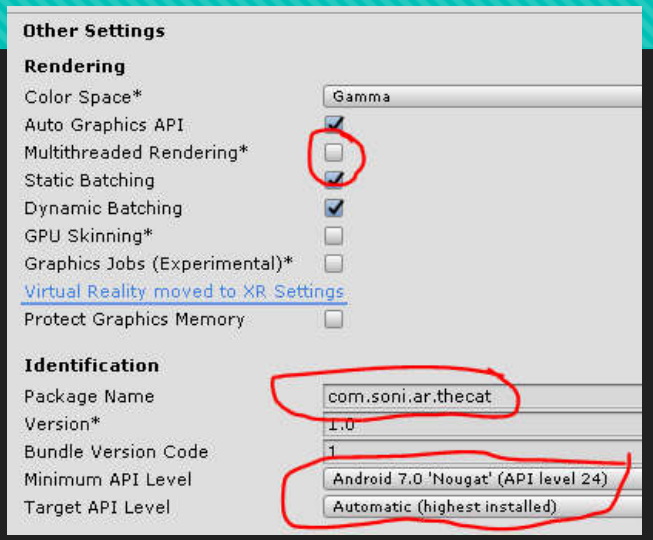




Configure the build settings

- Open the Build Settings window by clicking **File > Build settings.**
- Change the **target platform** to Android and click **Switch Platform**.
- Click Player Settings to open the Android Player Settings. Then change the following settings:
  - Other Settings > **Multithreaded Rendering: Off**
  - Other Settings > Package Name: a unique app ID that looks like a Java package name, such as **com.example.helloAR**
  - Other Settings > Minimum API Level: **Android 7.0 or higher**
  - Other Settings > Target API Level: **Android 7.0 or 7.1** or **Automatic**
- XR Settings > ARCore Supported: On
- The Scenes in Build lists the scenes from your project that will be included in your build. Add the scene from the HelloAR sample app to the build as follows:
- Locate the scene in the Project window under Assets > GoogleARCore > HelloARExample > Scenes.
- Click the scene and drop it on Scenes in Build.



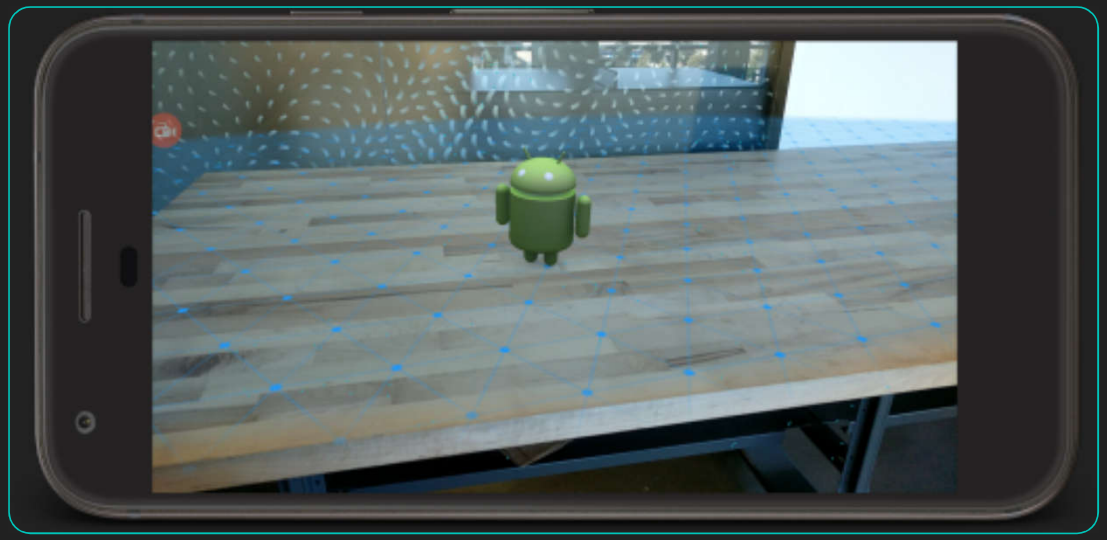


Build and run the sample app

- Make sure your device is connected to your machine and then click Build and Run. Unity builds your project into an Android APK, installs it on the device, and launches it.

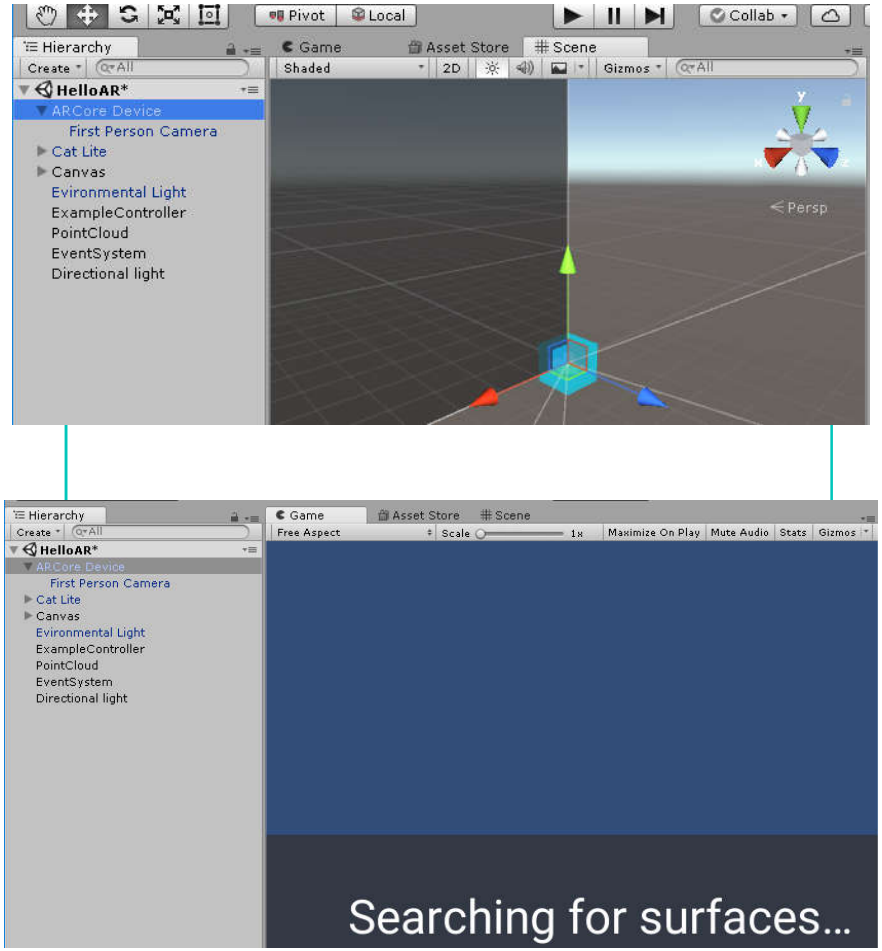
- As you move your device, the app automatically detects flat surfaces and draws a diamond grid. Tap the grid to place an Android robot on the surface.

CONGRATULATION

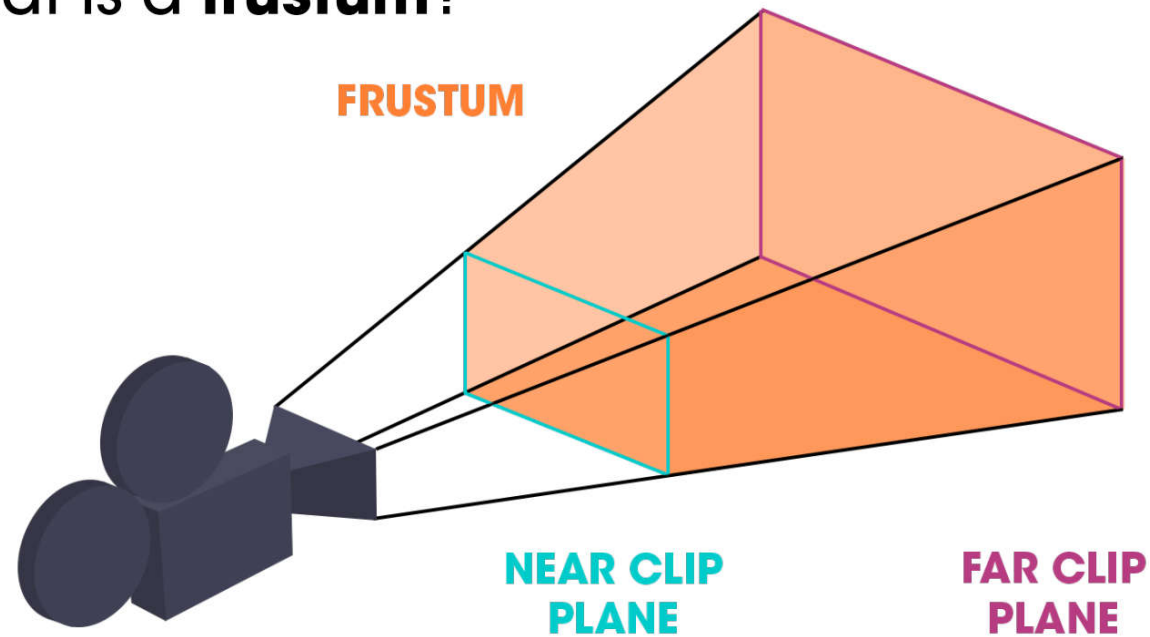


Understand our project

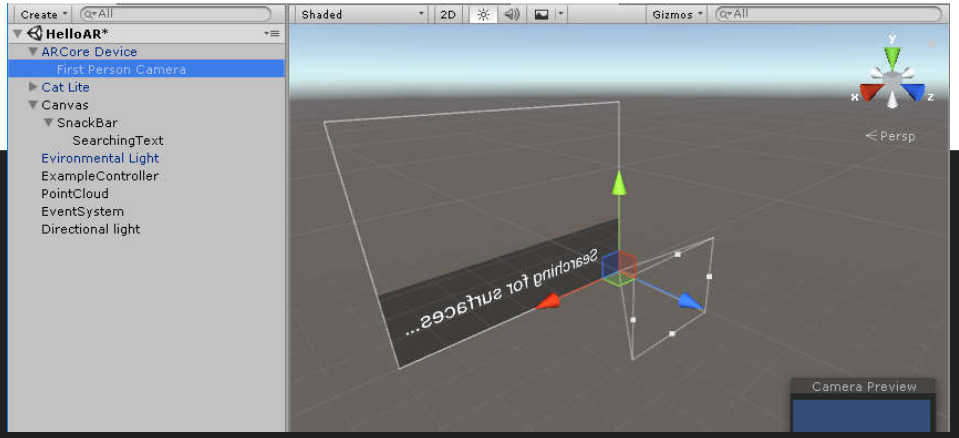
- ARCORE Device
- Scene tab
- Game Tab

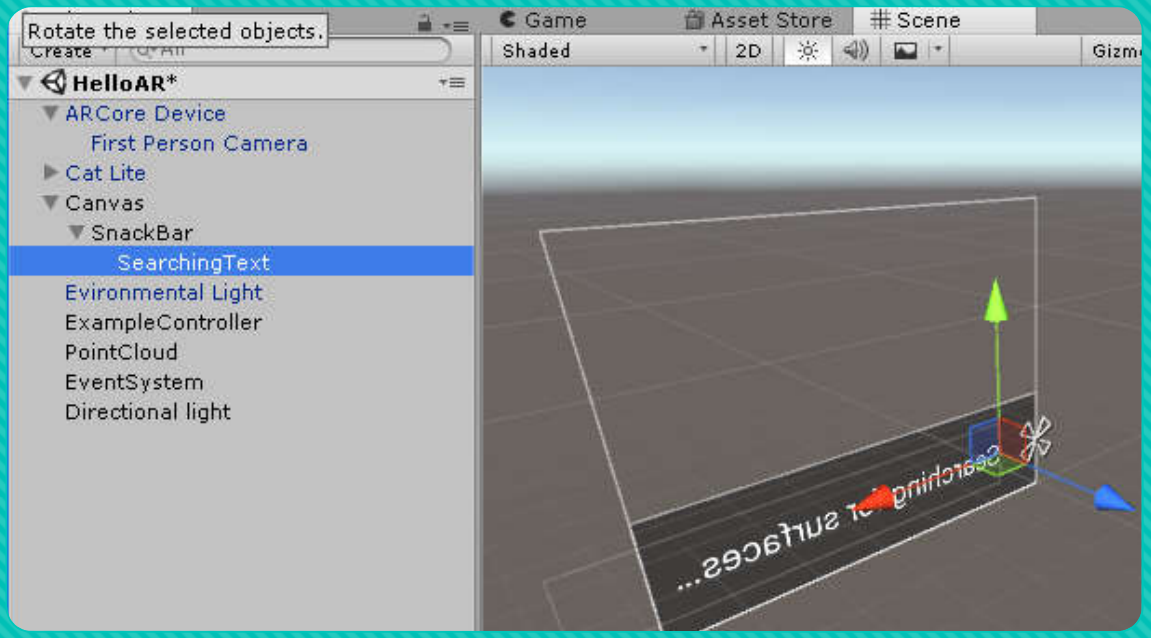


What is a **frustum**?



