

Unity3Dブロック崩し

角田研究室

奥崎弥

目次

- ・自己紹介
- ・Unity概要
- ・ゲーム作成

自己紹介

名前: 奥崎 弥(おくざき ひろ)

大学: 青森大学ソフトウェア情報学部4年生

趣味: 旅行

好きなアーティスト: TOMORROW X TOGETHER

クイズ(Forms)

10分くらい

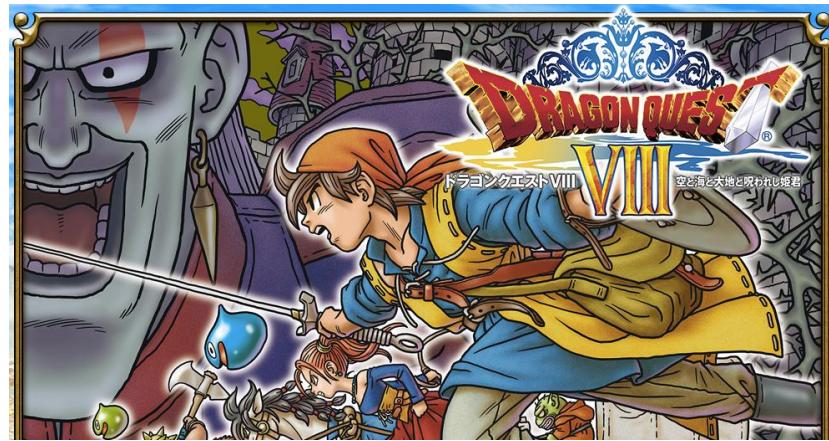
Unity概要

- ・活用事例
- ・Unityで開発されたゲーム例
- ・比較 Unreal Engine(アンリアル・エンジン)

Unityでできることは主に3つ

- ・ゲーム開発
- ・XRコンテンツ開発
- ・アプリ開発

Unityで開発されたゲーム例



XRコンテンツ開発

VR



AR



MR

XRとはVR・AR・MRの総称

他の活用事例

- ・自動車産業

- 車のデザイン設計にVR

- ・医療現場

- VRソフトでシミュレーション

- ・建築・不動産業

- 施工中の現場をバーチャル空間に再現するシステム

また宇宙産業、旅行業界、広告業界などもUnityを使用している。

比較 Unreal Engine(アンリアル・エンジン)



ゲーム作成

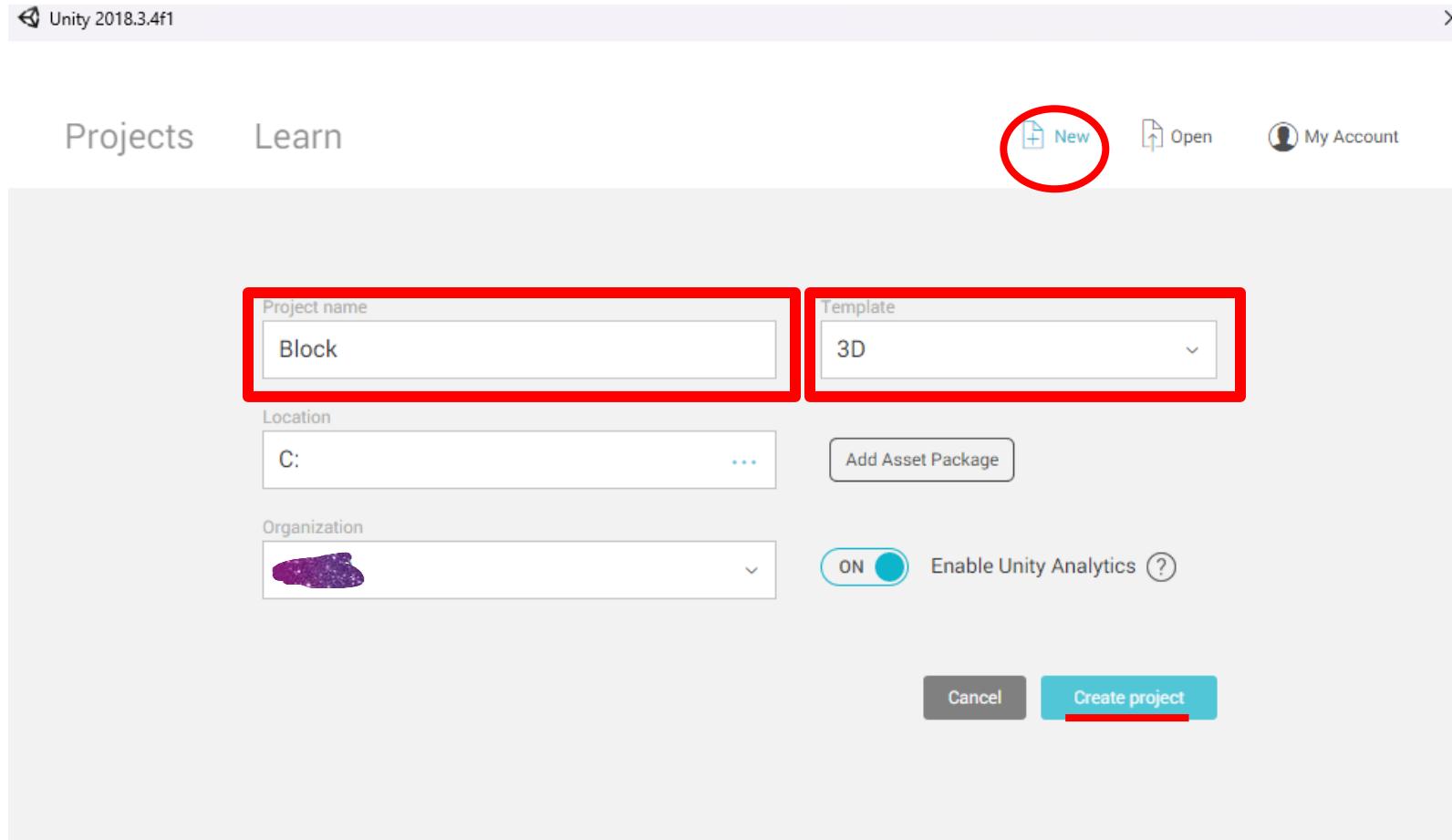
1日目

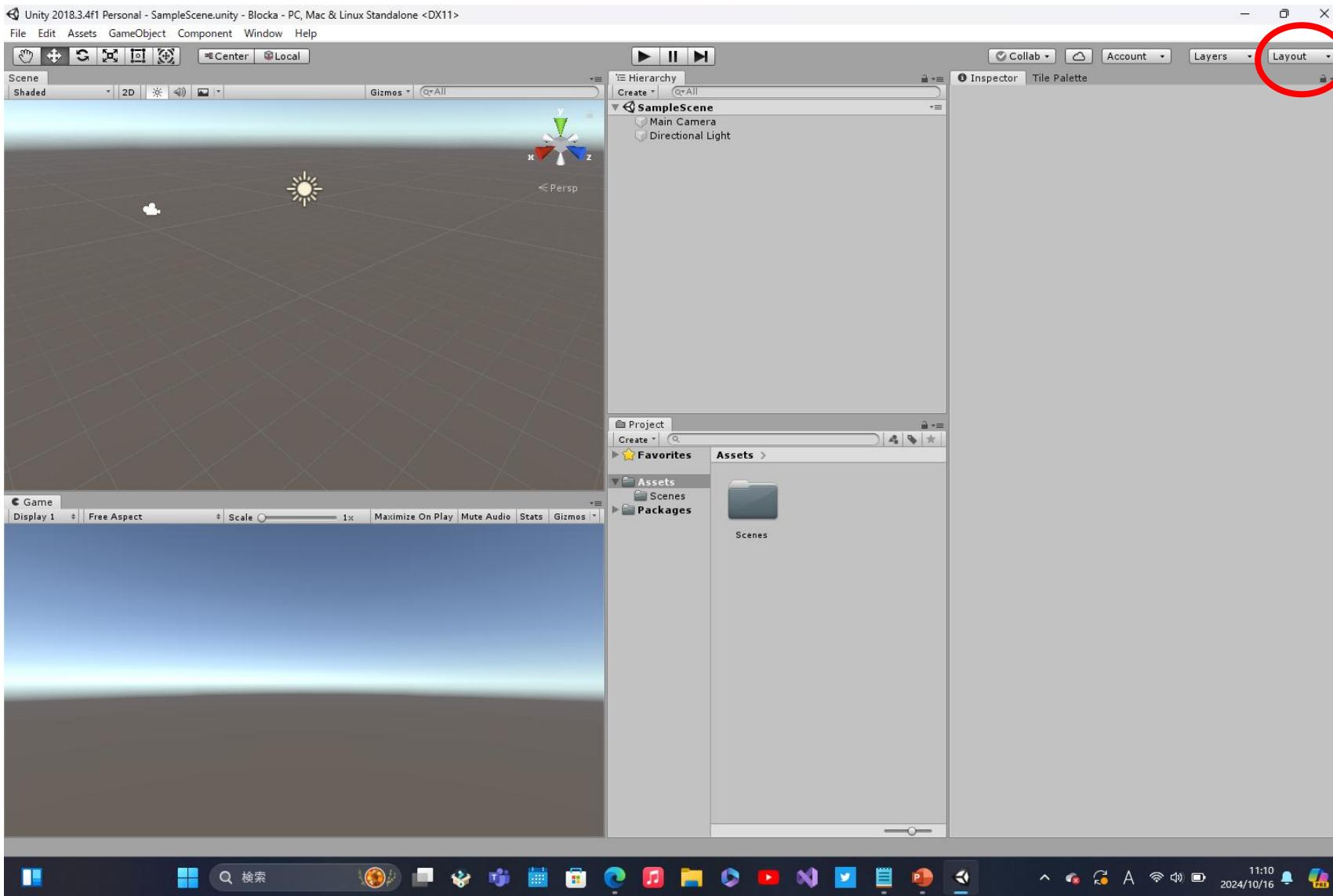
- Projectの新規作成
- シーンを保存
- Unity操作方法
- メインカメラの調整
- 壁の作成
- ボールの作成
- ボールに物理演算を加える

2日目

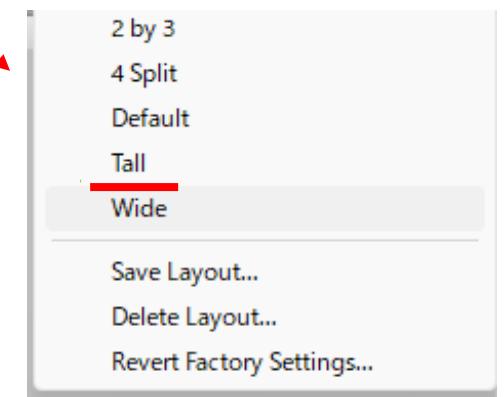
- ボールが動くスクリプト作成
- 正しく反射するように設定する
- パドル(バー)の作成
- ブロックの作成
- ゲームクリアの判定
- ゲームオーバーの判定
- テキストの設定
- マテリアルの設定(時間があったら)
- 自由制作

Projectの新規作成





このようになってない人



Scene
Shaded
2DGizmos
AllHierarchy
Create
SampleScene
Main CameraCollab
Account
InspectorLayers
Layout

