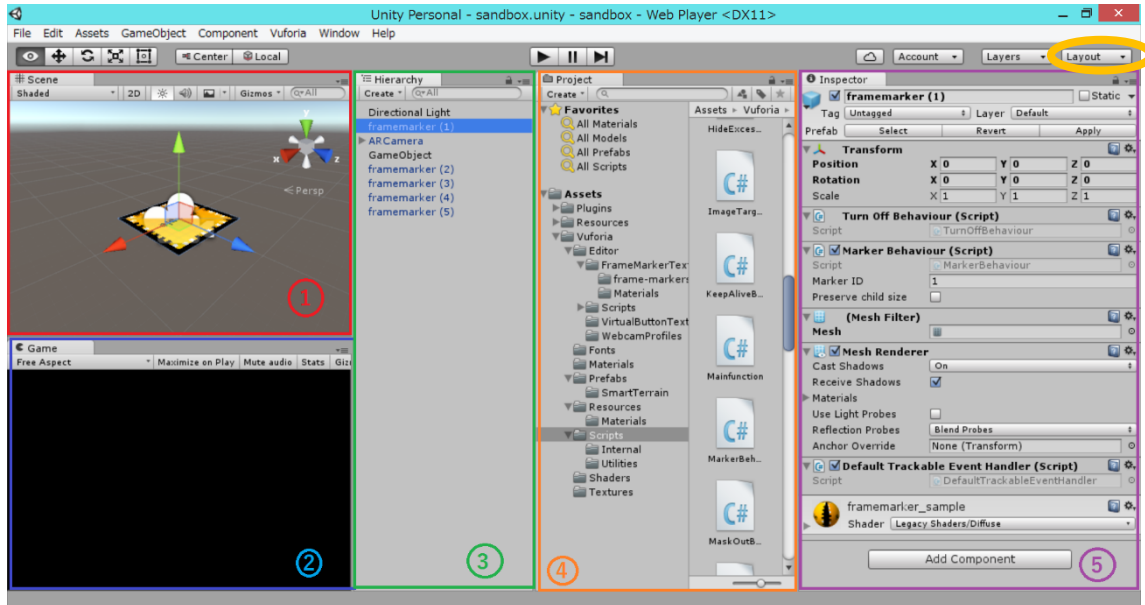


Unity & Vuforia

○基礎画面(黄色→2 by 3)

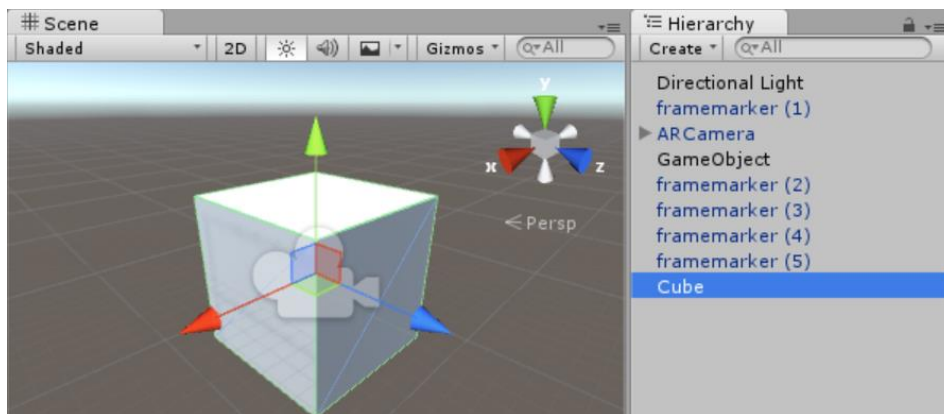


- 1…シーン。色々な角度からオブジェクトを見る
- 2…ゲーム。プログラムを動かした時に実際に見える画面
- 3…ヒエラルキー。ここにオブジェクトを置くと、シーンに反映される
- 4…プロジェクト。ファイル郡
- 5…インスペクター。選択しているオブジェクトの詳細

