

プロジェクト演習
インタラクティブ・ゲーム制作 III/V

<ゲームデザイン>
2022 前期

担当 安原広和
yasuharahk@edu.teu.ac.jp

第2回目

「Unity」の操作をしよう

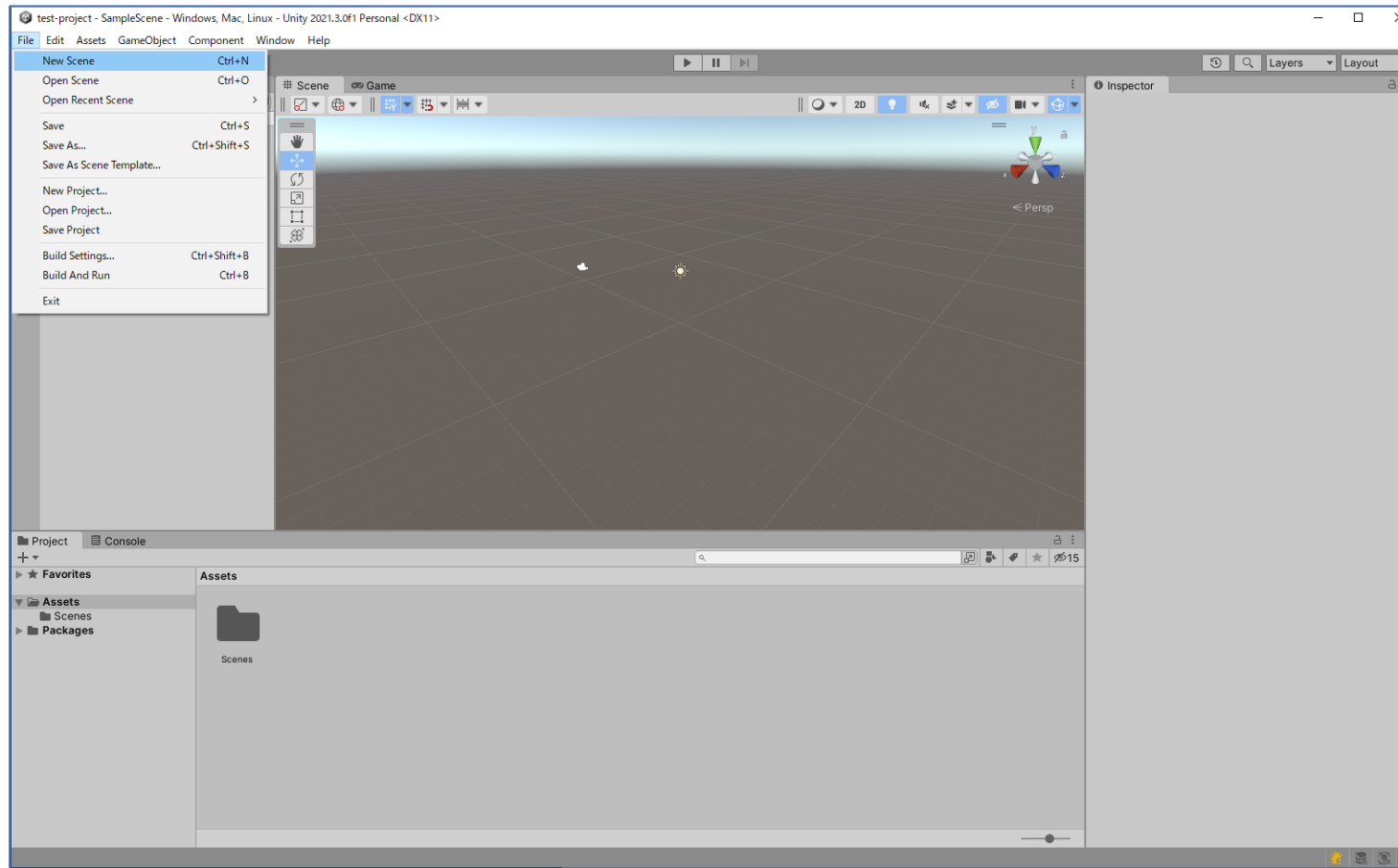
第2回の授業は「実習」の時間!

Unityでボールを転がしてみよう!

まずは「Unityの基本操作」です

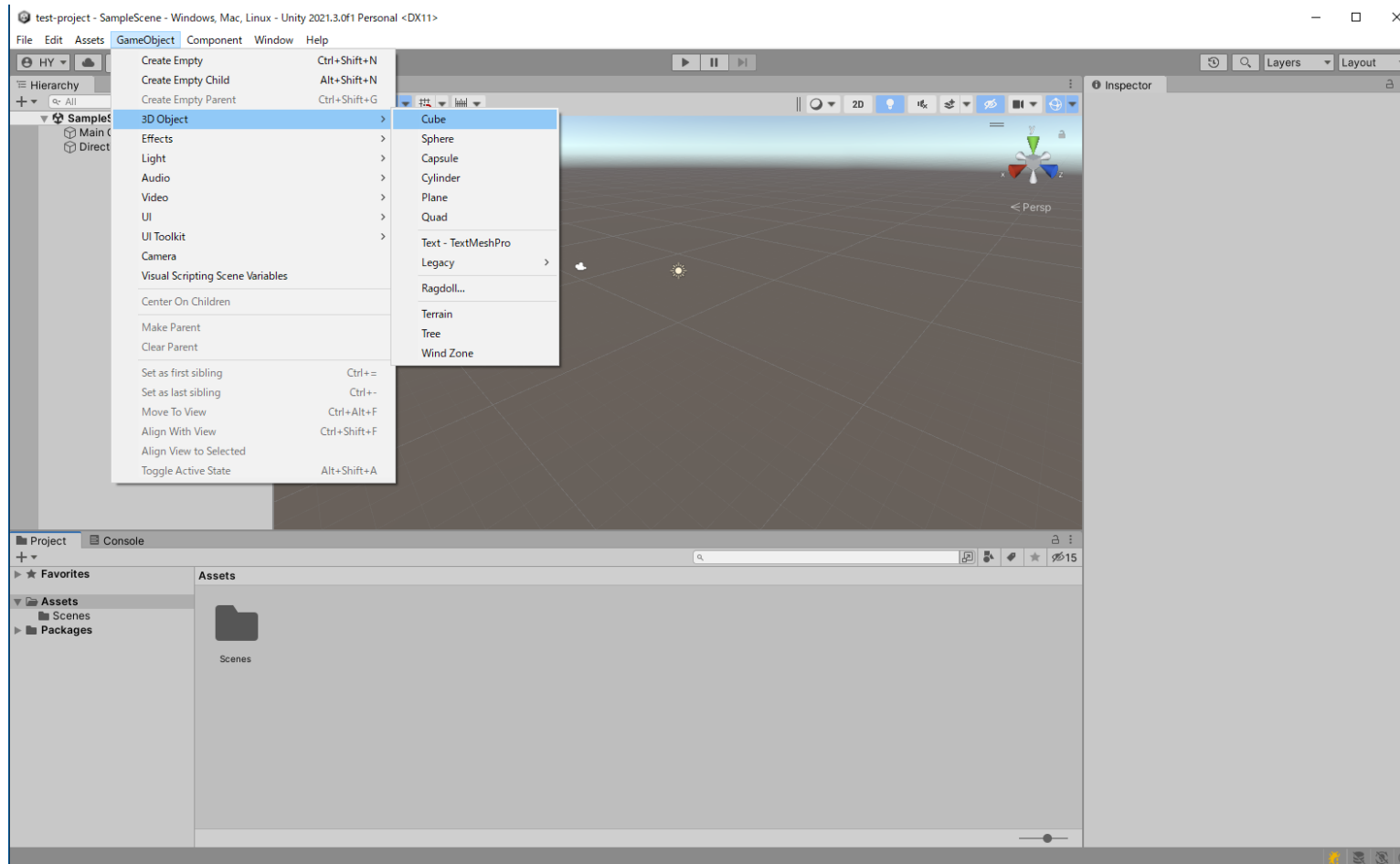


Unityをひらきます



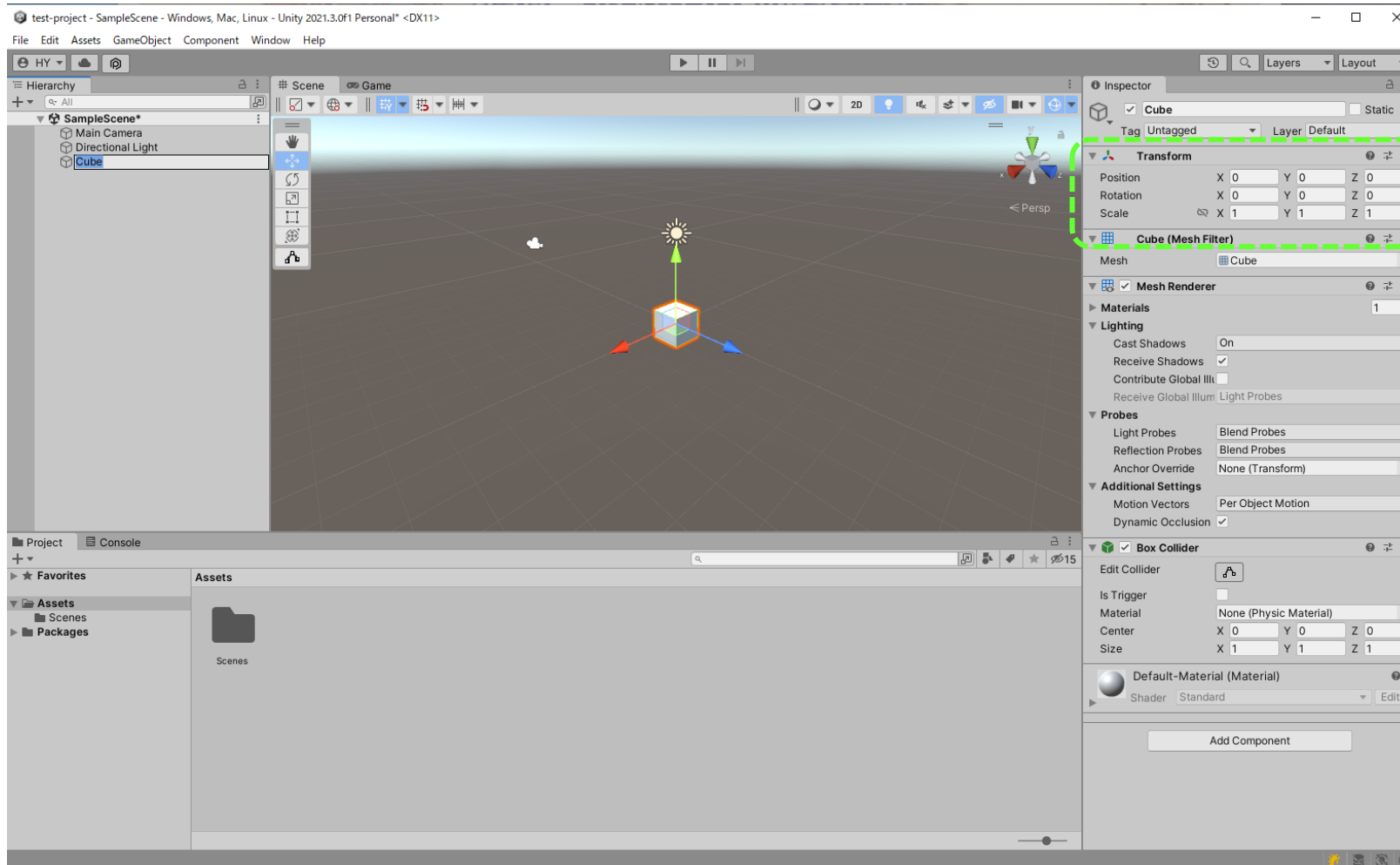
1. File > New Scene でSceneを作ります
2. 「Save」で名前をつけて、このSceneをセーブしておきましょう

「GameObject」を作しましょう



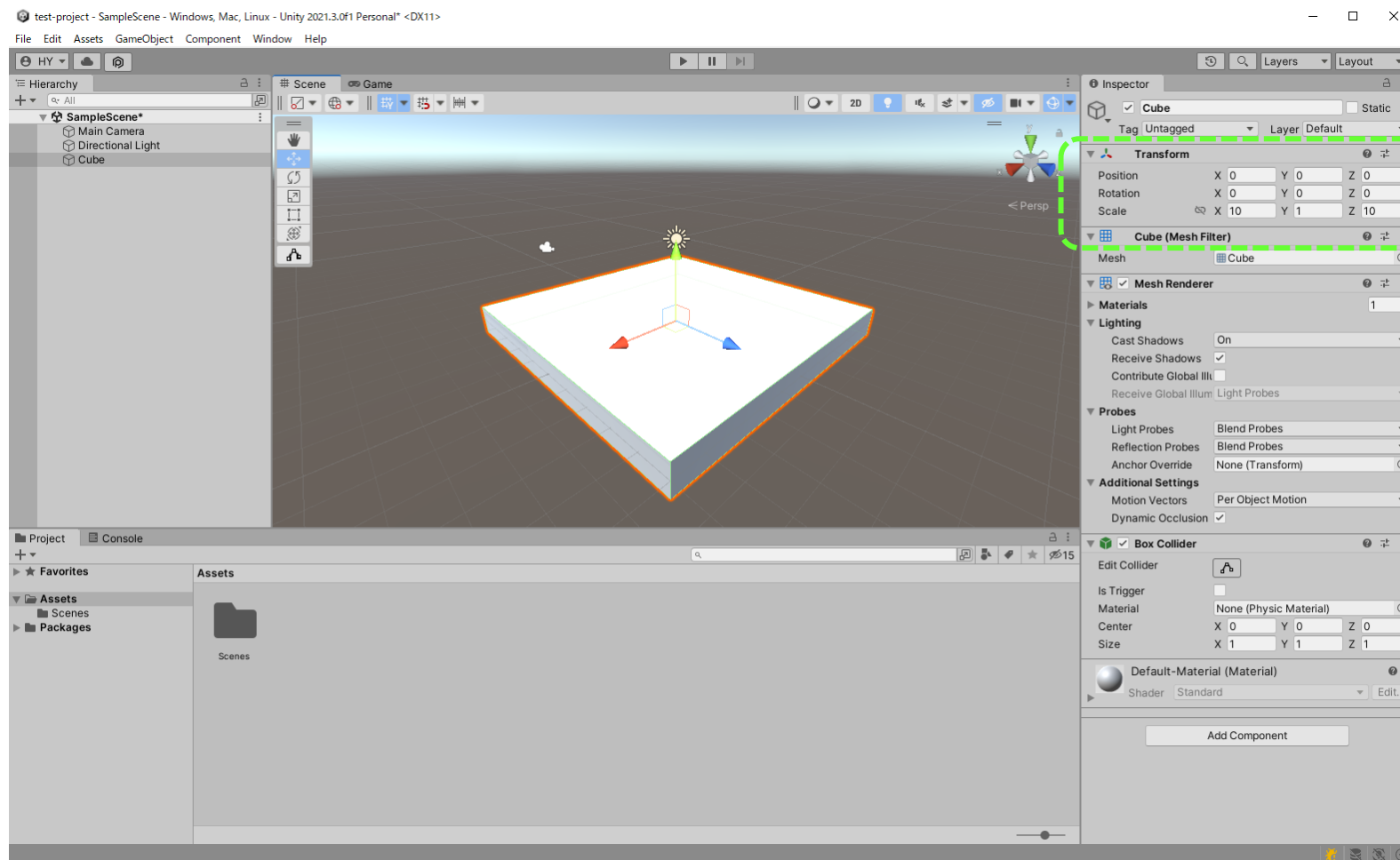
1. GameObject > 3D Object > Cube で四角の箱を作ります

「Cube」の場所とサイズを変えます



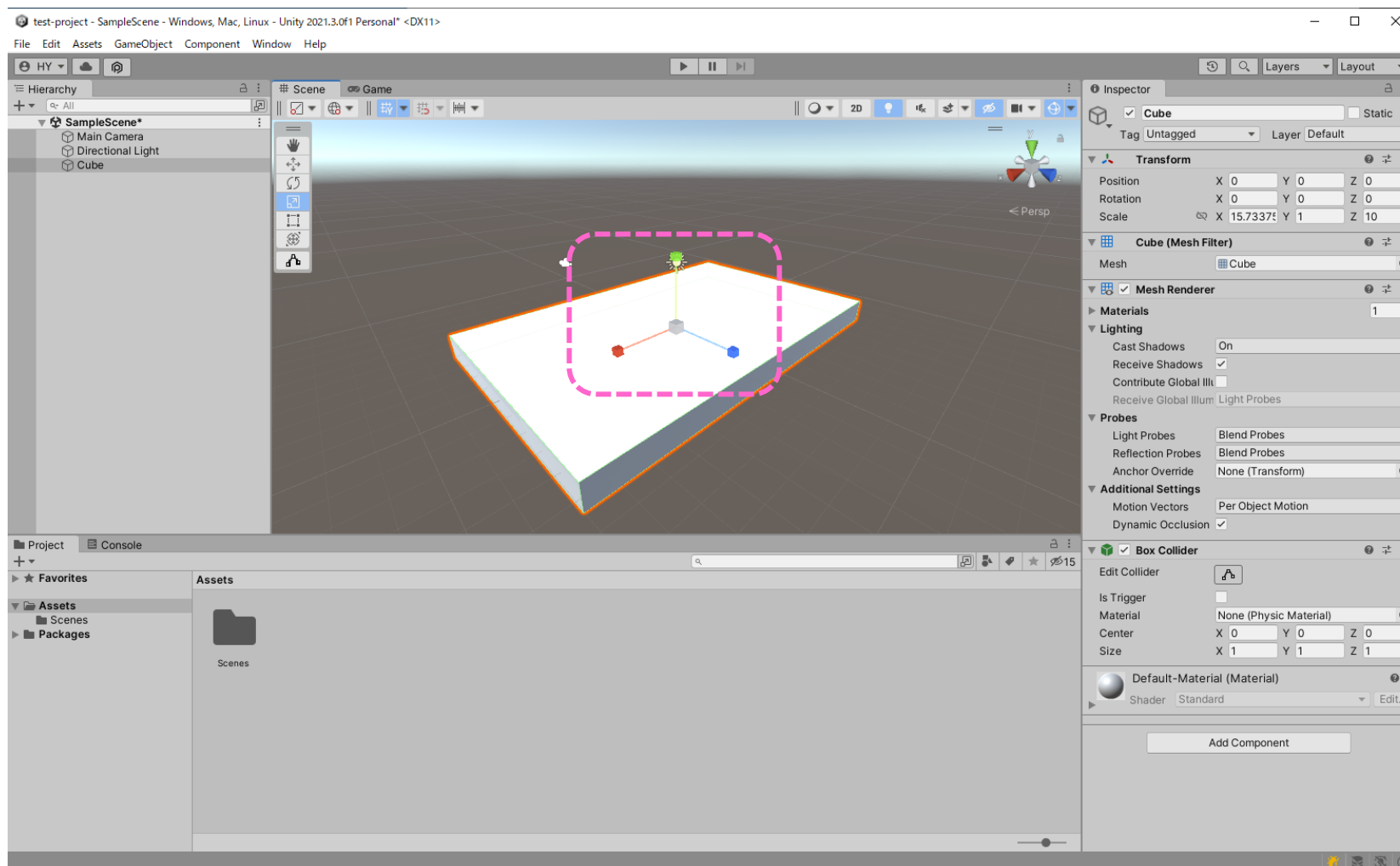
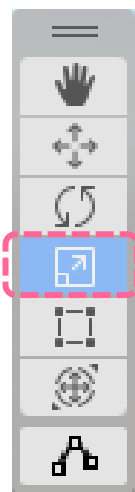
1. Cubeの「Inspector」欄にある「Transform」の「Position」の値と「Scale」の値の枠内に、直接数字を打ち込んでCubeの大きさを変えてみましょう

