

NCC Unity 講習会

~初めてのUnity~

By @fms_sabu

お品書き(前半)

1. 画面のレイアウト、操作説明
2. 球を落としてみよう
3. フィールドを作ってみよう
4. キャラを動かしてみよう

※注意

- このスライドは、あらかじめUnityのインストールを終わった方で、ほとんど使ったことのない方を対象としています。
- 本環境はUnity5.6.0f3で確認済みです。
- Windowユーザー向けの説明となるため、一部Macとは操作が違うことがあります。
- マウス必須

新しいProjectを作ろう

1.

NEW

OPEN

MY ACCOUNT

2.

New Unity Project

☒ 3D ☐ 2D

Add Asset Package

3.

C:\Users\ ...

☒ ON Enable Unity Analytics ?

Organization*

※Projectフォルダの場所を変更した場合、Unity側で認識できないため、改めてシーンから起動する必要がある

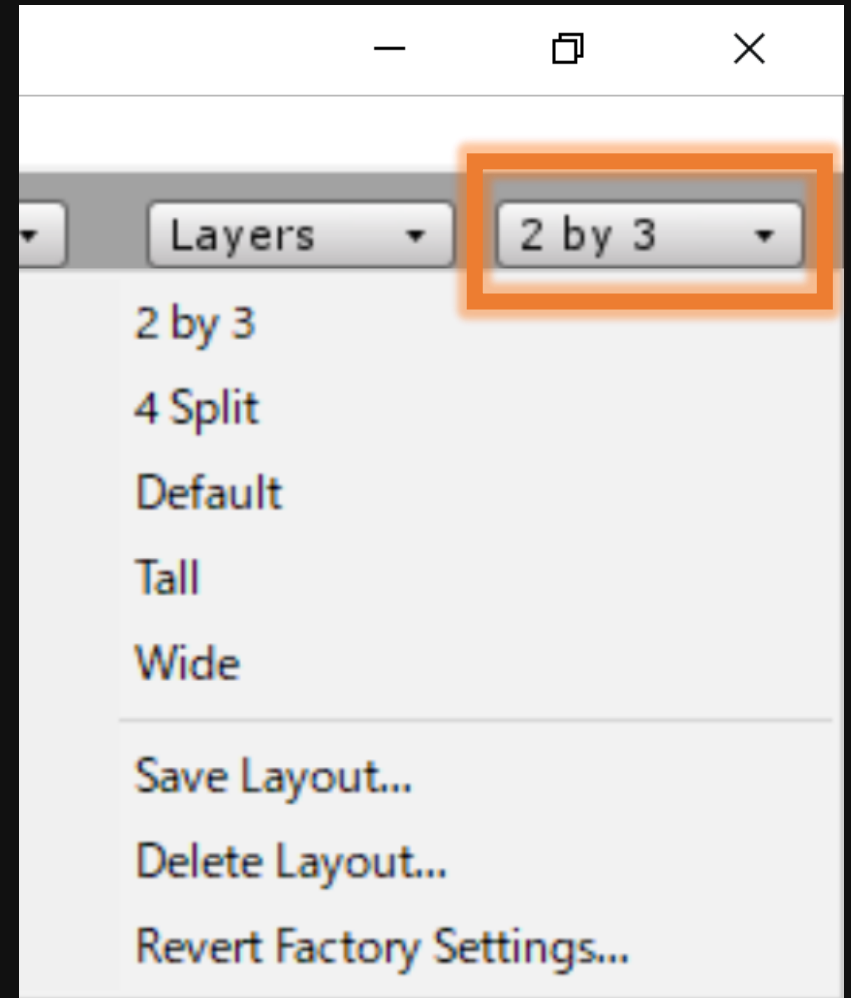
4.

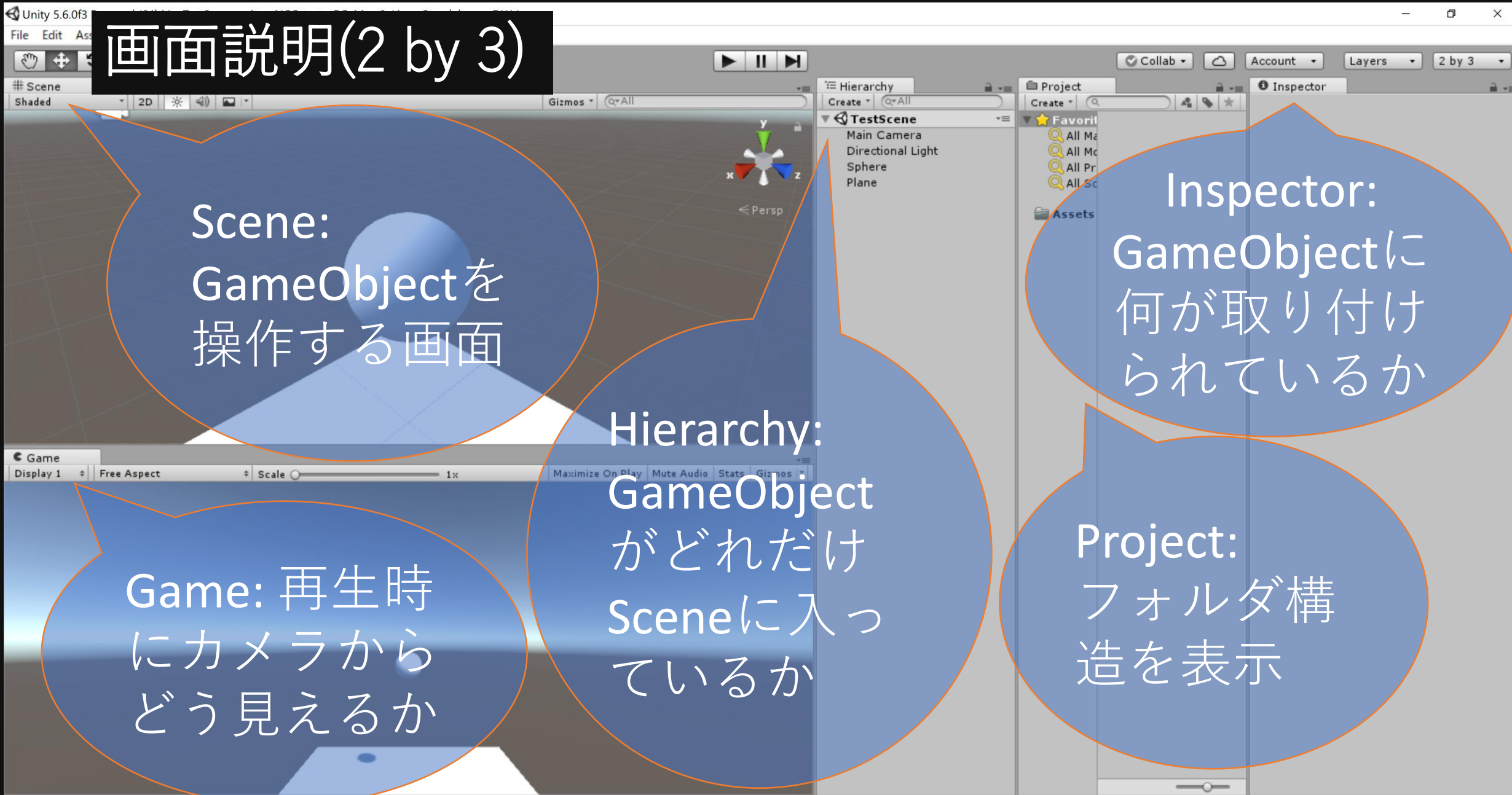
Cancel

Create project

初期設定

- 初期設定では画面レイアウトは”**Default**”になっている。
- おススメ設定 ”**2by3**”
(画面を俯瞰しながら再生ができる)