FUSTRUM

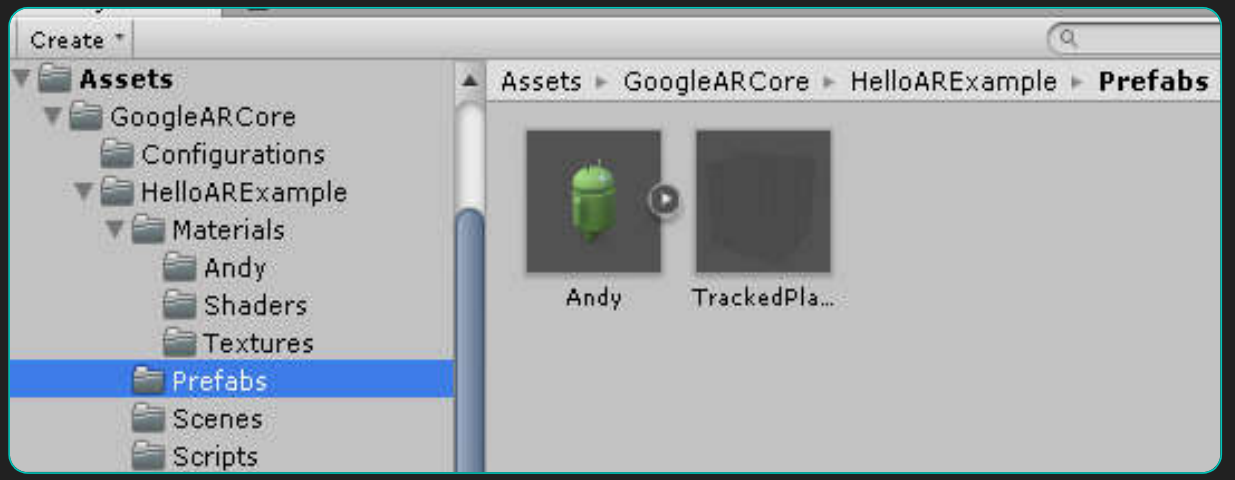


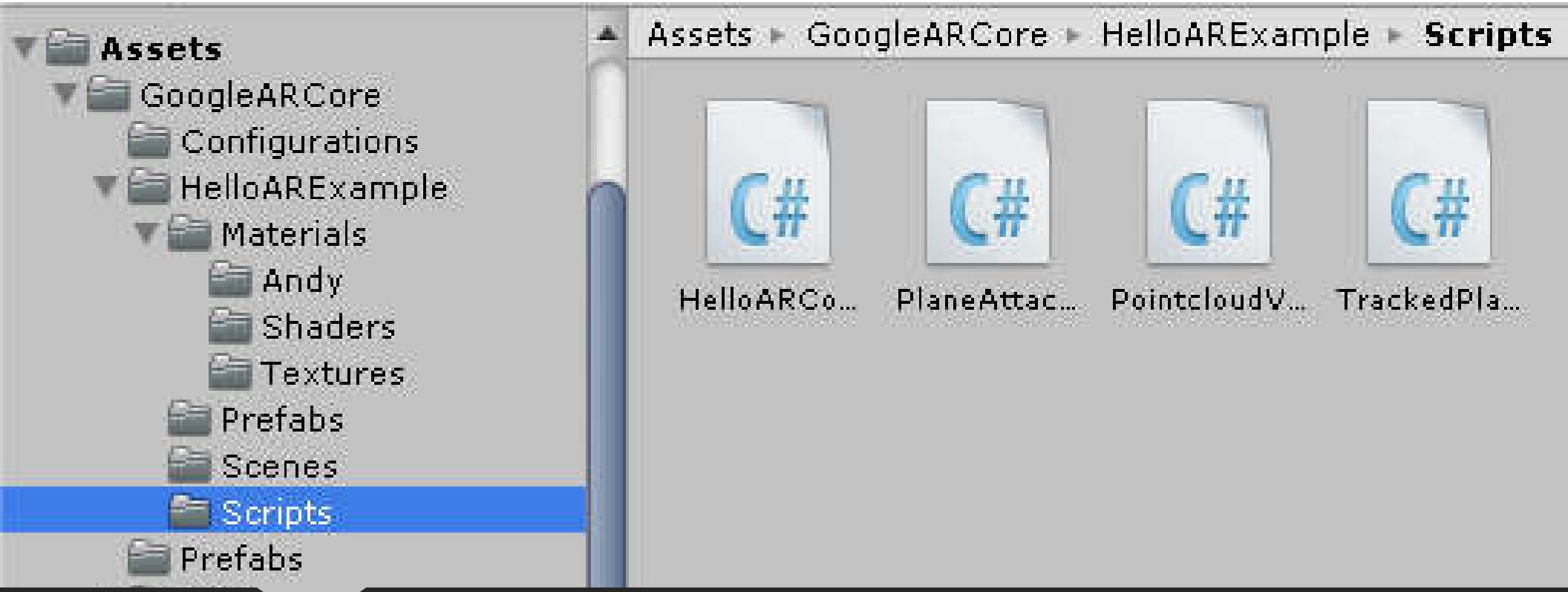


SnackBar and SearchingText

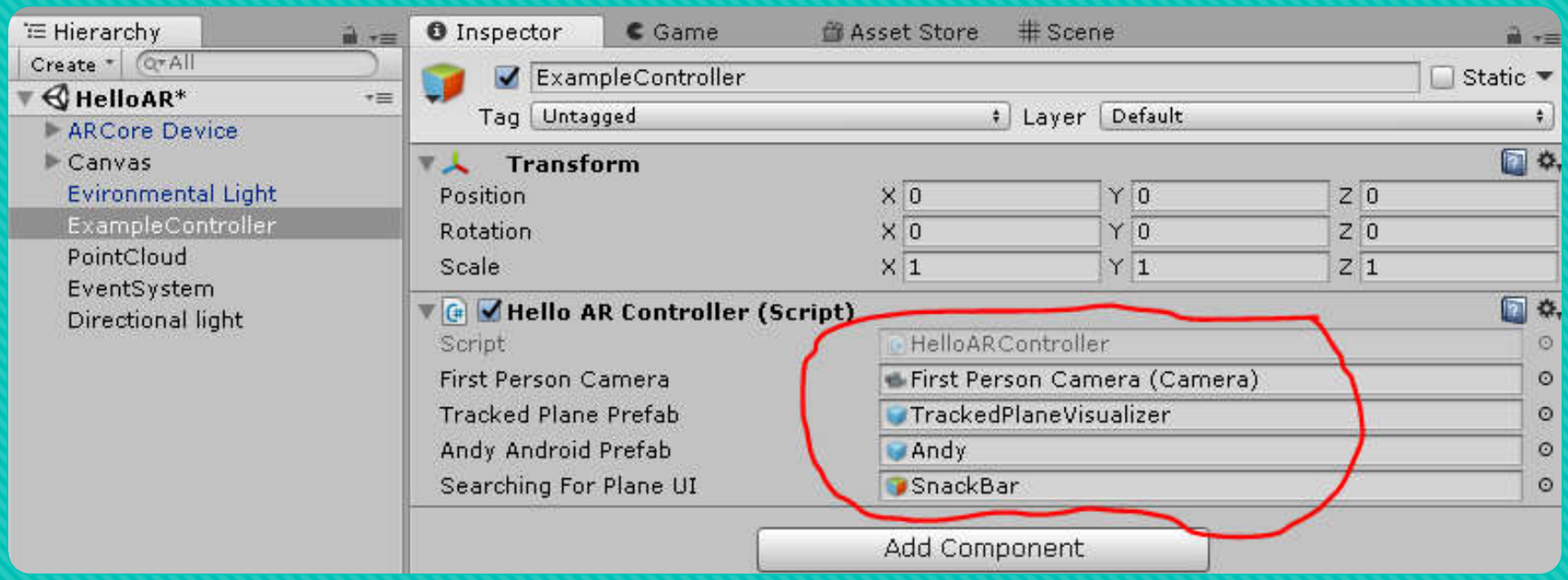
Prefabs

- is a type of asset
- a reusable GameObject stored in Project View. Prefabs can be inserted into any number of scenes, multiple times per scene. When you add a Prefab to a scene, you create an instance of it. All Prefab instances are linked to the original Prefab and are essentially clones of it.





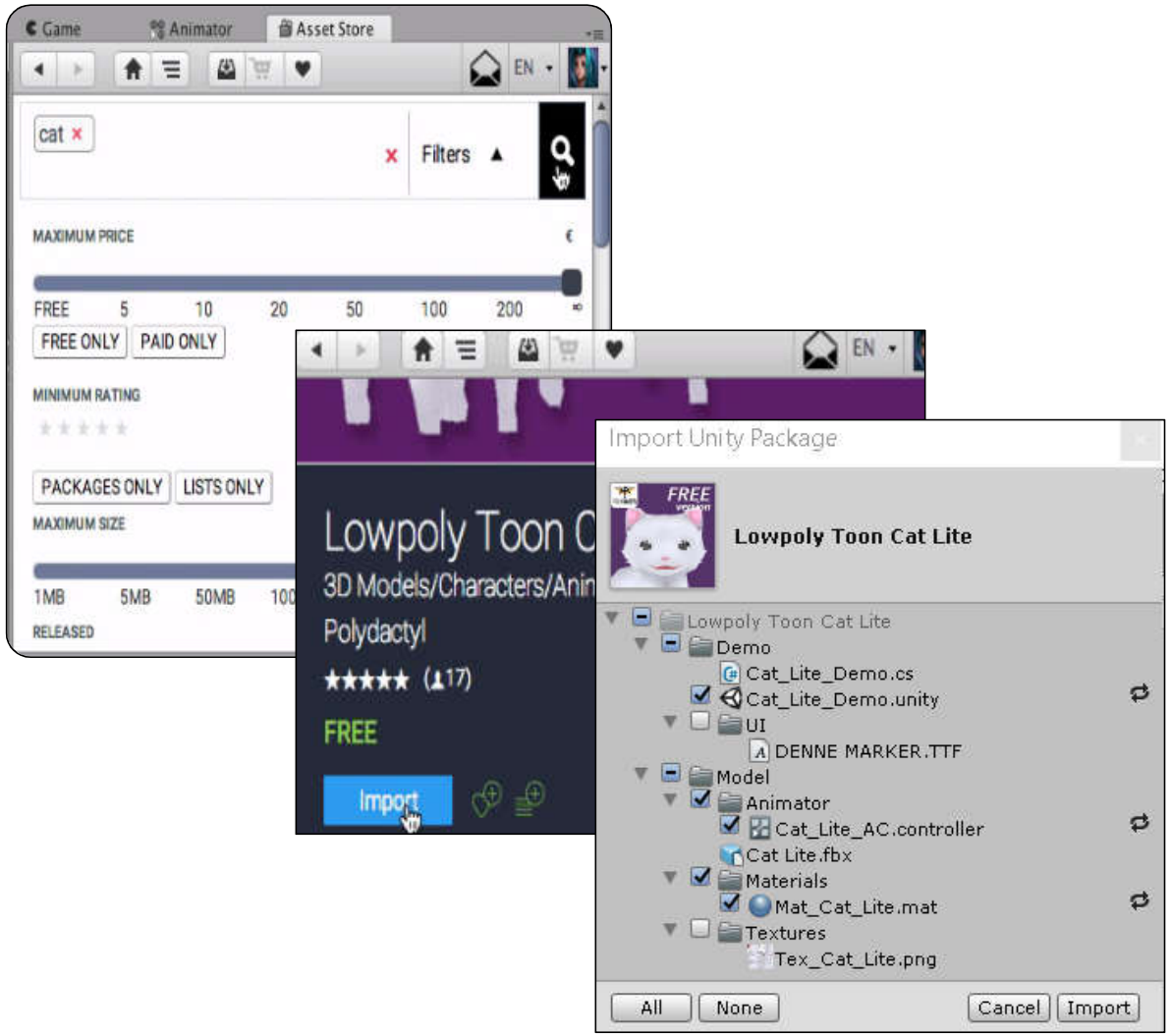
ARCORE BASIC SCRIPT (C#)