シーンビュー

ヒエラルキー

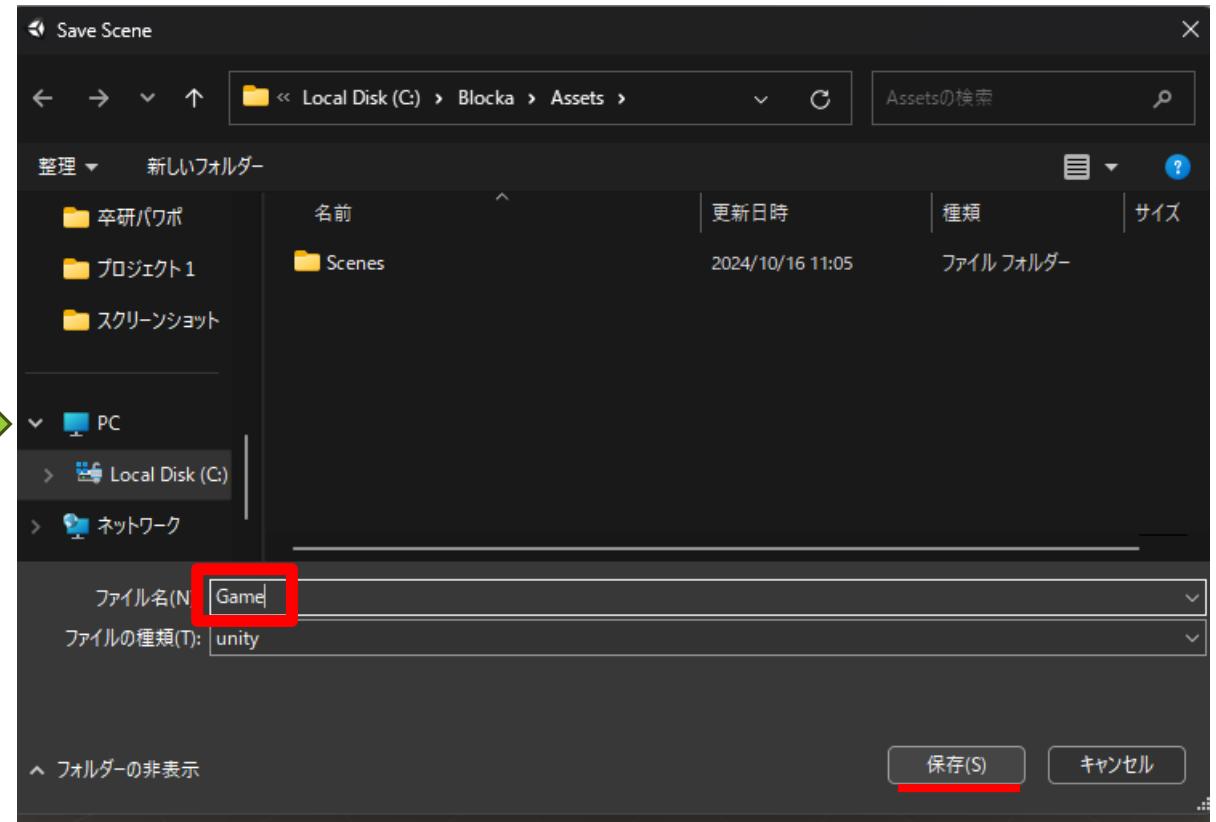
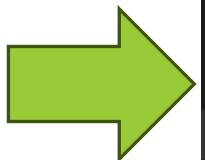
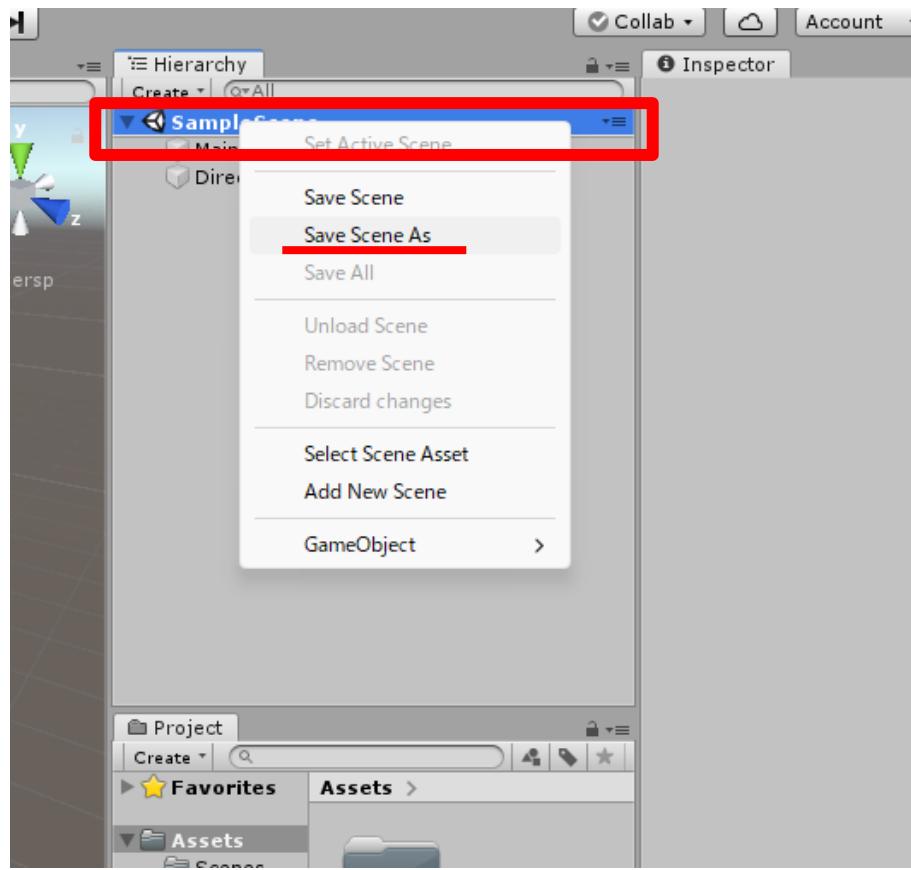
インスペクター

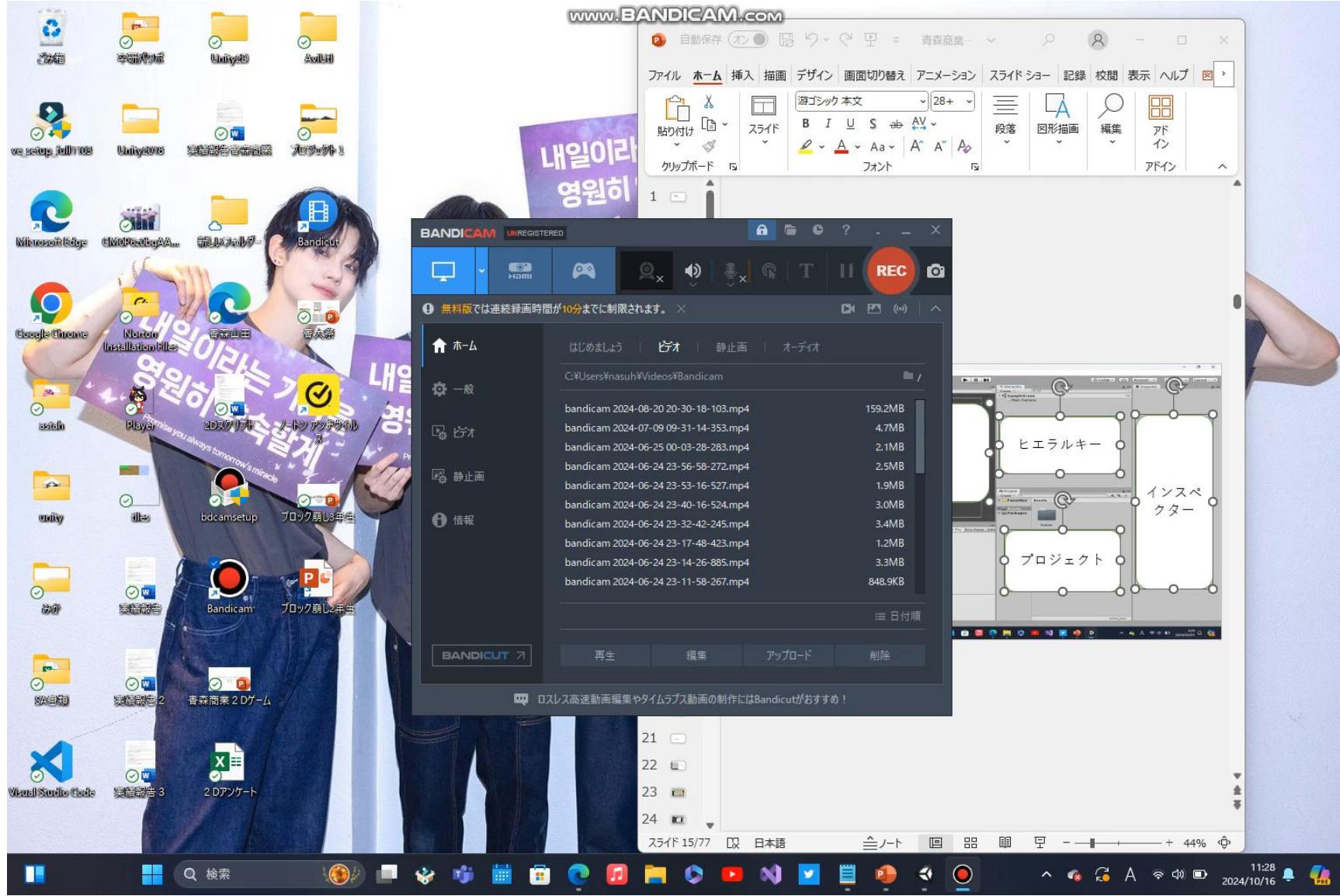
ゲームビュー

プロジェクト

Game
Display 1
Free Aspect
Scale 1x
Maximize On Play
Mute Audio
StopConsole
Clear
Collapse
Clear on Play
Error Pause
EditProject
Create
Favorites
Assets
Packages
Scenes

シーンを保存





PC、Unity操作方法

ショートカットキー

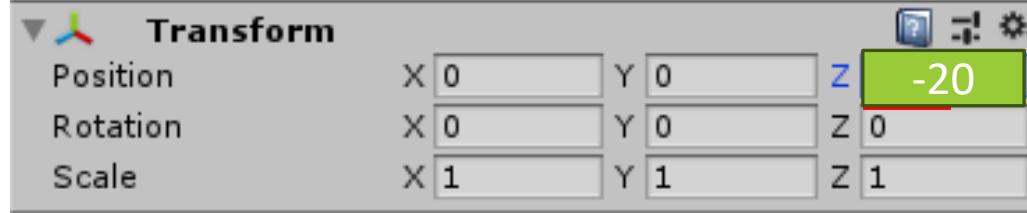
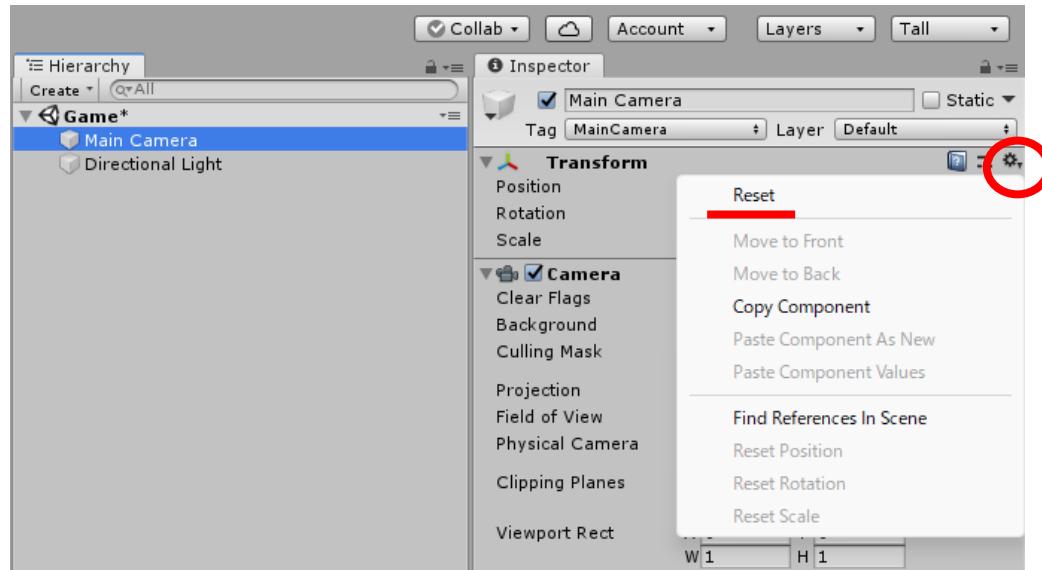
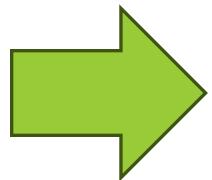
Ctrlキー+zキー ひとつ前の操作に戻る