「Cube」の場所とサイズを変えます



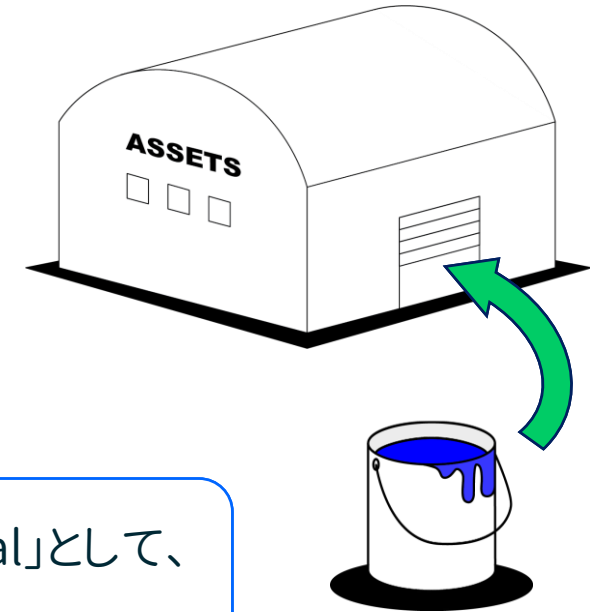
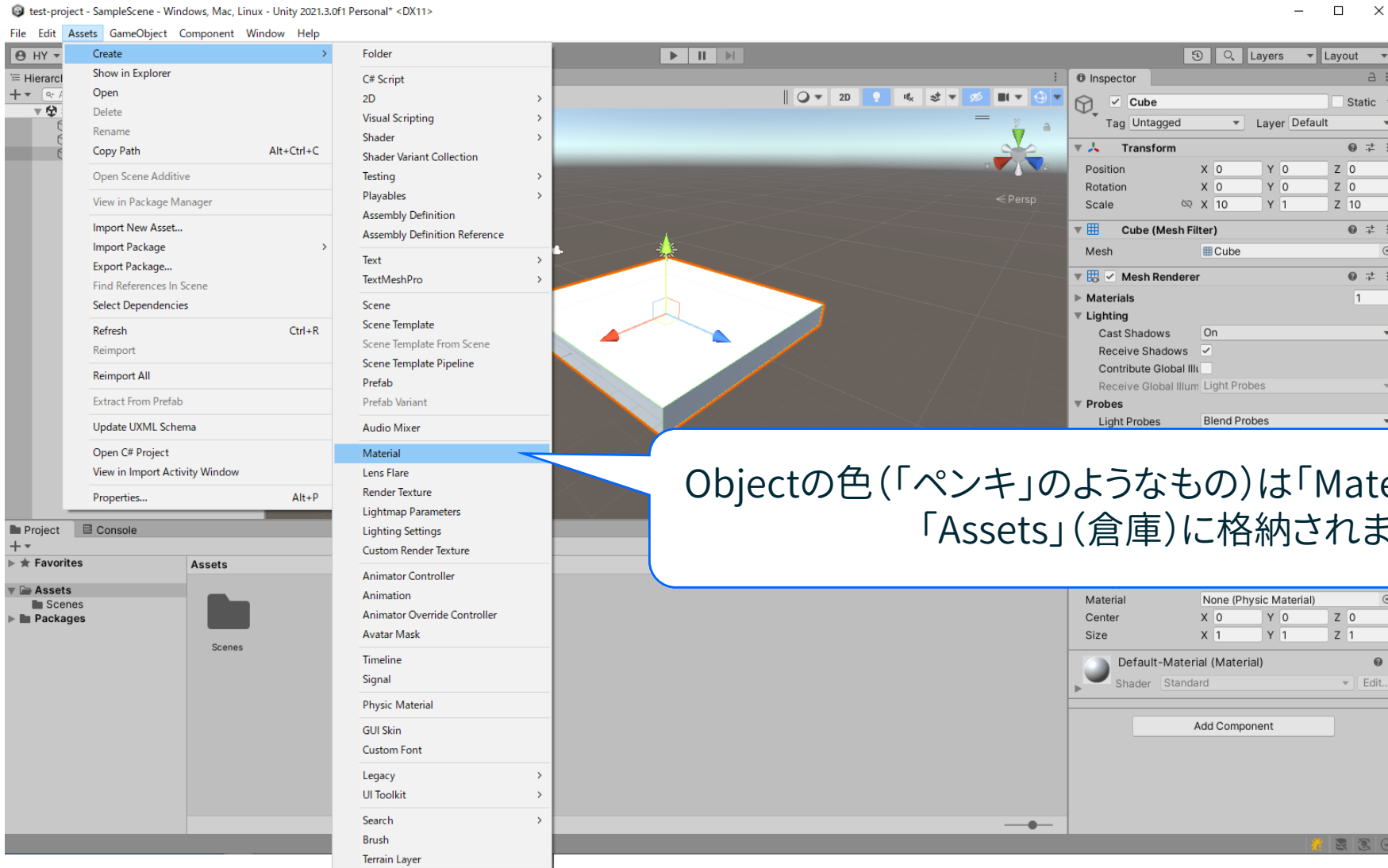
1. X軸側、Z軸側に値を大きくすると、平らな床になります

「Toolメニュー」から変形させる



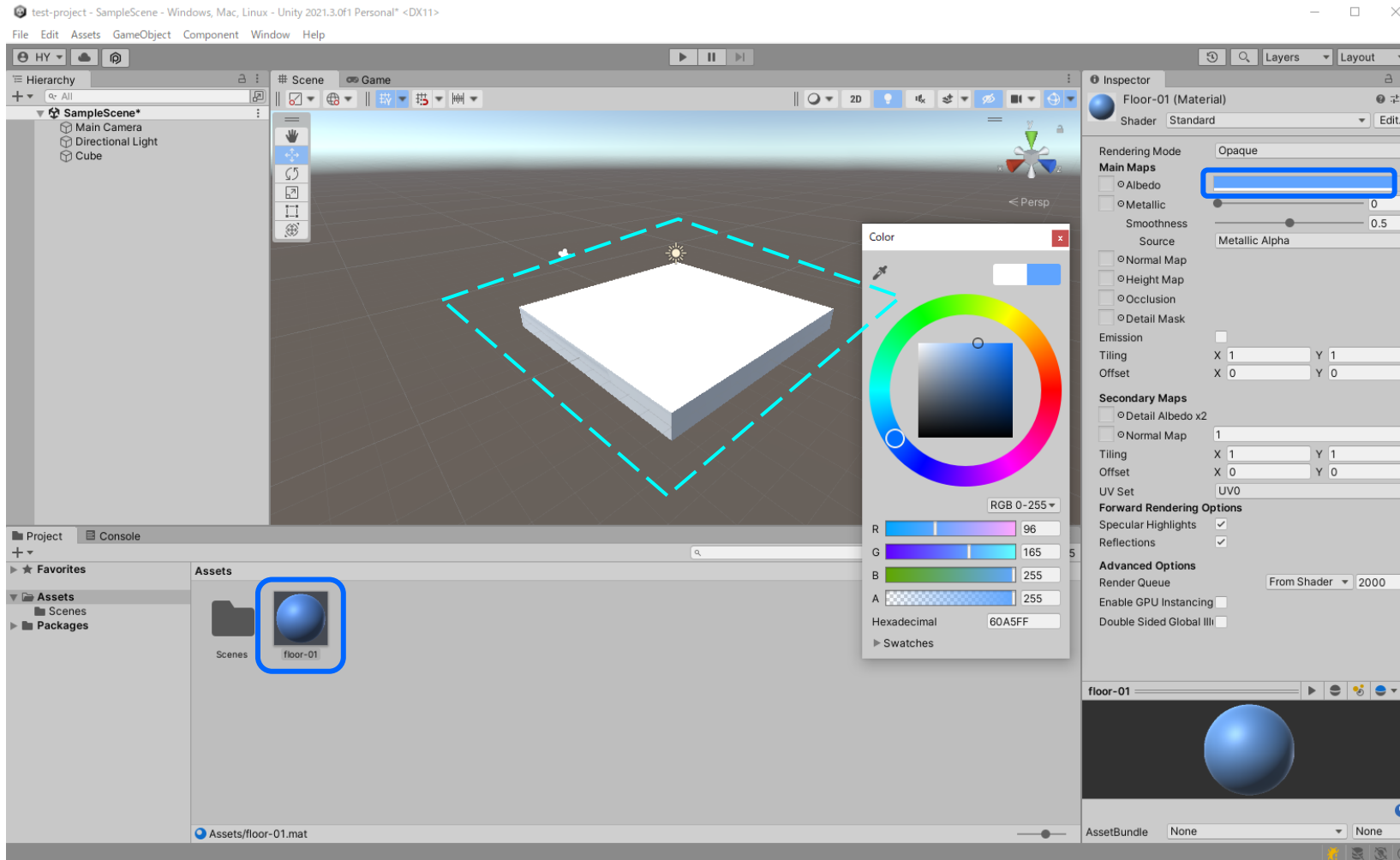
Toolメニューから「拡大縮小」を選ぶと、拡大縮小用のAxisに変わります
Axisの軸をマウスでドラッグすることで、変形させることができます

Cubeに色をつけます



Assets > Create > Material とタブを開いていきます

Cubeに色をつけます



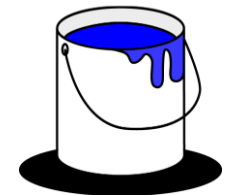
ここをクリック



カラーパレットが開
きます

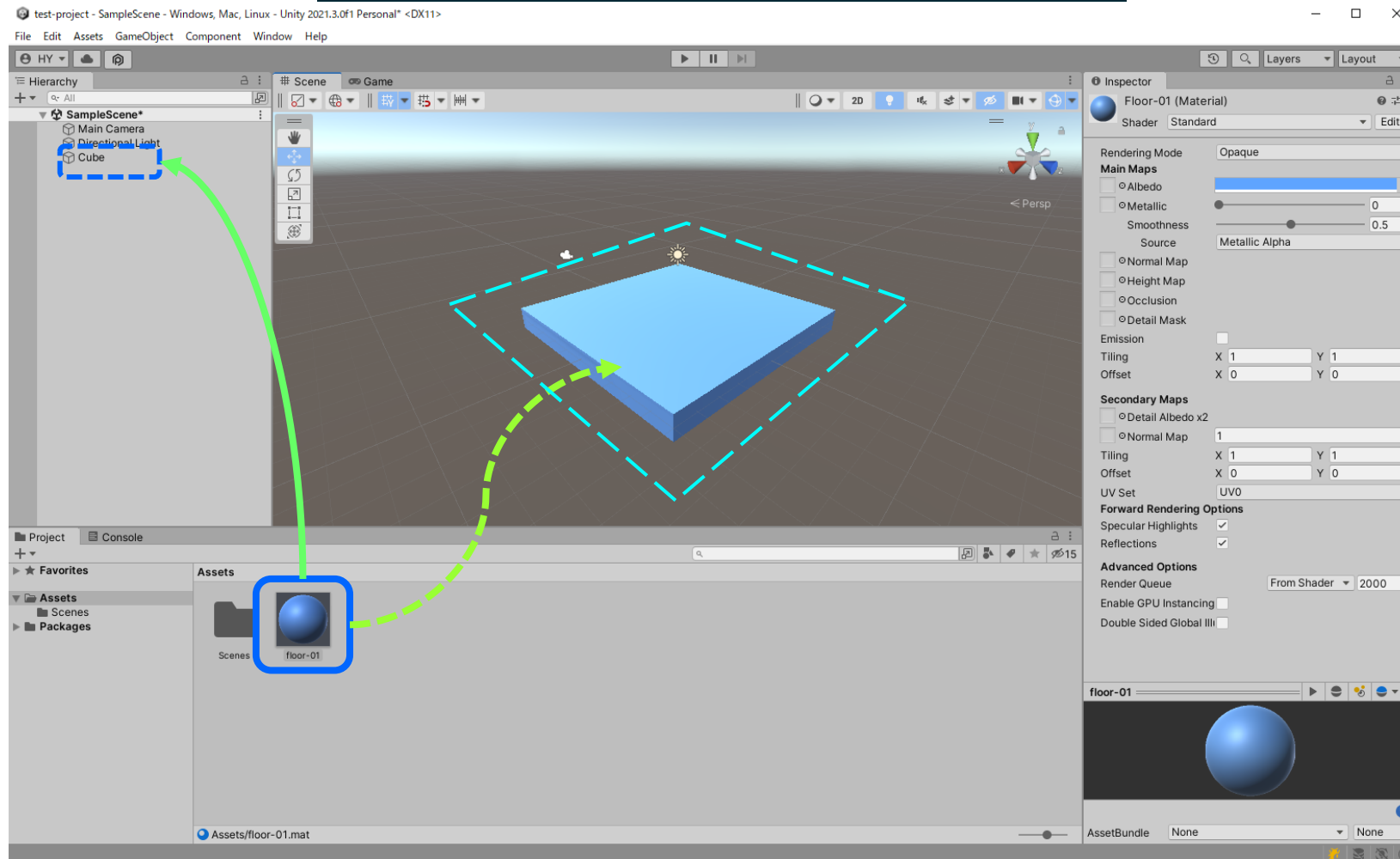


好きな色を
選びます



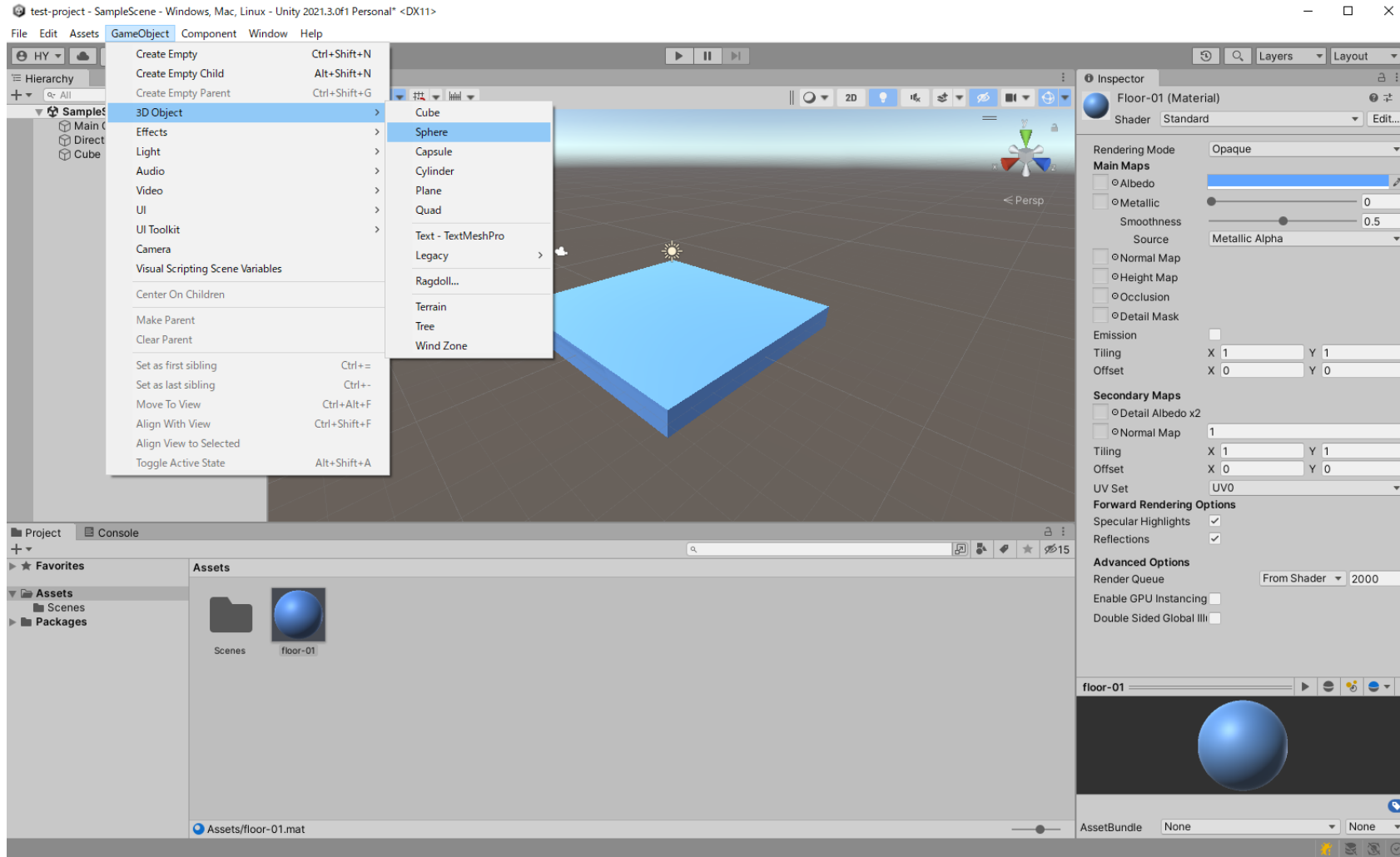
「Material」が「Asset」欄に加わり、「Material」の設定が「Inspector」欄に表示されます

Cubeに色をつけます



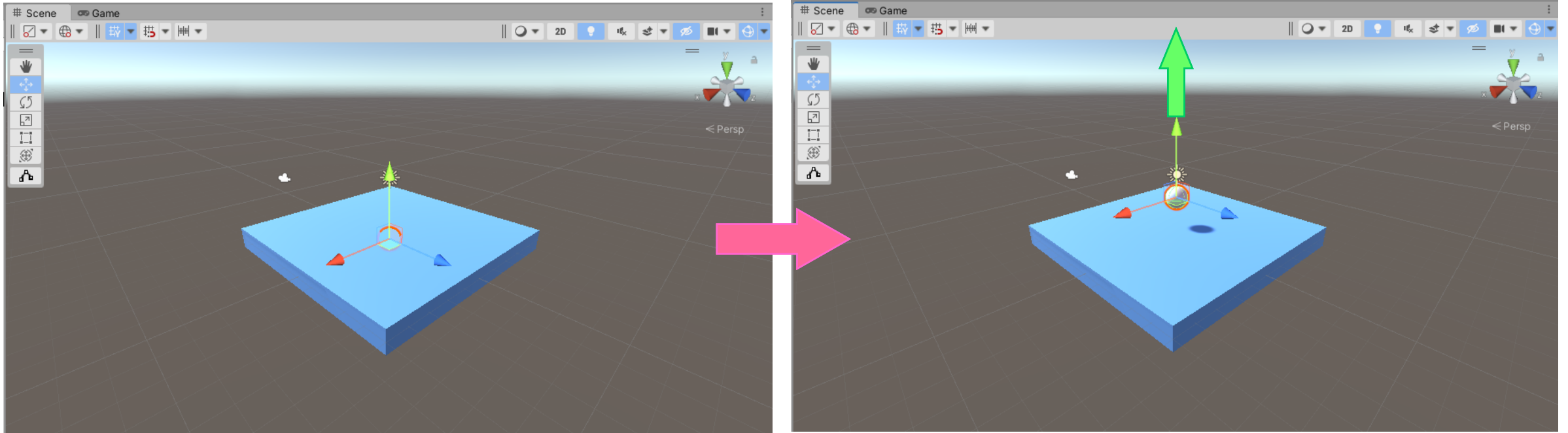
「Asset」欄に追加されたボール状の「Materialのアイコン」をドラッグして「Hierarchy」欄のCube、もしくは直接、「Scene」内のCubeに入れます