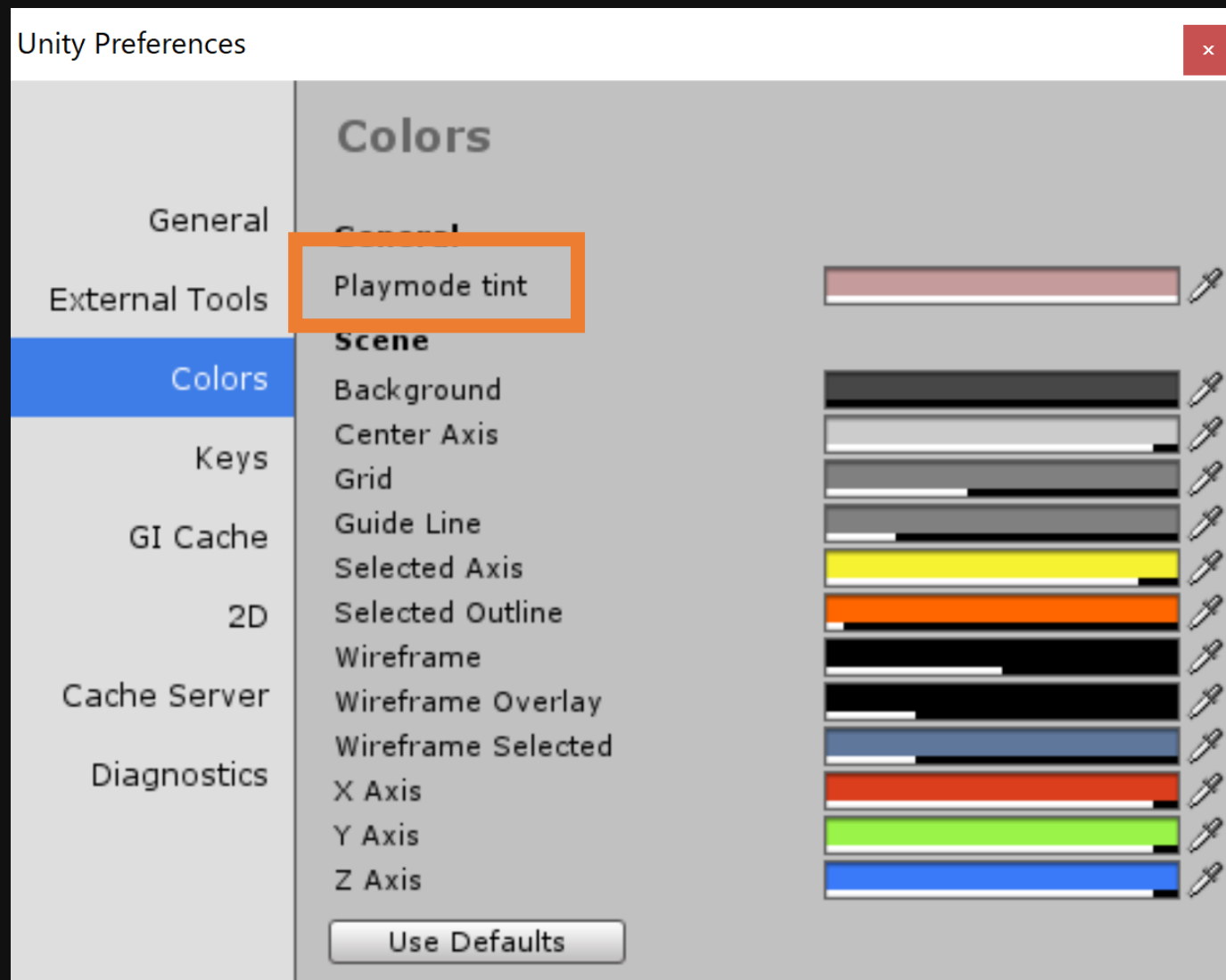
初期設定その 2

ヘッダーの

File ⇒ Preference ⇒

Colors ⇒ Playmode tint

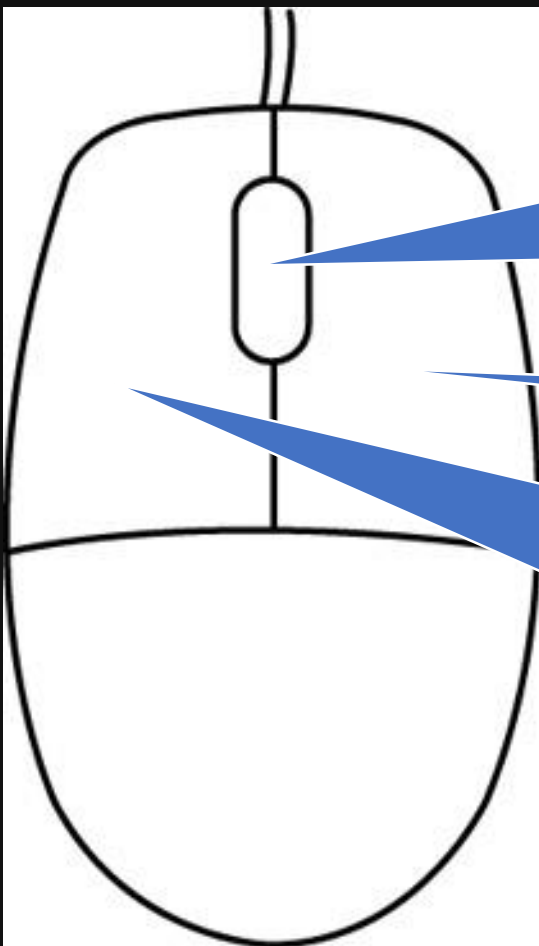
から再生時の色を変更
(再生しているかどうかを
はっきりさせる目的)



操作説明(ゲームシーン)

- GameObjectを動かすにはどうしたらいいか？
 1. **Scene** からマウスカーソルでドラッグする
 2. GameObjectの **Inspector** の Transform を
パラメータで設定する
 3. スクリプトを書いて自動的に動かすようにする

操作説明(マウスカーソル)



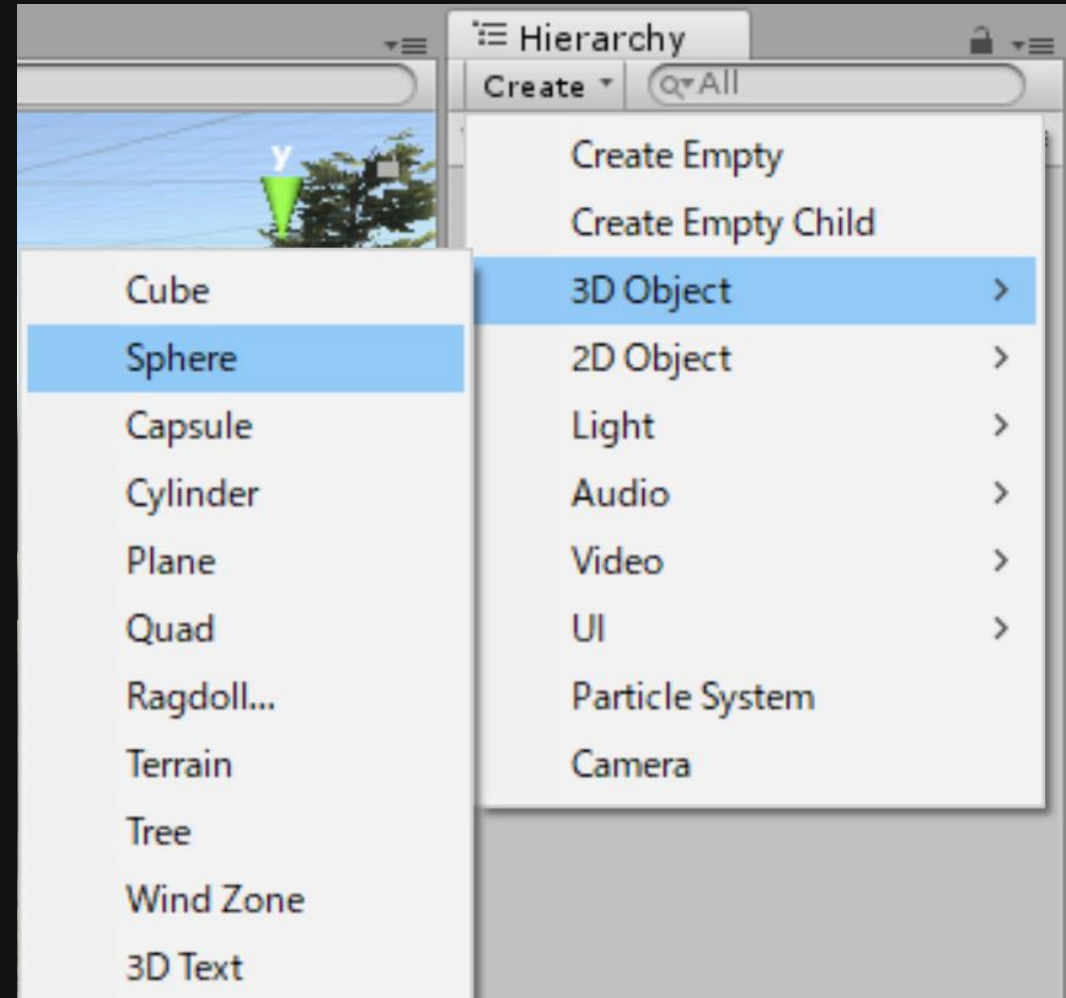
ホイール回転:拡大縮小
クリック:上下左右に移動
Alt(Option) or 右クリック
+ホイール回転:
マウス位置を中心に拡大縮小