Ctrlキー+sキー 保存

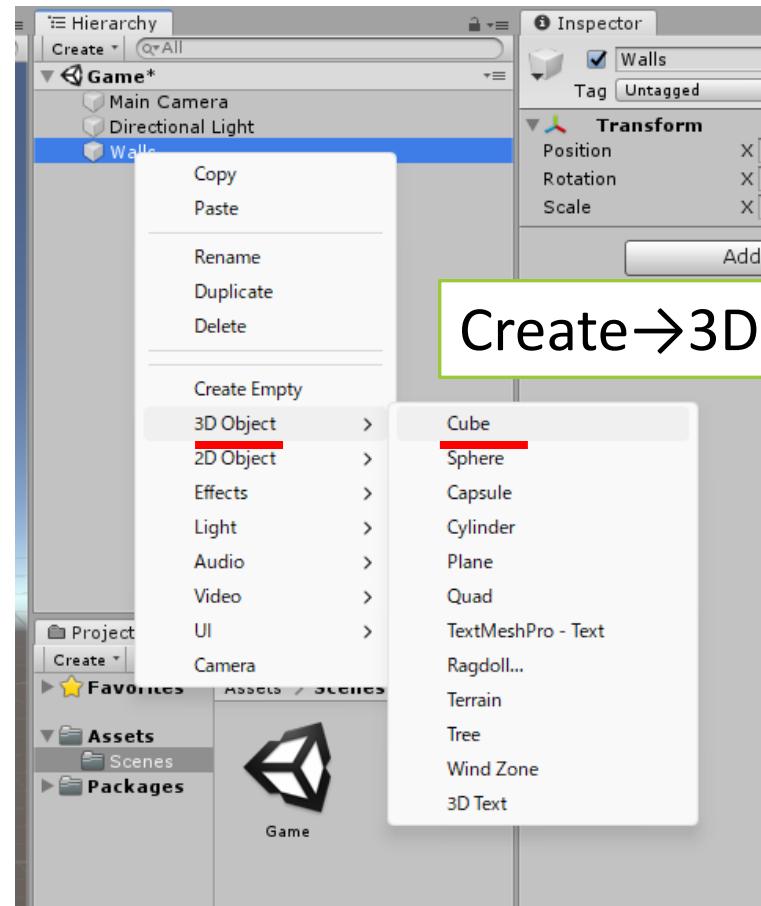
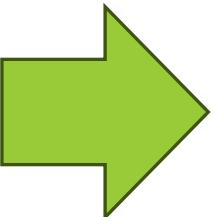
Ctrlキー+cキー コピー