○基本操作

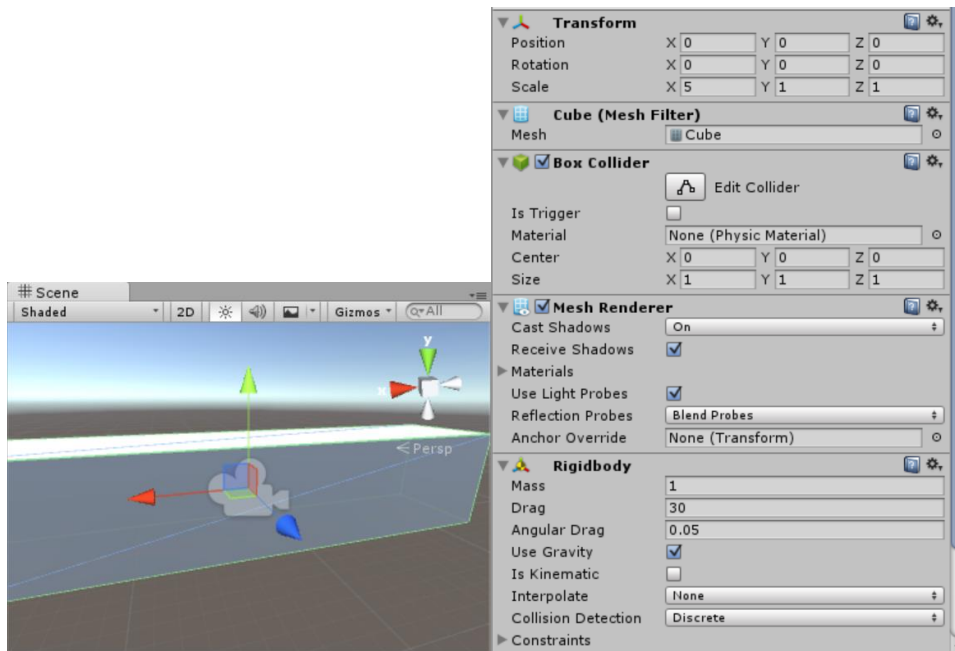
- ・オブジェクトの配置

1. ヒエラルキーで右クリック
2. 3D object→cube



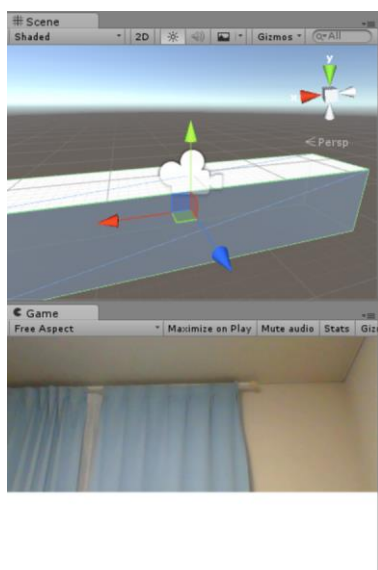
・インスペクターの操作

1. ヒエラルキーで cube を選択
2. Transform コンポーネントの Scale の X を 5 にする
3. 一番下の Add Component をクリック
4. Physics→Rigidbody を選択
5. Rigidbody コンポーネントの Drag (空気抵抗) を 30 にする