次に「Sphere」を作ります



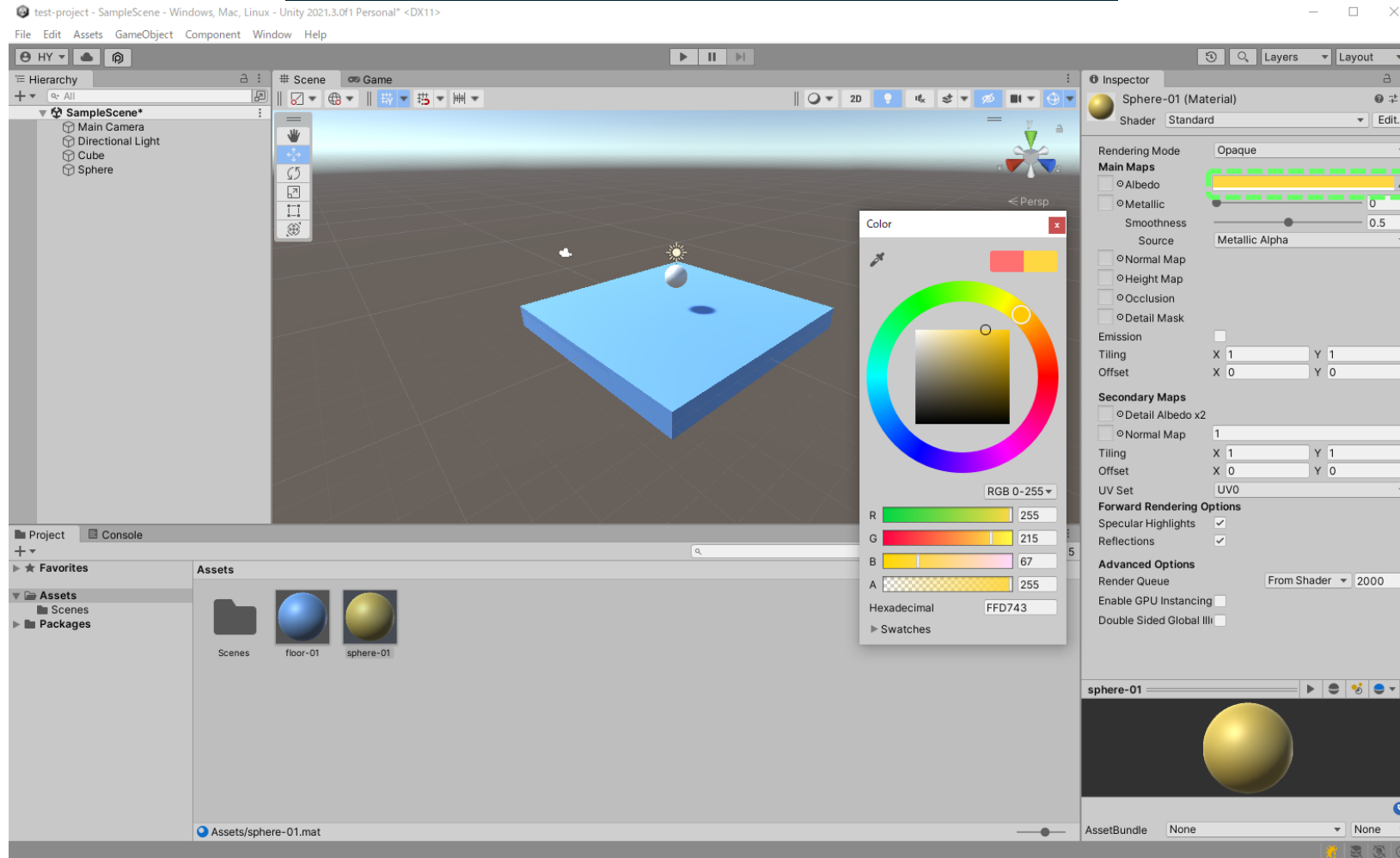
先ほどのCubeと同じように GameObject > 3D Object > Sphere とタブをひらき、球体を作ります

Sphereを作ります



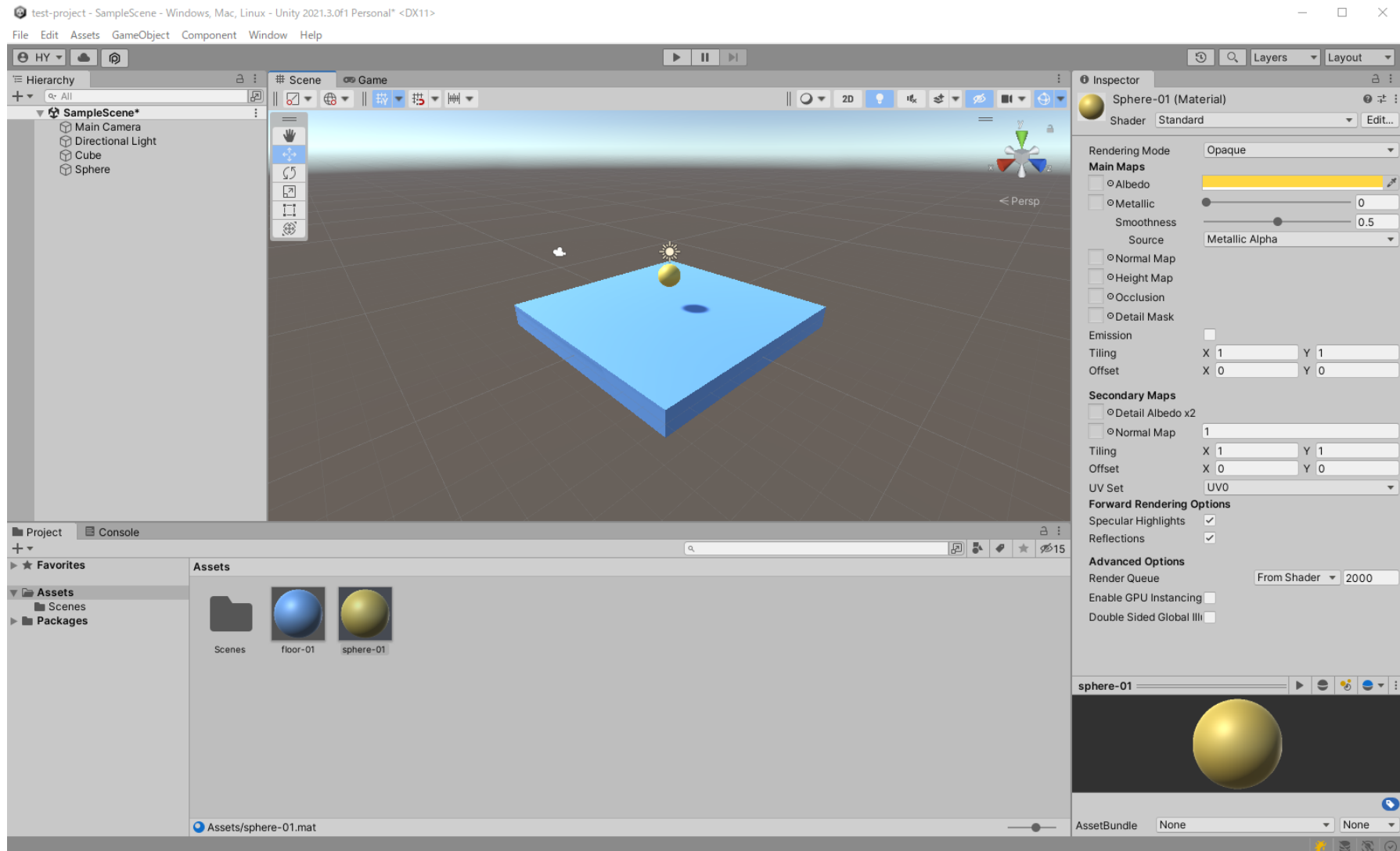
作られた「Sphere」はPositionが(0,0,0)でCubeに埋まっているので、矢印をy軸方向にドラッグしてSphereを引き上げます

Sphereに色をつけます



先ほどのCubeと同じように Assets > Create > Material と開いて「色」をつけます

フロアとボールができました

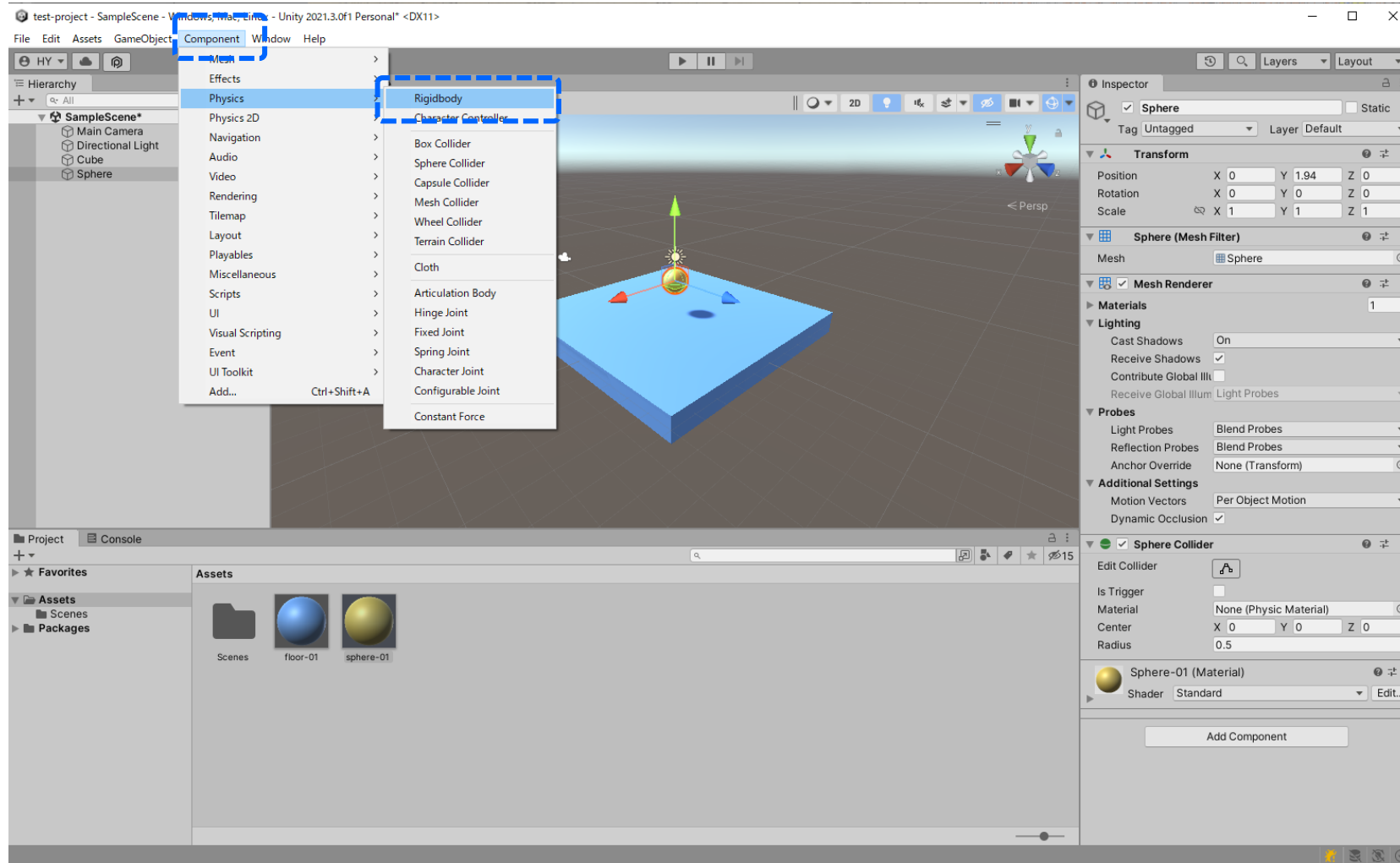


これに物理法則を加えます！

インタラクションの
始まりです！

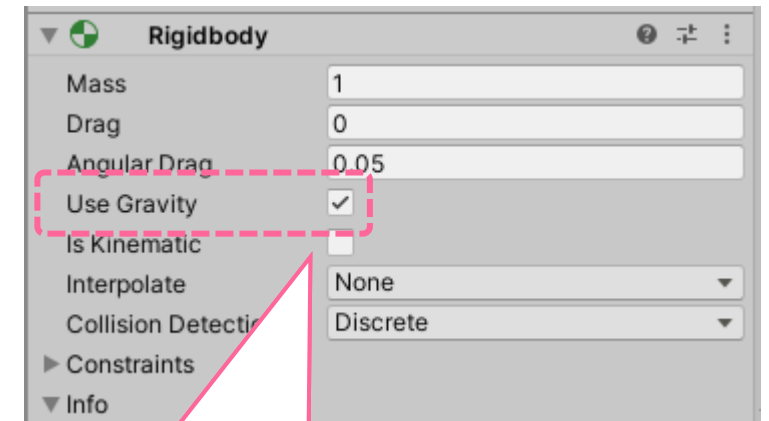
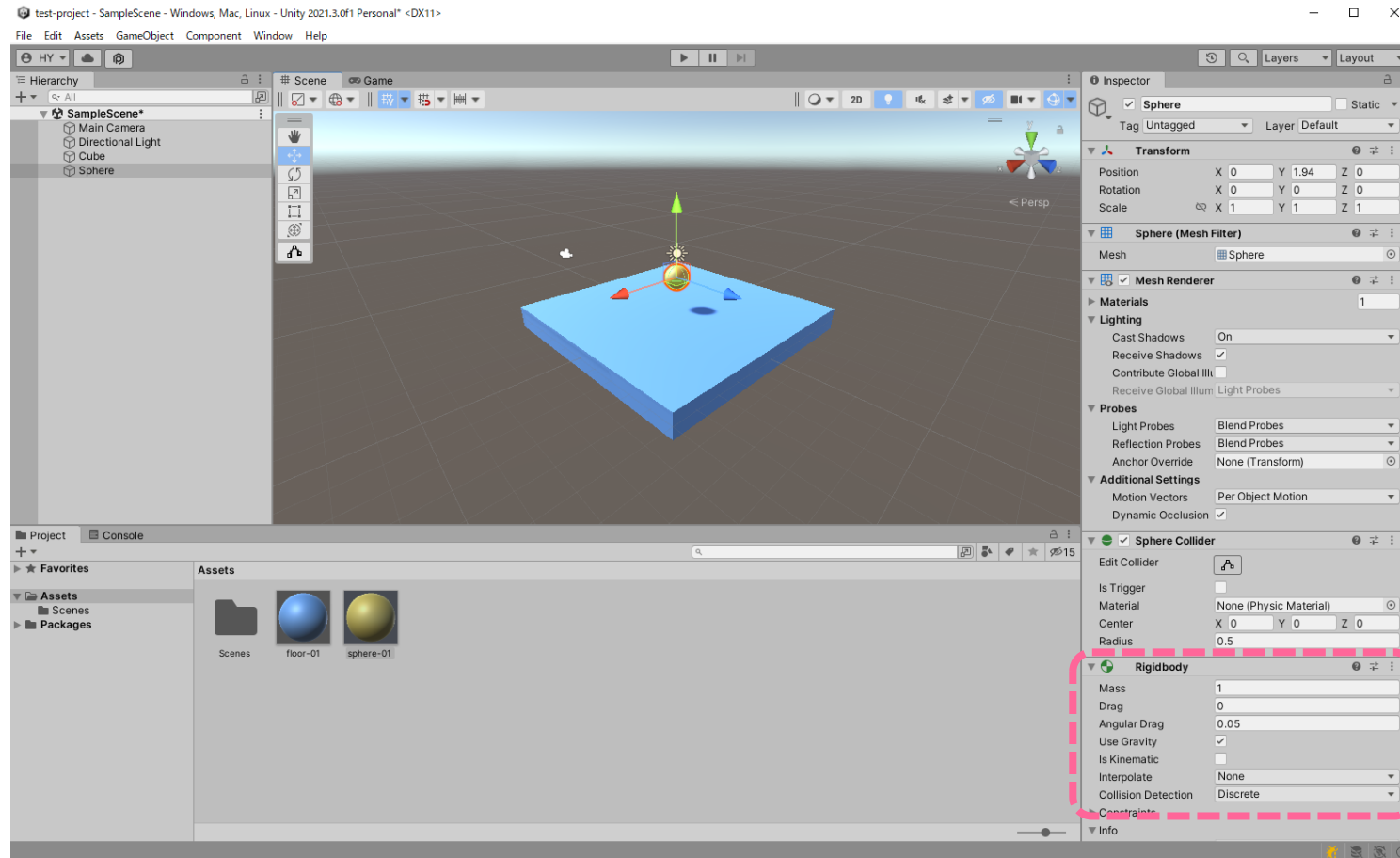


ボールに物理法則を起こす「Rigidbody」を追加します



ボールのObjectを選択して、 Component > Physics > Rigidbody とします

Sphereに足されたRigidbody欄の「Use Gravity」をチェックします

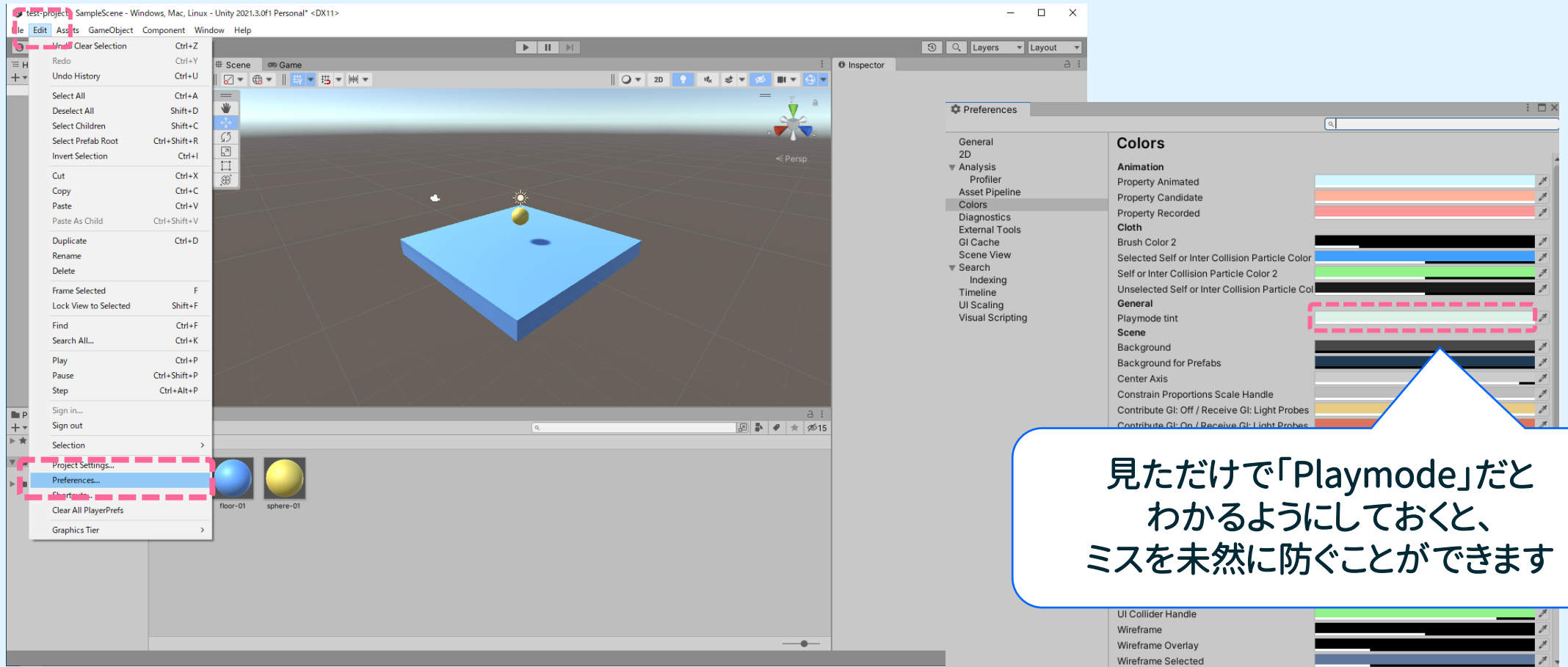


「重力」を働かせます

ボールのObjectに足された「Rigidbody」欄にある「Use Gravity」にチェックをつけます
これでボールに「重力」が働くようになります

参考

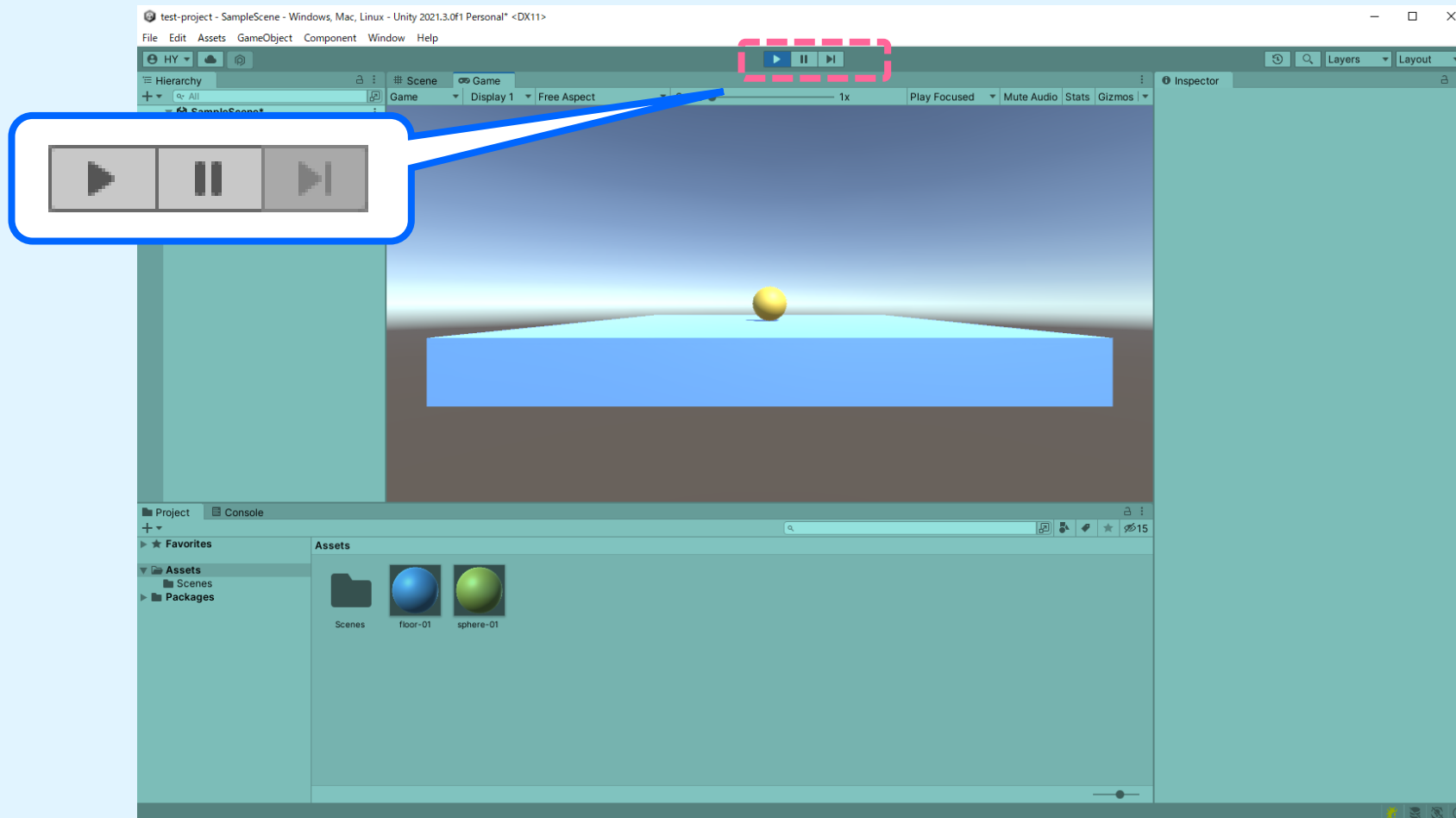
プレイモード時のUnityの色を自由に変えることができます



見ただけで「Playmode」だとわかるようにしておくと、ミスを未然に防ぐことができます

Edit > Preferenceを選ぶと「Preference」メニューが開きます
「Colors」内の「General」>「Playmode tint」の色を変えることができます