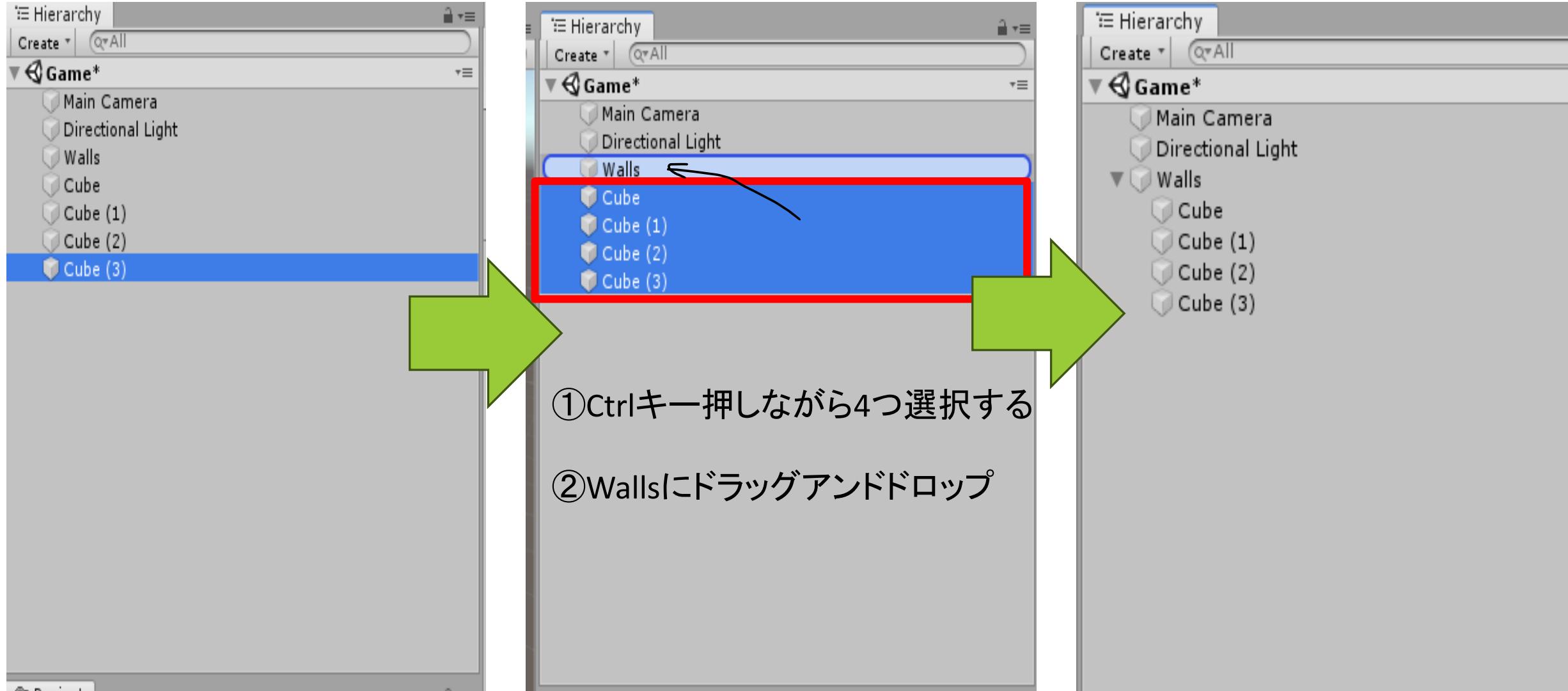
Ctrlキー+vキー 貼り付け

メインカメラの調整

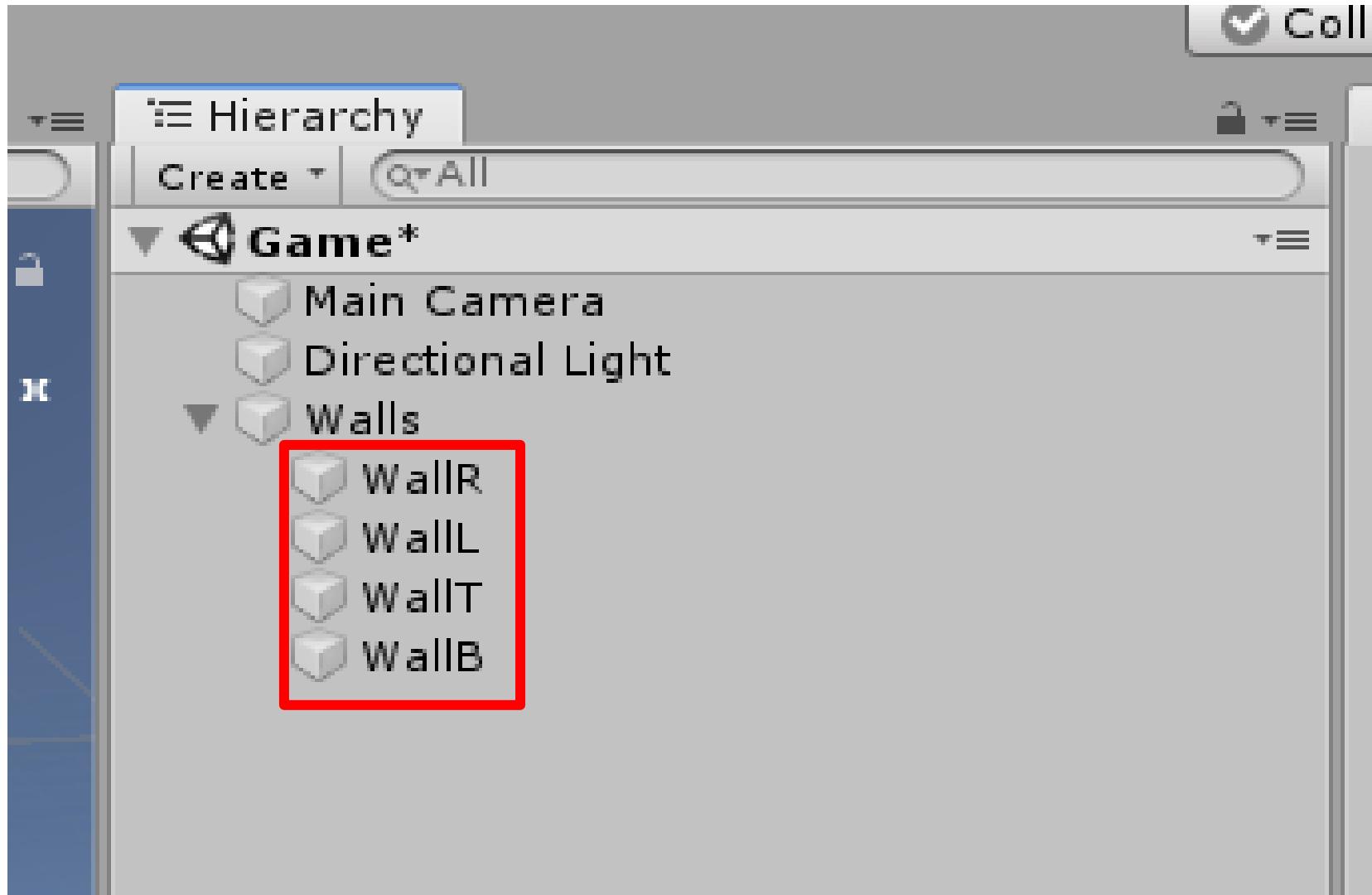


壁の作成

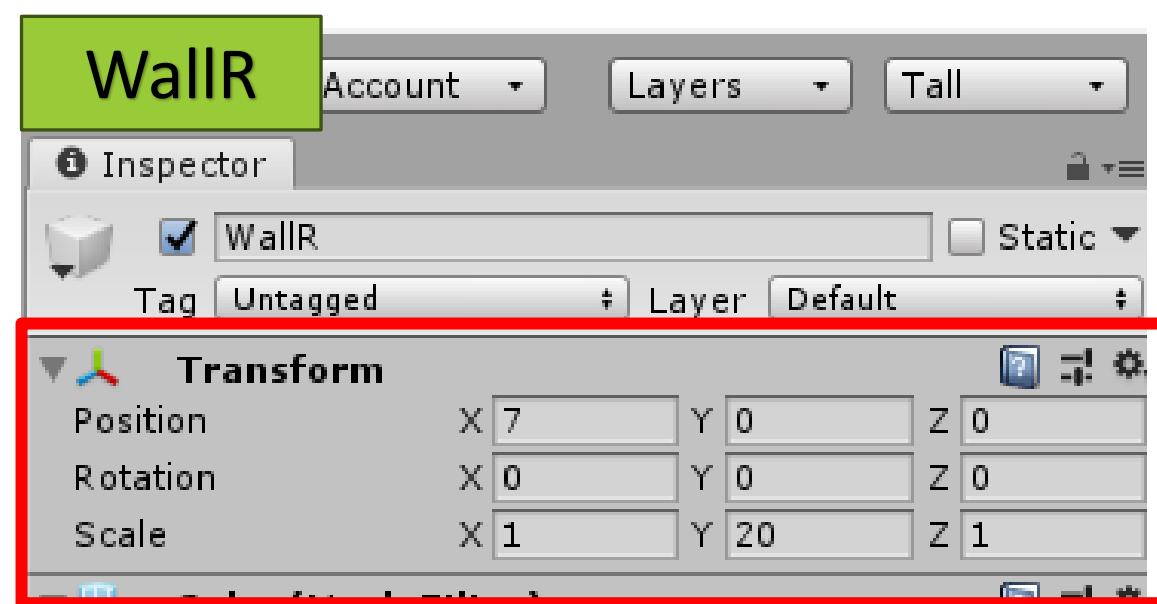




名前変更



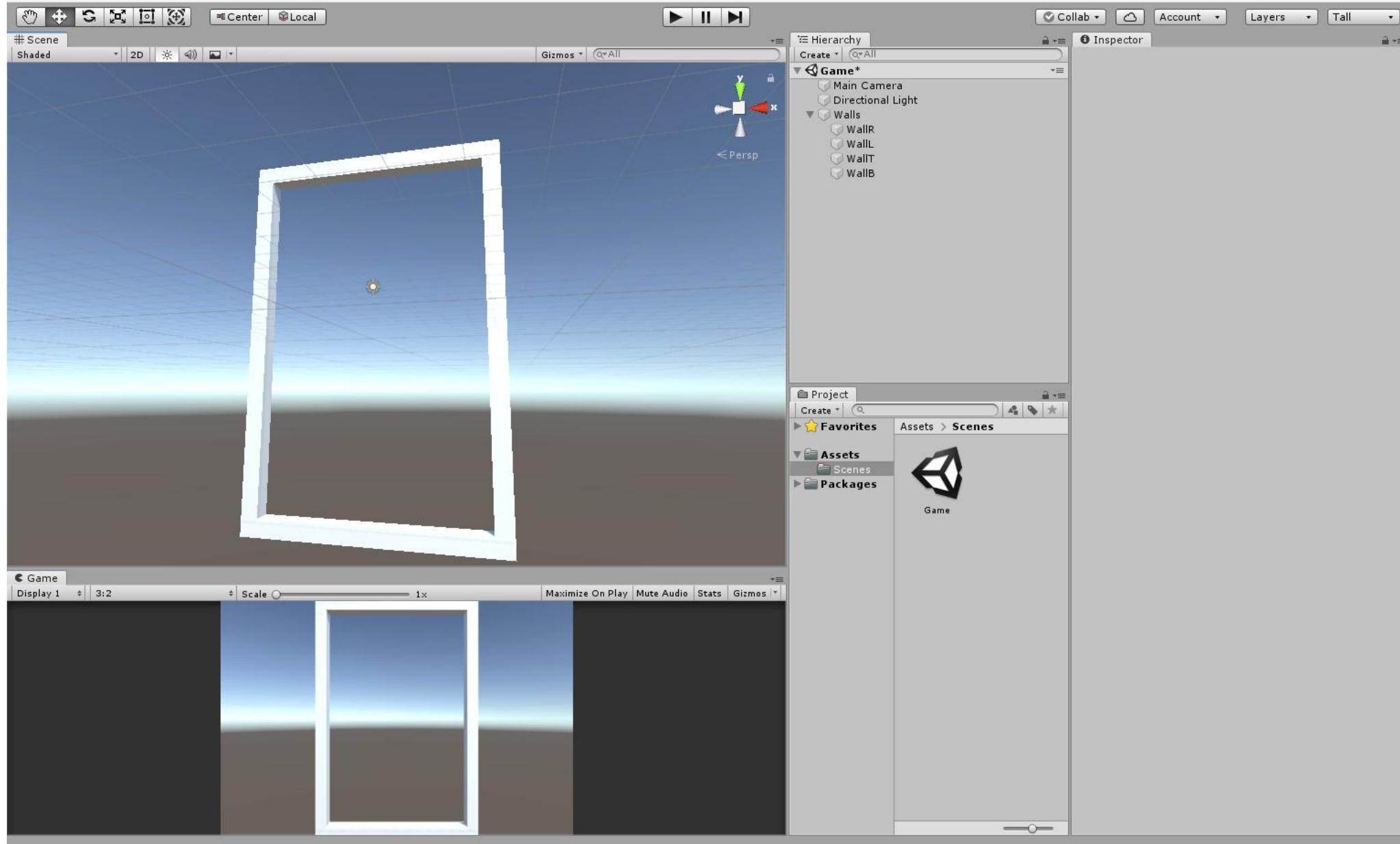
R(Right) ... 右
L(Left) ... 左
T(Top) ... 上
B(Bottom) ... 下



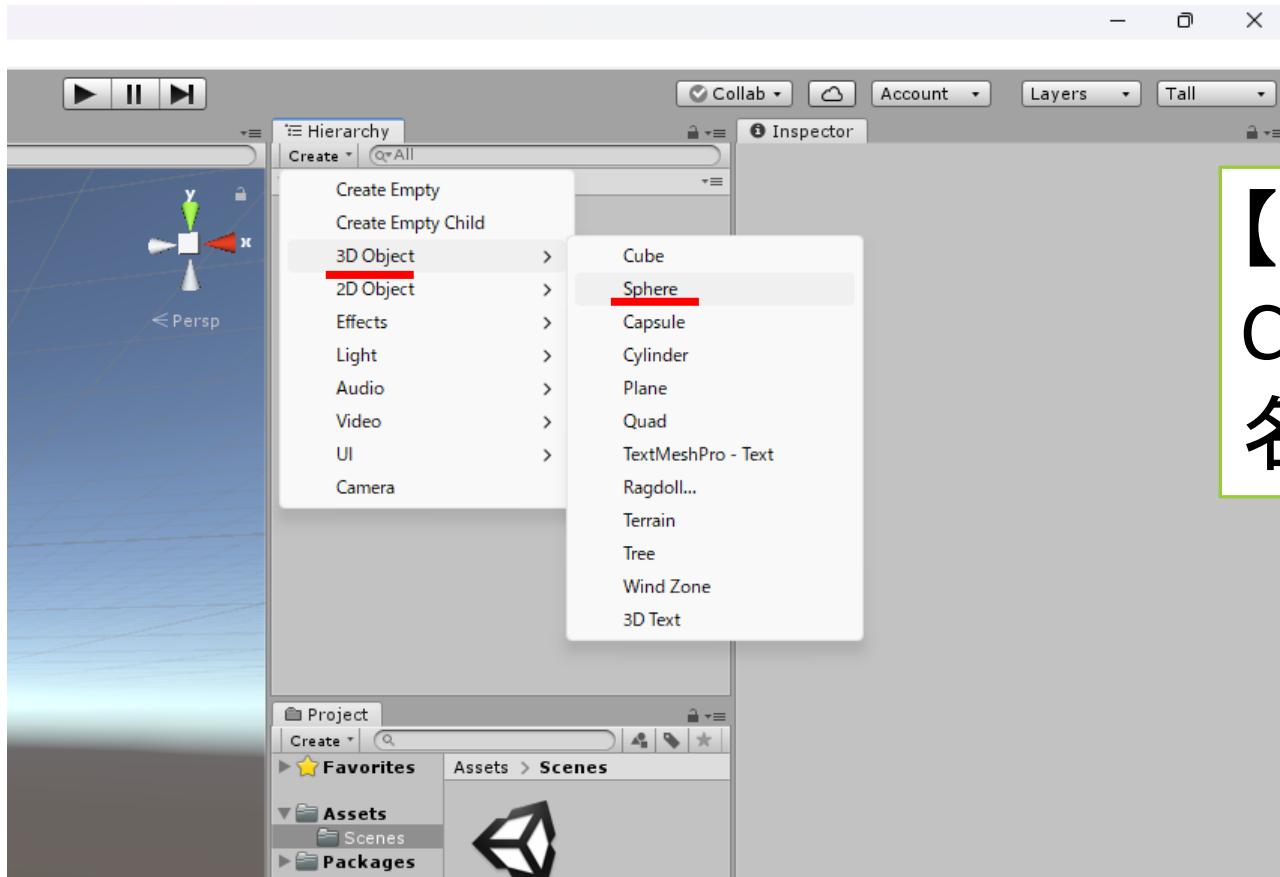
Position (位置)	X	-7	Y	0	Z	0
Rotation (回転)	X	0	Y	0	Z	0
Scale (大きさ)	X	1	Y	20	Z	1