ドラッグ:
その場を見渡す
Alt+ドラッグ:
少しずつ拡大縮小

クリック:GameObjectの選択
(非選択)Alt+ドラッグ:
マウス中心に回転
(選択中)Alt+ドラッグ:
GameObjectを中心に回転

オブジェクトの追加の仕方

- 標準で入っているもの:
Hierarchy の Create タブから作りたいものを選択して追加する
- 標準で入っていないもの:
Project の Create タブから追加、それを **Hierarchy** にドラッグドロップして追加する

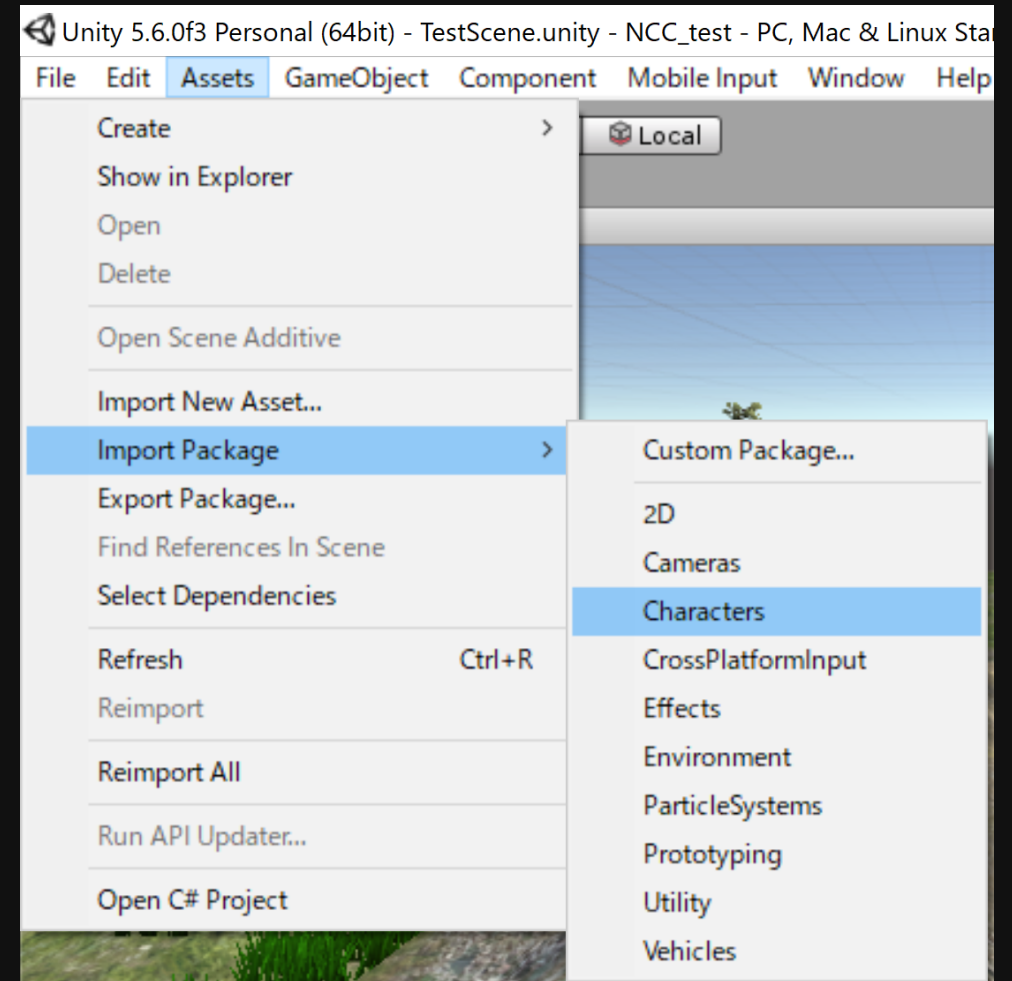


パッケージの追加の仕方

ヘッダーの

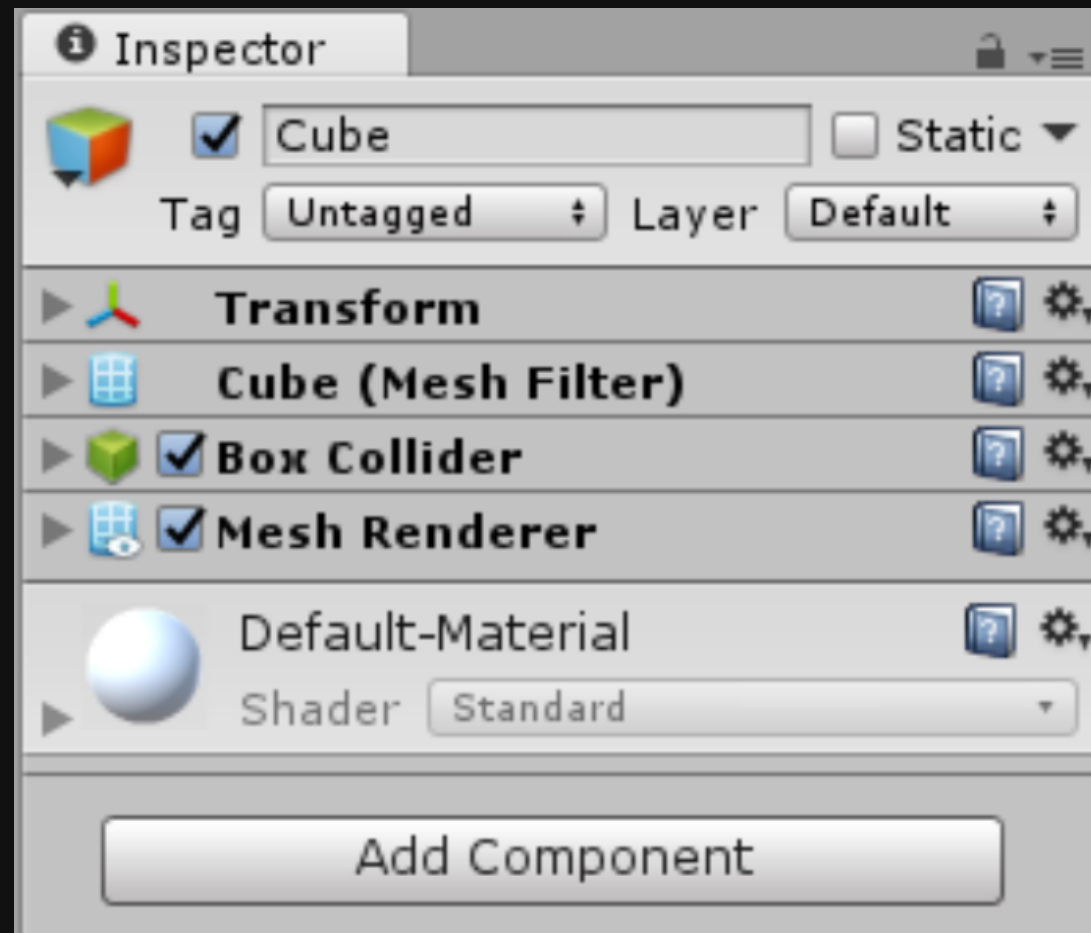
Assets ⇒ Import Package

の中にあるものを選択する
外部のパッケージを
追加する場合は
一番上のCustom Package
から**hoge.unitypackage**
を選択する



コンポーネントの追加の仕方

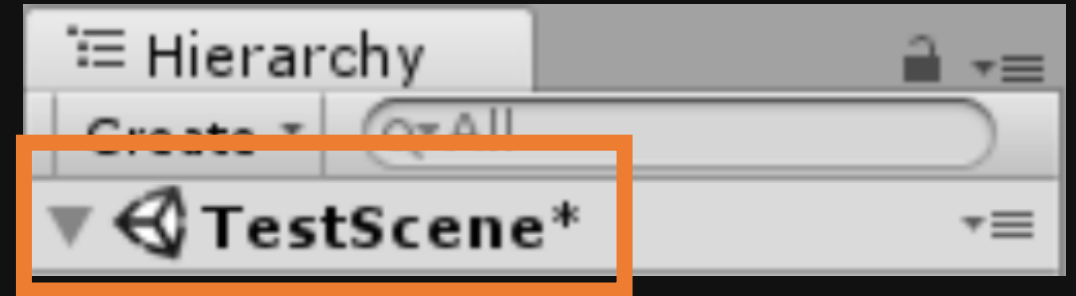
- GameObjectのInspectorには、それがどんな振る舞いをする要素を持つか、という単位が備わっていて、このことをComponentという
- Add Componentから要素を増やすことができる