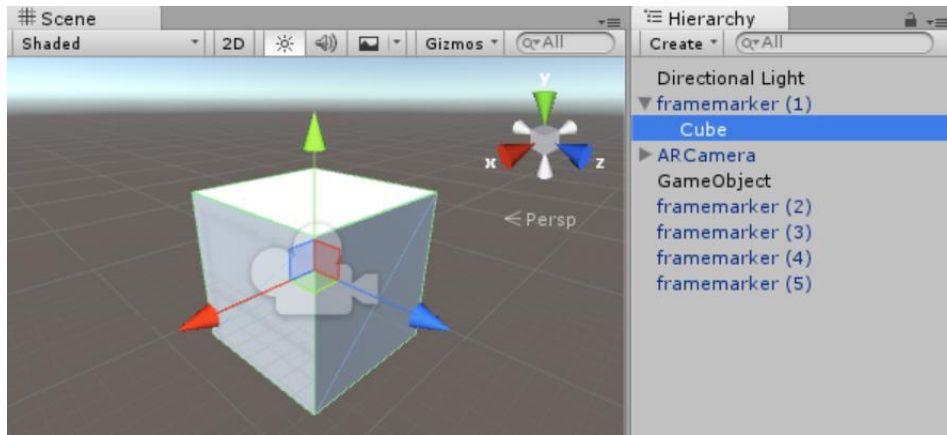
・実行

1. 上の三角か Ctrl+P
2. Unity Web Player～が出たら、Always～をクリック



・子要素の作成

1. 作成した cube を framemarker (1) に D&D

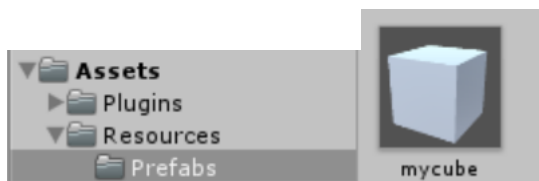


※ framemarker (X) の子要素は、そのマーカー上に表示される立体となる

・プレハブの作成

※ プレハブ…立体のコピー元と覚えておく

1. Assets フォルダを右クリック
2. Create→Folder
3. Resources フォルダを作成
4. Resources フォルダを右クリック
5. Create→Folder
6. Prefabs フォルダを作成
7. Prefabs フォルダを右クリック
8. Create→Prefab
9. mycube を作成
10. ヒエラルキーの cube をプロジェクトの mycube に D&D



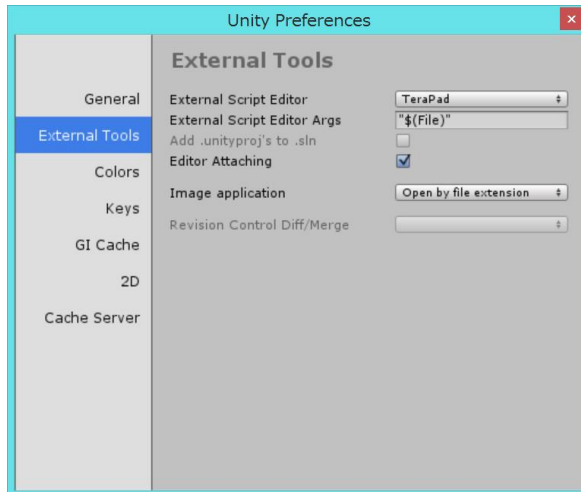
・オブジェクトの削除

1. ヒエラルキーの cube を右クリック
2. delete

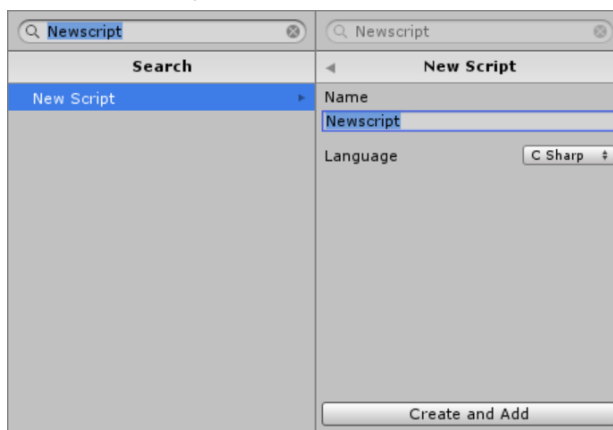
○スクリプト (C#)