Position (位置)	X	0	Y	10.5	Z	0
Rotation (回転)	X	0	Y	0	Z	0
Scale (大きさ)	X	15	Y	1	Z	1

Position (位置)	X	0	Y	-10.5	Z	0
Rotation (回転)	X	0	Y	0	Z	0
Scale (大きさ)	X	15	Y	1	Z	1

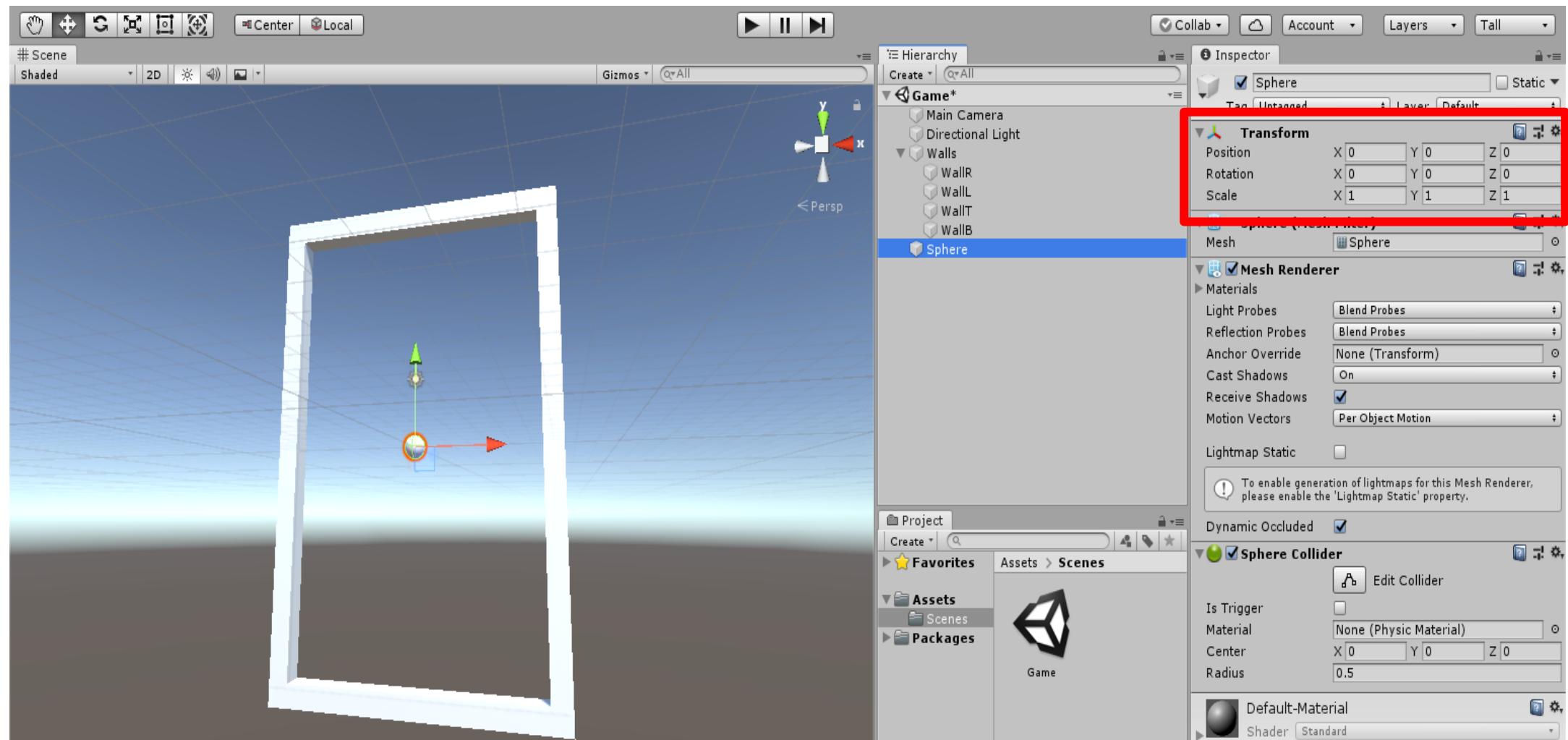


ボールの作成



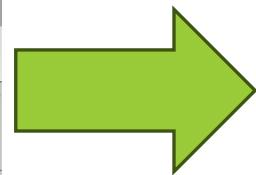
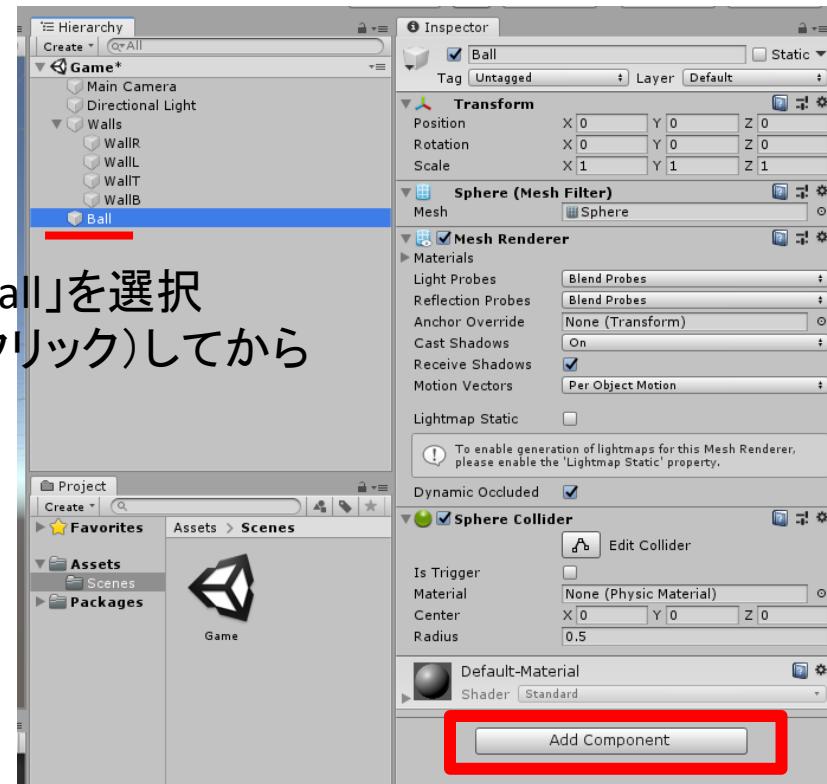
【hierarchy】
Create → 3DObject → Sphere
名前 : Ball

⌚ → reset



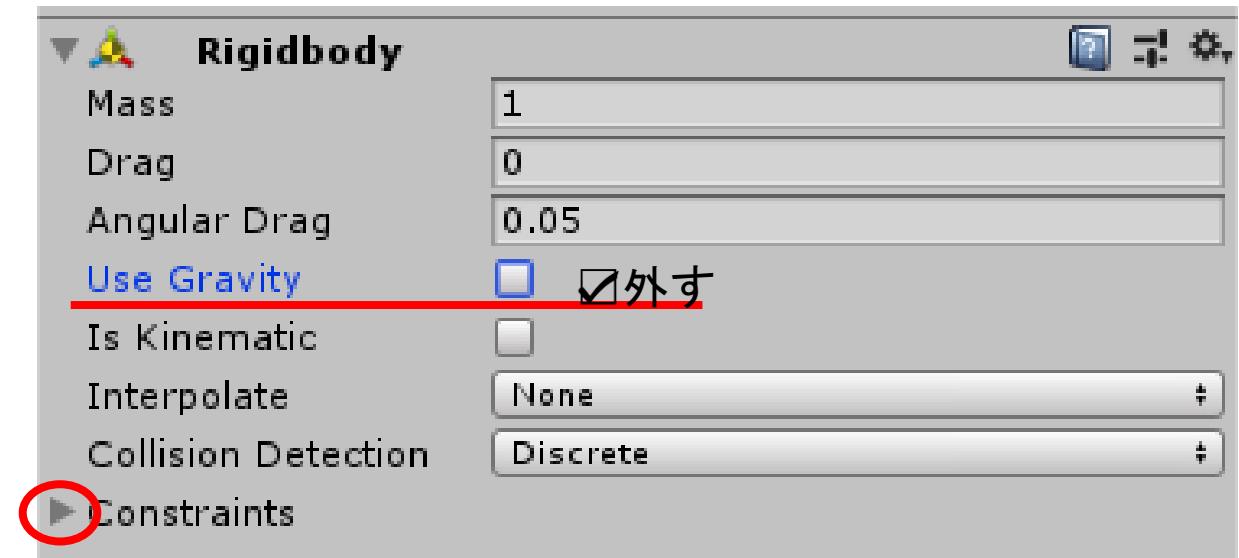
ボールに物理演算を加える

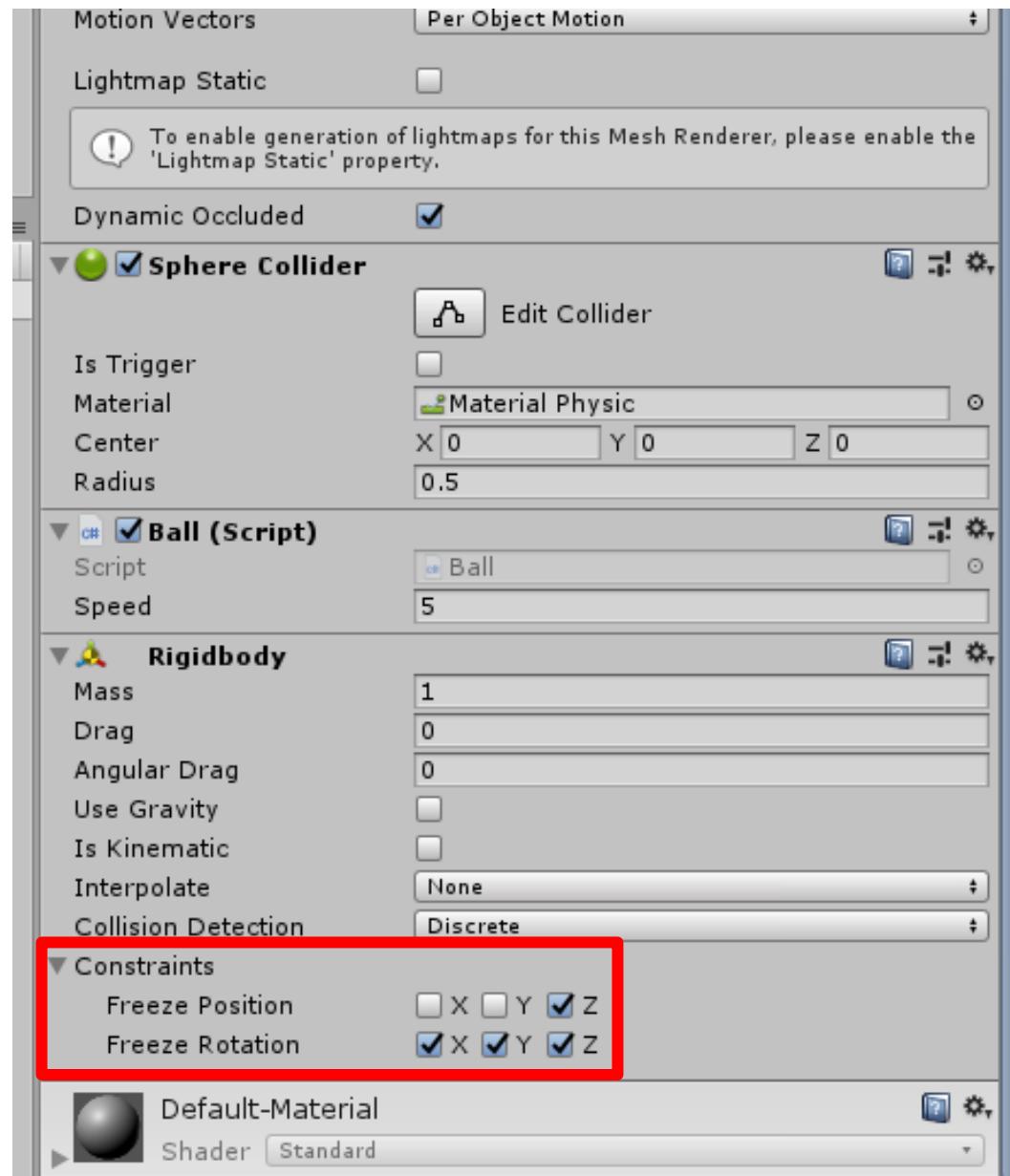
物理演算を付け加えて跳ね返ったりするようにする



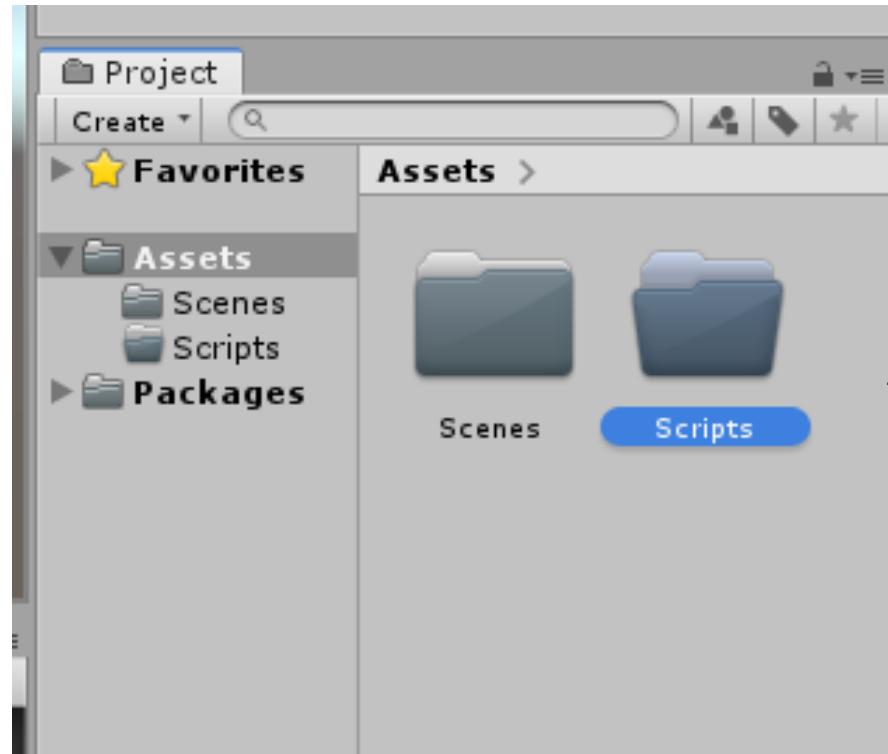
Rigidbodyを追加
(最初の文字の「ri」と打てば出てくる)