シーンを保存しよう

- **Unity**をいじっていると、たまにアプリが落ちる
(忘れたところにやってくる)

- 初回のみ Ctrl + S で名前を設定して保存

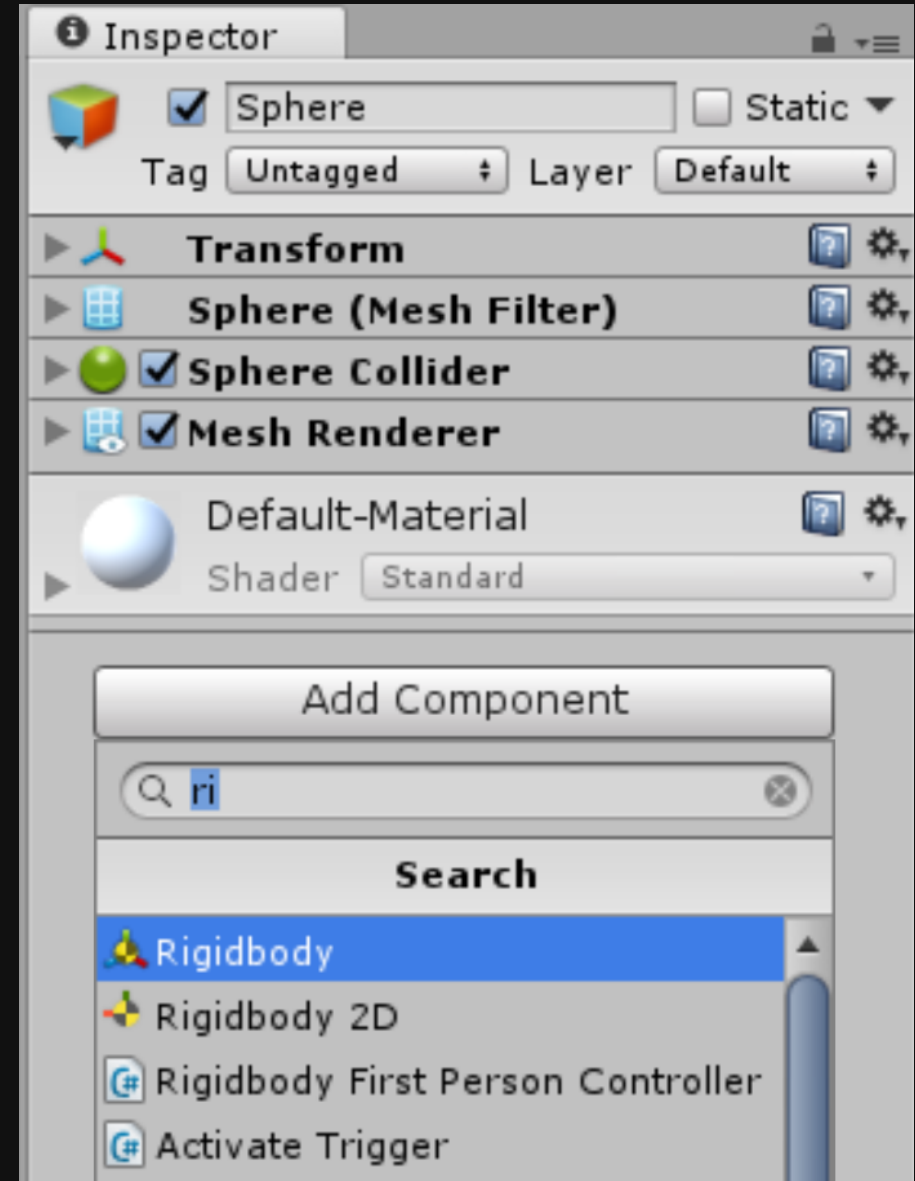
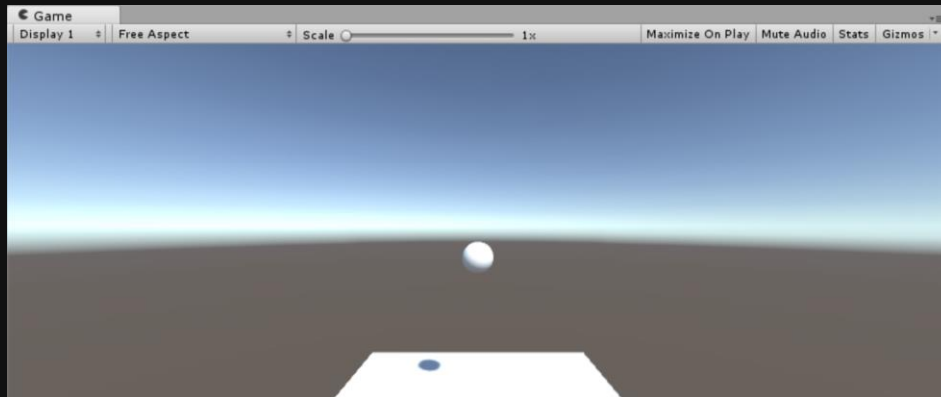


- **Hierarchy** のシーン名横に*がつくと前回から何か変更点のある印なのでそれを参考にする

球を落としてみよう

1. Sphereを **Hierarchy** に追加
2. PlaneをSphereの下に追加
3. Sphereの **Inspector** から Add Component ⇒ 検索バーで **Rigidbody** と入力 ⇒ 追加
4. 画面中央上の再生ボタンをクリック
球が落ちるデモの完成！