・準備

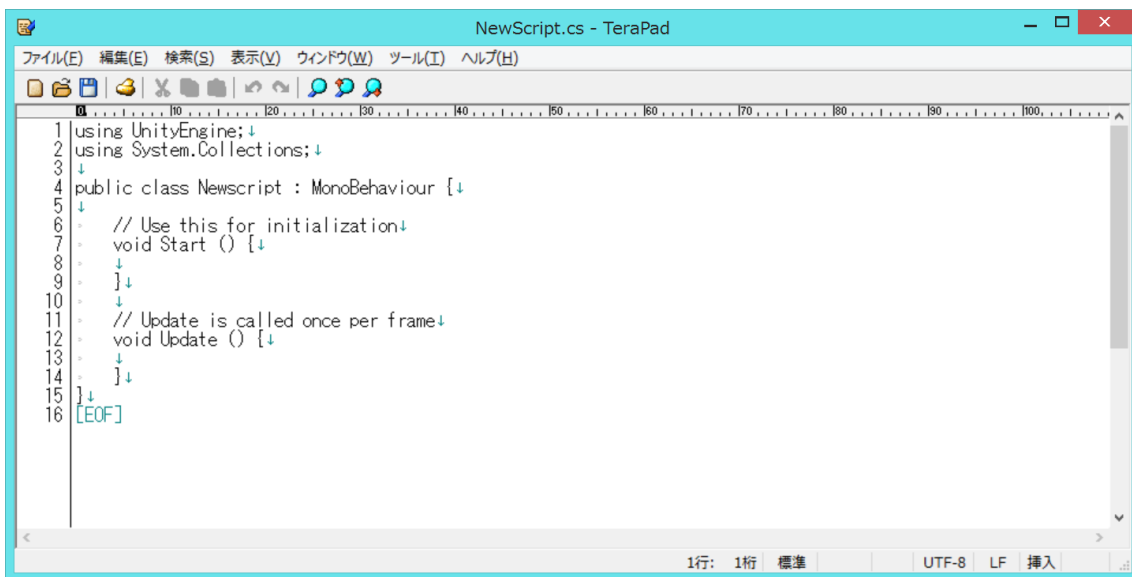
1. メニューバーの Edit→Preferences
2. External Script Editor で使いたいエディタを選択(デフォは MonoDevelop)



3. ×で Preferences を閉じる
4. ヒエラルキーを右クリック
5. Create Empty (すでに GameObject があれば不要)
6. GameObject を選択
7. インスペクターで Add Component
8. 検索ボックスで「NewScript」と入力
9. New Script→Create and Add

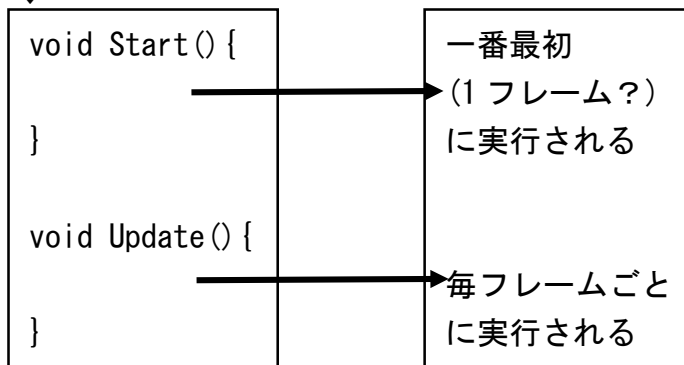


10. Assets/にある NewScript をダブルクリック



```
1 using UnityEngine;
2 using System.Collections;
3
4 public class Newscript : MonoBehaviour {
5
6     // Use this for initialization
7     void Start () {
8
9     }
10
11    // Update is called once per frame
12    void Update () {
13
14    }
15 }
16 [EOF]
```

↓



・ 試し書き

```
void Start () {
    Debug.Log("Start");
}

void Update () {
    Debug.Log("Update");
}
```

1. メニューの Window→Console または Ctrl+C
2. 実行
3. コンソールに右が出力(他の出力はあるかもしれないが)
4. GameObject のインスペクターで NewScrip コンポーネントを Remove & プロジェクトの NewScript を Delete

```
! Start
. . .
! Update
! Update
. . .
```

・ 出力

```
Debug.Log(<文字列や数値>);  
もしくは、  
print(<文字列や数値>);
```

```
Debug.Log("Hello, World!");  
print(1+1);
```

・ オブジェクトの検索

```
GameObject <変数名> = GameObject.Find(<オブジェクト名(シーン)>);
```

```
GameObject marker = GameObject.Find("framemarker (1)");
```

※オブジェクトの変数型は GameObject

・ プレハブのロード

```
GameObject <変数名> =  
    (GameObject)Resources.Load ("Prefabs/" + <プレハブ名>);
```

```
GameObject prefab =  
    (GameObject)Resources.Load ("Prefabs/mycube");
```