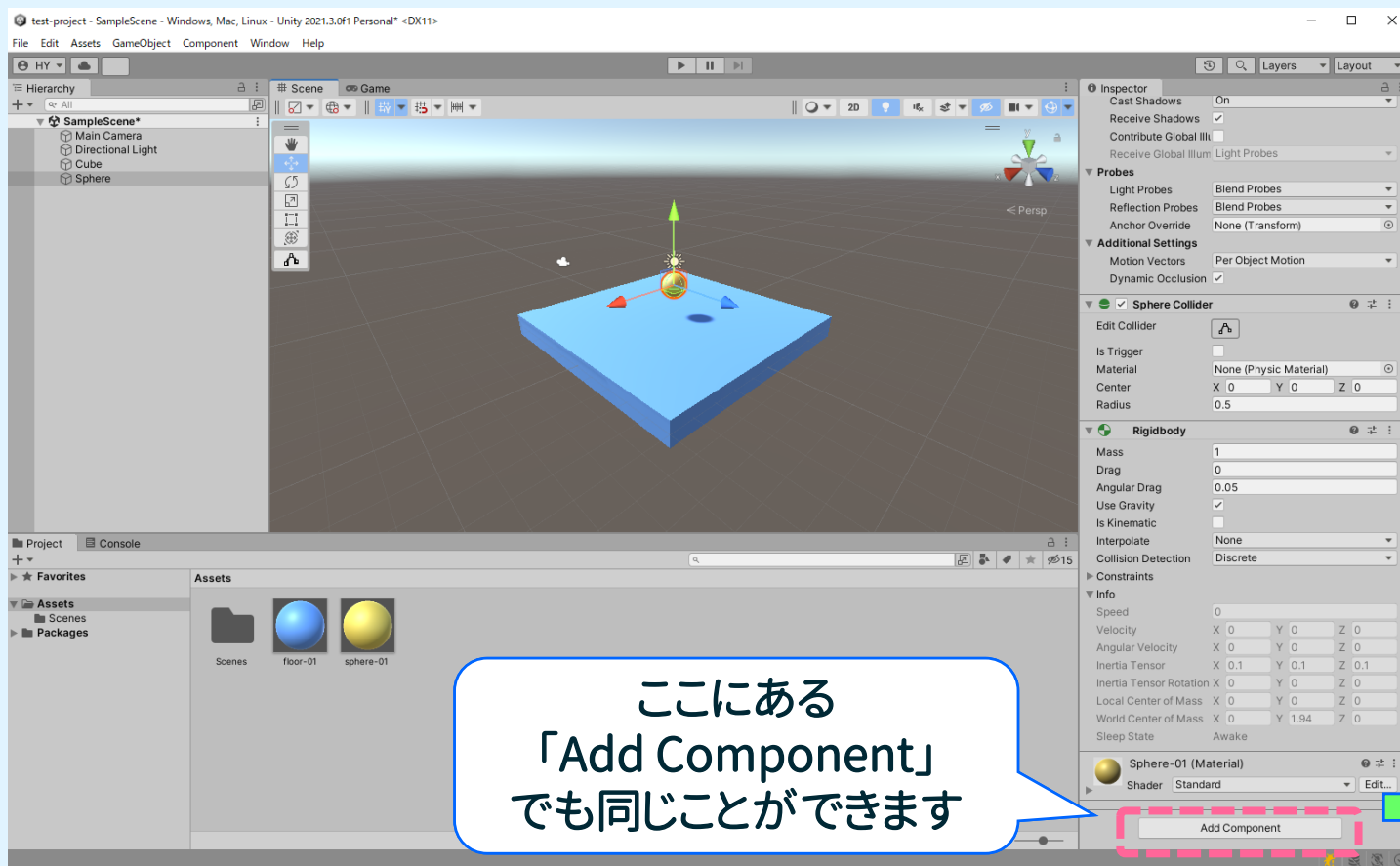
プレイモードのボタンを押して確認します



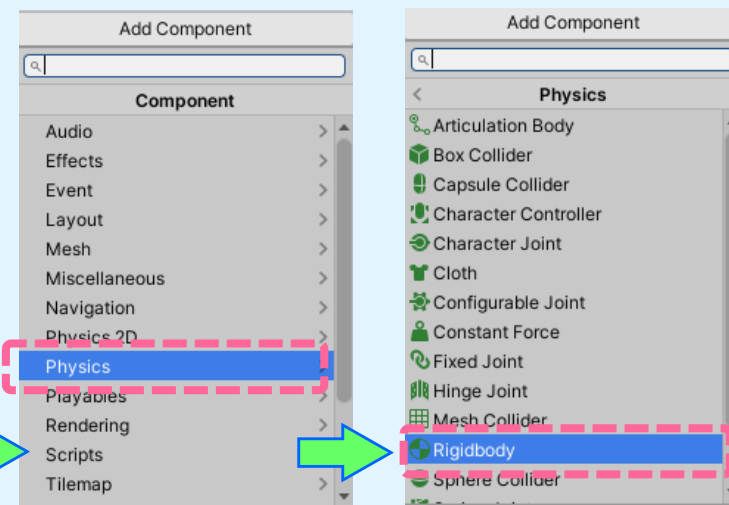
Scene画面の上にある「Play」ボタンを押した時にエディター全体の色が変わることを確認します

追加情報

「Rigidbody」を追加する別の方法

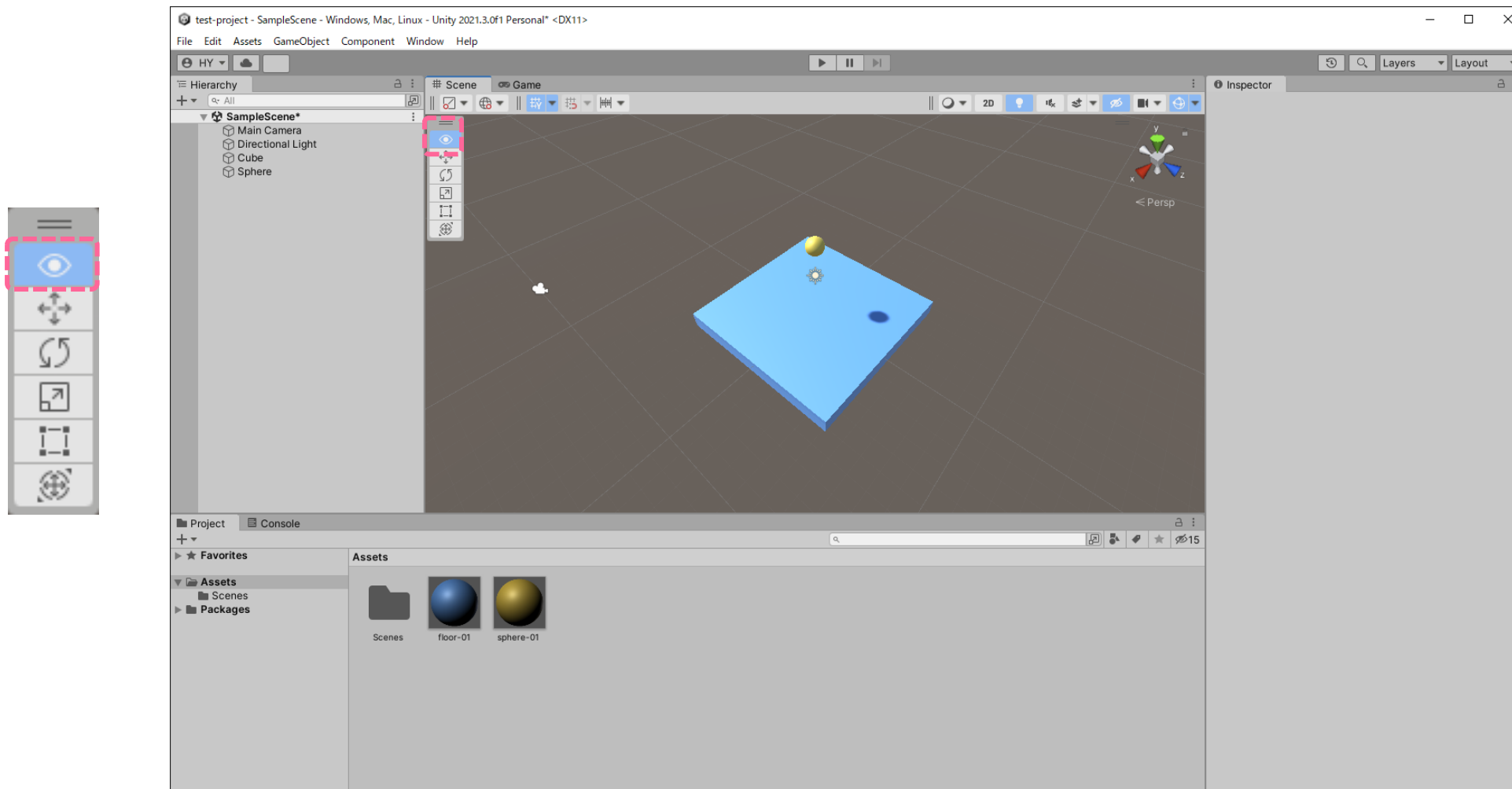


ここにある
「Add Component」
でも同じことができます



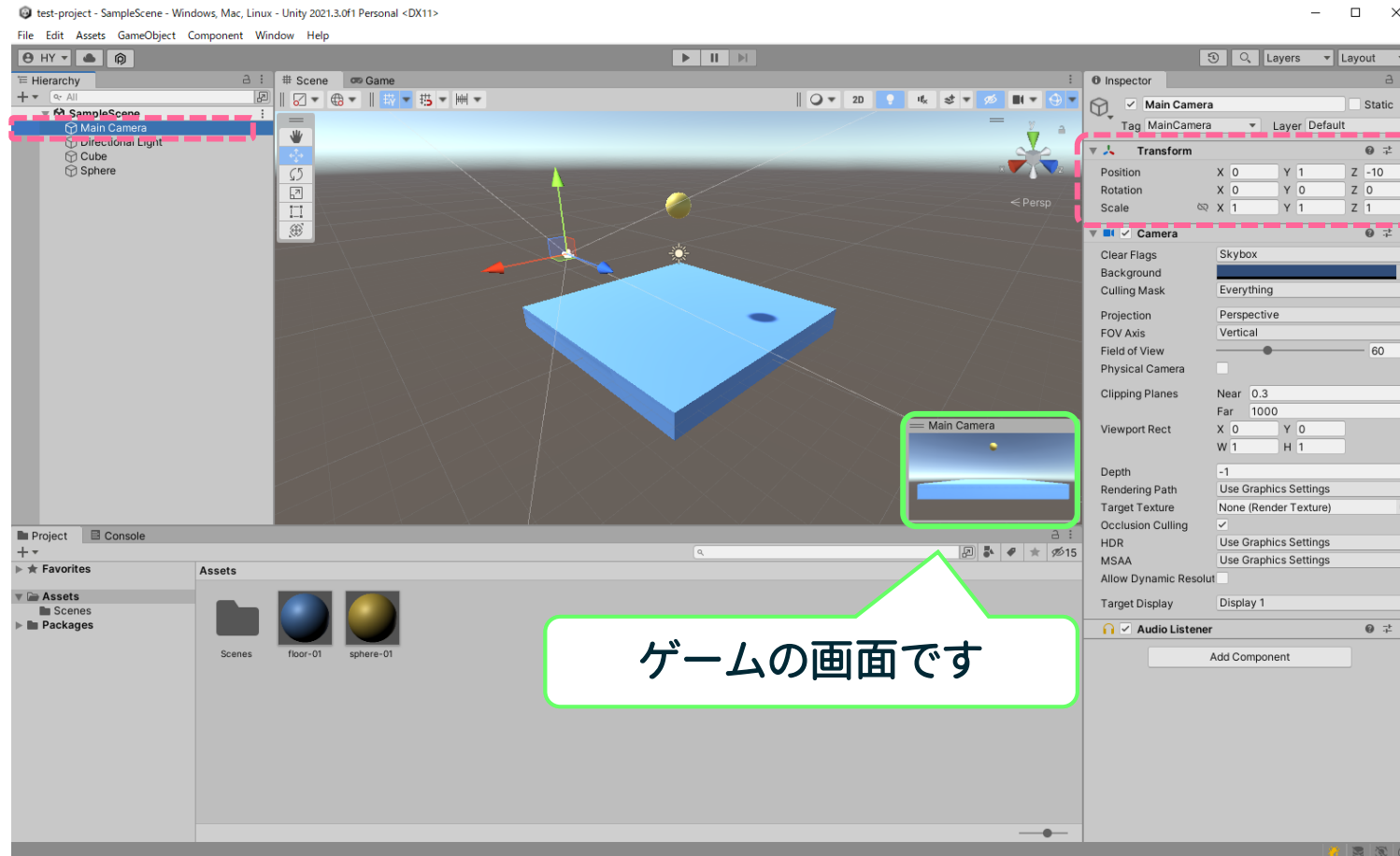
Add Component > Physics > Rigidbody
メニューがひらき同じ手順でRigidbodyをつけることができます

Sceneのカメラを移動させる方法



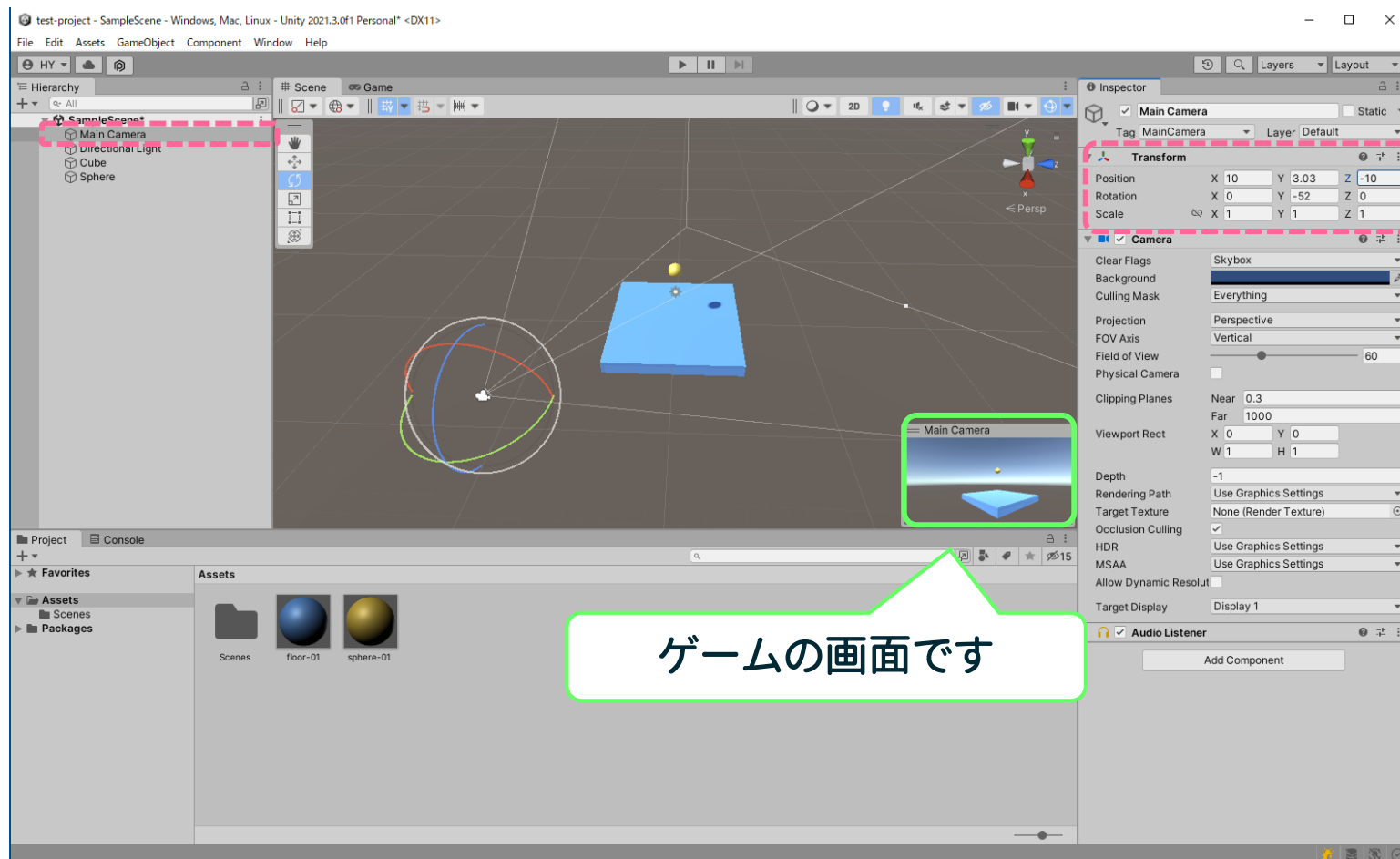
Sceneのカメラ位置は、「Alt」キーを押しながら、マウスの移動で行います
「Alt」キーを押すと、コマンドメニューの「手」表示が「👁」になります