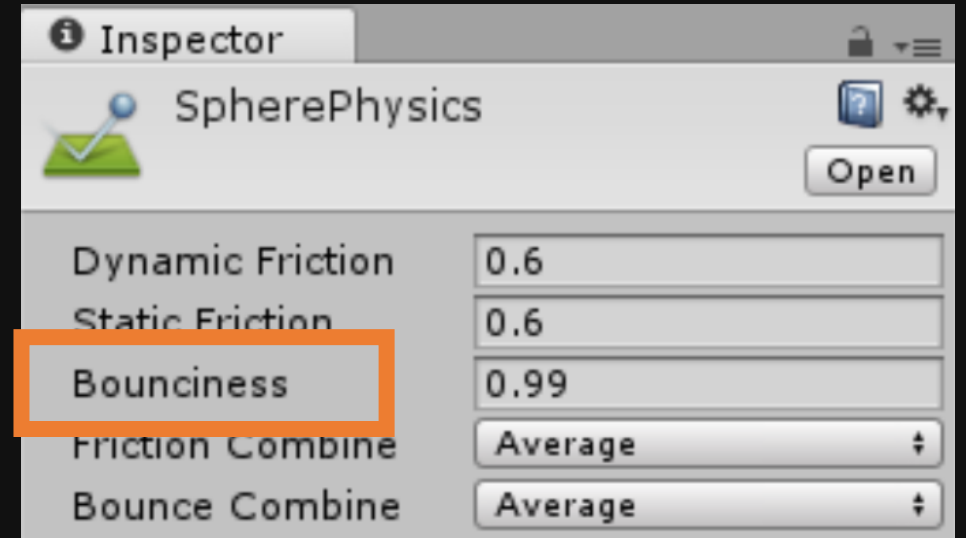
完成図 ⇒



ちょっとアレンジしてみよう

- 球をバウンドさせてみる

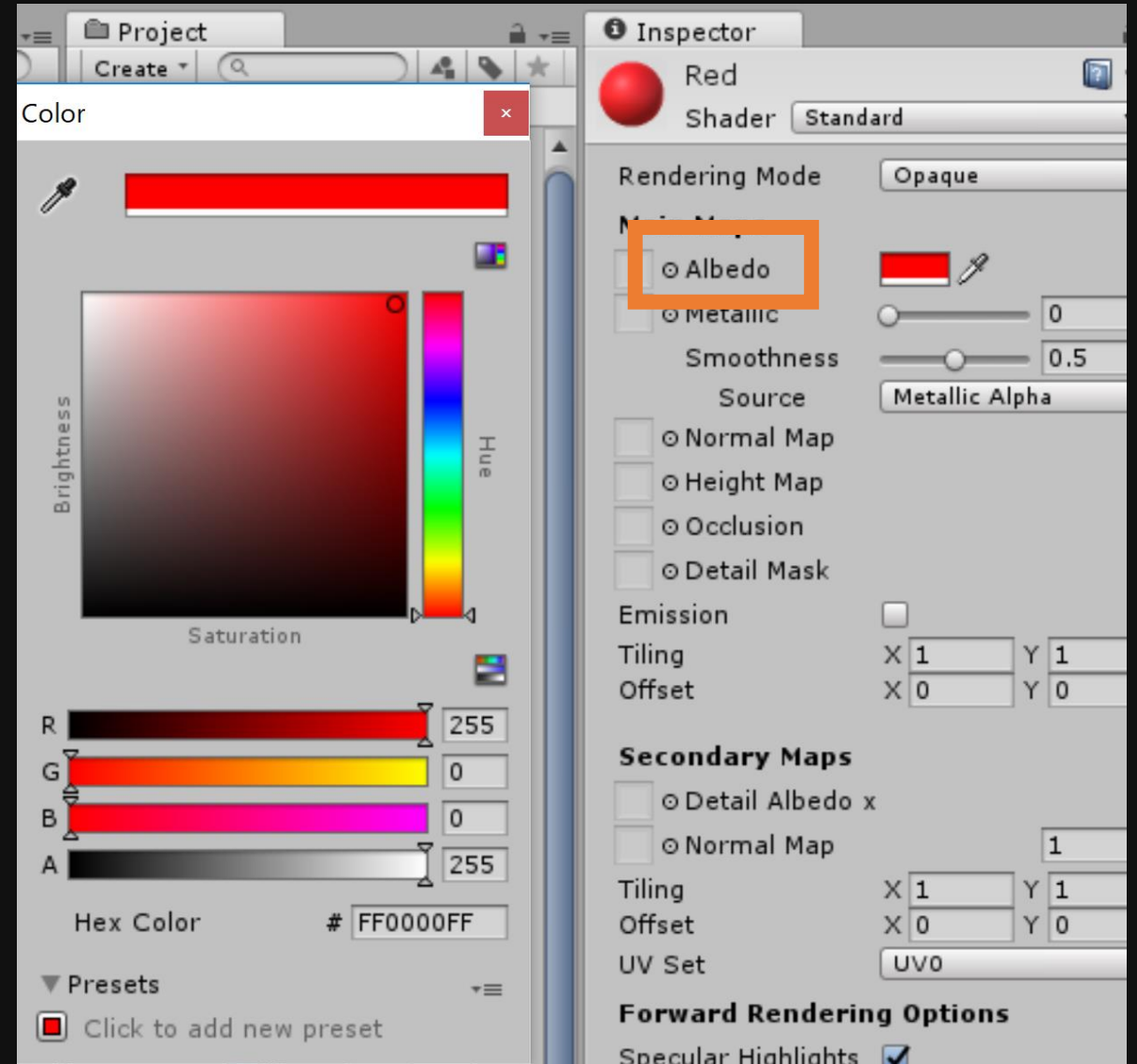
1. **Project** ⇒ **Create** ⇒ **Physic Material** を追加
2. 作られた Physic Material から Bounciness の値を 0.99 ほどに設定
3. Physic Material を Sphere にドラッグドロップ



アレンジその2

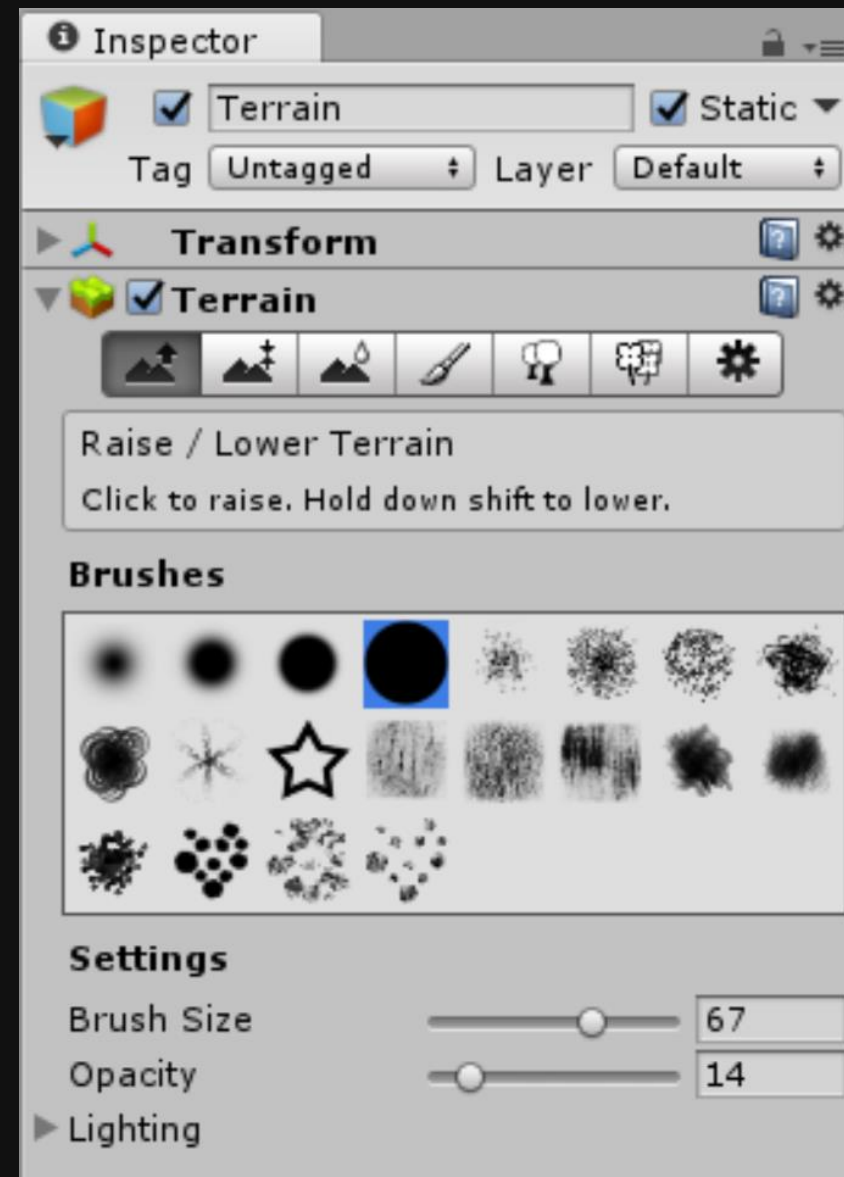
- 色を変化させてみる

1. **Project** ⇒ **Create** ⇒ **Material**
から Material を作る
2. 作成したものの **Inspector** から
Albedo の色を変更する
3. 作成した Material を
Sphere にドラッグドロップ



世界を作ろう

- Hierarchy ⇒ Create ⇒ 3D Object ⇒ Terrainを追加
- Brushes : 筆選択
- BrushSize : 筆の大きさ
- Opacity : 筆の影響率
- Scene から直接ペイントすることで地形を形成できる