Example Controller

MODIFY SAMPLE PROJECT

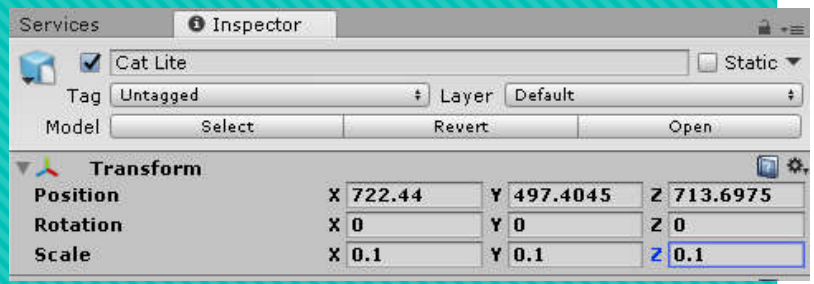
CHANGE - ROBOTO

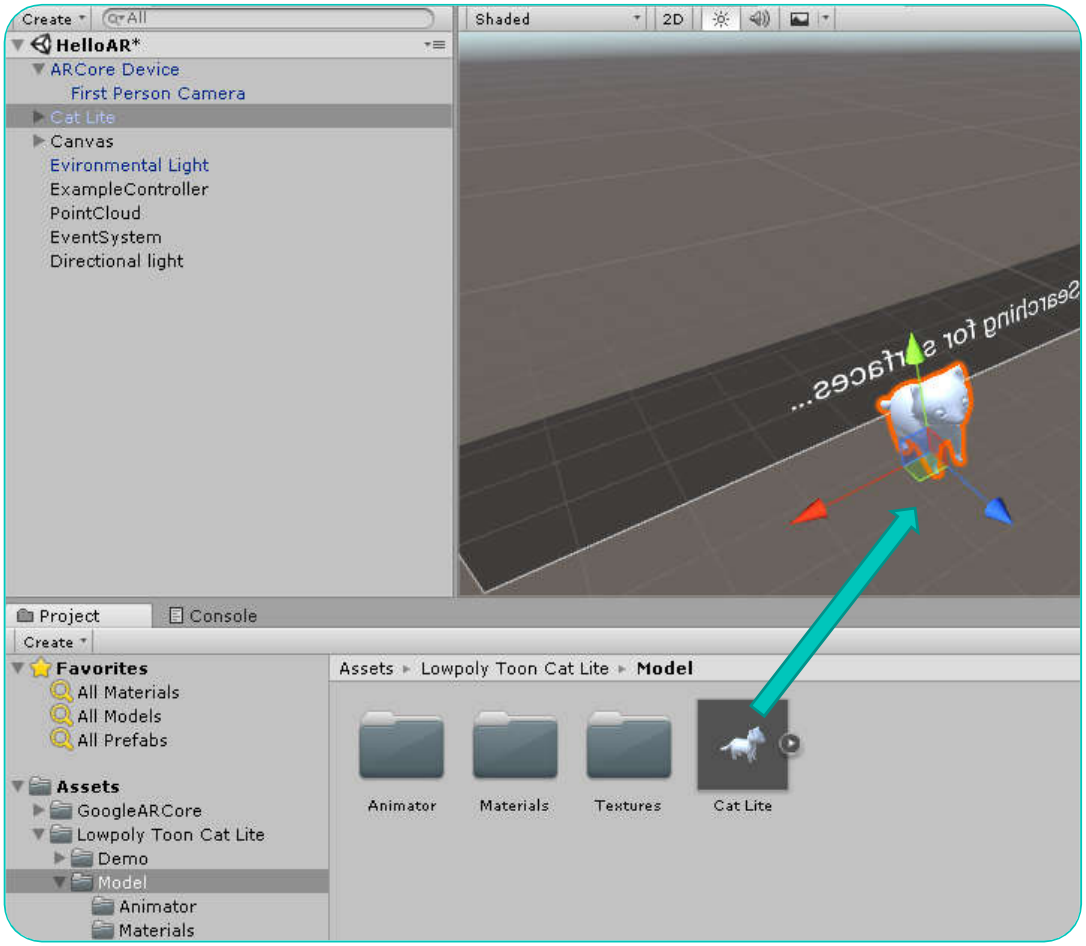


Download From Unity ASSET STORE

then import

Add Object From Asset to preview and modify

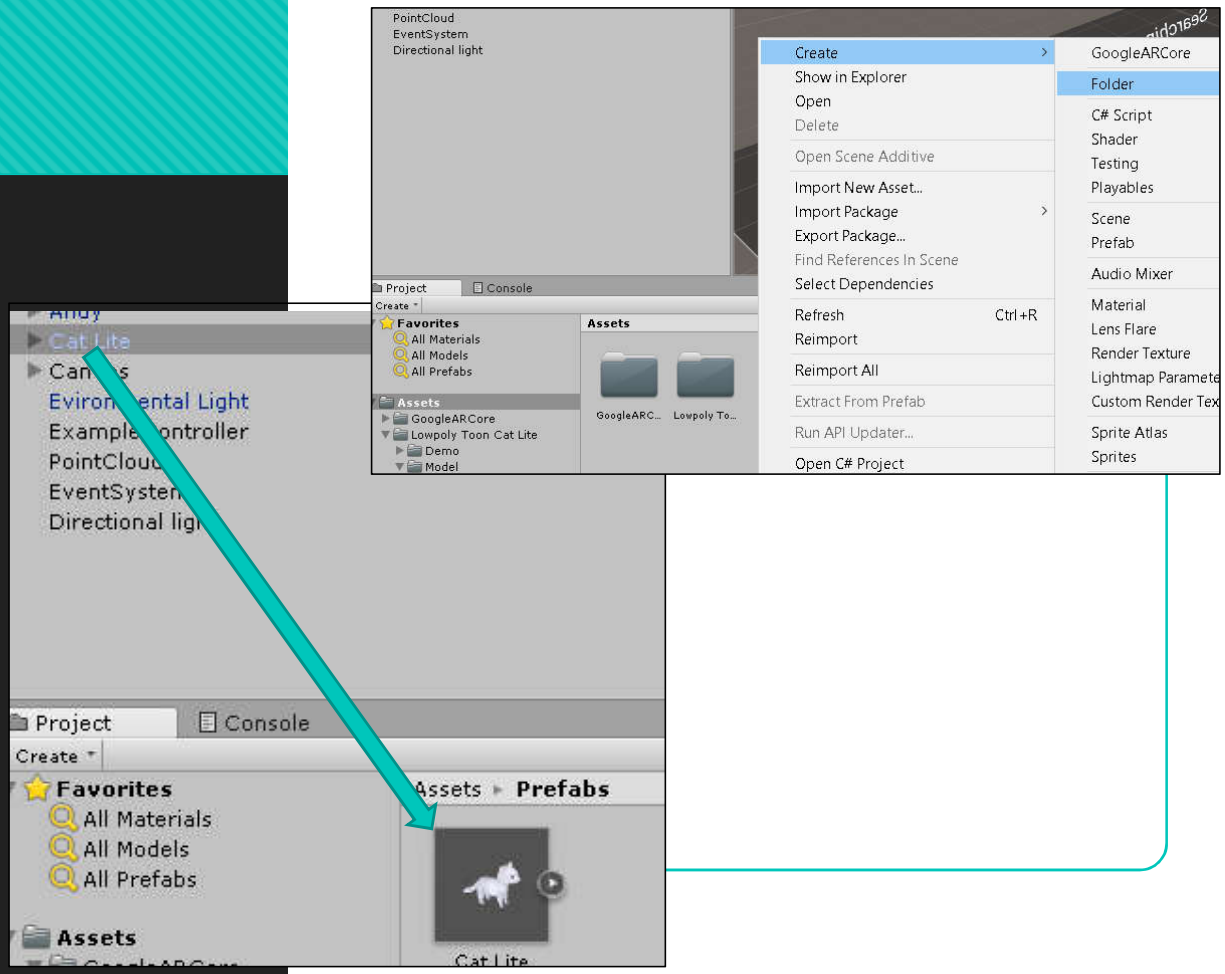




Create Own Prefabs

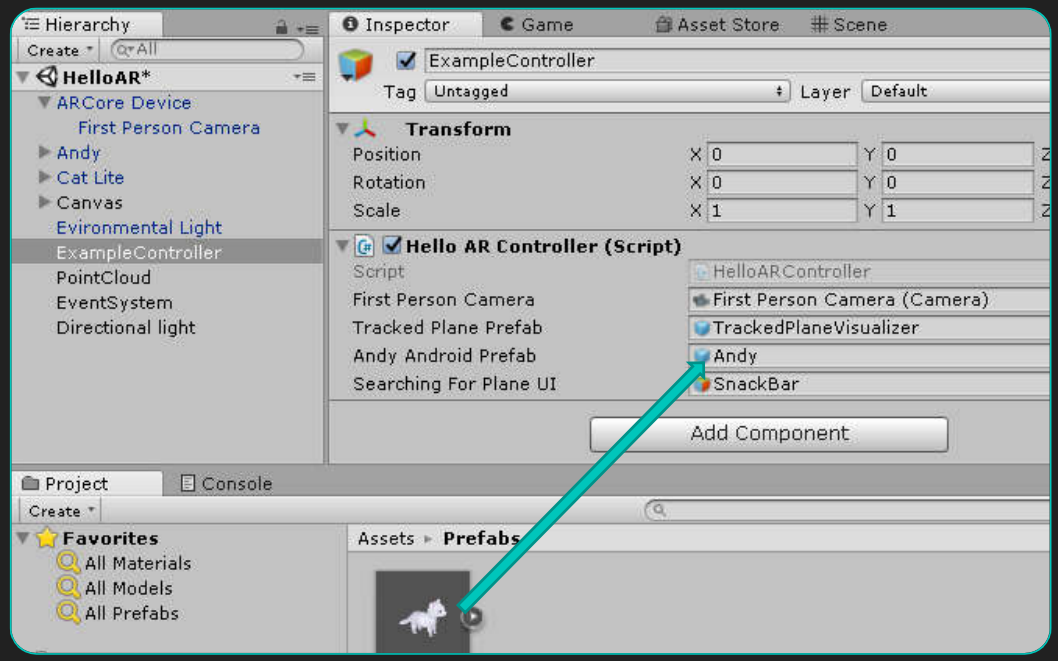
- Open Asset Folder in Project
- Right Click > Create > Folder
- Rename > Prefabs