「Ball」を選択
(クリック)してから





スクリプトフォルダ作成

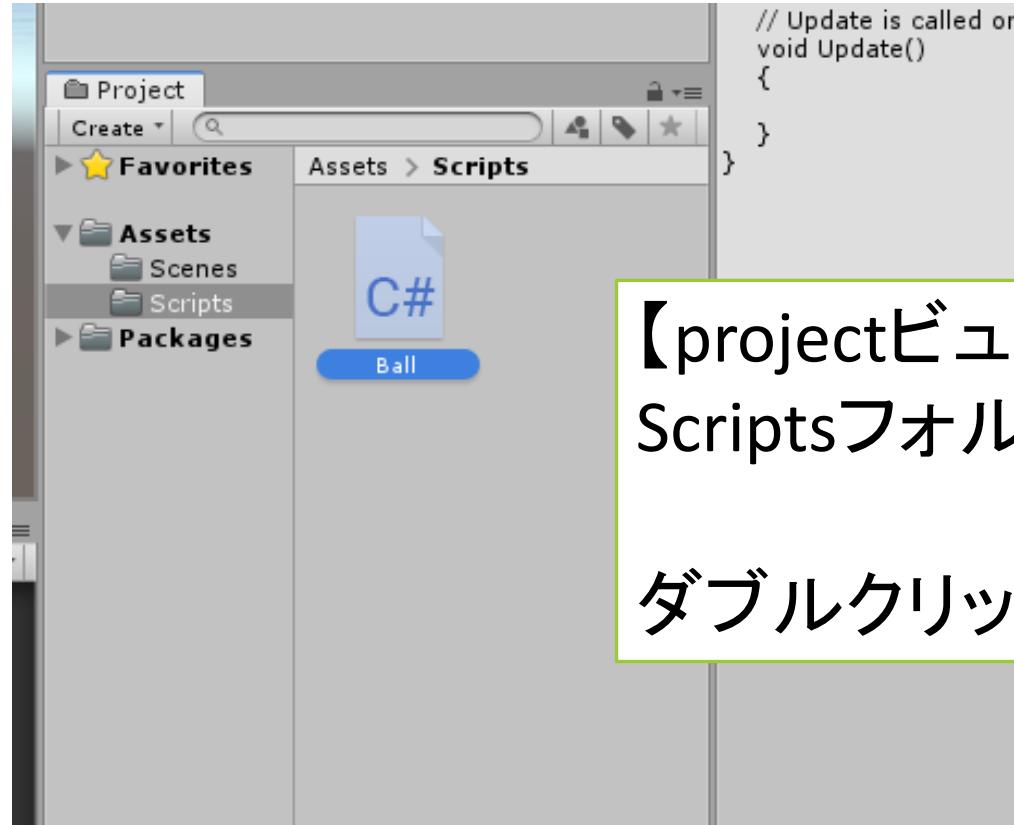


* scripts フォルダ作成 *

【projectビュー】

Assets選択→Create→Folder→フォルダ名「Scripts」

Scriptsフォルダの中にボールを動かすためのスクリプトを作成する



【projectビュー】

Scriptsフォルダ選択→Create→C#Script→名前「Ball」

ダブルクリックで開く

The screenshot shows the Visual Studio IDE interface with a C# script named `Ball.cs` open. The code defines a `Ball` class that inherits from `MonoBehaviour`. It contains two methods: `Start()` and `Update()`. The `Start()` method is annotated with a comment indicating it is called before the first frame update. The `Update()` method is annotated with a comment indicating it is called once per frame.

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

public class Ball : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {

    }
}
```

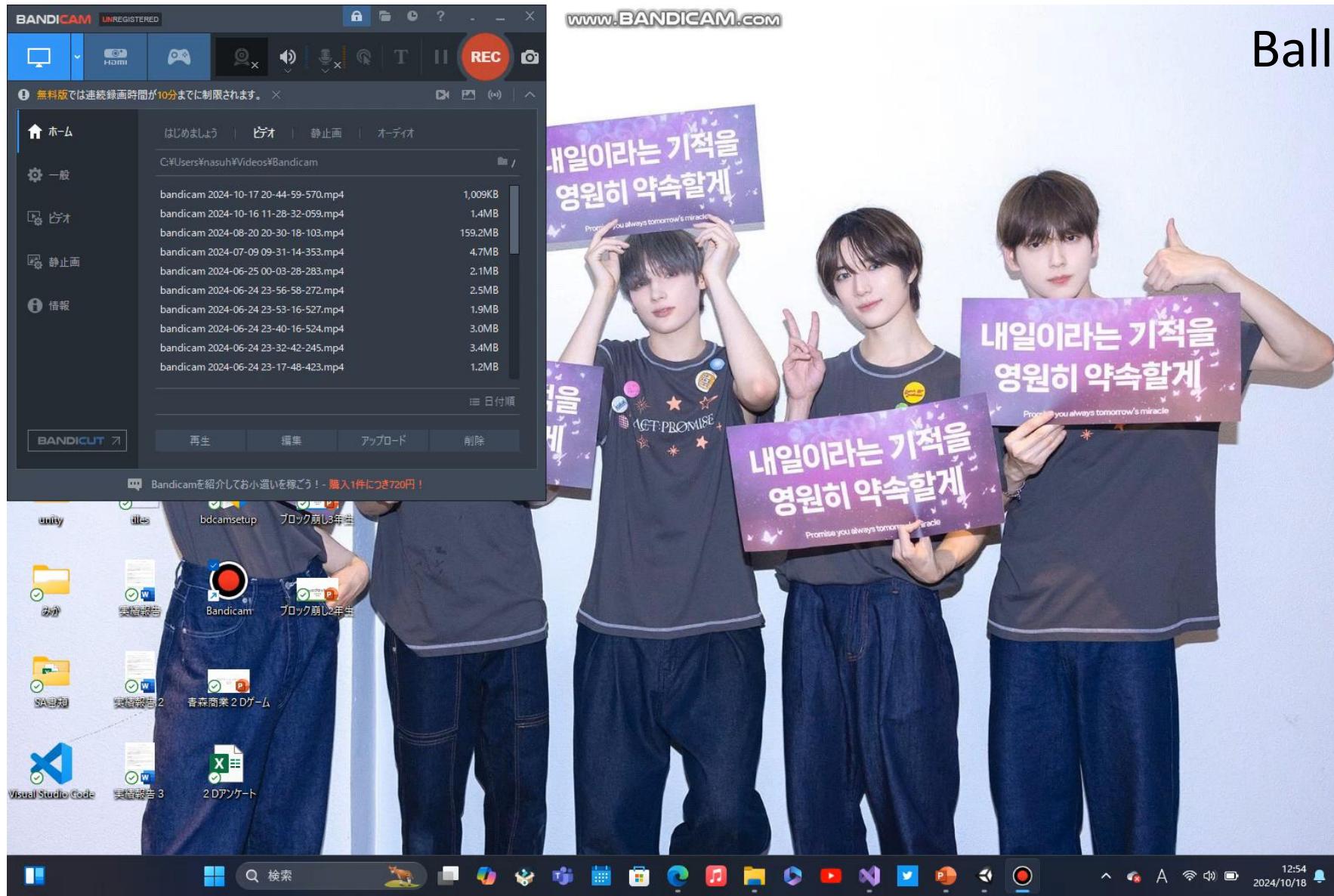
Below the code editor, the Error List window is visible, showing 0 errors, 0 warnings, and 0 messages. The status bar at the bottom indicates the zoom level is 189%.

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

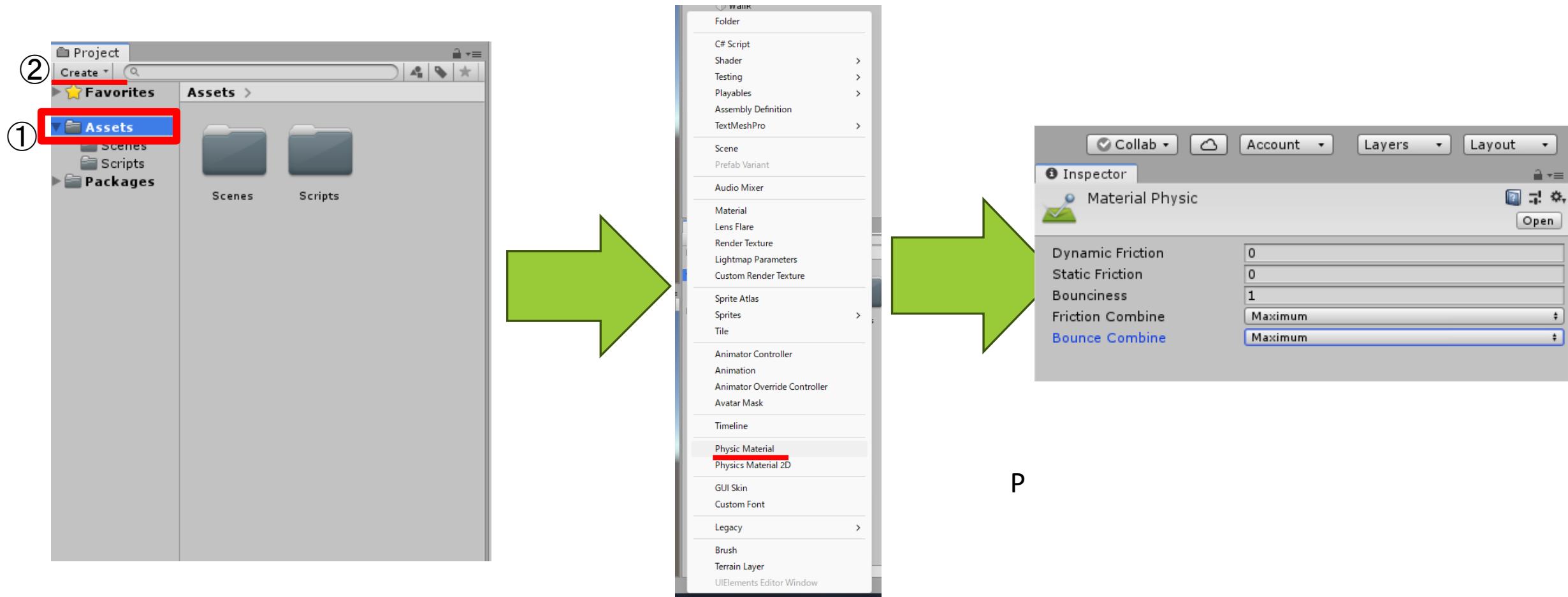
public class Ball : MonoBehaviour
{
    // ボールの移動の速さを指定する変数
    public float speed = 5f;
    Rigidbody myRigidbody;

    private void Start()
    {
        // Rigidbodyにアクセスして変数に保持しておく
        myRigidbody = GetComponent<Rigidbody>();
        // 右斜め45度に進む
        myRigidbody.velocity = new Vector3(speed, speed, 0f);
    }
}
```