カメラの位置を調整します



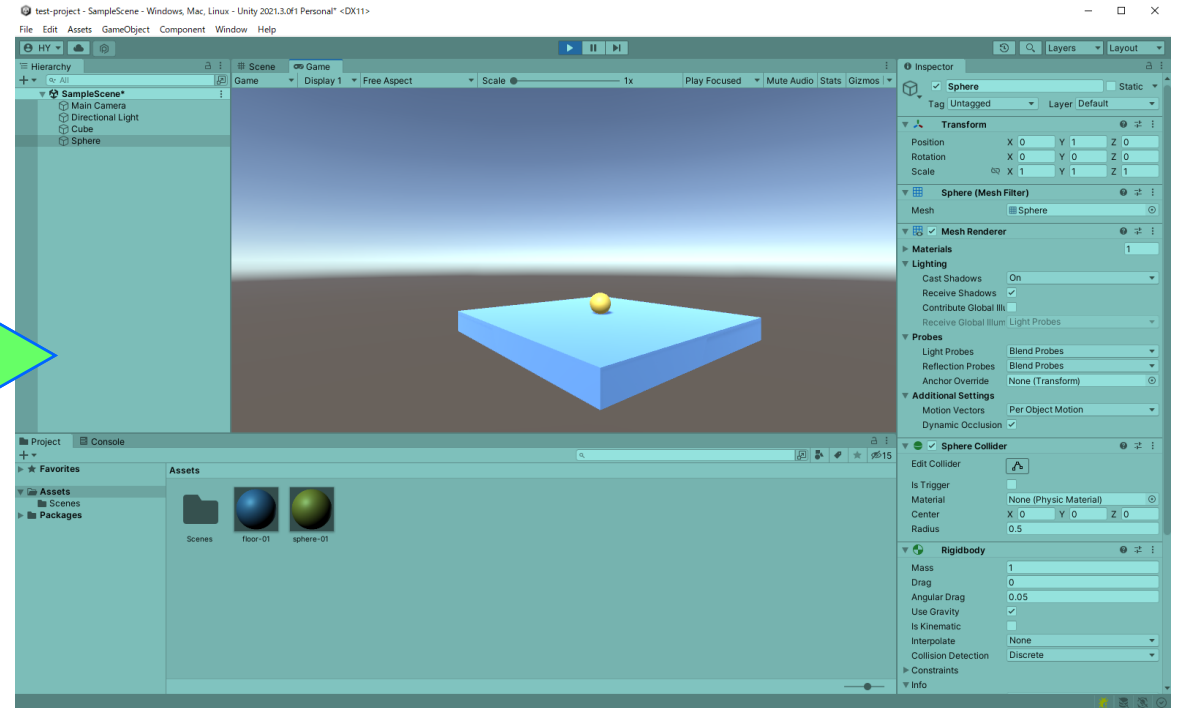
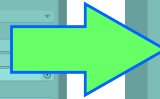
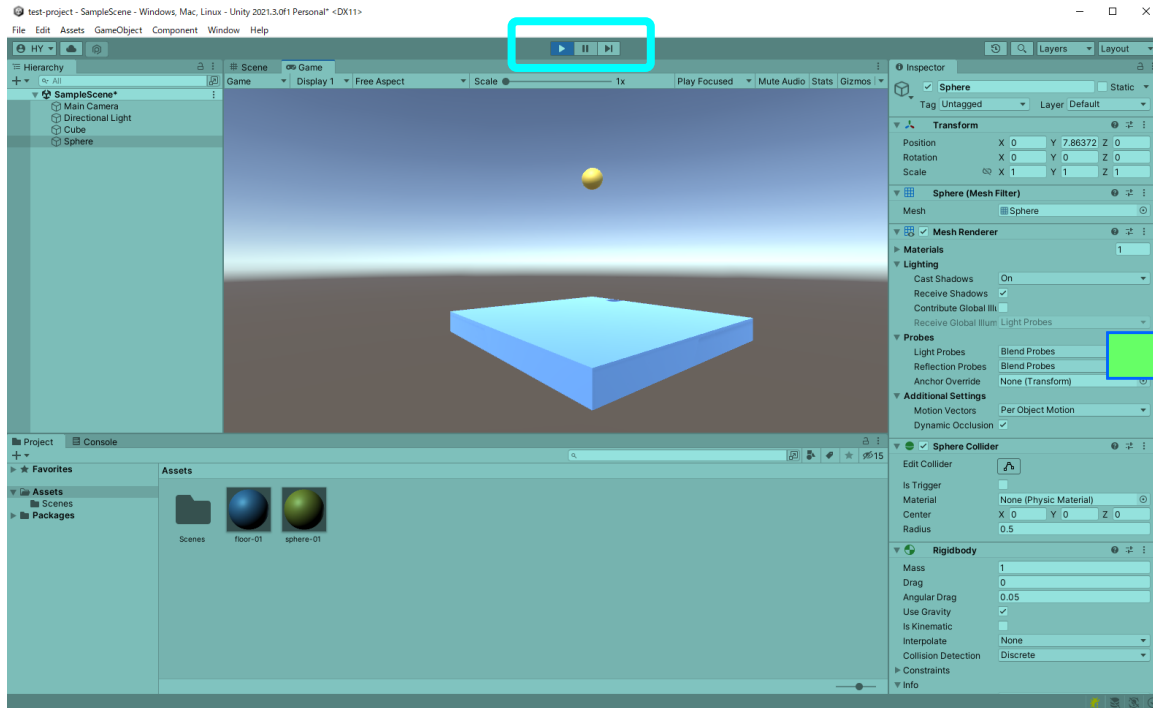
ゲームのカメラビューは、「Hierarchy」欄にある「Main Camera」の位置を変えることで、変化させます

カメラの位置を調整します



ゲームの「Main Camera」を角度を変えるなどしながら良い位置に、移動させます

ボールの落下を確認します



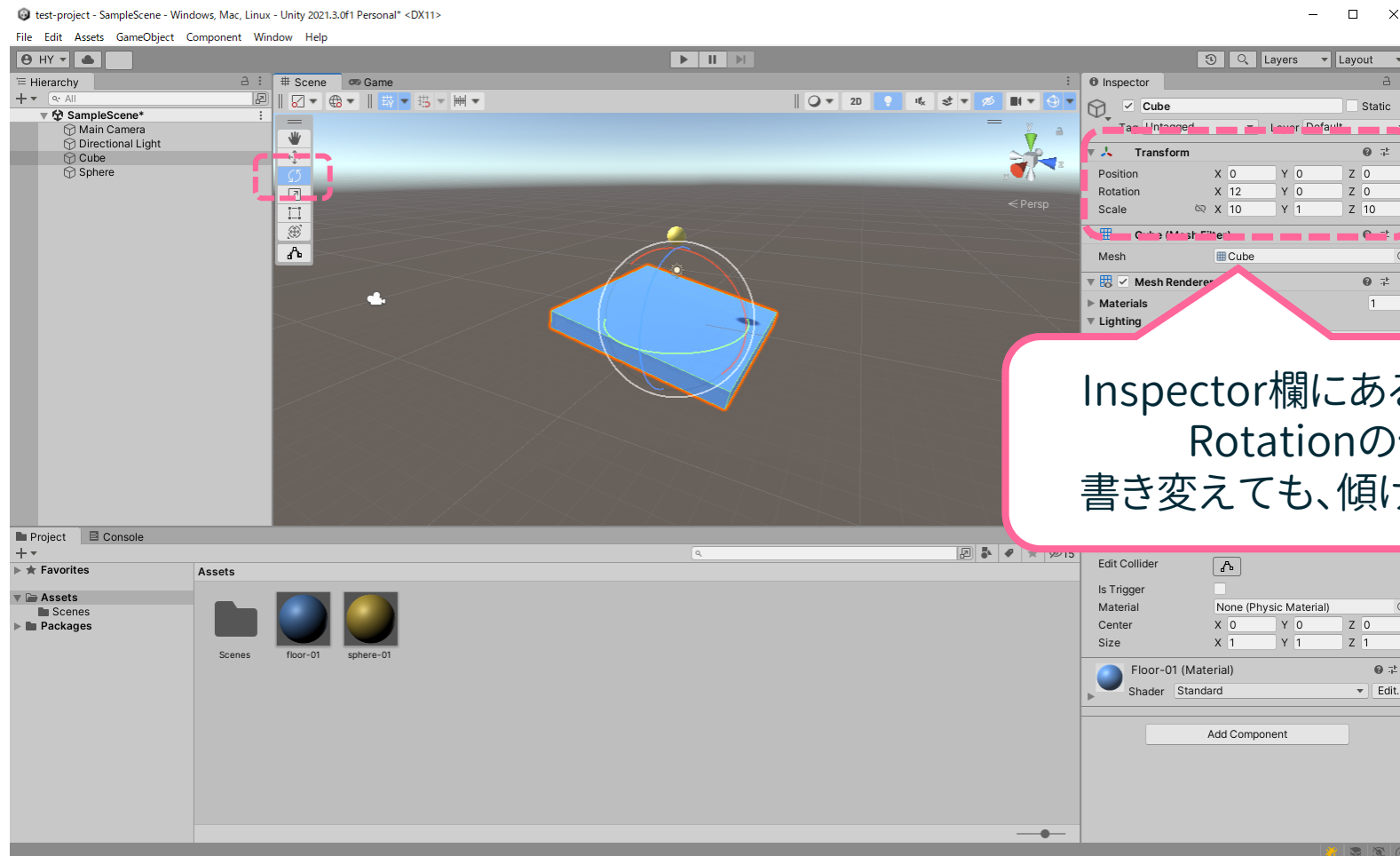
「Play」ボタン▶を押して、ボールがフロアに落下するのを確認します

ボールを転がします

ボールはフロアが
傾いていれば転がります

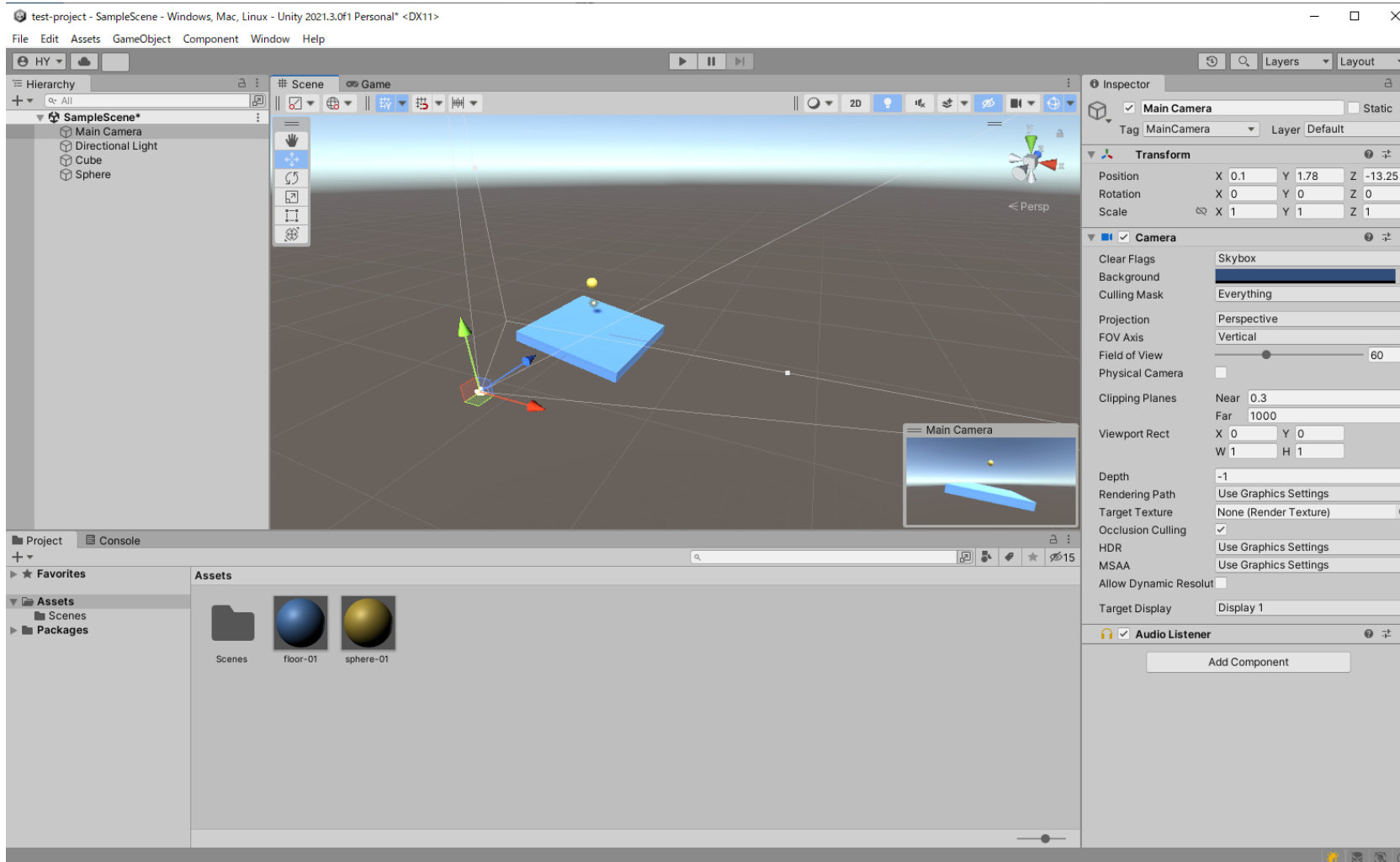


フロアを傾けて「ボール」を転がします



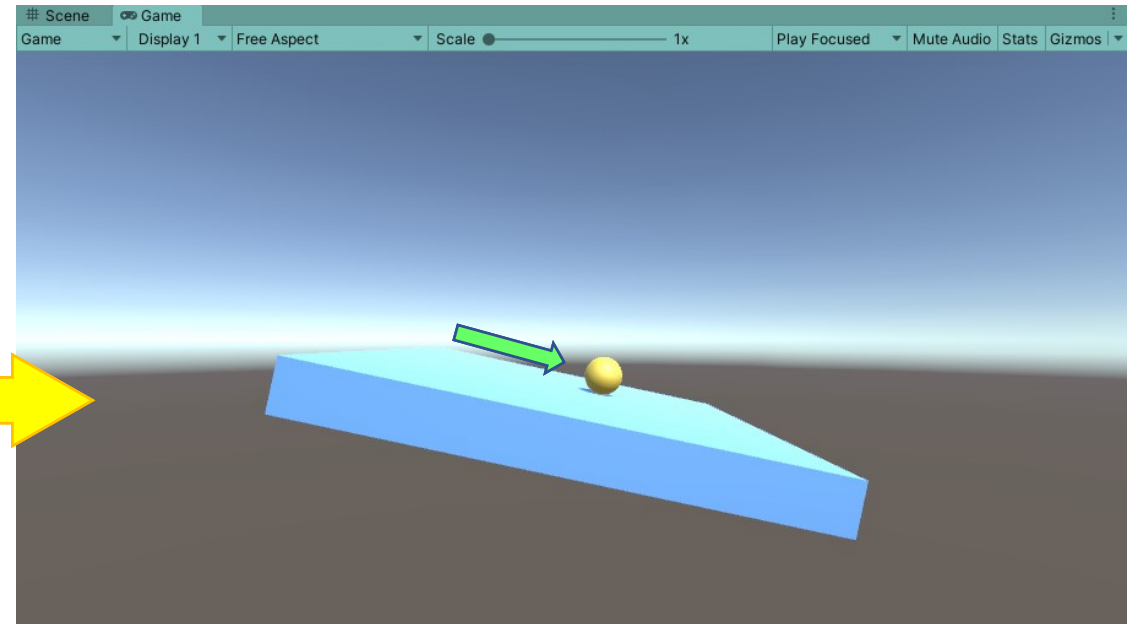
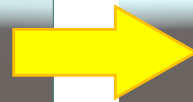
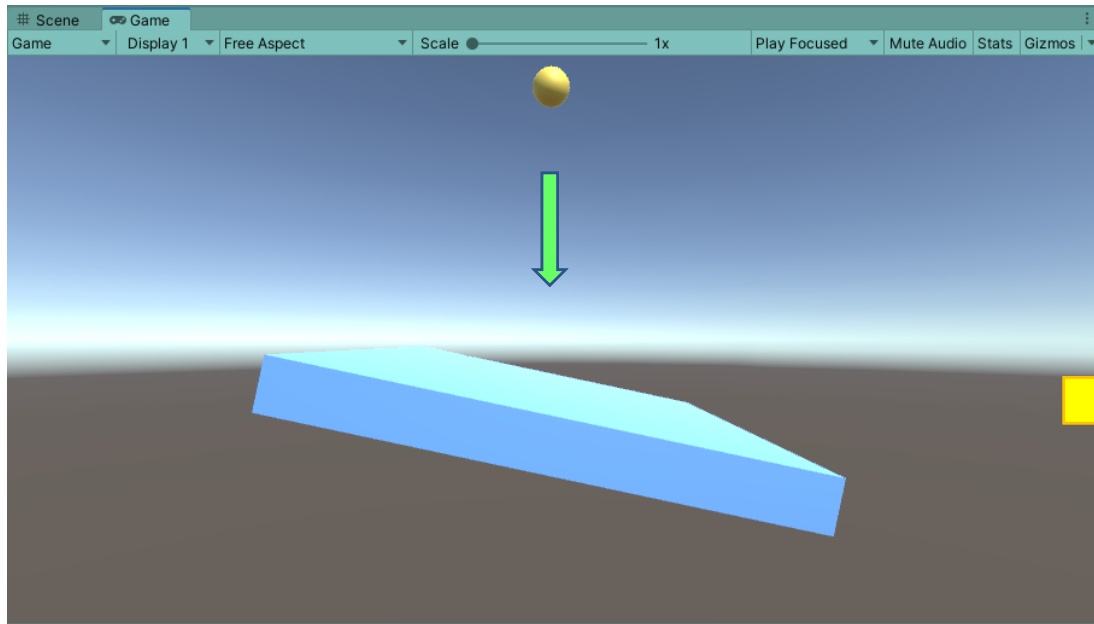
「フロア」(Cube)を選び、回転コマンドを使って傾けます
表示された円状の曲線を、マウスの左ボタンでドラッグして傾けることができます