- Click And Drag Cat to Prefabs

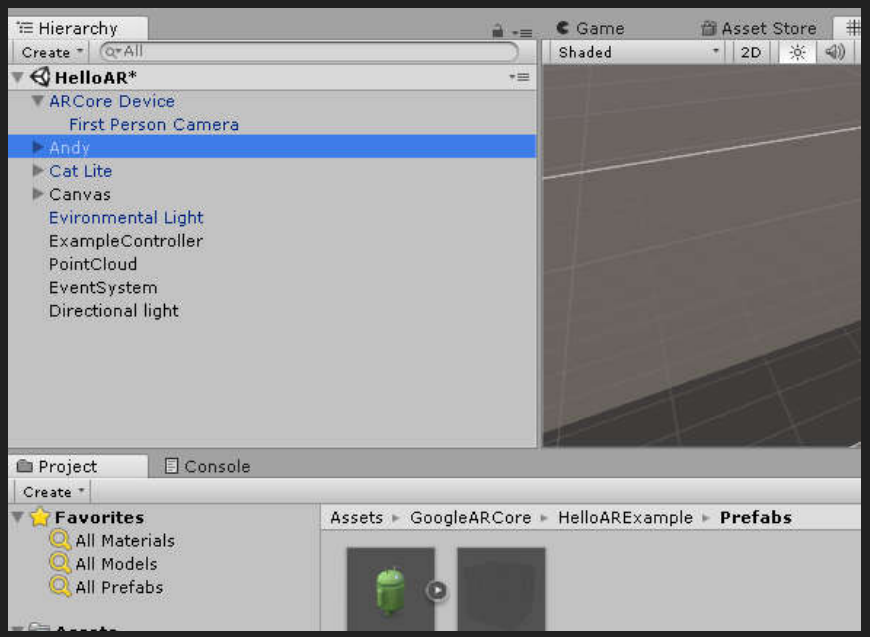


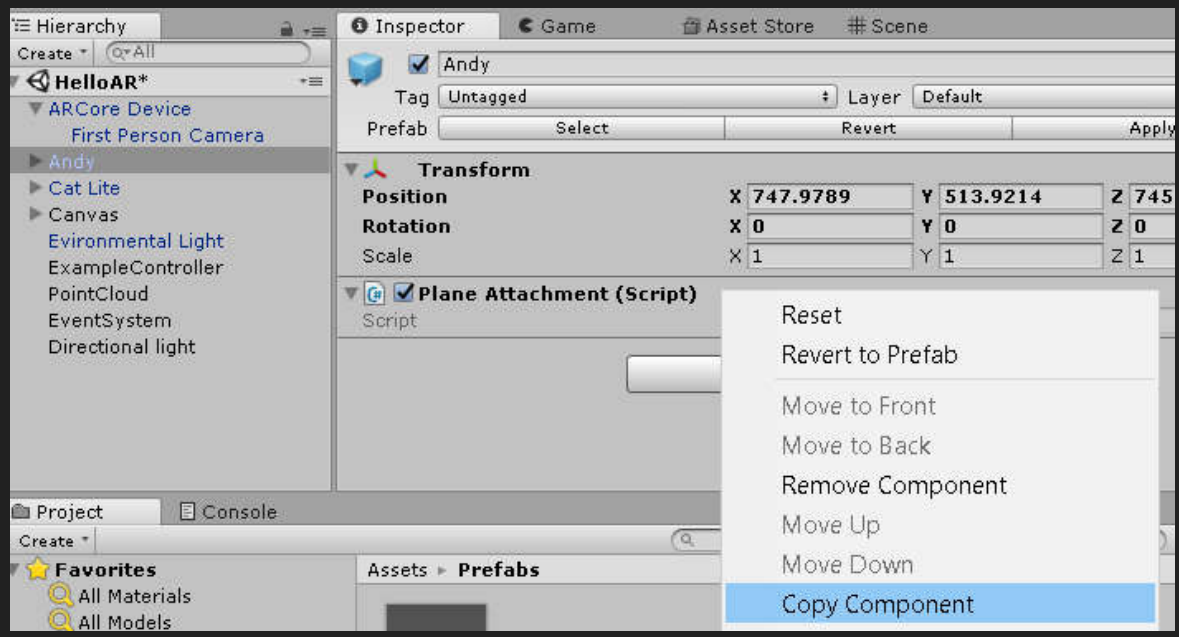
Change Cat as the Actor in scene

- In ExampleController
- Click Drag Cat Prefab to Andy Android Prefab

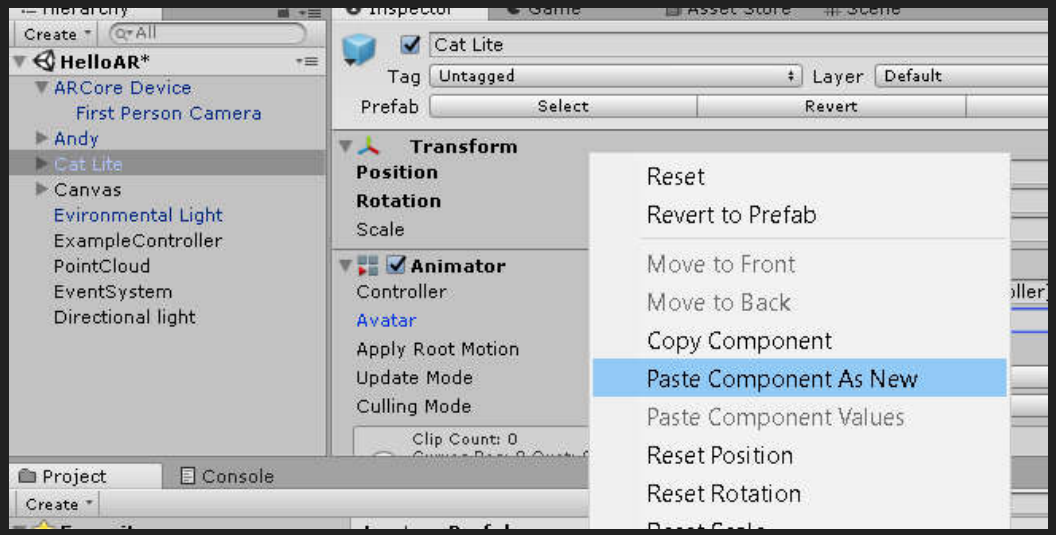


Not Only the Object But Also its behaviour





Copy Component from Andi and Paste to Cat

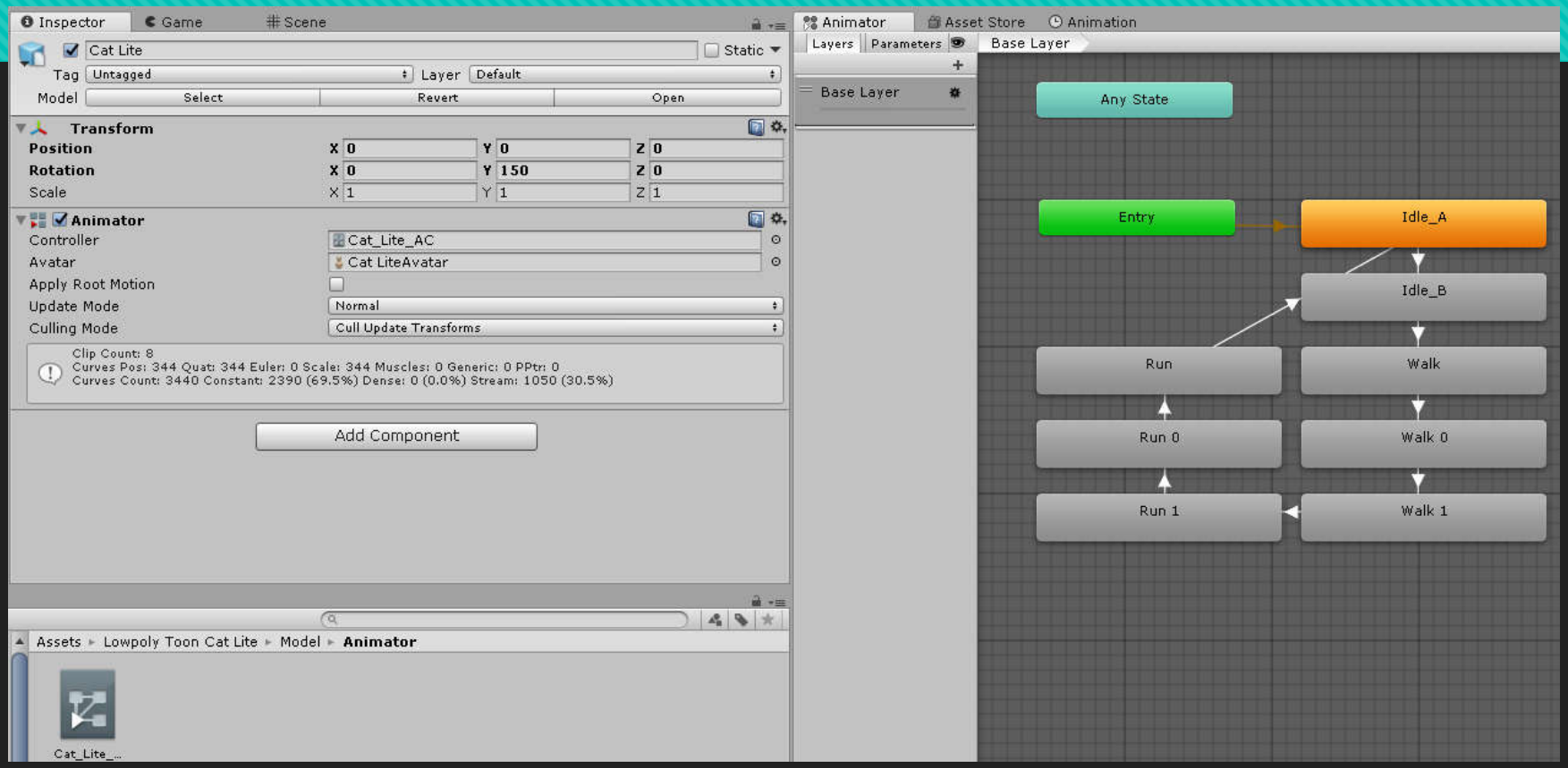


BUILD AND RUN - again

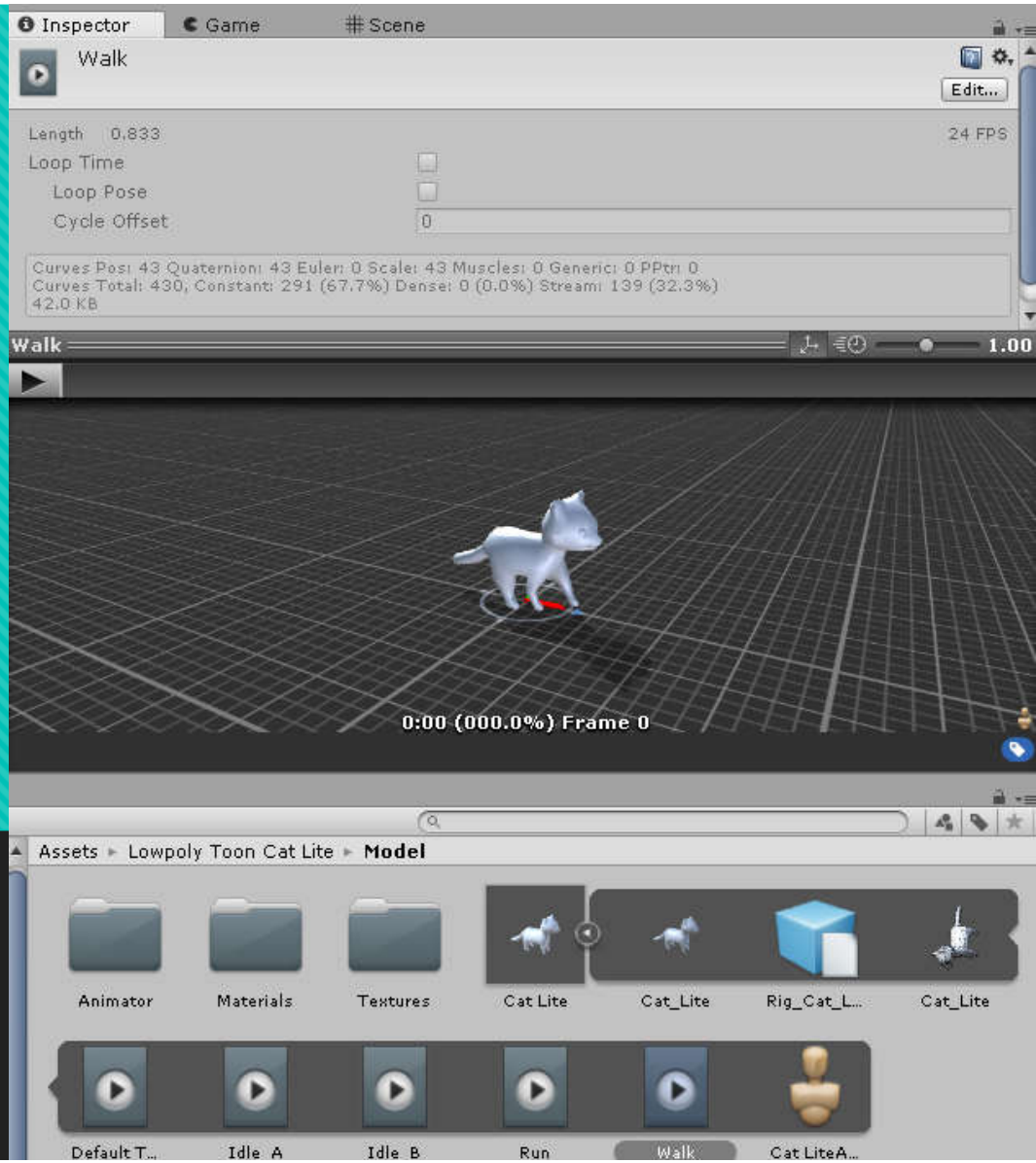
ADD ANIMATION

PREDEFINED IN UNITY

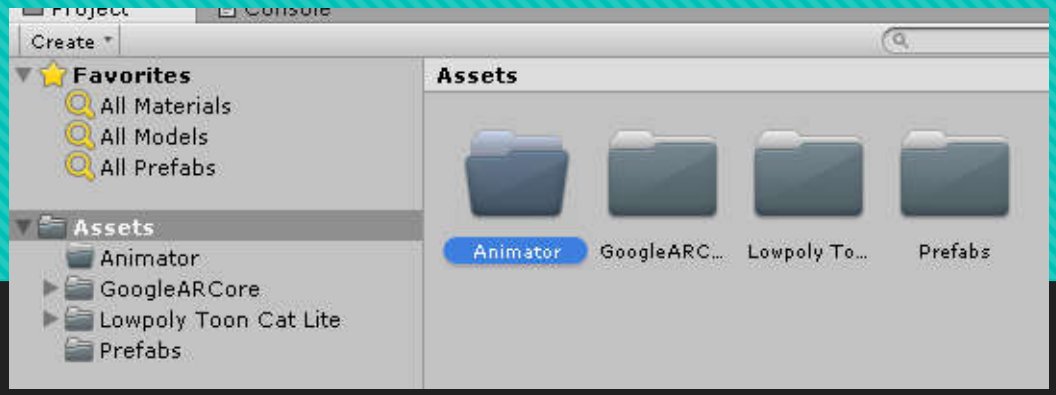
ANIMATION is already given



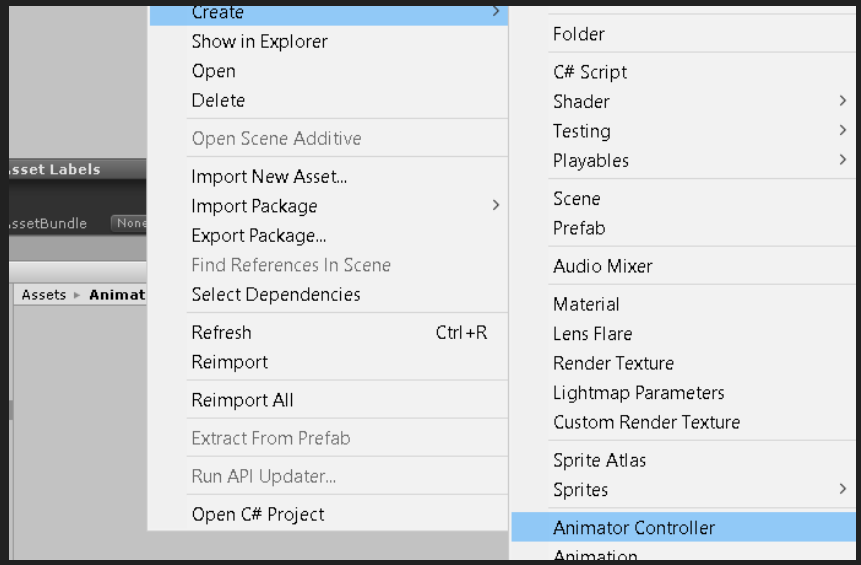
Open Model and inspect “WALK” animation in inspector tab