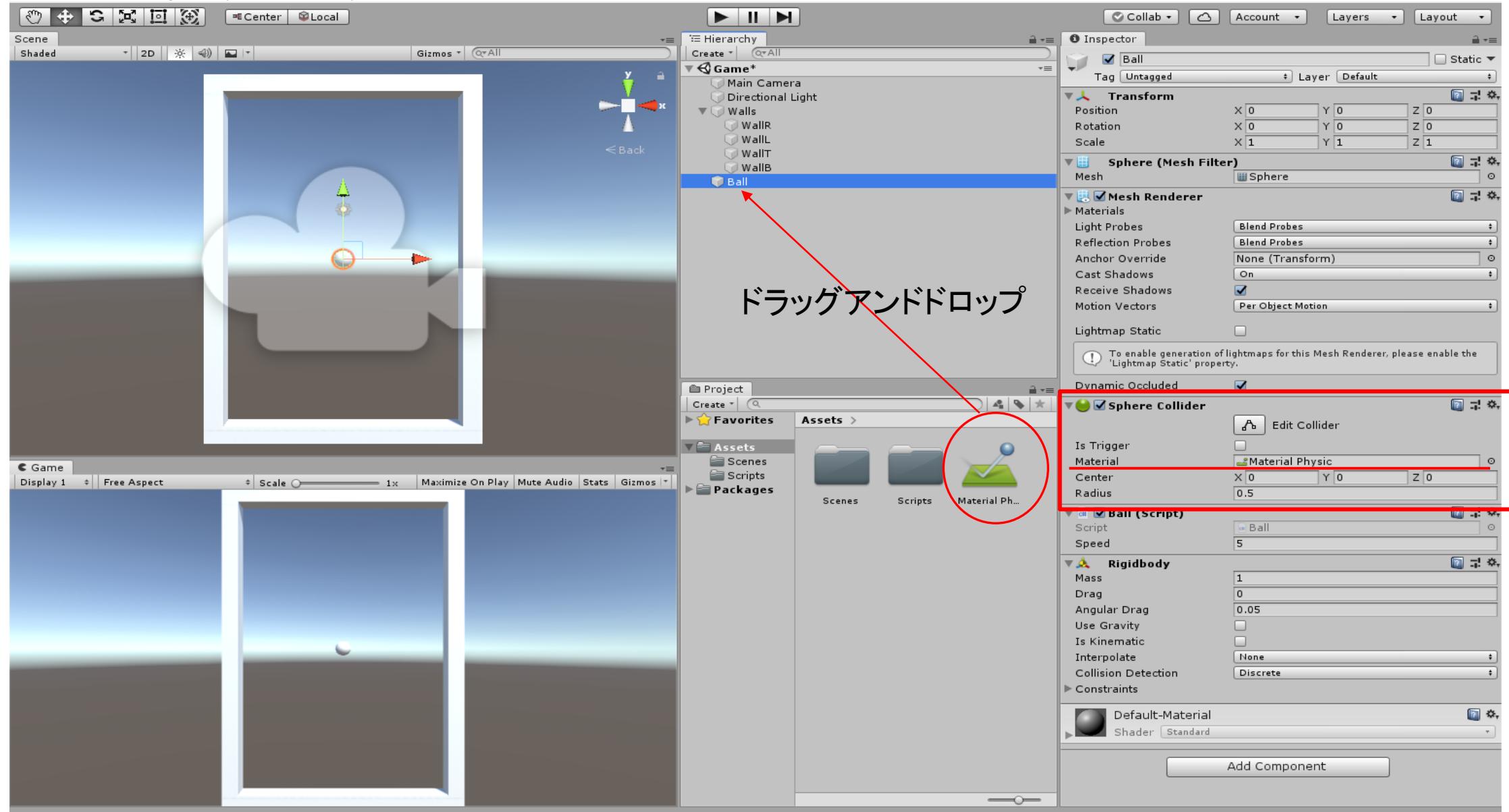
Ballスクリプトを入れる



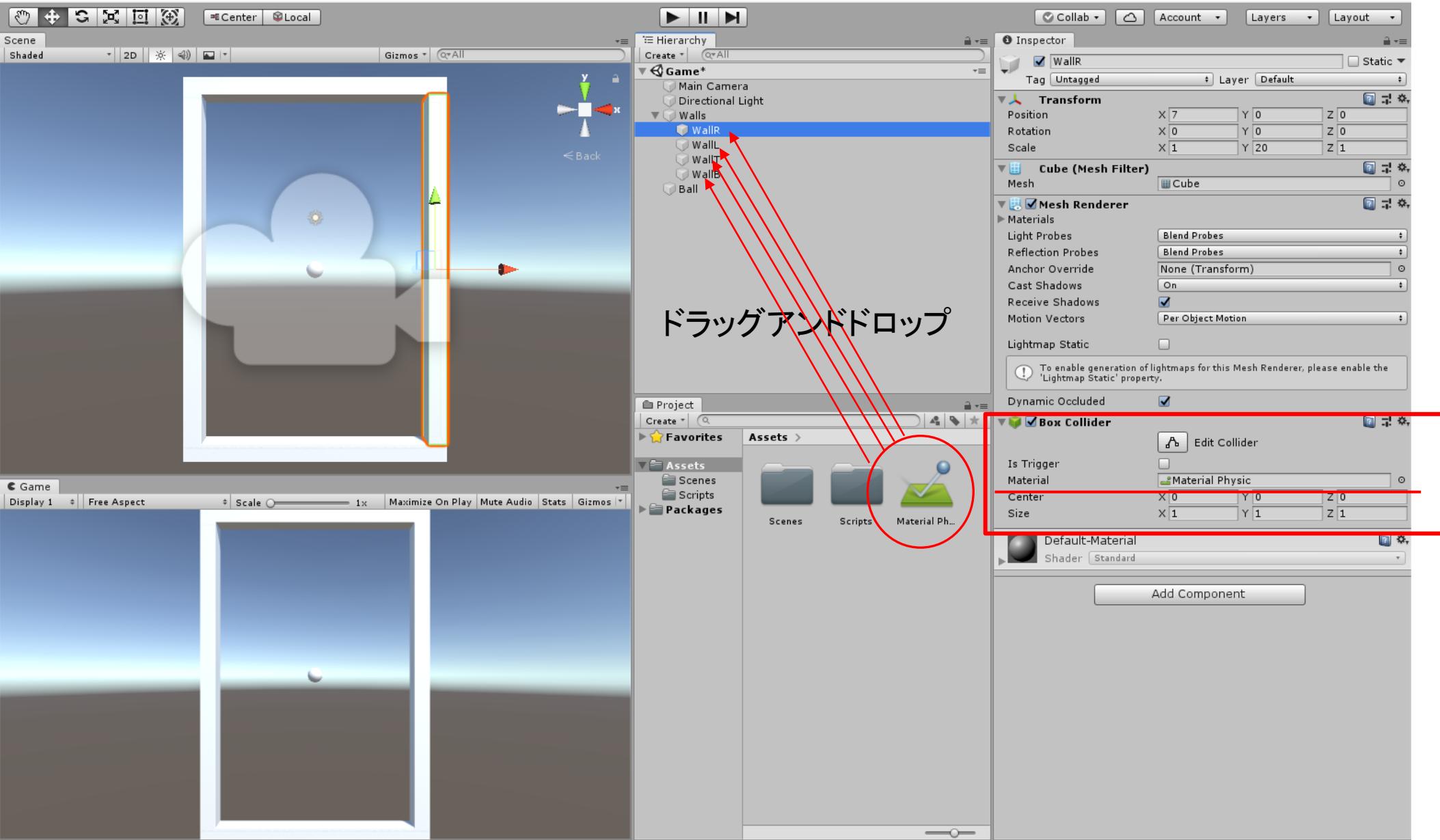
正しく反射するように設定



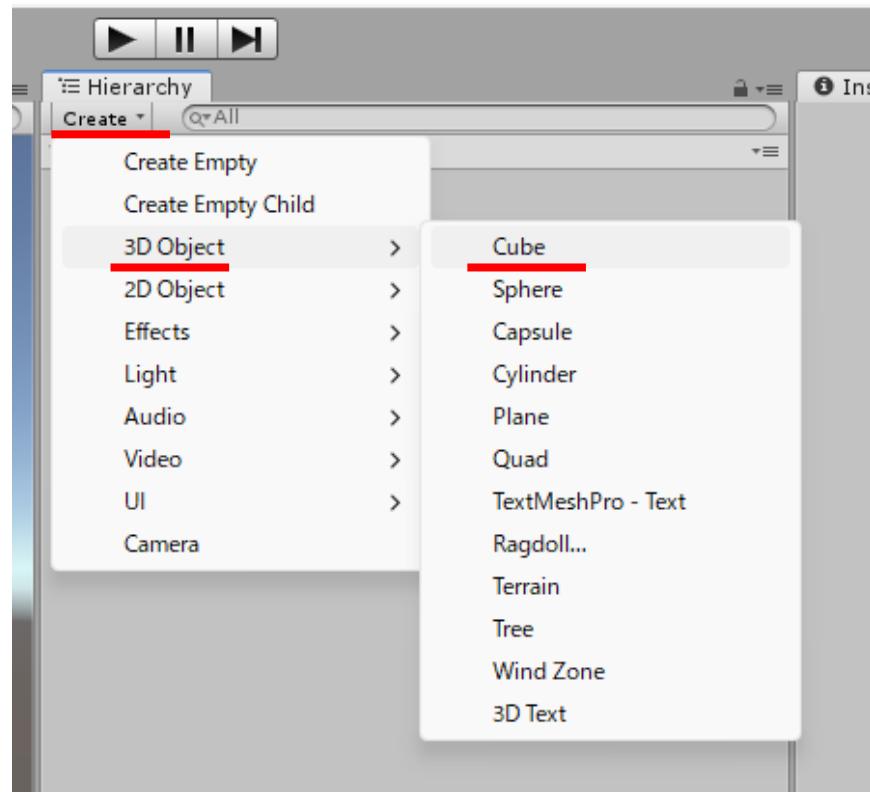
P



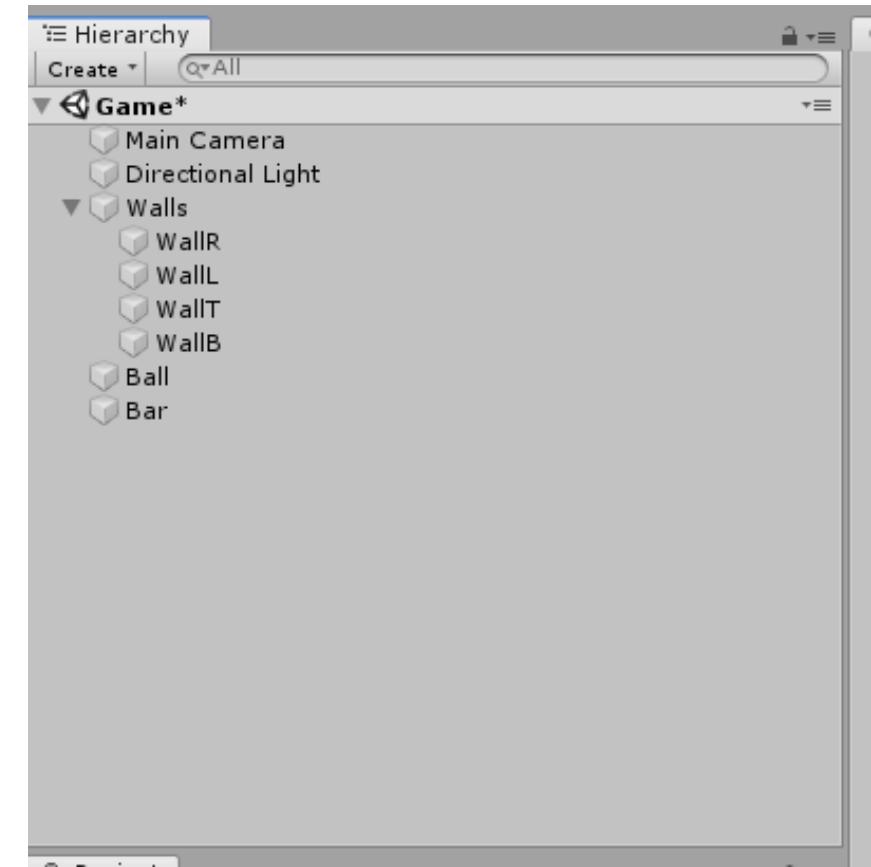
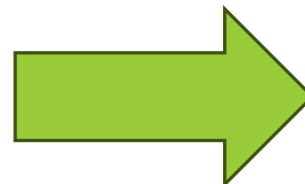
ドラッグアンドドロップ