・ オブジェクトの生成

```
GameObject <オブジェクト名> =  
    Instantiate(<プレハブ変数名>,  
        transform.position, transform.rotation) as GameObject;
```

```
GameObject newcube =  
    Instantiate(prefab, transform.position, transform.rotation)  
    as GameObject;
```

・オブジェクトの子要素化

```
<子オブジェクト>.transform.parent = <親オブジェクト>.transform;
```

```
newcube.transform.parent = marker.transform;
```

・オブジェクトの名前変更

```
<オブジェクト>.name = <値>;
```

```
newcube.name = "newcube";
```

・コンポーネントの取得

```
<コンポーネントクラス> <変数名> =  
    <オブジェクト>.GetComponent.<<コンポーネントクラス>>;
```

```
Rigidbody rigidbody = newcube.GetComponent.<Rigidbody>;
```

※覚えておいたほうがいいコンポーネントクラス

- ・ Transform…座標関係
- ・ Rigidbody…質量や空気抵抗など物理演算関係

・コンポーネントパラメータの変更

```
<コンポーネント変数>.<パラメータ名> = <値>
```

```
rigidbody.drag = 1
```

○フレームマーカー

- ・フレームマーカーとは？

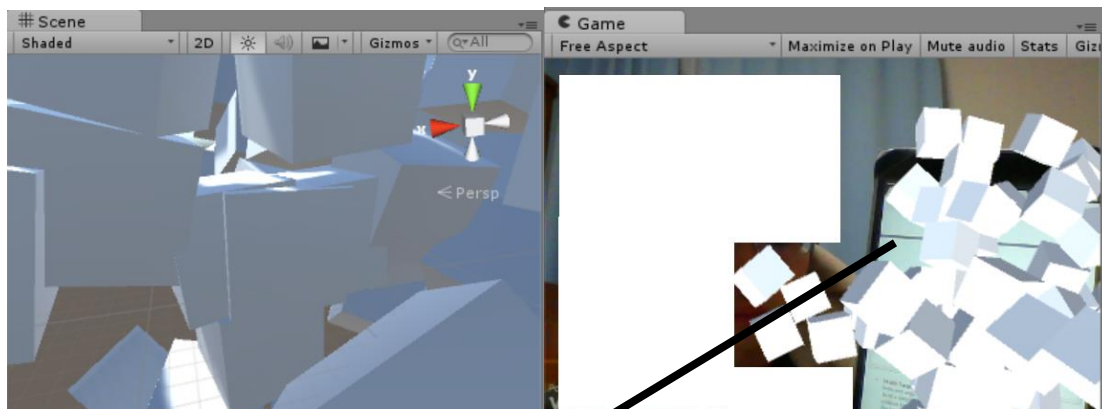
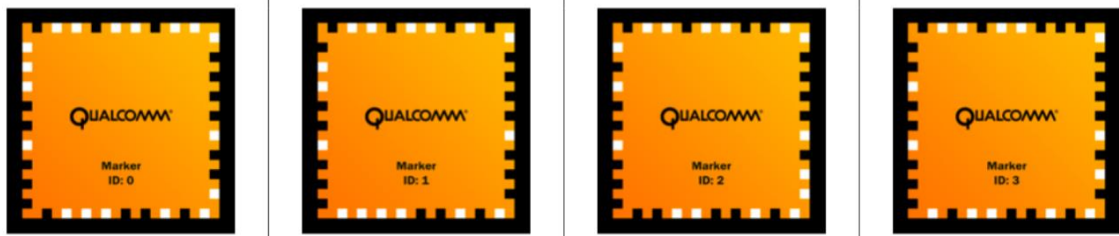
Vuforia に登録されている 512 種類のマーカーのこと。縁の点線が重要らしく、中はわりとどうでもいい？認識精度はかなり高いが、イメージマーカー(写真をマーカーにする)とは違い、一部が隠れると認識されにくくなる。