Create custom Animation

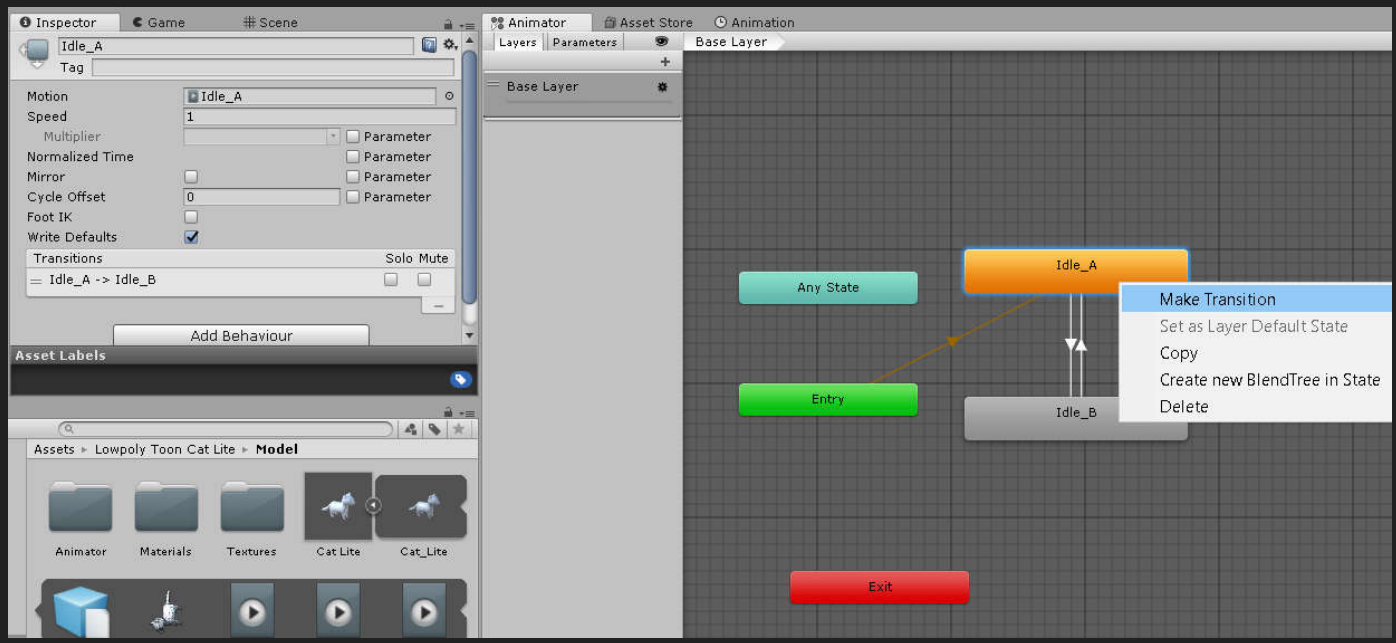


- Create New Folder at Asset
- Right Click and Select Animation Controller



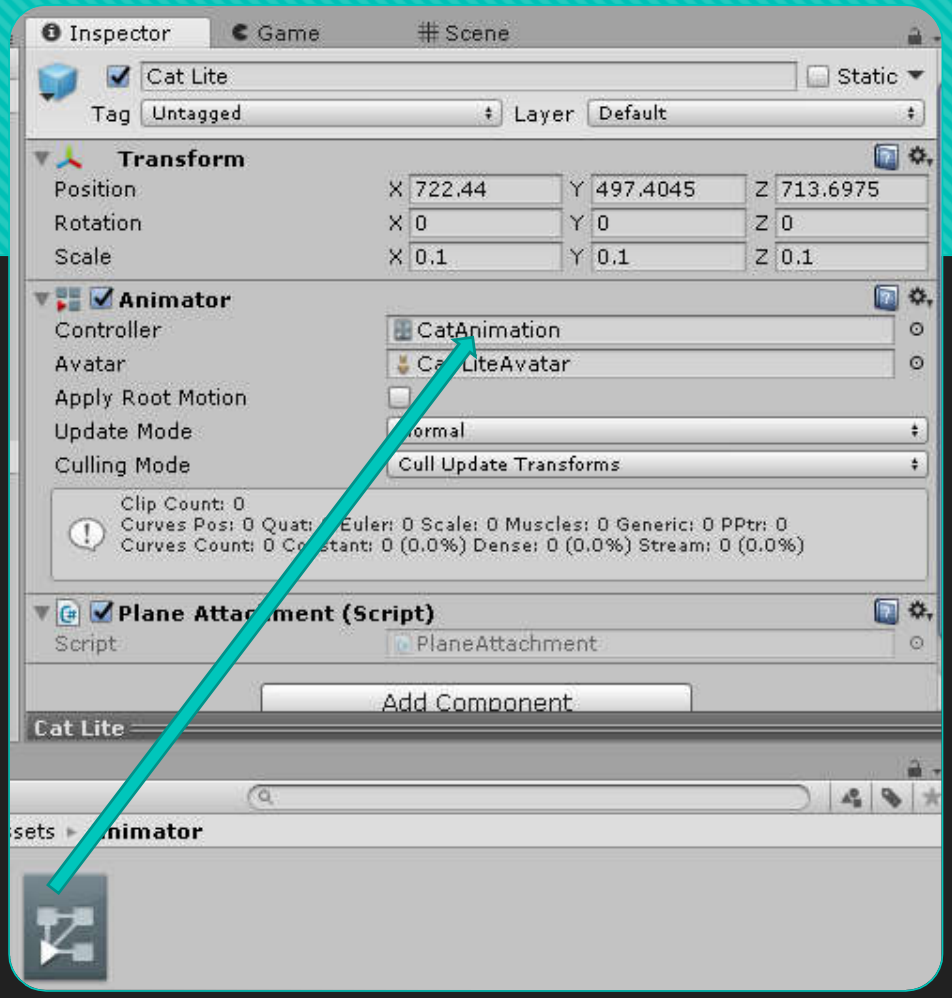
Pick Animation to be added

- Click And Drag Animation types from MODEL to Anomator
- Right Click > Make Transition from one STATE to other STATE



Place into Cat PREFAB

- Click And Drag the ANIMATOR CONTROLLER as Controller CAT



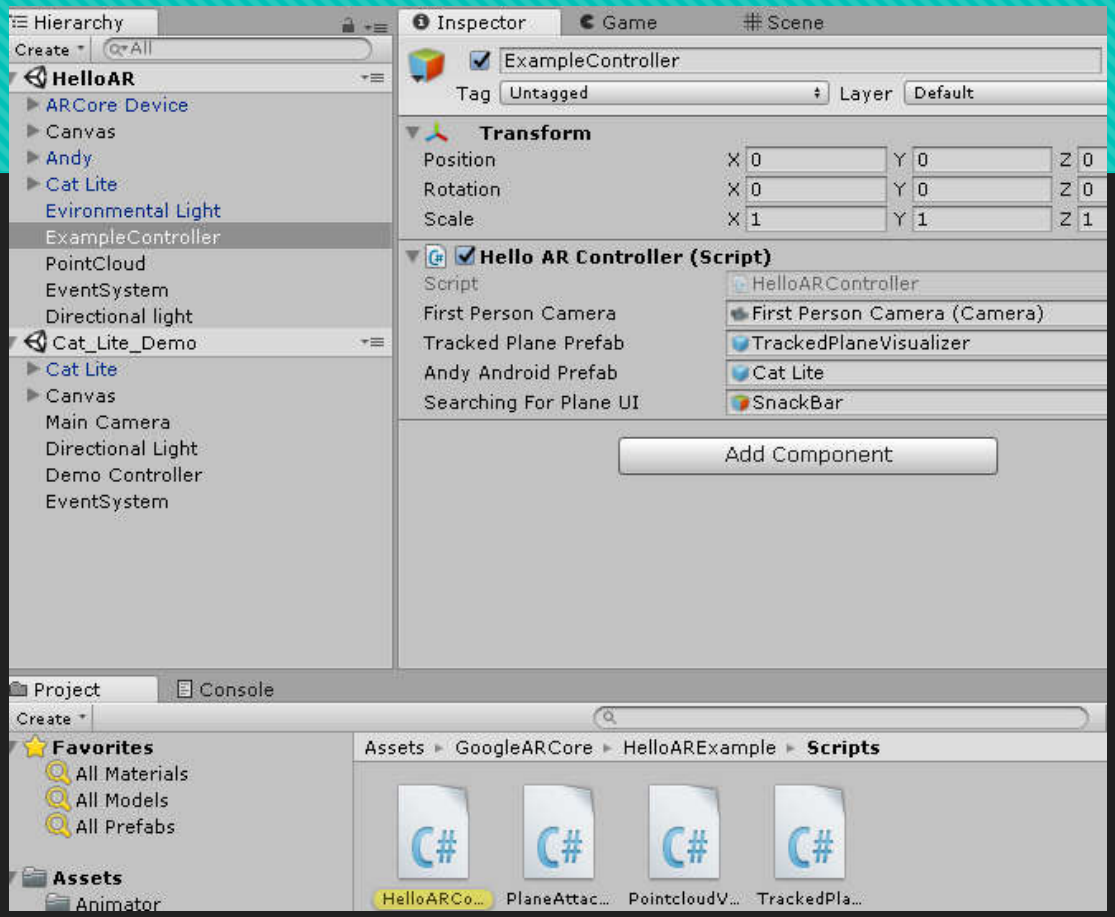
MODIFY SCRIPT

Simple Change

Open Script

- Find ExampleController
  - In script section, it was typed as HelloARControlller
  - Open HelloARControlller