フロアを傾けて「ボール」を転がします



カメラ位置なども、ボールが良く見える場所に調整します

「プレイ」ボタンで「ボール」を転がしてみます

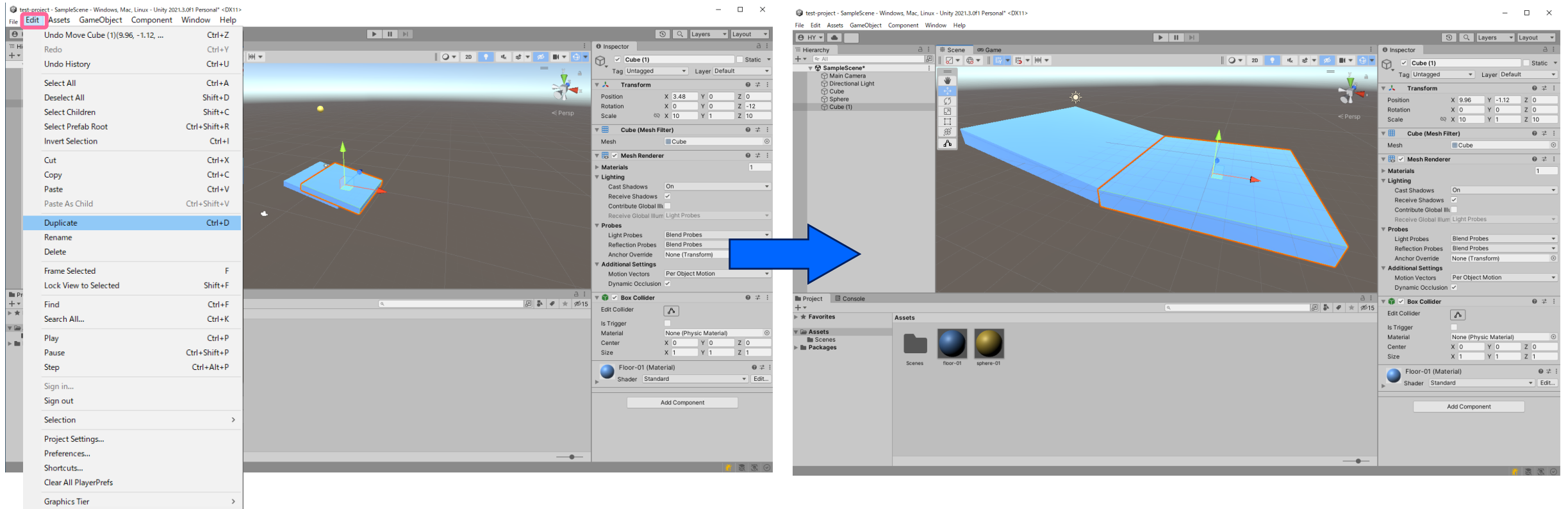


これを使ってコースをつくります

うまく転がるように
フロアを並べます

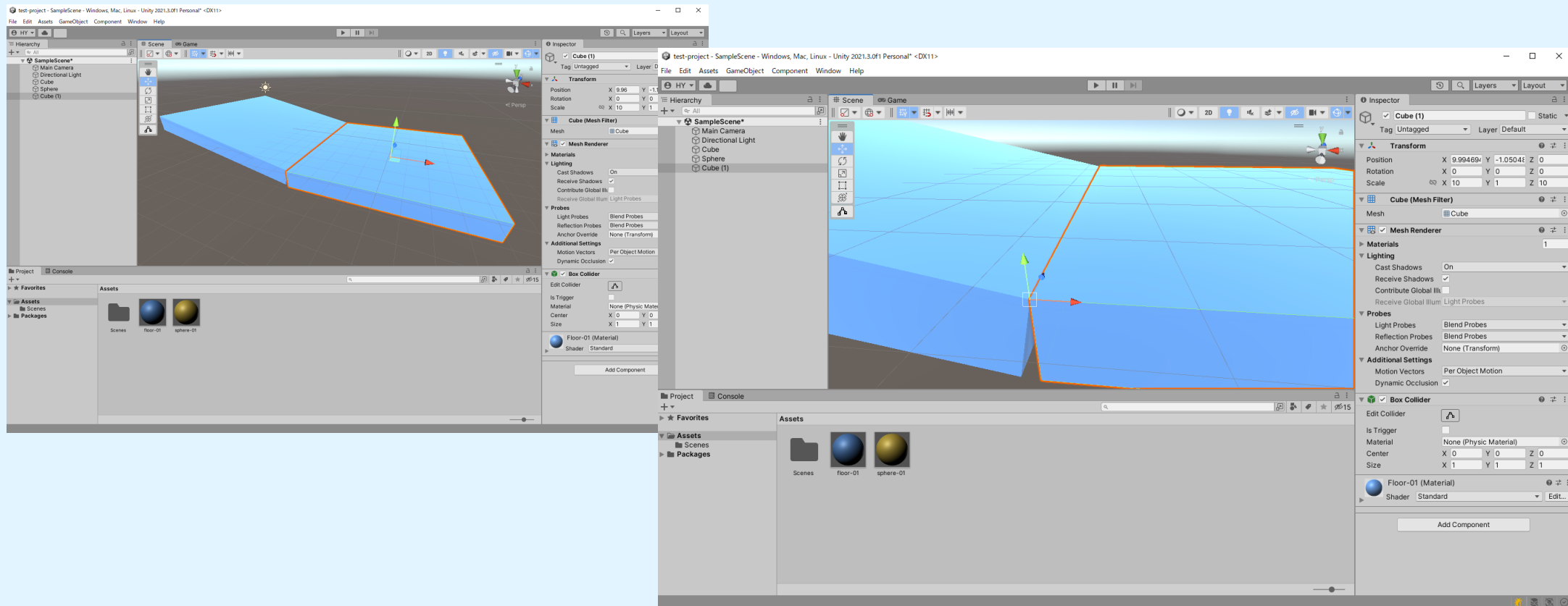


フロアを増やして「コース」をつくります



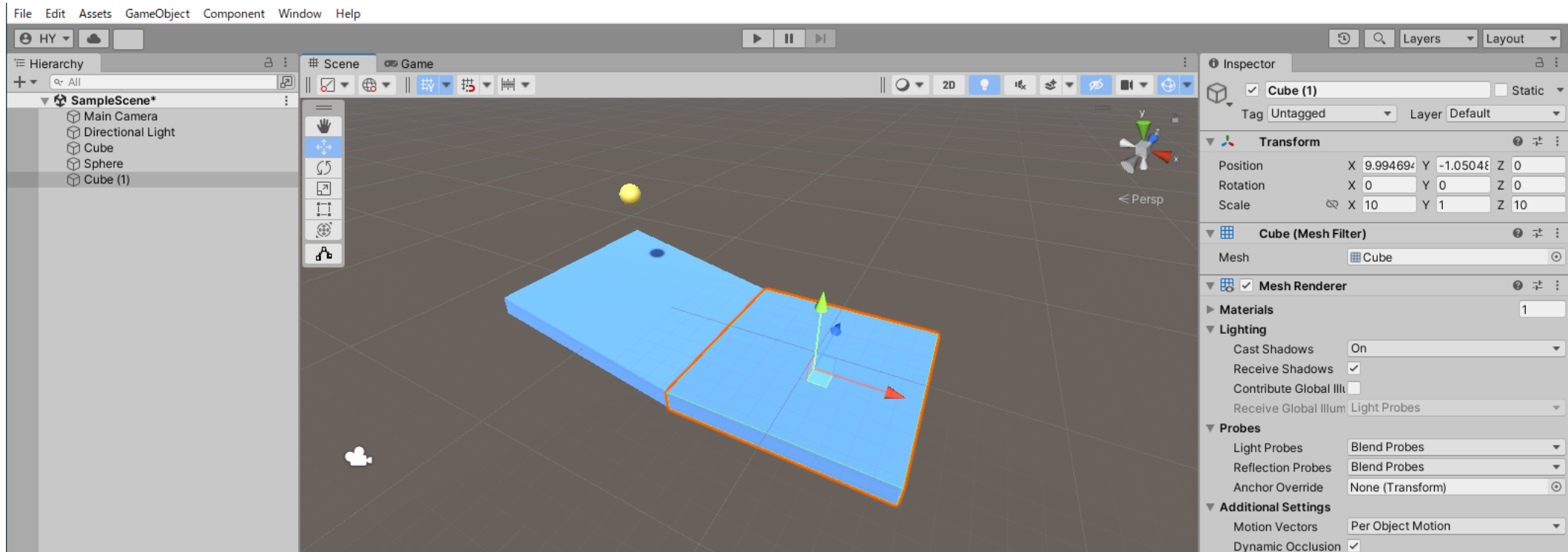
フロアのGameObjectを選択して、 Edit>「Duplicate」 で複製して位置を調整します
複製は(“Ctrl+D”でも行うことができます)

便利な「V」を押しながらのセット



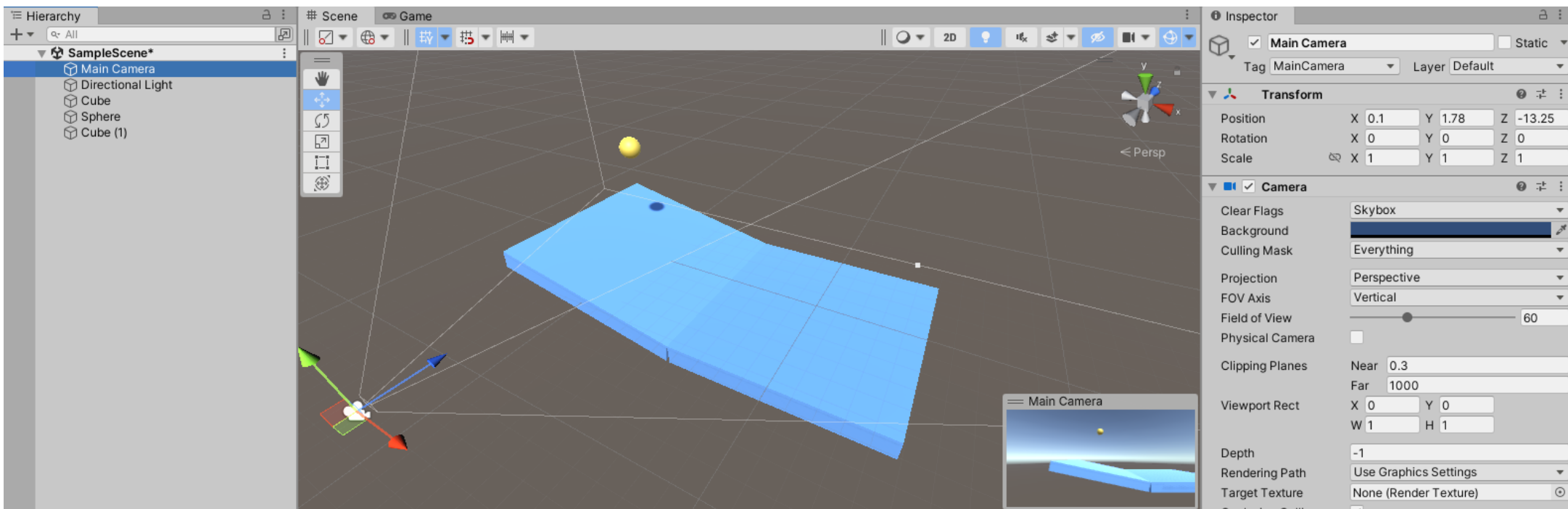
キーボードの「V」を押しながら移動させると、オブジェクト同士の角と角をピッタリと合わせることができます

コース全体が見えるように「カメラ」を合わせます



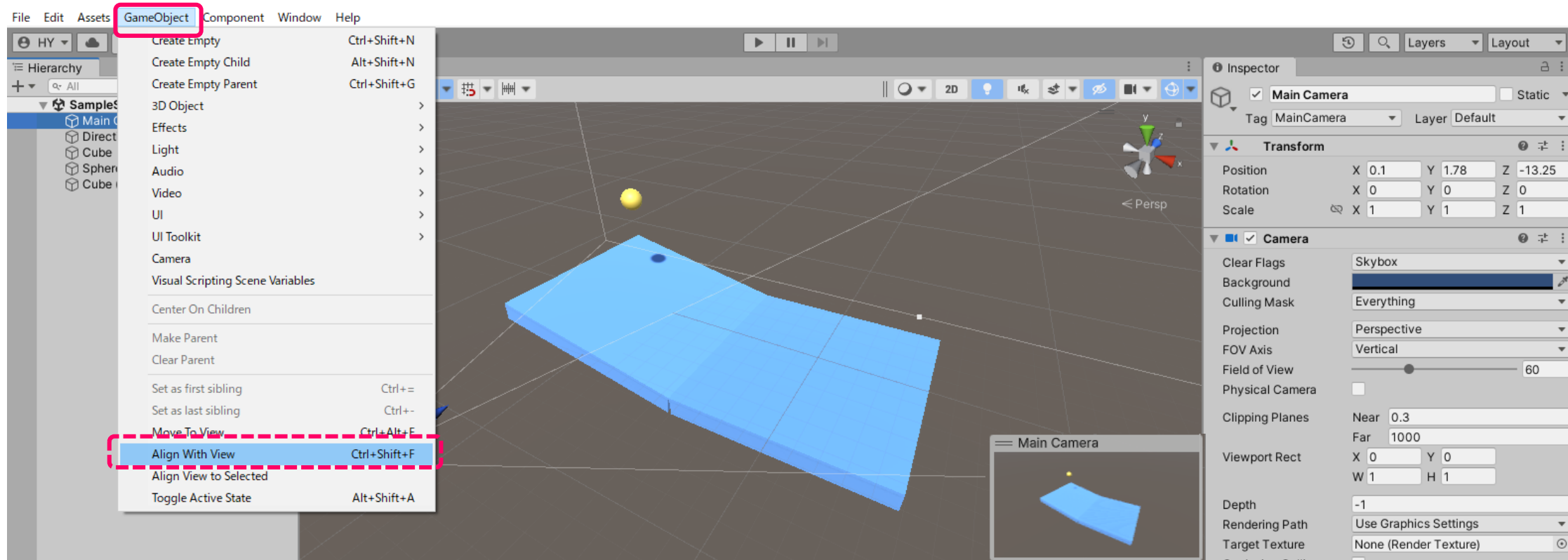
コース全体が入るように画面を移動します

「カメラ」をエディタビューに合わせる方法



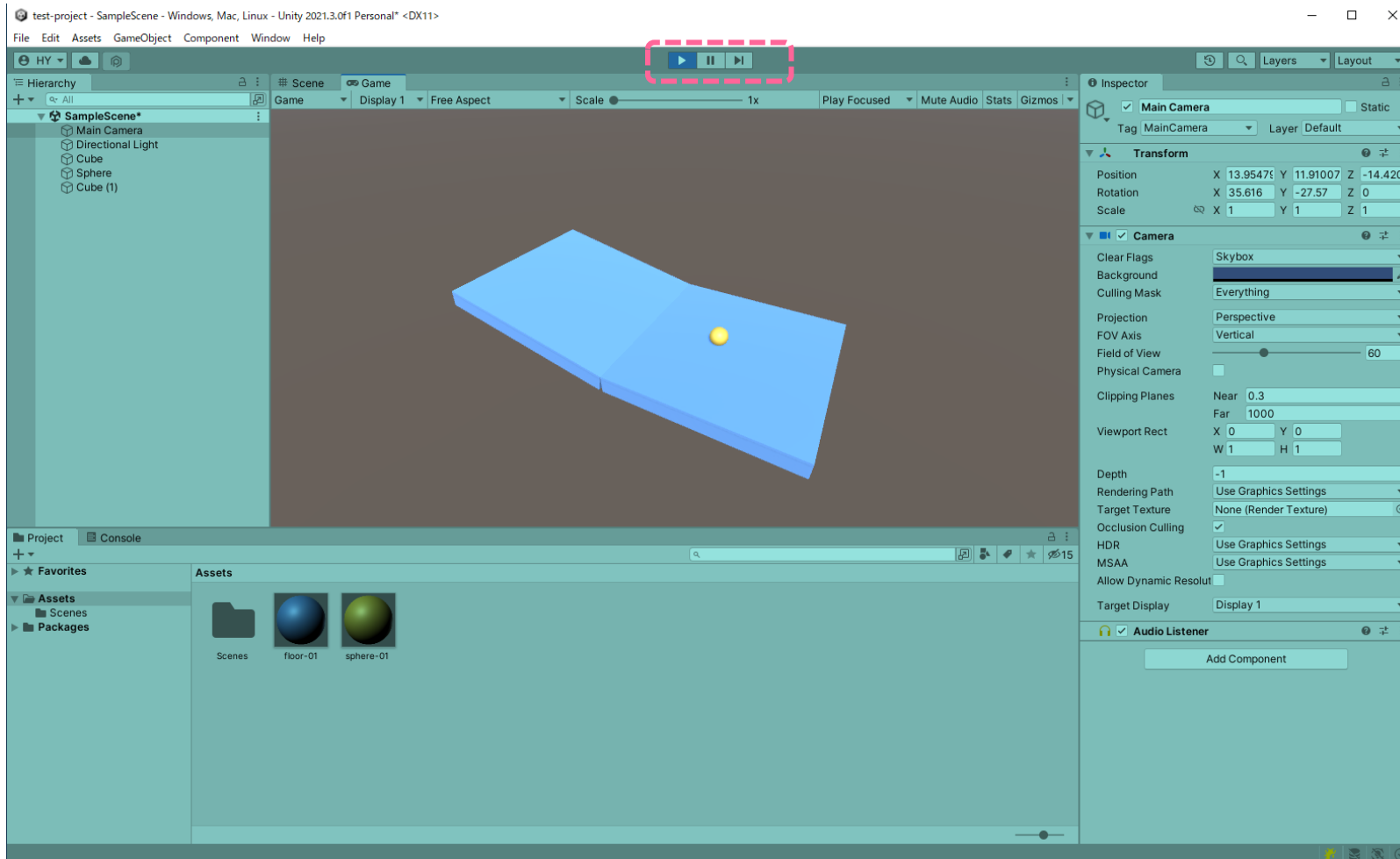
Hierarchyの「Main Camera」を選びます

プレイ画面をエディタービューに一致させます



そのまま「GameObject」>「Align With View」を選びます
「MainCamera」の位置が、エディタービューと同じ位置に移動します

「プレイ」ボタンを押して確認します



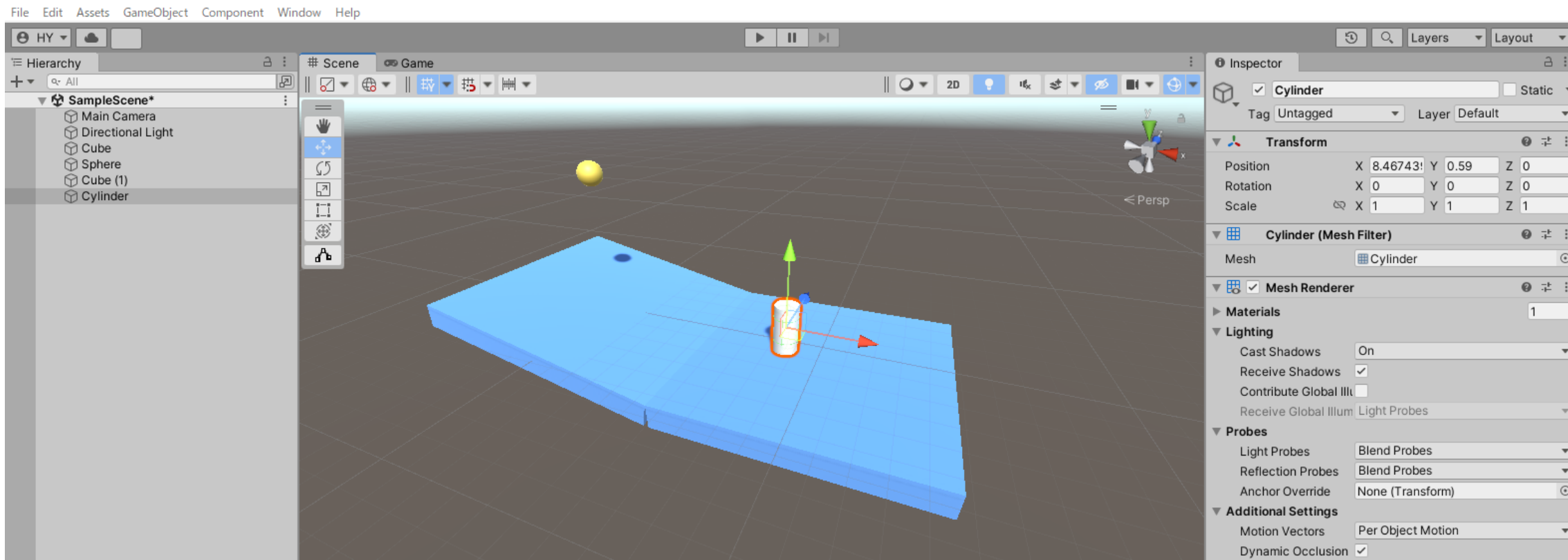
プレイ画面がコース全体を写しているか確認します

コースに障害物をセットします

転がるボールに
「ストーリー」を加えます

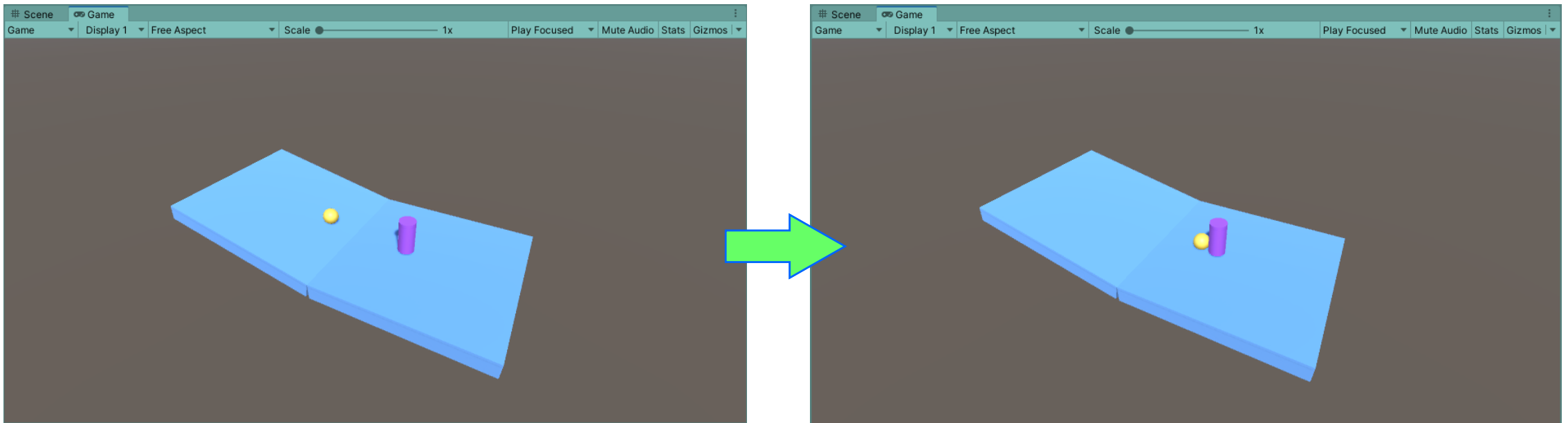


ほかのGameObjectをセットします



GameObject>3D Object>Cylinder で「円柱(シリンダー)」をセットします
そして同じく Assets > Create > Material と開いて「色」をつけます