- ・インスペクター



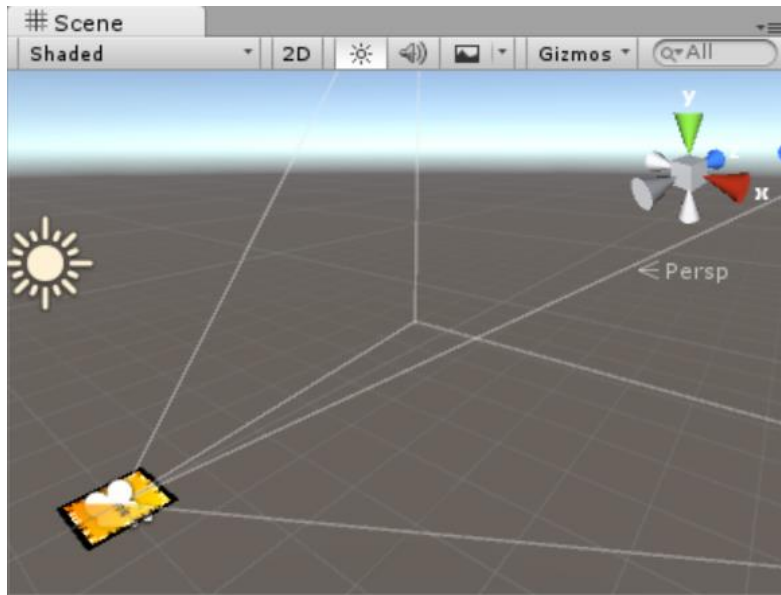
Marker ID がマーカーの種類を表す。0～511 まで設定可能。

- ・マーカーと実行例

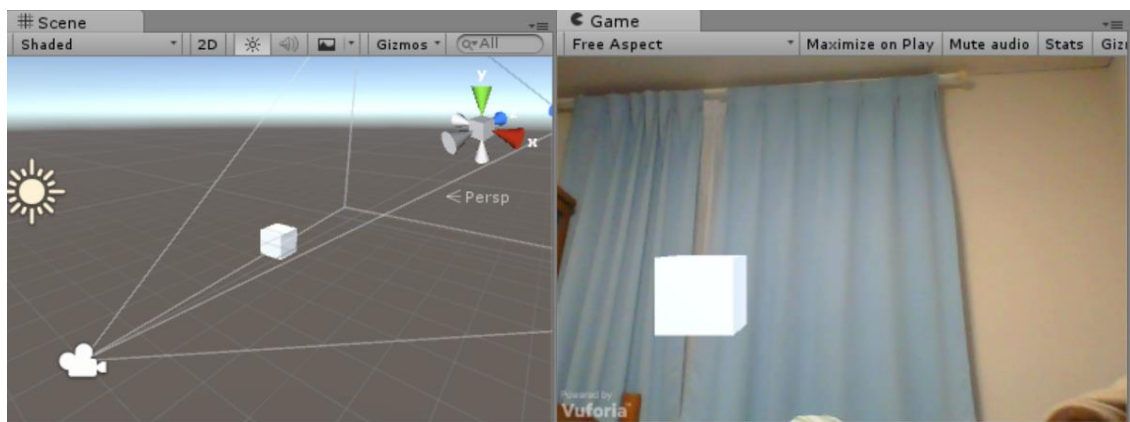


タブレット(マーカーを映している)

○マーカに頼らない AR (?)



AR マーカーを選択すると放射状に広がる線が見える (見つらかったら、右クリックしながら or ホイールを押しながら or ホイールを回して視点変更可能)。
これはカメラが捉える範囲である。この範囲内に立体を置くと、



カメラ上に常に立体が配置される。

○保存について

Unity はプロジェクトという単位で保存される。1 つのプロジェクトはファイル（スクリプト、プレハブ、テクスチャ、シーン…）の集まりである。

シーンとはプロジェクトの中に複数存在できるヒエラルキーやインスペクターなどの状態である。

つまり、1 つのプロジェクトでも異なるプログラム（シーン）を保存することができる。ただし、ファイルは共通なので、他シーンで使うスクリプトを消したり、（他のシーンにとって）余計な unityproject ファイルをインポートすると困ることになるかもしれない。