Terrainで世界を作ろう



地形の隆起



地形の標高を合わせる



地形をなだらかにする



テクスチャを追加



木を追加



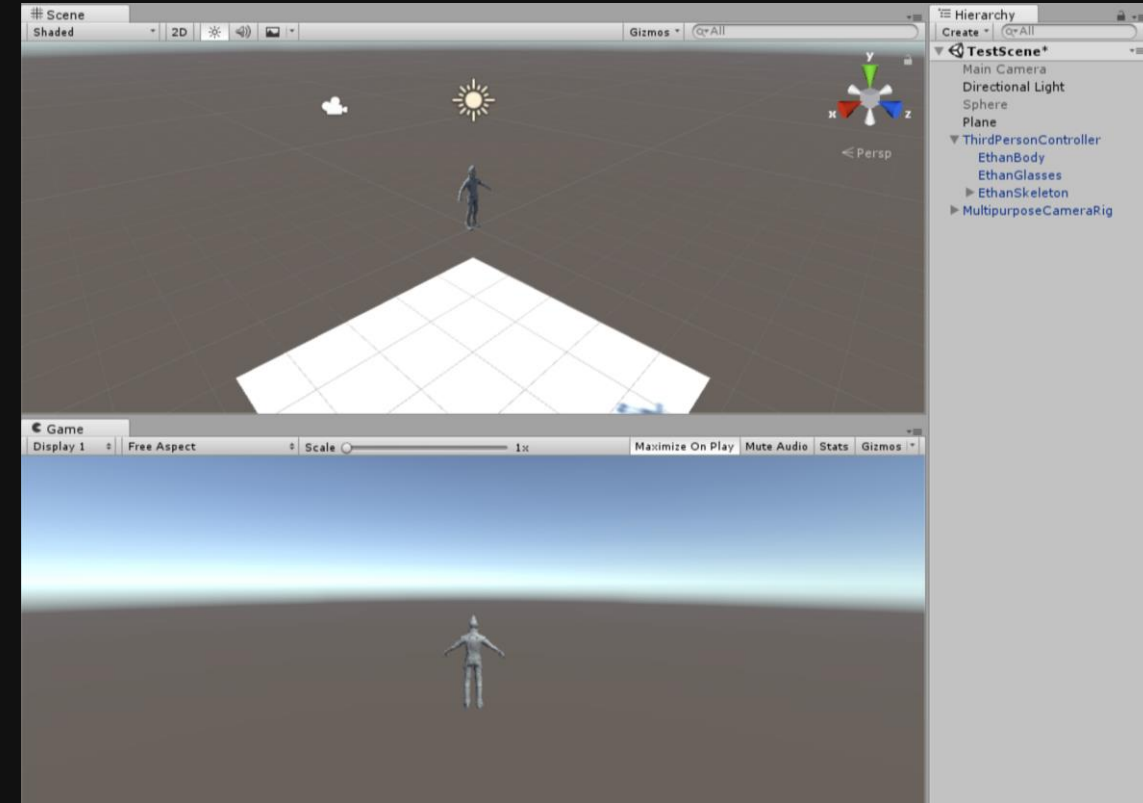
草を追加



詳細設定

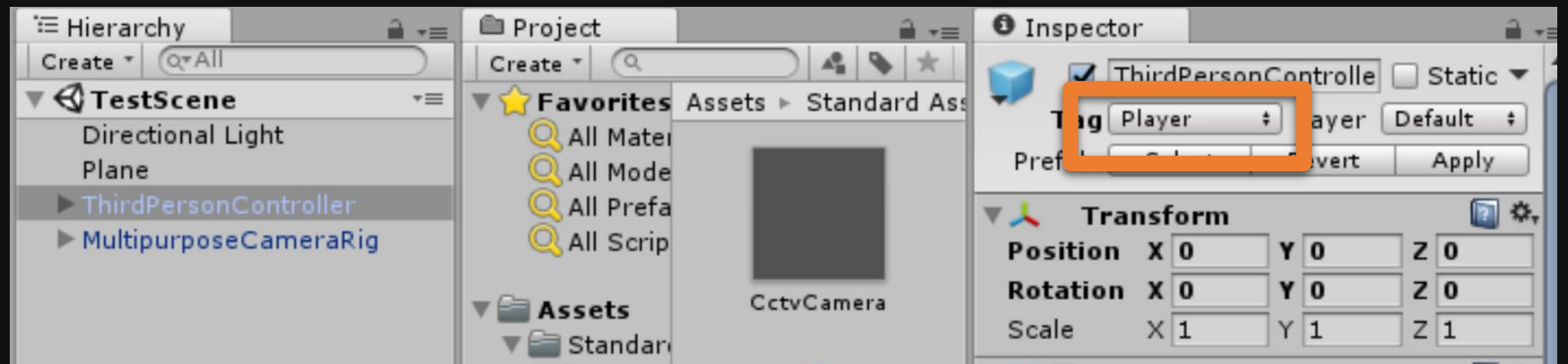
キャラクターを動かそう

1. Characters パッケージを
インポート
2. Project フォルダの
Assets ⇒ Standard Assets ⇒
Characters ⇒
ThirdPersonCharacter ⇒
Prefabs ⇒ ThirdPersonController
を Hierarchy に追加
3. Ethan を方向キーで移動、Spaceキーでジャンプできる



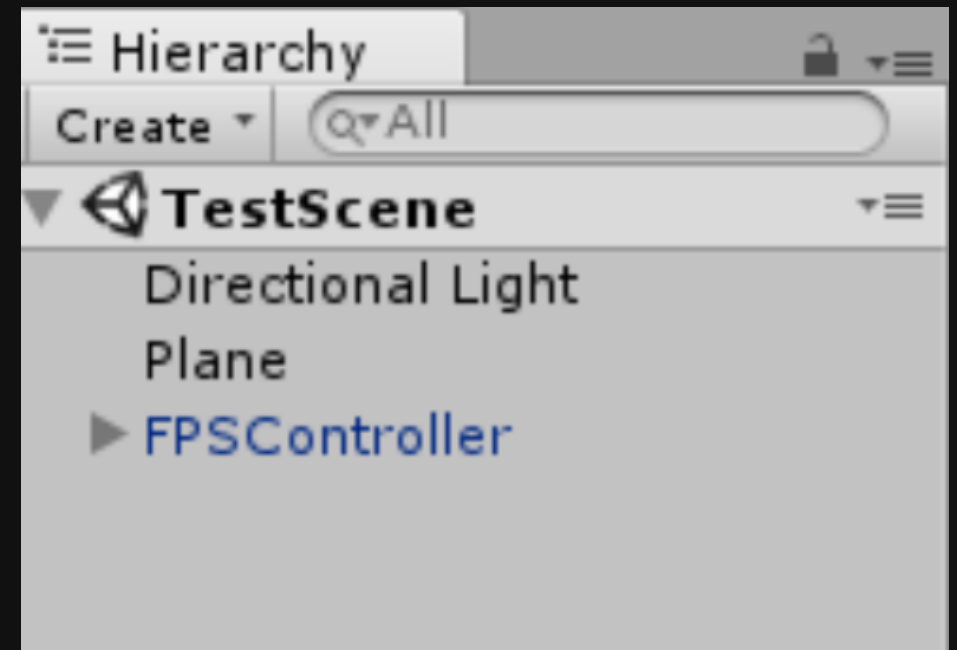
カメラをキャラクターに追従させる

1. **Hierarchy** の Main Camera を削除する
2. Cameras パッケージをインポートする
3. **Assets** ⇒ **Standard Assets** ⇒ **Cameras** ⇒ **Prefabs** にある **MultipurposeCameraRig** を **Hierarchy** に追加
4. **ThirdPersonController** の **Inspector** の Tag を Untagged から **Player** に変更



TPS(三人称視点)からFPS(一人称視点)に変更

Assets ⇒ StanderdAssets ⇒
Characters⇒FirstPersonCharacter
⇒ Prefabs ⇒ FPSController
を **Hierarchy** に追加



お品書き(後半)