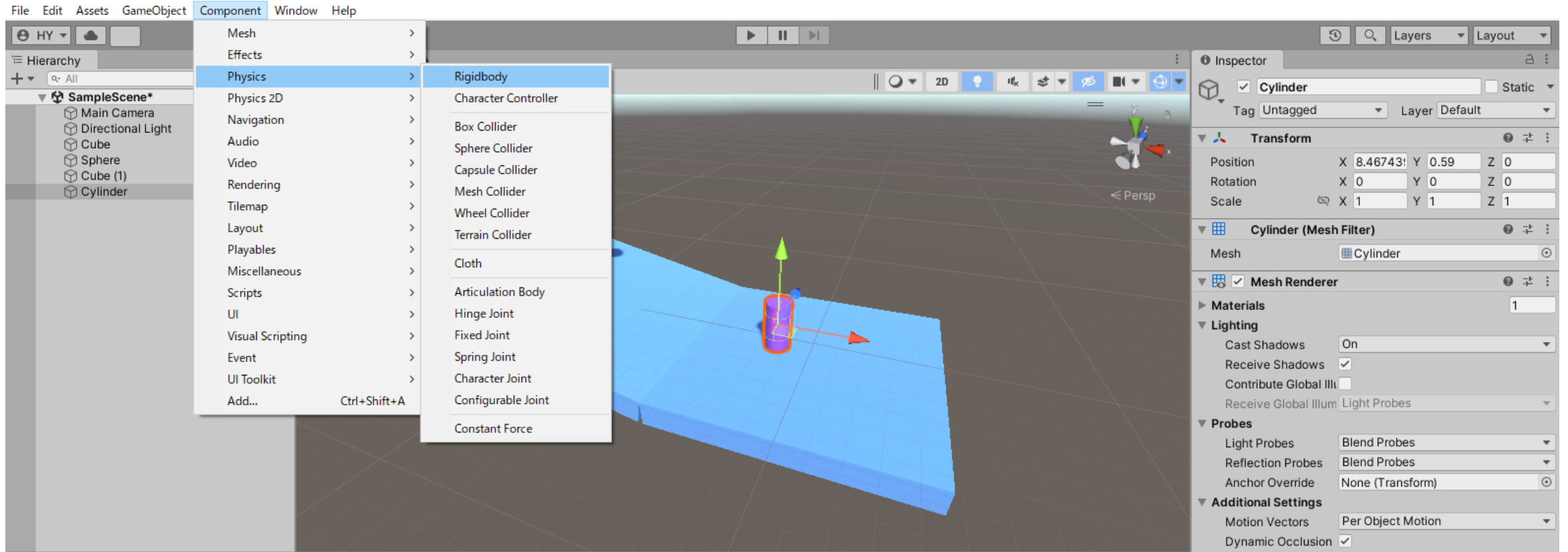
「プレイ」を押してみます



円柱にあたると、ボールは止まります

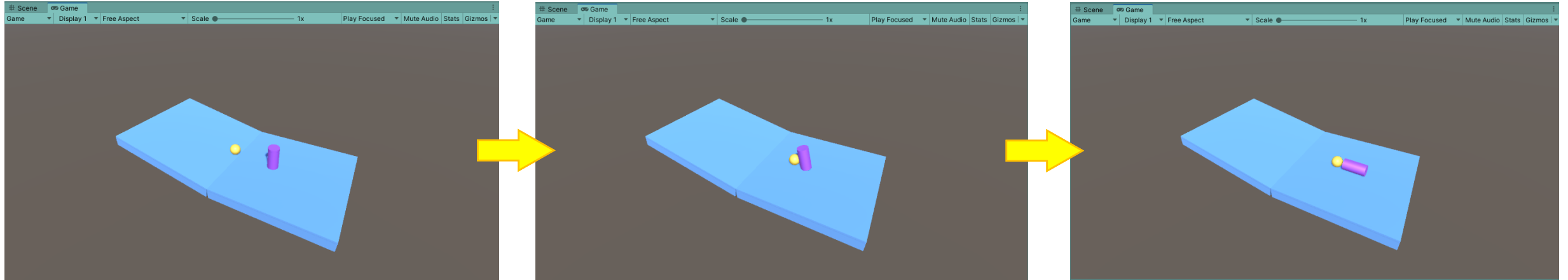
円柱に「Rigidbody」をつけて物理特性をあたえます

「円柱」にRigidbodyを付けます



Component > Physics > Rigidbody で
「円柱」にRigidbodyコンポーネントを付けて「物理法則」が働くようにします

「プレイ」を押してみます



今度はボールが当たった「円柱」は倒れるようになります

「Rigidbody」をつけるとObjectには「物理法則」が働くようになります

これで基本の操作は完了です

基本の操作は
以上です



基本の操作のまとめです

マウス・キーボード操作



画面の移動「Q」、Objectの移動「W」、回転「E」、拡大縮小「R」、カメラ(視点)の移動「Alt」のキーに対応しています

クリエイトできるオブジェクトの種類、配置、エディットの方法

GameObject > 3D Object でオブジェクトをつくる Inspector欄から数値を代入する変形の方法

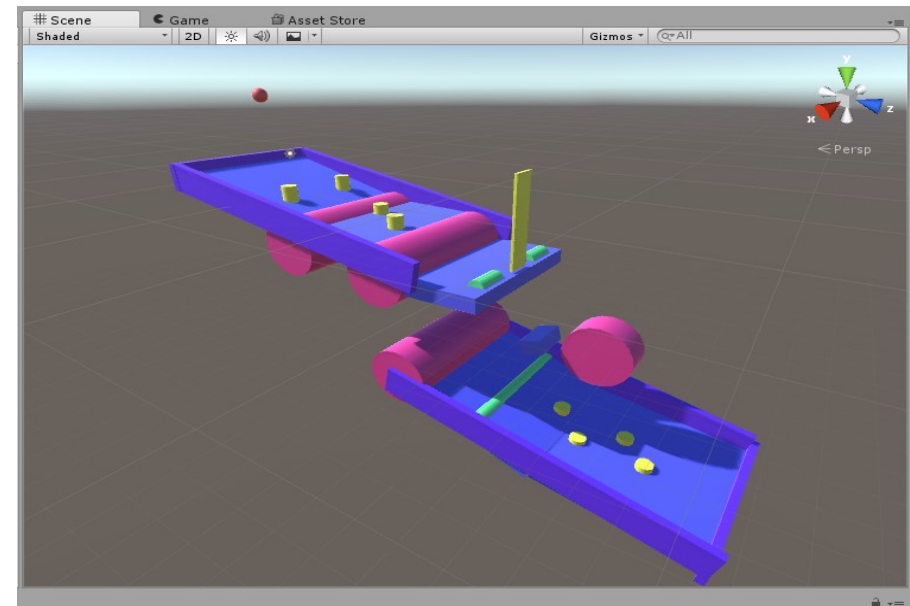
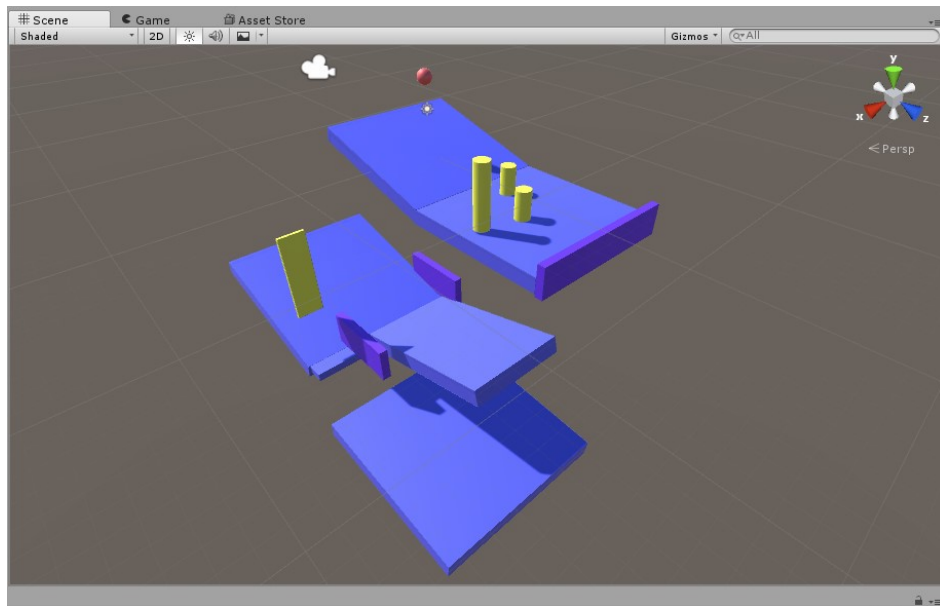
そのほかにもっと詳しく学びたいときのために
「Unity Manual」のweb参照ページ

<https://docs.unity3d.com/Manual/PrimitiveObjects.html>

<https://docs.unity3d.com/Manual/UnityHotkeys.html>



第2回の課題 球を転がしてゴールまで行くコースを作る



1、はじめにやってみるデザインのヒント

- いろいろな形のGameObjectを作って配置します
- ボーリングのようにオブジェクトを倒して転がり続けさせます
- 細く狭い場所をうまく転がして通り抜けさせます
- 単純に凸凹道を転がる挙動をたのしみます
- 倒したオブジェクトを利用します
- 違う形のものや複数のボールを転がします・・・などなど

ためしながら楽しみながら
コースをつくりましょう



2、初心者からステップアップするためのデザインのヒント

- この「コース」ではボールが「プレイヤー(自分)」だと考えます
- ボールが倒すオブジェクトは「エネミー」です
- 「ボール(自分)」の目的(ゴール)は何かを考えます
- ボールになった自分が「やりたいこと」をイメージしましょう
- コースは自分が冒険する一つの「ストーリー」だと考えます

- ボールは「プレイヤー」、当たるオブジェクトは「エネミー」です
- オブジェクトに「当たって止まった = ボールはやられた」、と考えてみます
- はね返されても、倒し続けてボールを「ゴール」までたどり着かせるコースを作りましょう

転がるボールに「冒険」や「チャレンジ」を感じさせるコースなら、
きっと見る人をたのしい気持ちにさせることができます

これは物語りの登場人物に観客を感情移入させるテクニックです
「ストーリーを感じさせる」

映画やマンガ、アニメを見てしまうPassive(受け身)なコンテンツの
メカニズムと同じものです

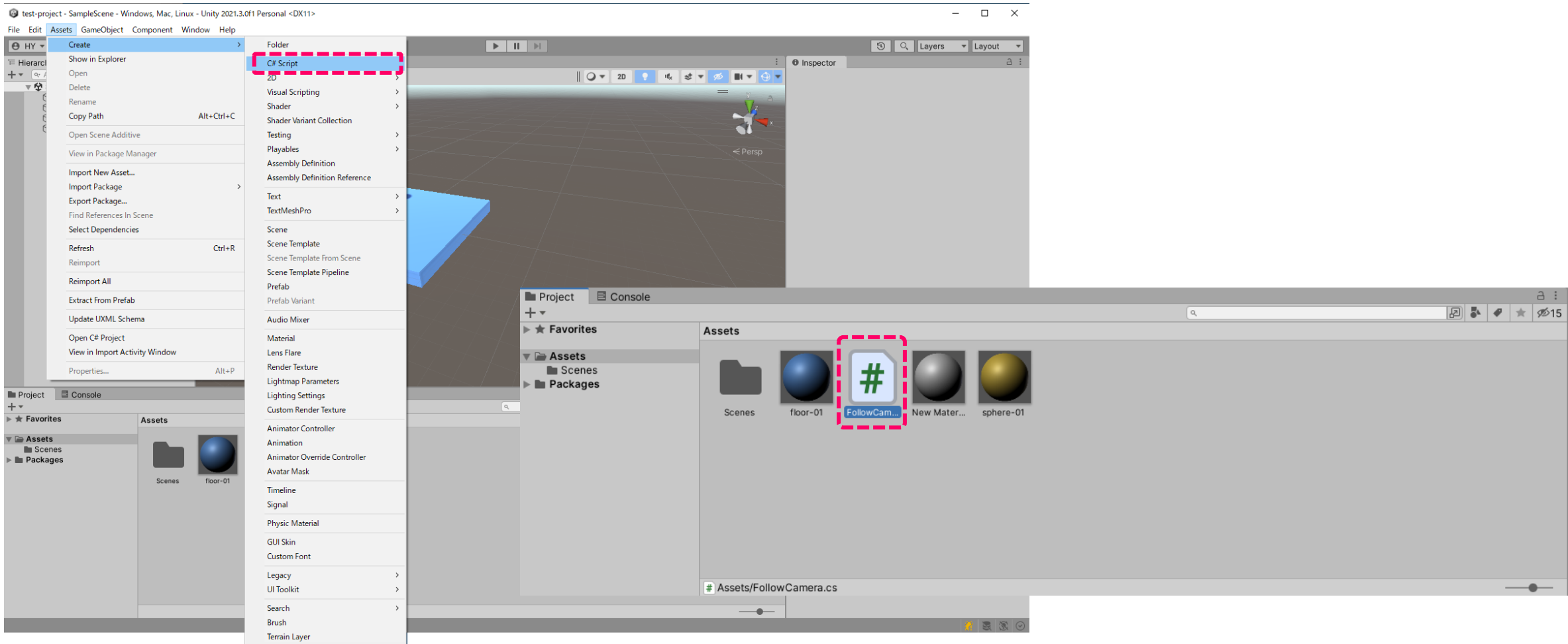
おまけ

もしも、レベルが長くなり過ぎた場合

ボールを追いかける
カメラを作ります

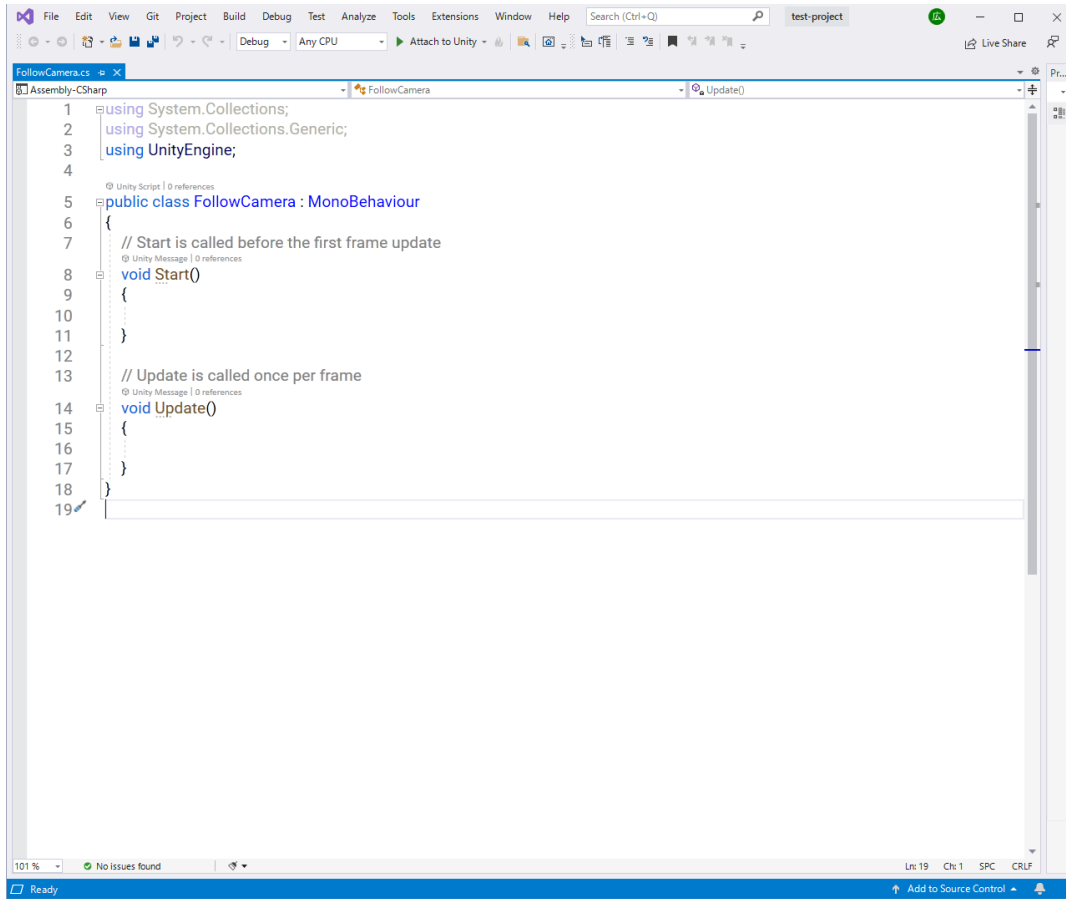


C#スクリプトを作ります

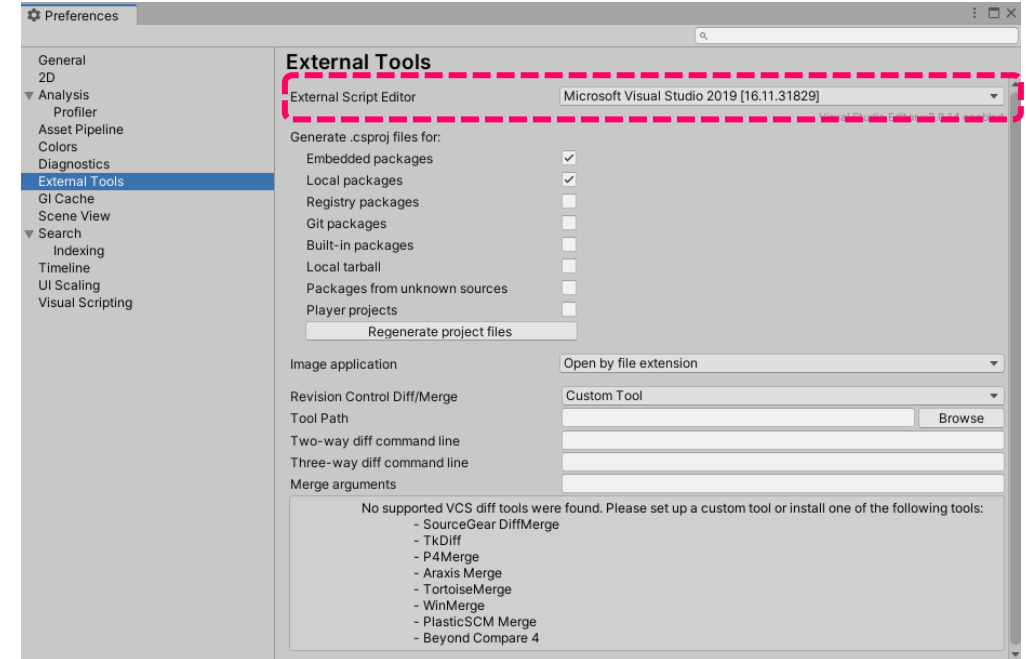


Assets > Create > C#Script で、C#スクリプトを作ります
Assets欄にC#ファイルができるので、その時忘れずに「ファイル名前」を付けます

作ったC#ファイルをダブルクリックすると、
VisualStudio(スクリプトエディター)が開きます



```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class FollowCamera : MonoBehaviour
6 {
7     // Start is called before the first frame update
8     void Start()
9     {
10
11     }
12
13     // Update is called once per frame
14     void Update()
15     {
16
17     }
18 }
19
```



通常は Edit > Preferences > ExternalTools のなかにある
External Script Editorが「Microsoft Visual Studio 2019[***]」になっています

FollowCameraという名前で 新しいクラスを作りました

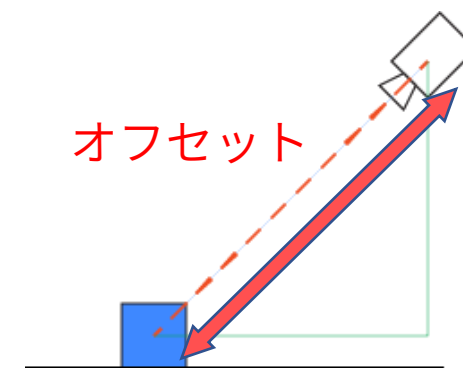
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class FollowCamera : MonoBehaviour
{
    public GameObject player; //プレイヤーとなるオブジェクトを格納する変数playerを用意します
    private Vector3 offset; //Vector3型の変数offsetを用意します

    void Start()
    {
        //プレイヤーの位置とカメラの位置同士を引いて、両者の距離を変数offsetに格納します
        offset = this.transform.position - player.transform.position;
    }