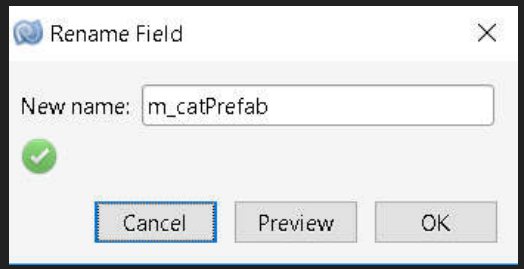
And **Mono Developer** will be automatically opened and show the codes



Renaming variable

```
41        public GameObject m_trackedPlanePrefab;
42
43        /// <summary>
44        /// A model to place when a raycast from a user touch hits a plane.
45        /// </summary>
46        public GameObject m_andyAndroidPref
47
48        /// <summary>
```


- Refactor (change all related)
  - Locate the m_andyAndroidPreab
  - Right Click > Refactor > Rename
  - Change to m_catPrefab



  - Locate the andyObject
  - Right Click > Refactor > Rename
  - Change to catObject

```
138             // Intanstiate an Andy Android object as a child of the anchor; it's transform
139             // from the anchor's tracking.
140             var andyObject = Instantiate(m_catPrefab, hit.Point, Quaternion.identity,
141                 anchor.transform);
142
143             // Andy should look at the camera but still be flush with the plane.
144             andyObject.transform.LookAt(m_firstPersonCamera.transform);
145             andyObject.transform.rotation = Quaternion.Euler(0.0f,
146                 andyObject.transform.rotation.eulerAngles.y, andyObject.transform.rotation.
147
148             // Use a plane attachment component to maintain Andy's y-offset from the plane
149             // (occurs after anchor updates).
150             andyObject.GetComponent<PlaneAttachment>().Attach(hit.Plane);
```

Just One Cat at time (1)

- Add a new Variable named “numberOfCatsAllowed “ as an integer and set into 1
- Add a new Variable named “currentNumberOfCats“ as an integer and set into 0
- Placed it in the HelloARController class

```
53        private List<TrackedPlane> m_newPlanes = new List<TrackedPlane>();
54
55        private List<TrackedPlane> m_allPlanes = new List<TrackedPlane>();
56
57        public int numberOfCatsAllowed = 1;
58        public int currentNumberOfCats = 0;
59
```

We create a class variable/property, by convention place with other variables/properties

Do not place it in a method

Just One Cat at time (2)

- Locate the code to detect touch event
- Check whether it is still allowed to add a new cat by comparing “current number of cats” and “number of cats allowed”
- If it’s true we add cat object and add current number of cats

```
133             TrackableHitFlag raycastFilter = TrackableHitFlag.PlaneWithinBounds
134
135             if (Session.Raycast(m_firstPersonCamera.ScreenPointToRay(touch.posit
136             {
137                 if (currentNumberOfCats < numberOfCatsAllowed) {
138                     currentNumberOfCats++;
139
140
141                 }
```

Just One Cat at time (3) – move the code

```
if (Session.Raycast(m_firstPersonCamera.ScreenPointToRay(touch.position), raycastFilter, out hit))
{
    if (currentNumberOfCats < numberOfCatsAllowed) {
        currentNumberOfCats++;


    }
        // Create an anchor to allow ARCore to track the hitpoint as understanding of the physical
        // world evolves.
        var anchor = Session.CreateAnchor(hit.Point, Quaternion.identity);

        // Intanstiate an Andy Android object as a child of the anchor; it's transform will now bene
        // from the anchor's tracking.
        var catObject = Instantiate(m_catPrefab, hit.Point, Quaternion.identity,
            anchor.transform);

        // Andy should look at the camera but still be flush with the plane.
        catObject.transform.LookAt(m_firstPersonCamera.transform);
        catObject.transform.rotation = Quaternion.Euler(0.0f,
            catObject.transform.rotation.eulerAngles.y, catObject.transform.rotation.z);

        // Use a plane attachment component to maintain Andy's y-offset from the plane
        // (occurs after anchor updates).
        catObject.GetComponent<PlaneAttachment>().Attach(hit.Plane);
}
```

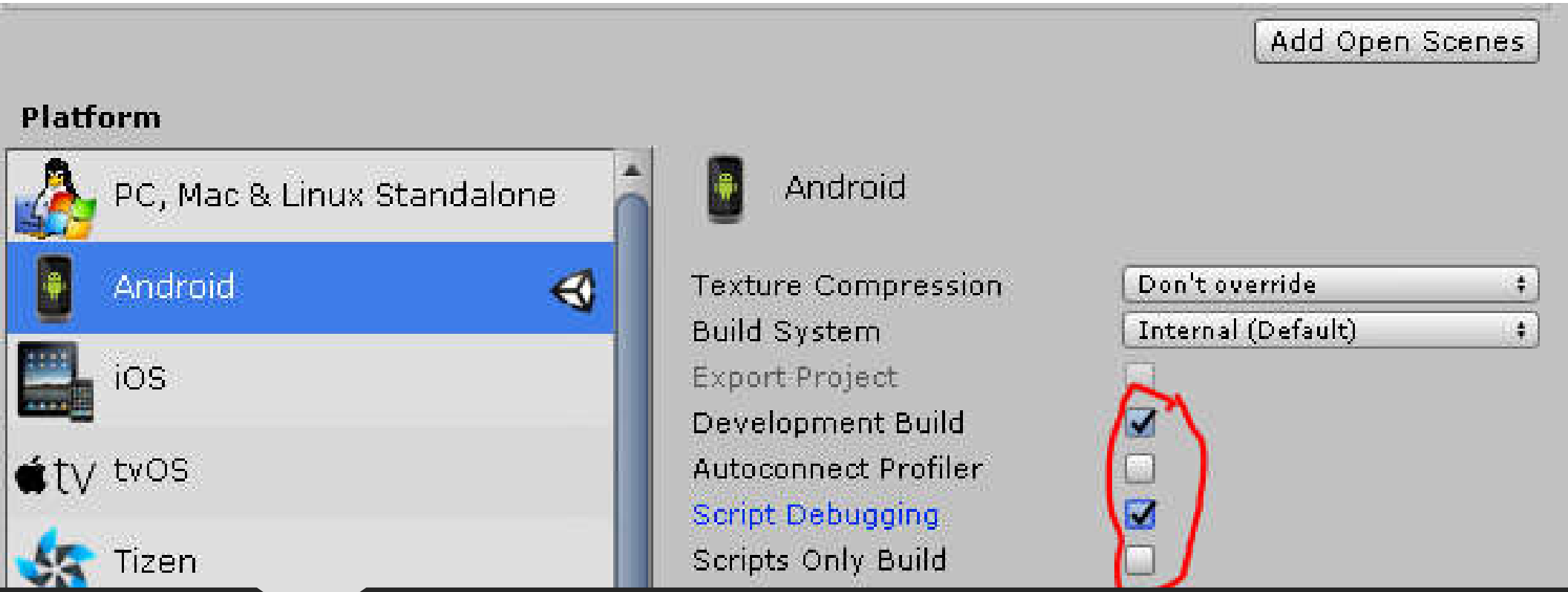
```
if (Session.Raycast(m_firstPersonCamera.ScreenPointToR
{
    if (currentNumberOfCats < numberOfCatsAllowed) {
        currentNumberOfCats++;
        // Create an anchor to allow ARCore to track
        // world evolves.
        var anchor = Session.CreateAnchor(hit.Point,

        // Intanstiate an Andy Android object as a ch
        // from the anchor's tracking.
        var catObject = Instantiate(m_catPrefab, hit.
            anchor.transform);

        // Andy should look at the camera but still b
        catObject.transform.LookAt(m_firstPersonCamera
        catObject.transform.rotation = Quaternion.Eul
            catObject.transform.rotation.eulerAngles.

        // Use a plane attachment component to mainta
        // (occurs after anchor updates).
        catObject.GetComponent<PlaneAttachment>().Att
    }

}
```



Debugging

enable (check) Development Build and Script Debugging

Put Log while suspecting BUG

- Print out when user touch the phone
- Print out value of
  - currentNumberOfCats
  - numberOfCatAllowed
- Print out cat object

```
if (Session.Raycast(m_firstPersonCamera.ScreenPointToRay(touch.positio
{
    Debug.Log ("TOUCH PHONE");
    Debug.Log ("currentNumberOfCats:"+currentNumberOfCats);
    Debug.Log ("numberOfCatsAllowed:"+numberOfCatsAllowed);
    if (currentNumberOfCats < numberOfCatsAllowed) {
        currentNumberOfCats++;
        // Create an anchor to allow ARCore to track the hitpoint as u
        // world evolves.
        var anchor = Session.CreateAnchor(hit.Point, Quaternion.identi

        // Intanstiate an Andy Android object as a child of the anchor
        // from the anchor's tracking.
        var catObject = Instantiate(m_catPrefab, hit.Point, Quaternion
            anchor.transform);
        Debug.Log ("catObject" + catObject);
        // Andy should look at the camera but still be flush with the
        catObject.transform.LookAt(m_firstPersonCamera.transform);
```

Project | Console

Clear | Collapse | Clear on Play | Error Pause | Connected Playe

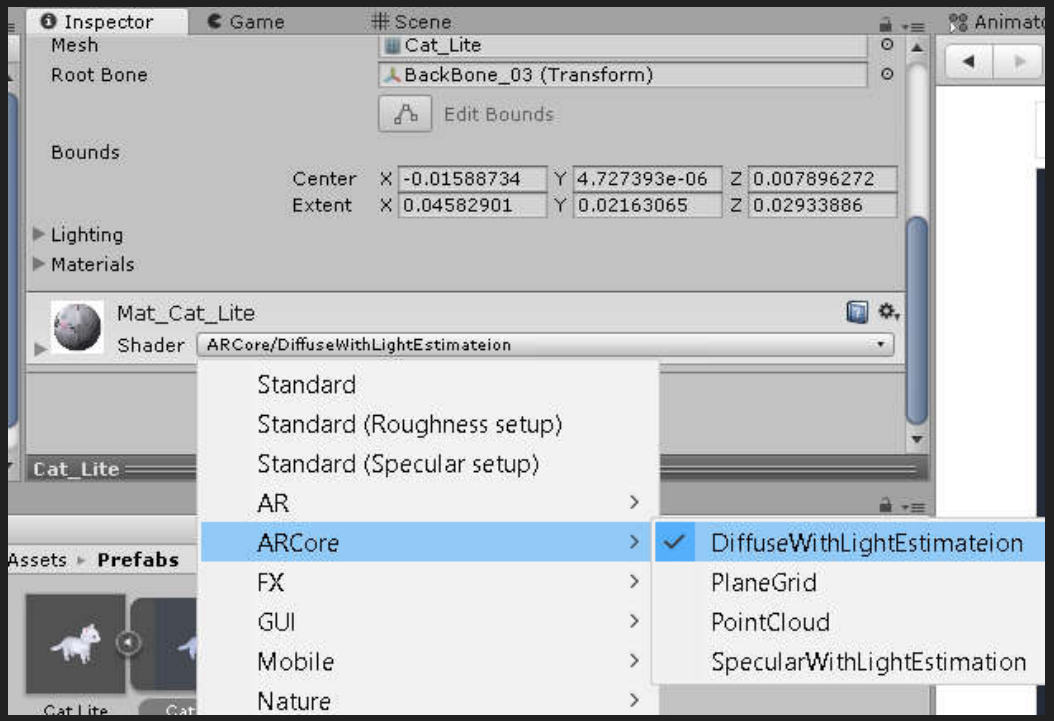
AndroidPlayer(ADB@127.0.0.1:34999) TOUCH PHONE
UnityEngine.DebugLogHandler:Internal_Log(LogType, String, Object)