- 実際にC#でプログラムを書いてみる

1. Hello,world!

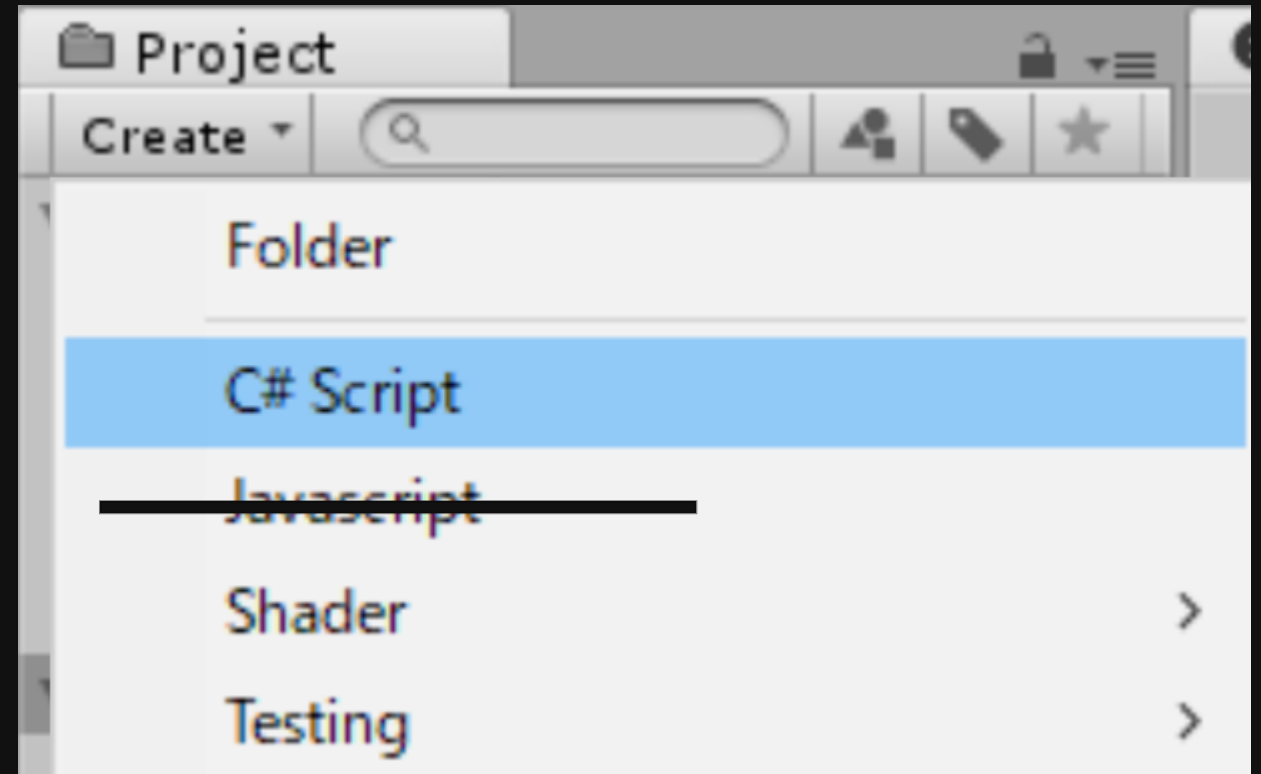
2. GameObjectの生成、
キー入力を受け付ける

3. GameObjectの破棄

初めてのUnityプログラミング

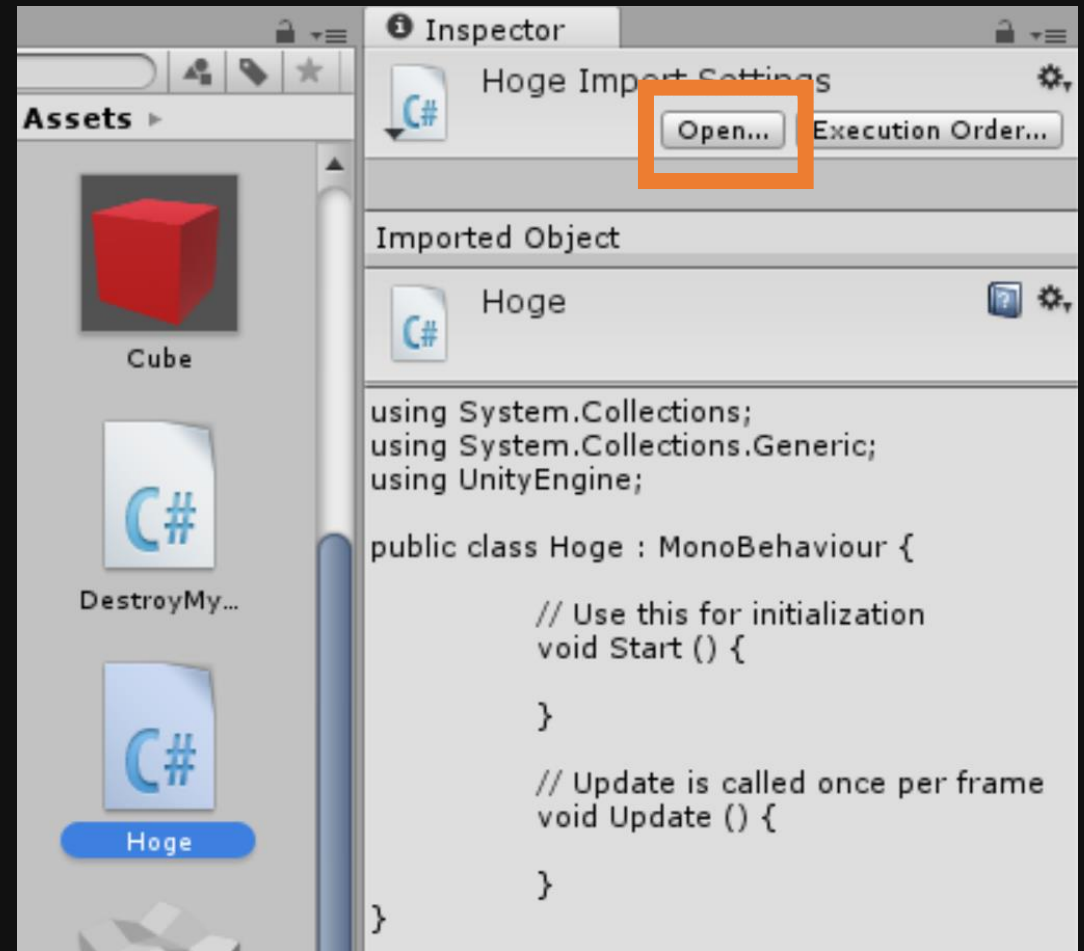
Unityでは、C# を
用いてソースコードが
書ける

Project ⇒ Create ⇒
C# Script から新しく
プログラムを
追加することができる



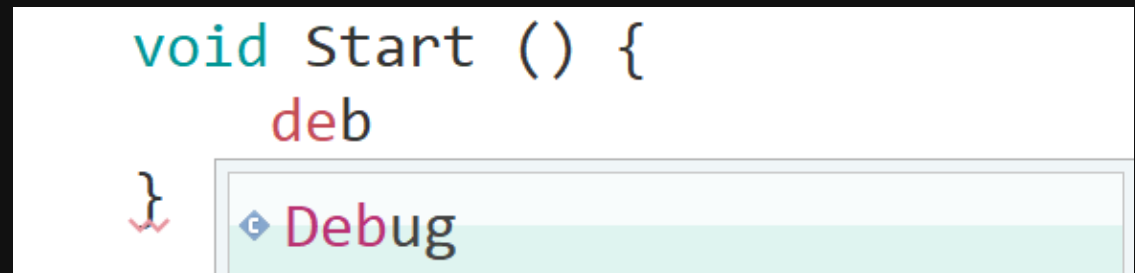
初めてのUnityプログラミング

作成した C# Script
から Open をクリック
すると、エディタが
立ち上がりプログラ
ムの編集ができる



Unityでプログラミングの注意事項

- 予測変換(サジェスト)の活用
 - 長いメソッドをドットでつなげてプログラムを書いていくため、ミスタイプが命取り
→Tabキーの有効活用
 - Tabキーの効果: 予測変換で一番上に出てきたものを自動補完してくれる



The screenshot shows a code editor window with a C# script. The code is as follows:

```
void Start () {  
    deb  
}
```

A dropdown menu is open below the 'deb' text, showing a suggestion for 'Debug' with a small icon to its left. The 'Debug' suggestion is highlighted in a light blue background.

Unityで Hello, world!