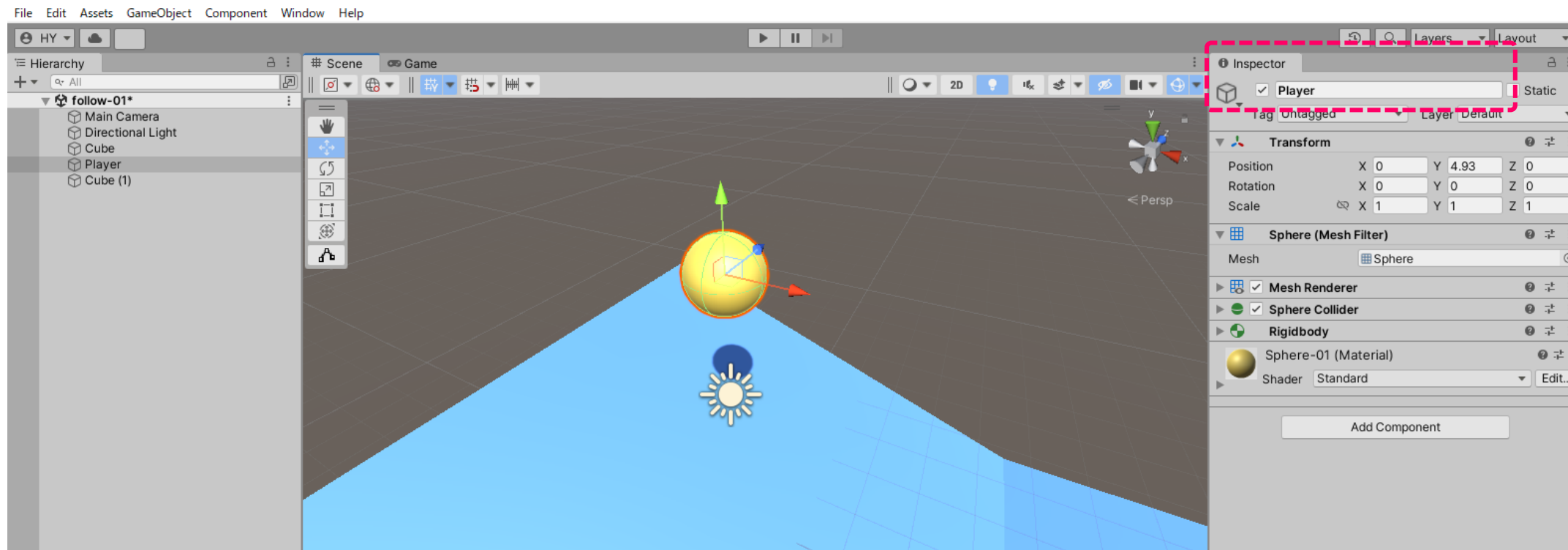
    void Update()
    {
        //object名が"Player"のオブジェクトを探して、変数playerに格納します
        player.transform.position = GameObject.Find("Player").transform.position;

        //このオブジェクトの位置を、プレイヤーの位置と変数offsetの位置を足したものにします
        this.transform.position = player.transform.position + offset;
    }
}
```



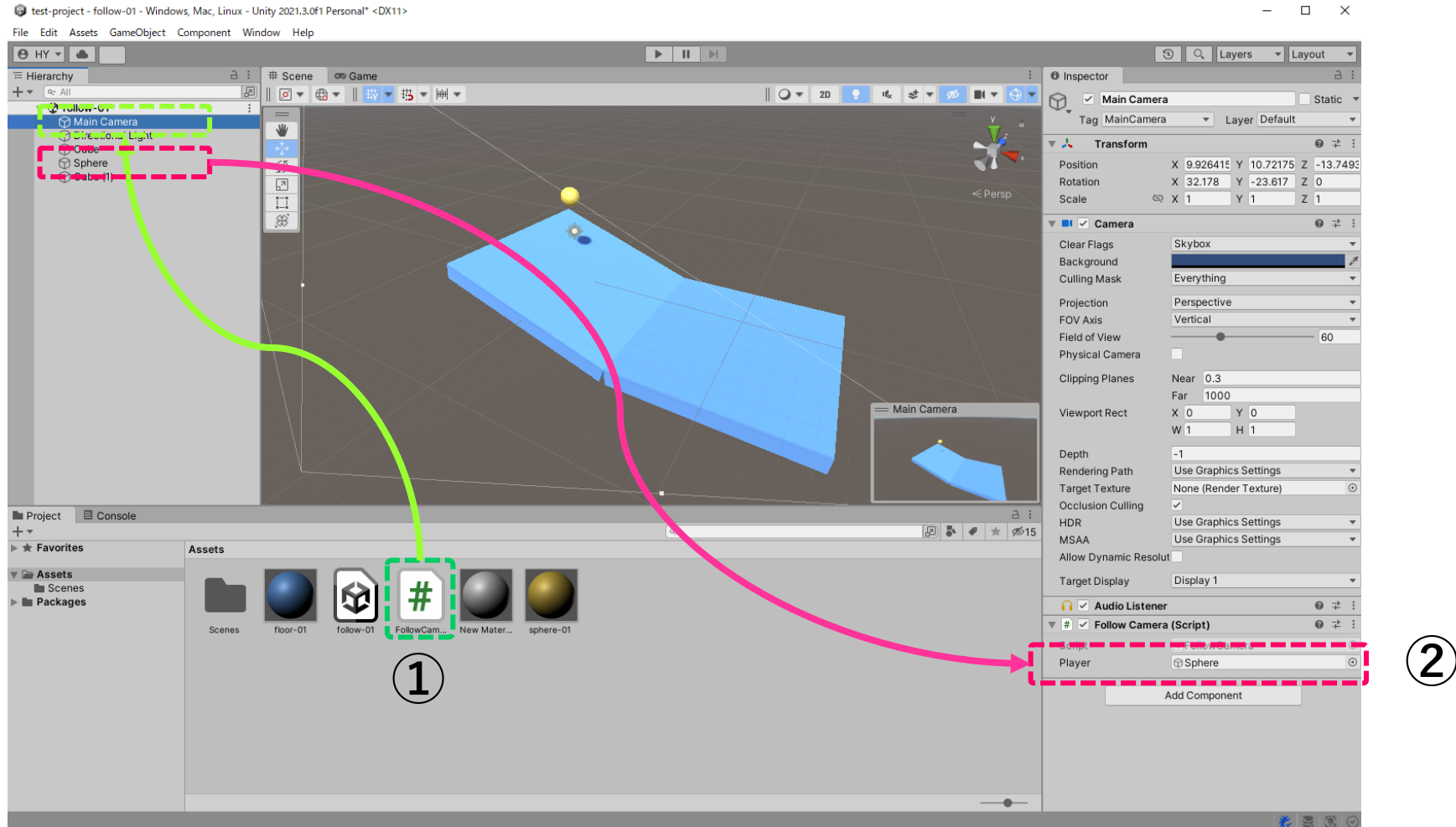
カメラとプレイヤーの距離を
各座標の引き算で求めたものが
offset (オフセット) の値です

球体のTagを”Player”にします



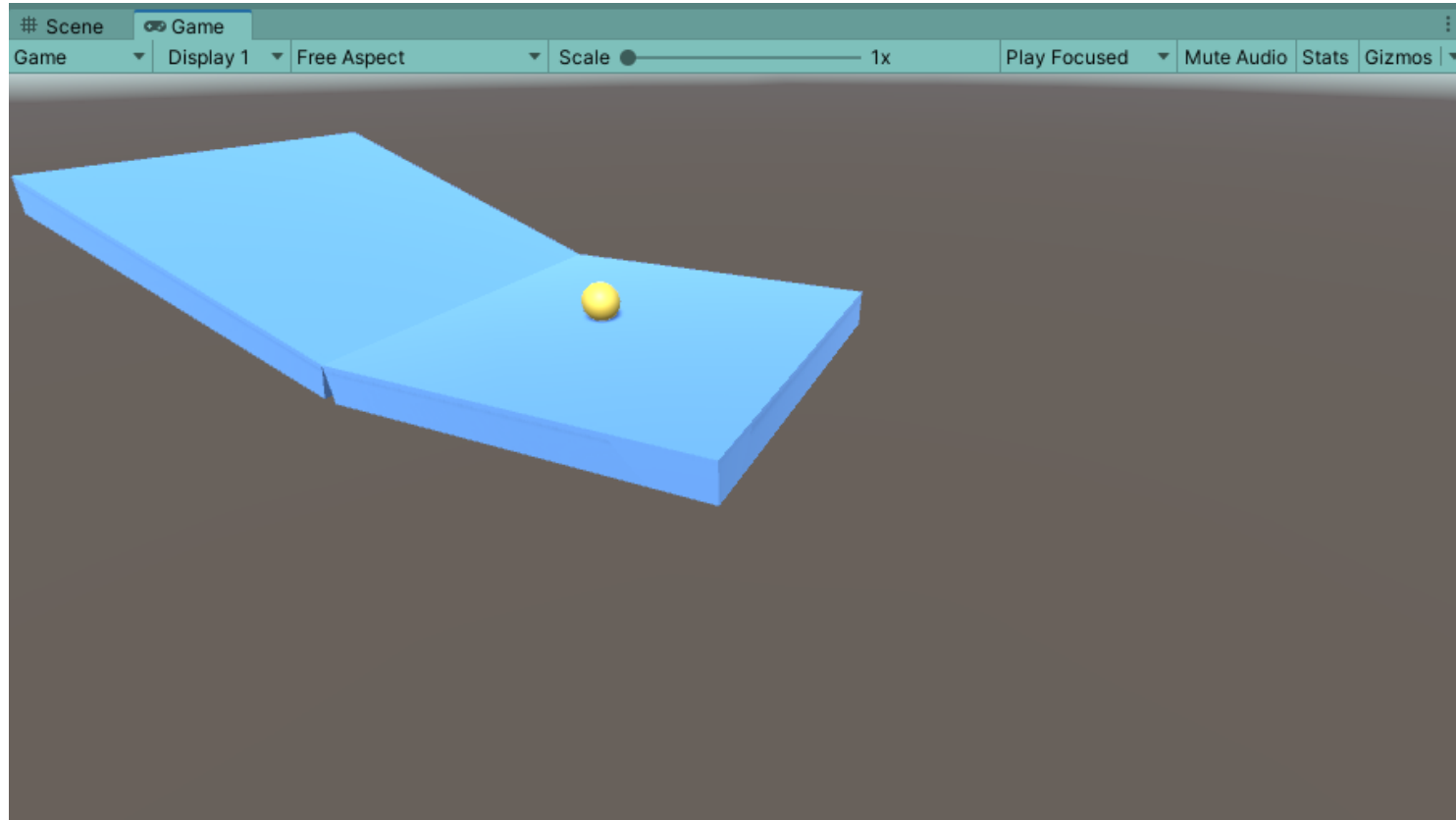
球体をクリックして、名前を「Player」にします

MainCameraにFollowCameraスクリプトを入れます



- 1、 MainCameraに「FollowCamera」スクリプトをいれると、Inspector欄に、スクリプトが表示されます
- 2、 FollowCameraスクリプトの「Player」欄に「Sphere (球体)」をドラッグして、「Player」にセットします

プレイしてカメラが球に追従することを確認します



カメラの位置を自分のレベルが見やすいよう調整します

おまけ

さまざまな外部ツールの利用方法

PackageManagerから
さまざまなツールを
インストールできます



Package Manager

test-project - 01-exam-01 - Windows, Mac, Linux - Unity 2021.3.0f1 Personal <DX11>

File Edit Assets GameObject Component Window Help

Panels

- Next Window Ctrl+Tab
- Previous Window Ctrl+Shift+Tab
- Layouts
- Search
- Plastic SCM
- Collaborate
- Asset Store
- Package Manager**
- Asset Management
- Text
- TextMeshPro
- General
- Rendering
- Animation
- Audio
- Sequencing
- Analysis
- AI
- UI Toolkit
- Visual Scripting

Package Manager

Packages: Unity Registry

- Unity Registry 12.1.6
- In Project 4.1.4
- My Assets 1.3.0
- Built-in 1.0.0

Package Name	Version
Magic Leap XR Plugin	6.4.1
Mathematics	1.2.6
ML Agents	2.0.1
Mobile Notifications	2.0.0
Oculus XR Plugin	3.0.1
OpenXR Plugin	1.3.1
Polybrush	1.1.3
Post Processing	3.2.2
ProBuilder	5.0.4
Profile Analyzer	1.1.1
Recorder	3.0.3
Remote Config	2.1.2
Scriptable Build Pipeline	1.19.6
Sequences	1.0.4
Shader Graph	12.1.6
Terrain Tools	4.0.3
Test Framework	1.1.31
TextMeshPro	3.0.6
Timeline	1.6.4
Tutorial Authoring Tools	1.0.2
Tutorial Framework	2.1.2
Unity Distribution Portal	2.2.5
Unity Profiling Core API	1.0.2
Unity UI	1.0.0
Universal RP	12.1.6
Version Control	1.15.15
Visual Effect Graph	12.1.6
Visual Scripting	1.7.6
Visual Studio Code Editor	1.2.5
Visual Studio Editor	2.0.14
WebGL Publisher	4.2.3
XR Plugin Management	4.2.1

ProBuilder Release

Unity Technologies
Version 5.0.4 - February 07, 2022
Registry Unity
[View documentation](#) · [View changelog](#) · [View licenses](#)

Build, edit, and texture custom geometry in Unity. Use ProBuilder for in-scene level design, prototyping, collision meshes, all with on-the-fly play-testing.

Advanced features include UV editing, vertex colors, parametric shapes, and texture blending. With ProBuilder's model export feature it's easy to tweak your levels in any external 3D modelling suite.

If you are using URP/HDRP, be careful to also import the corresponding sample project in the section below to get the proper materials.

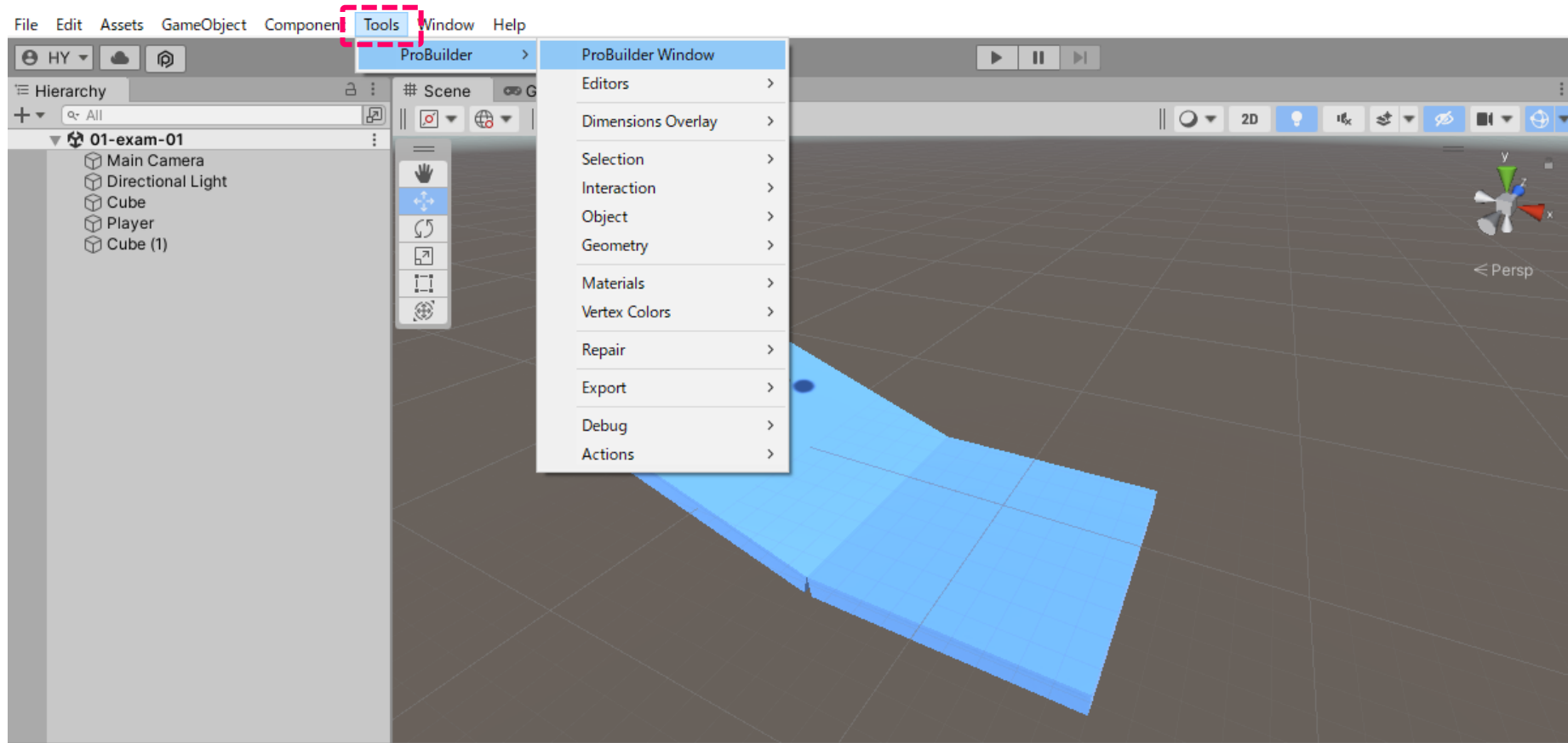
「Pro Builder」を選んでみます

インストールボタンを押します

Install

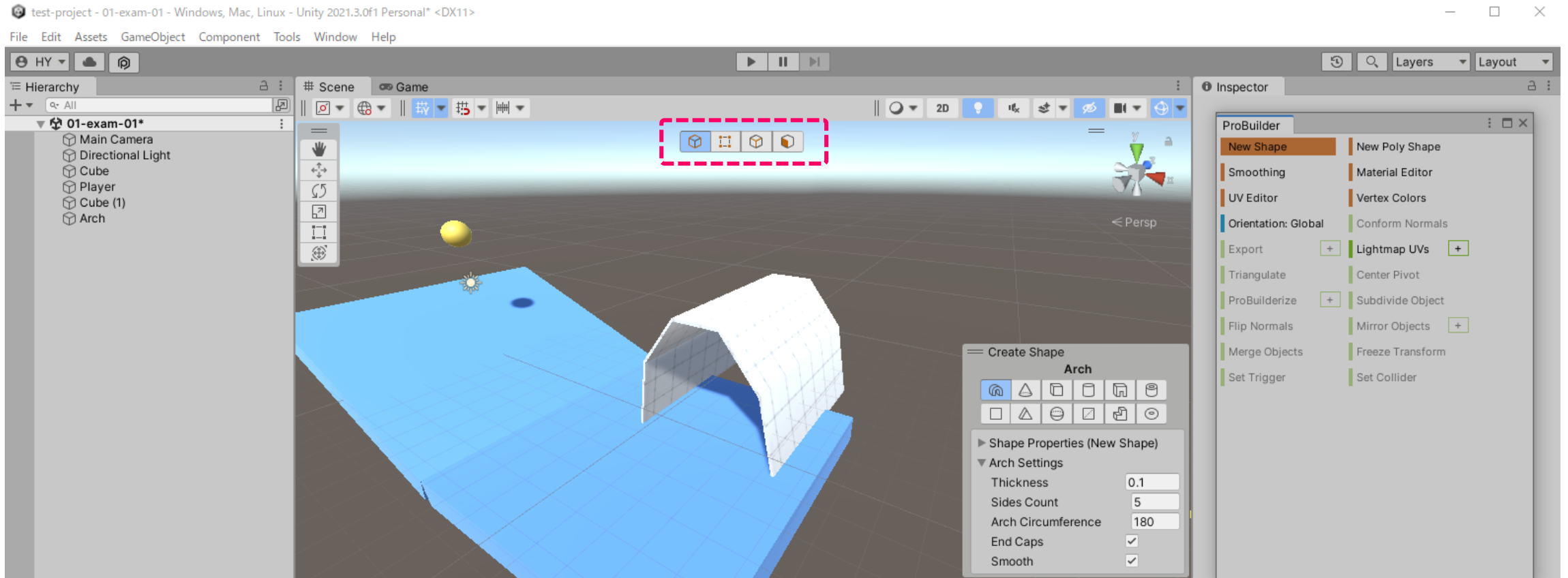
Window > Package Manager で、ウィンドウが開きます
左上の「Packages」を「UnityRegistry」にするとインストール
可能なたくさんのツールが表示されます

インストールが完了すると「Tools」がメニューに加えられます



Toolsメニューのなかに「ProBuilder」が入ります

さまざまな図形を使ったレベルを作成することができます



「Pro Builder」はとても強力なモデリングツールです
<https://www.youtube.com/watch?v=HUeQeSB45PI>