AndroidPlayer(ADB@127.0.0.1:34999) currentNumberOfCats:1
UnityEngine.DebugLogHandler:Internal_Log(LogType, String, Object)

AndroidPlayer(ADB@127.0.0.1:34999) numberOfCatsAllowed:1
UnityEngine.DebugLogHandler:Internal_Log(LogType, String, Object)

BONUS….. Light Estimate

- Object can dime base on lightning
- Open the prefab object > set Materials
- Shader > ARCORE > DiffuseWithLightEstimation

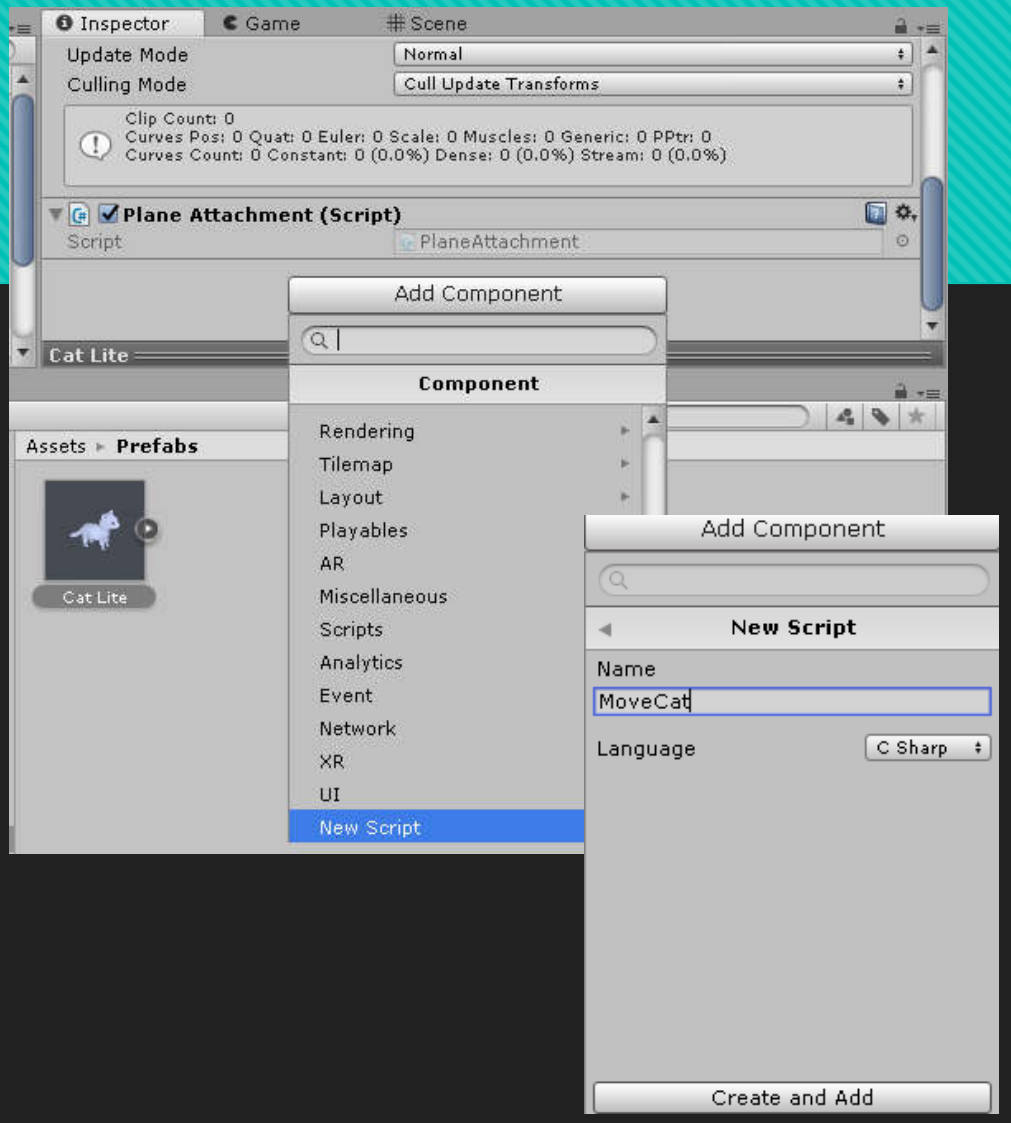


MOVE OBJECT

Tapping the phone and move object to

Add new Behaviour to Object

- Using Script, from prefabs
  - **Add Component > New Script**
  - Give Name, **MoveCat**, select **Csharp** as the language
  - Click **Create and Add**



Using Documentation "SCRIPTING API" Vector3.Lerp

- public static Vector3 Lerp(Vector3 a, Vector3 b, float t);
- Description
  - Linearly interpolates between two vectors.
  - Interpolates between the vectors a and b by the interpolant t. The parameter t is clamped to the range [0, 1]. This is most commonly used to find a point some fraction of the way along a line between two endpoints (e.g. to move an object gradually between those points).
- When t = 0 returns a. When t = 1 returns b. When t = 0.5 returns the point midway between a and b.
- COPY + PASTE

```
using UnityEngine;
using System.Collections;

public class ExampleClass : MonoBehaviour {
    public Transform startMarker;
    public Transform endMarker;
    public float speed = 1.0F;
    private float startTime;
    private float journeyLength;
    void Start() {
        startTime = Time.time;
        journeyLength = Vector3.Distance(startMarker.position, endMarker.position);
    }
    void Update() {
        float distCovered = (Time.time - startTime) * speed;
        float fracJourney = distCovered / journeyLength;
        transform.position = Vector3.Lerp(startMarker.position, endMarker.position, fracJourney);
    }
}
```

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class MoveCat : MonoBehaviour {
6      public Transform startMarker;
7      public Vector3 endMarker;
8      public float speed = 0.1F;
9      private float startTime;
10     private float journeyLength;
11     void Start() {
12
13     }
14     void Update() {
15         float distCovered = (Time.time - startTime) * speed;
16         float fracJourney = distCovered / journeyLength;
17         transform.position = Vector3.Lerp(startMarker.position, endMarker, fracJourney);
18     }
19     public void moveTo(Vector3 endPosition)
20     {
21         startMarker = this.transform;
22         endMarker = endPosition;
23         startTime = Time.time;
24         journeyLength = Vector3.Distance(startMarker.position, endMarker);
25     }
26 }
```

Modify **MoveCat.cs** to code above

Connect HelloARController and MoveCat

In HelloARController.cs we need two case if user HIT

- when there is no cat, place cat
- Otherwise, move cat to desire HIT position

So we need to Promote variable named "catObject" as a GameObject

Add else code to call moveTo method in MoveCat class

```
56
57          public int numberOfCatsAllowed = 1;
58          public int currentNumberOfCats = 0;
59          GameObject catObject;
60
```

```
        catObject = Instantiate (m_catPrefab, hit.Point, Quaternion.identity,
                                 anchor.transform);
        Debug.Log ("catObject" + catObject);
        // Andy should look at the camera but still be flush with the plane.
        catObject.transform.LookAt (m_firstPersonCamera.transform);
        catObject.transform.rotation = Quaternion.Euler (0.0f,
            catObject.transform.rotation.eulerAngles.y, catObject.transform.rotation.z);

        // Use a plane attachment component to maintain Andy's y-offset from the plane
        // (occurs after anchor updates).
        catObject.GetComponent<PlaneAttachment> ().Attach (hit.Plane);
    } else {
        catObject.GetComponent<MoveCat> ().moveTo (hit.Point);
    }
```

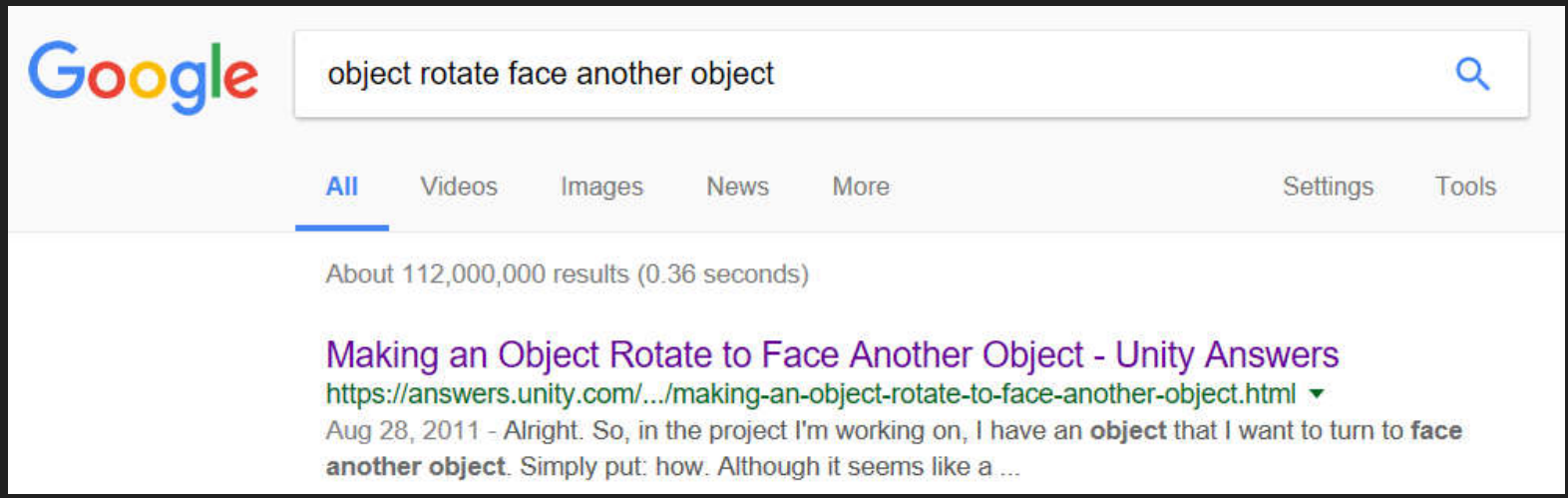
We create a class variable/property, by convention place with other variables/properties

Remember : Do not place it in a method

TURNING OBJECT

Object move and face direction

Google it with appropriate KeyWord



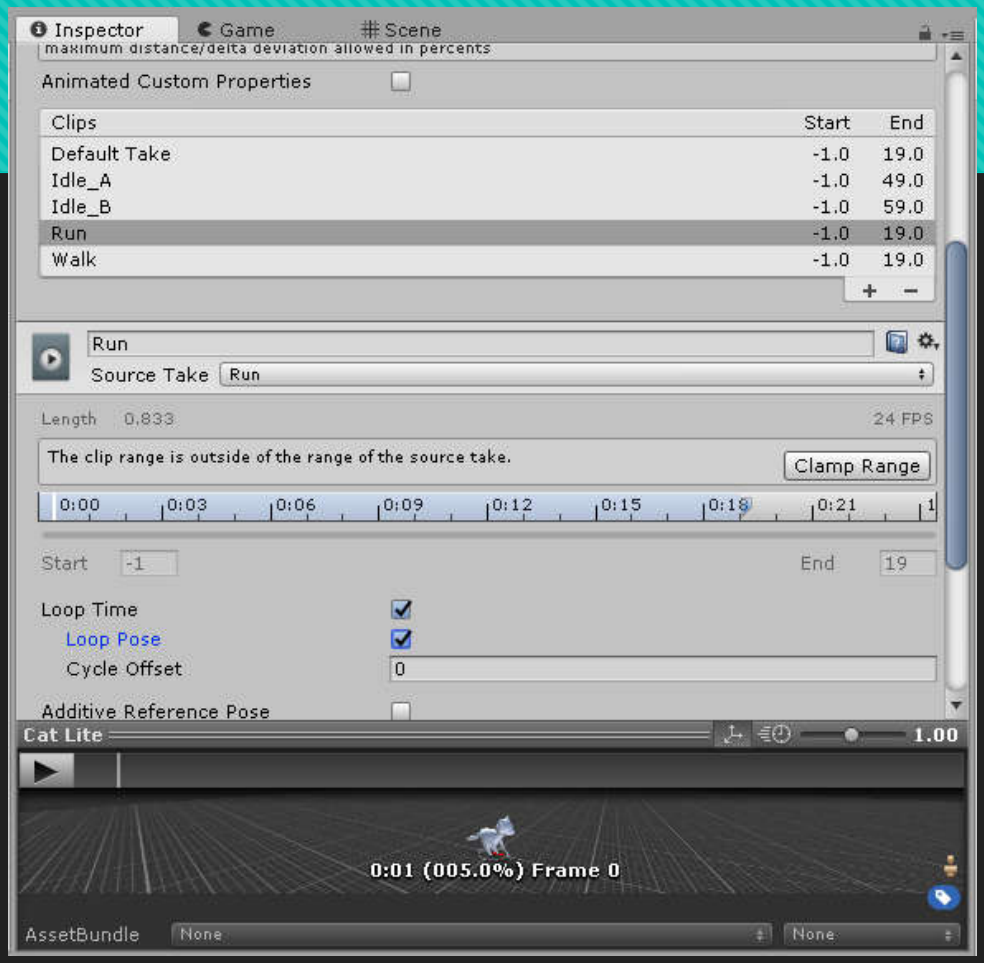