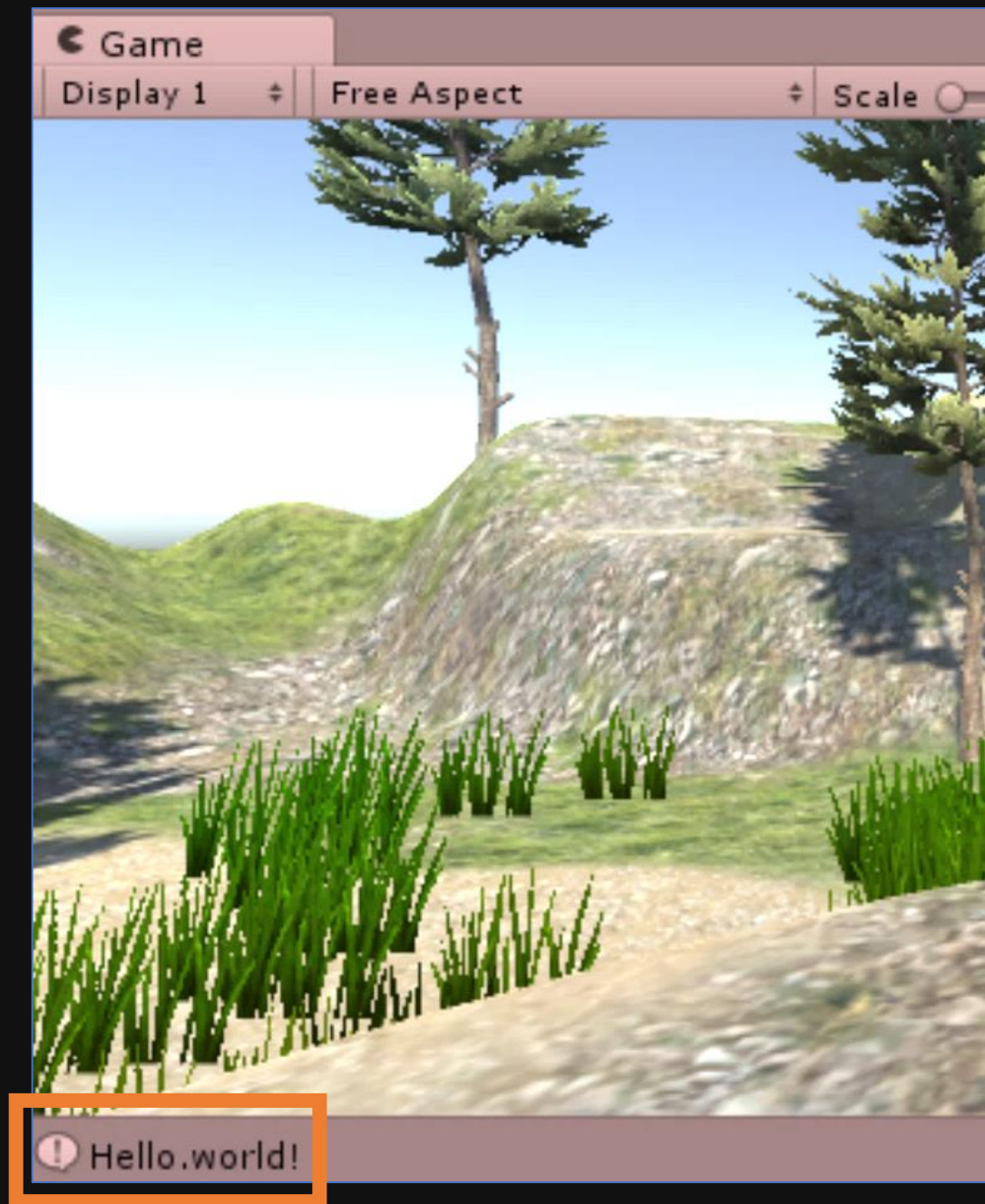
このボタンでプログラムのデバッグが行える

この一行を加えてCtrl+Sで保存

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class Hoge : MonoBehaviour {
6
7     // Use this for initialization
8     void Start () {
9         Debug.Log ("Hello.world!");
10    }
```

Unityで Hello, world!

1. **Hierarchy** ⇒ **Create** ⇒ **Create Empty**から空の GameObject を作る
2. その GameObject の **Inspector** に先ほどのスクリプトを追加する
3. ゲームを再生する
4. 左下の Console に
"Hello, world!" が表示される



GameObjectを複製する

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Hoge : MonoBehaviour {

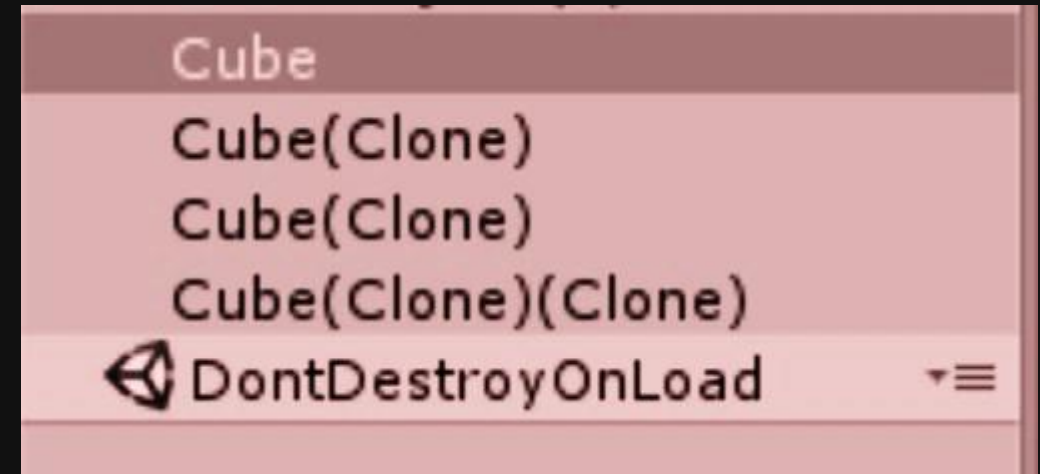
    void Start () {

    }

    void Update () {
        if (Input.GetKeyDown(KeyCode.A)) { // Aキーの入力受け付け
            Instantiate(this.gameObject); // 自身を複製する
        }
    }
}
```

GameObjectを複製する

1. **Hierarchy** ⇒ **Create** ⇒ **Cube** を追加する
2. 前スライドのスク립トを作成し、Cube の **Inspector** に追加する
3. 再生する
4. Aキーを押すと、同じ場所に同じGameObject が複製される
5. ただし、今回は自身を対象に複製しているため、指数関数的に増えていく



↑二回複製した図
 $2^2 = 4$

新しくGameObjectを生成する

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class HogeHoge : MonoBehaviour {

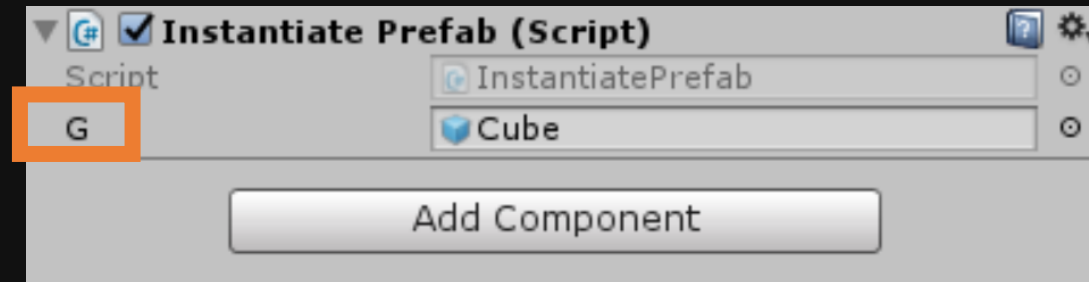
    public GameObject g; // gameobjectを指定
    void Start () {

    }

    void Update () {
        if (Input.GetKeyDown(KeyCode.B)) { // Bキーが押されたか判定
            Instantiate (g); // gを生成する
        }
    }
}
```

新しくGameObjectを生成する

1. **Hierarchy** ⇒ **Create** ⇒ **Create Empty**を追加する
2. 前スライドのスク립トを1に追加する
3. **Hierarchy** ⇒ **Create** ⇒ **Cube**を追加後、**Hierarchy** から **Project** ⇒ **Assets** にドラッグドロップ
4. 追加したスク립ト内のGameObject に **Project** ⇒ **Assets** 内のCubeをドラッグドロップ
5. 再生する
6. **Project** 内の指定したものが生成できるようになる



新しく GameObject を生成する

- 本操作で何をしていたか？
 - 空の GameObject に〇〇を作るスクリプト(ふるまい)を追加していた
 - **Hierarchy** から **Project** ⇒ **Assets** フォルダに落とす作業は、作りたいもののテンプレートを保存していた
 - 本来はモデルを作ってそこから引っ張ってくるが、そのモデルがないため、本操作でモデルを作成した
 - スクリプト内の作成物は最初は空だが、後付けで作成物を指定していた

Unityをこれから使っていくには？

- 作りたいものを作って発信しよう！！
 - お手軽に作れる環境が揃っているので活用を
- **Asset Store** を積極的に活用しよう！！
 - 一人でやるには分野が広すぎて**限界がある**
- プログラムの基礎はちゃんと学ぼう！！
 - ツールに使われるだけではダメ

ご清聴ありがとうございました