```
if(fracJourney<0.1){
    var lookPos = endMarker - transform.position;
    lookPos.y = 0;
    var rotation = Quaternion.LookRotation(lookPos);
    transform.rotation = Quaternion.Slerp(transform.rotation, rotation, Time.deltaTime * 10f);
}
```

OBJECT / animation STATE

CHANGE game state

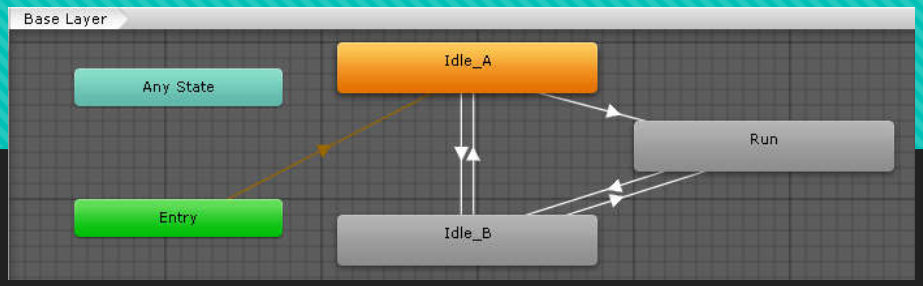
Change RUN Style

- From model
  - Select prefab and find desired animation
  - Chek Loop Time and Loop Pose

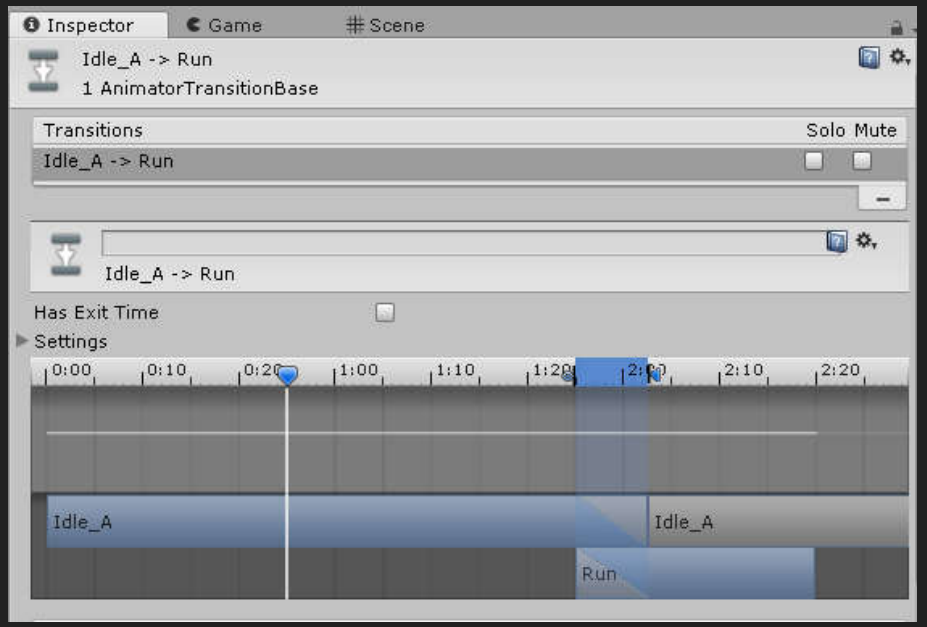


Run State

- Add Run State to Base Layer



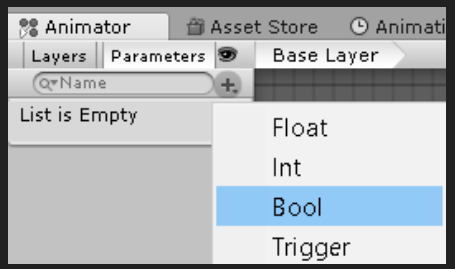
- Make Transition from Idle_A to RUN
- Set Transition without waiting Idle_A, by uncheck “Has Exit Time”



- Make Transition from Idle_B to RUN
- Set Transition without waiting Idle_A, by uncheck “Has Exit Time”

- Make Transition from RUN to Idle_B
- Set Transition without waiting RUN, by uncheck “Has Exit Time”

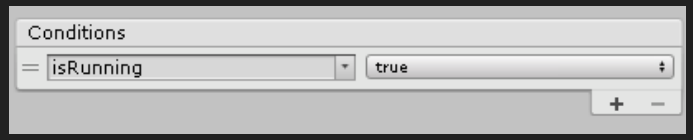
Add Parameter in Base Layer



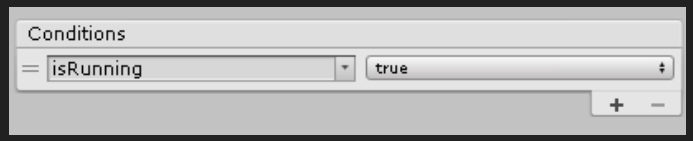
- Click Parameters Tab
- Click " + " > Bool
- Name "isRunning"

Add Conditions

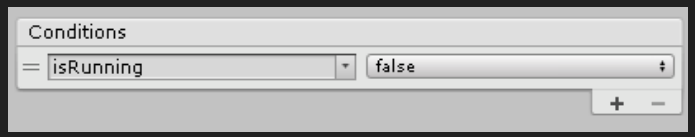
Transition from Idle_A to RUN



Transition from Idle_B to RUN



from RUN to Idle_B



Change State by Code

MoveCat class

- Add variable class named CatAnim as GameObject

moveTo method (when user tap phone)

- Set Boolean isRunning to true

update method (check cat is close to target)

- Set Boolean isRunning to true

```
5  public class MoveCat : MonoBehaviour {
6      public Transform startMarker;
7      public Vector3 endMarker;
8      public float speed = 0.1F;
9      private float startTime;
10     private float journeyLength;
11     private Animator catAnim;
12     void start()
```

```
public void moveTo(Vector3 endPosition)
{
    catAnim.SetBool ("isRunning", true);
    startMarker = this.transform;
    endMarker = endPosition;
    startTime = Time.time;
    journeyLength = Vector3.Distance(startMarker.position, endMarker);
}
```

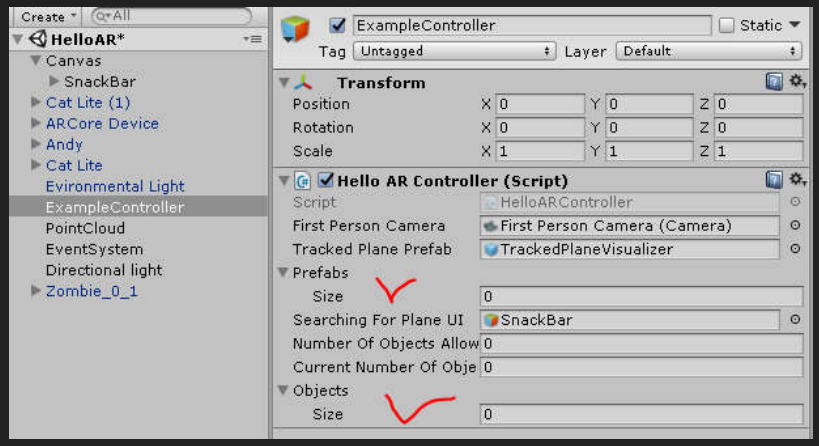
```
if(Vector3.Distance(startMarker.position, endMarker)<0.1)
    catAnim.SetBool ("isRunning", false);
```

MULTI OBJECTs

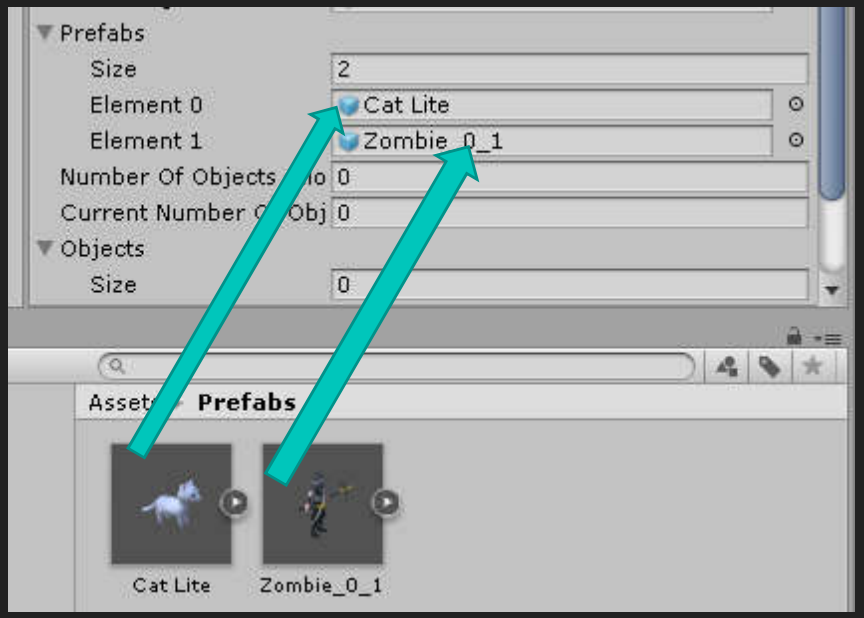
Using ARRAY

Add more prefab
Change code to Array

```
public GameObject[] m_Prefabs;
public int numberOfObjectsAllowed;
public int currentNumberOfObjects = 0;
public GameObject[] objects;
```



Example Controller will be change automatically by unity



Change prefabs as new PREFAB COLLECTION for each element

Initiate & Access Array

Constructor

```
public void Start()
{
    numberOfObjectsAllowed = m_Prefabs.Length;
    objects = new GameObject[numberOfObjectsAllowed];
}
```

```
if (currentNumberOfObjects < numberOfObjectsAllowed) {
    // Create an anchor to allow ARCore to track the hitpoint as understanding of the physical
    // world evolves.
    var anchor = Session.CreateAnchor (hit.Point, Quaternion.identity);
    // Intanstiate an Andy Android object as a child of the anchor; it's transform will now benefit
    // from the anchor's tracking.
    objects[currentNumberOfObjects] = Instantiate (m_Prefabs[currentNumberOfObjects],
        hit.Point, Quaternion.identity, anchor.transform);
    Debug.Log ("Object" + objects[currentNumberOfObjects]);

    objects[currentNumberOfObjects].transform.LookAt (m_firstPersonCamera.transform);
    objects[currentNumberOfObjects].transform.rotation = Quaternion.Euler (0.0f,
        objects[currentNumberOfObjects].transform.rotation.eulerAngles.y,
        objects[currentNumberOfObjects].transform.rotation.z);
    objects[currentNumberOfObjects].GetComponent<PlaneAttachment> ().Attach (hit.Plane);
    currentNumberOfObjects++;
} else {
    //objects[0].GetComponent<MoveCat> ().startMove (hit.Point);
    objects[1].GetComponent<MoveCat> ().startMove (hit.Point);
}
```