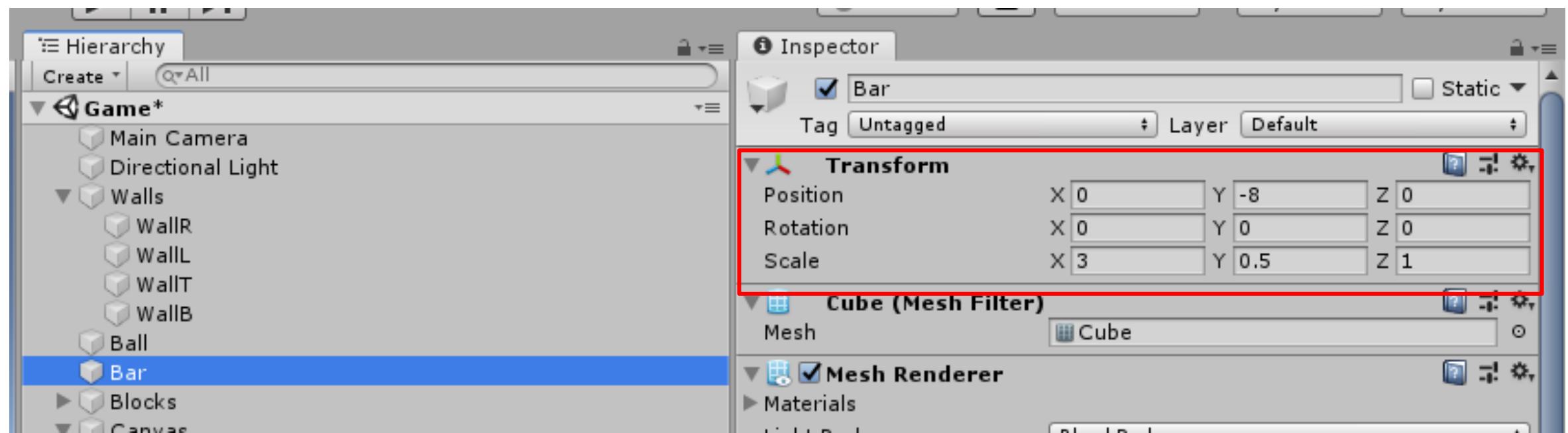
パドル(Bar)の作成



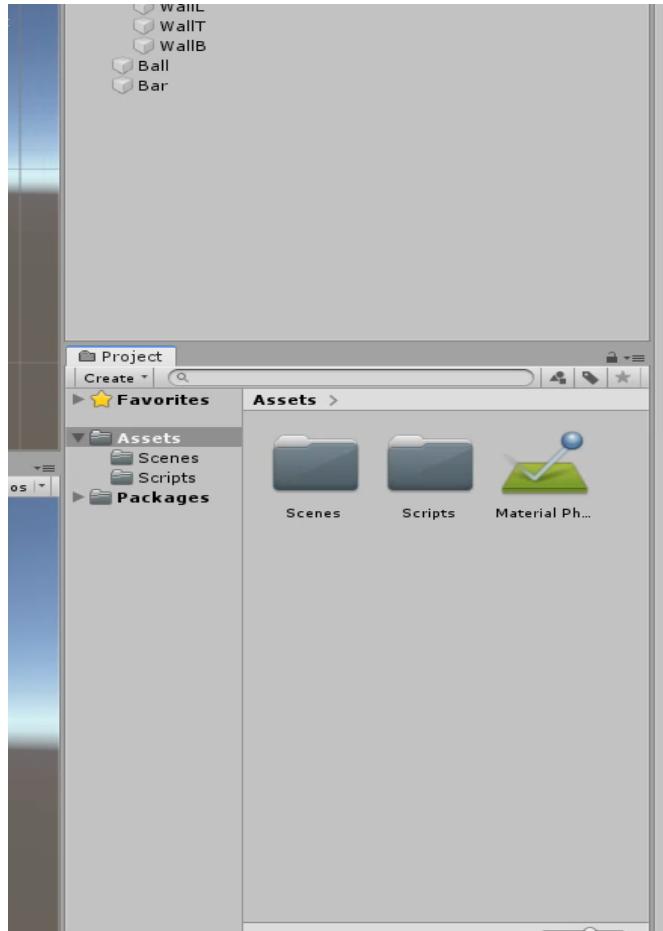
名前 : Bar



パドルサイズ変更



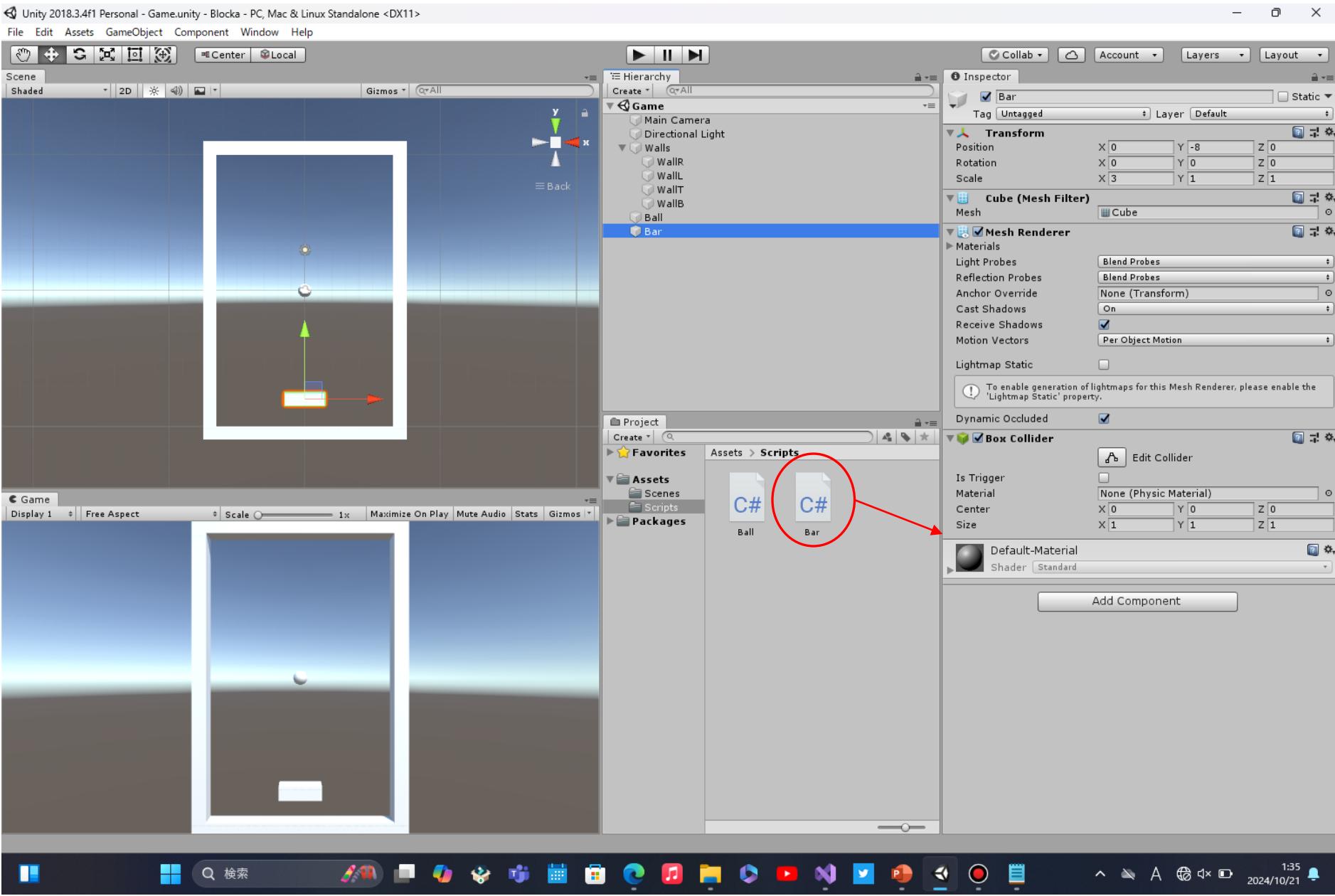
パドル(Bar)のスクリプト作成

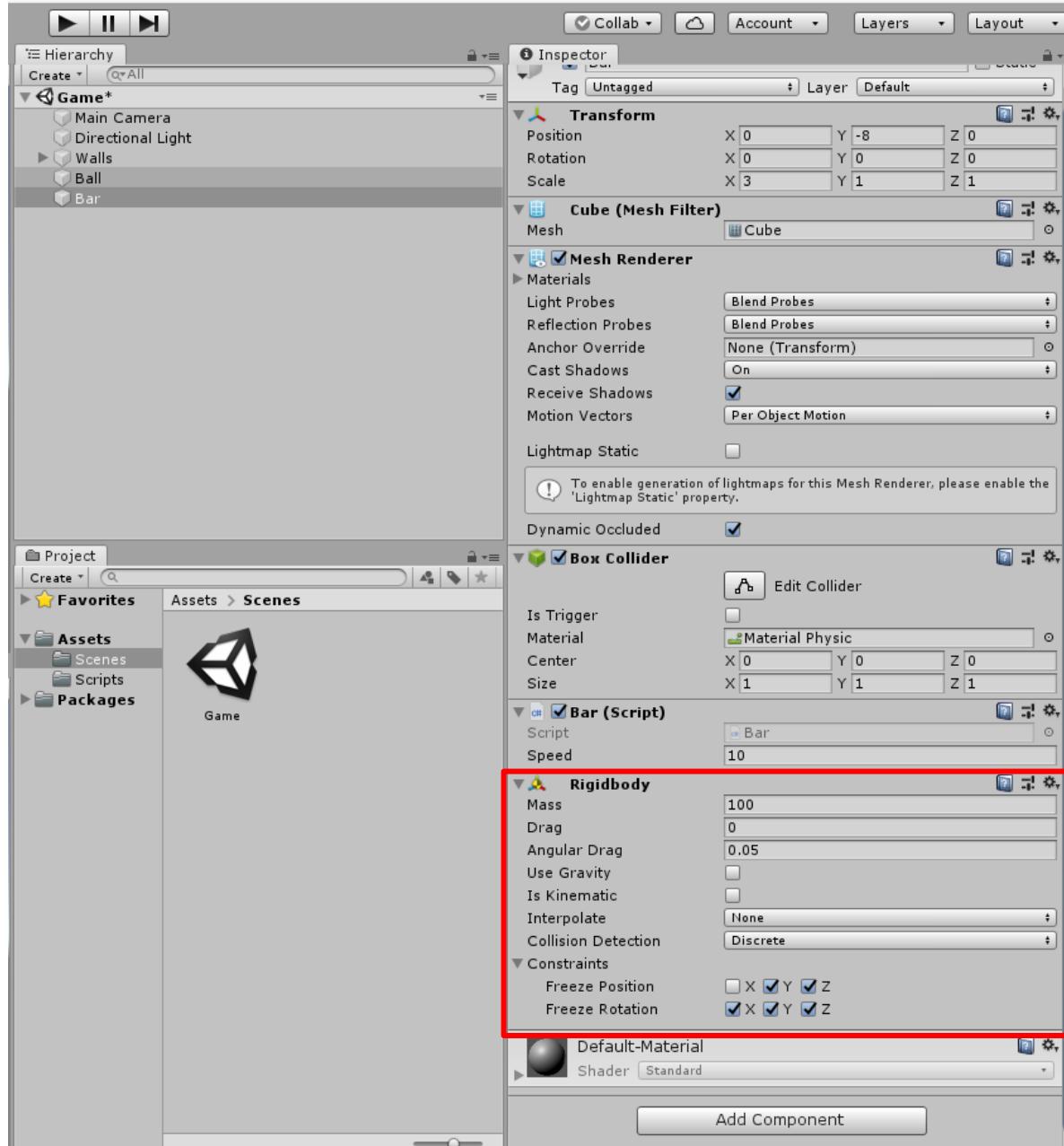


Scriptsフォルダの中にBarスクリプト作成
【Project】
Create→C#Script 名前Bar

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

public class Bar : MonoBehaviour
{
    void Update()
    {
        //左キーを入力すると左にBarが動く
        if (Input.GetKey(KeyCode.LeftArrow))
        {
            this.transform.Translate(-0.5f, 0f, 0f);
        }
        //右キーを入力すると右にBarが動く
        if (Input.GetKey(KeyCode.RightArrow))
        {
            this.transform.Translate(0.5f, 0f, 0f);
        }
    }
}
```





BarにRigidbody追加

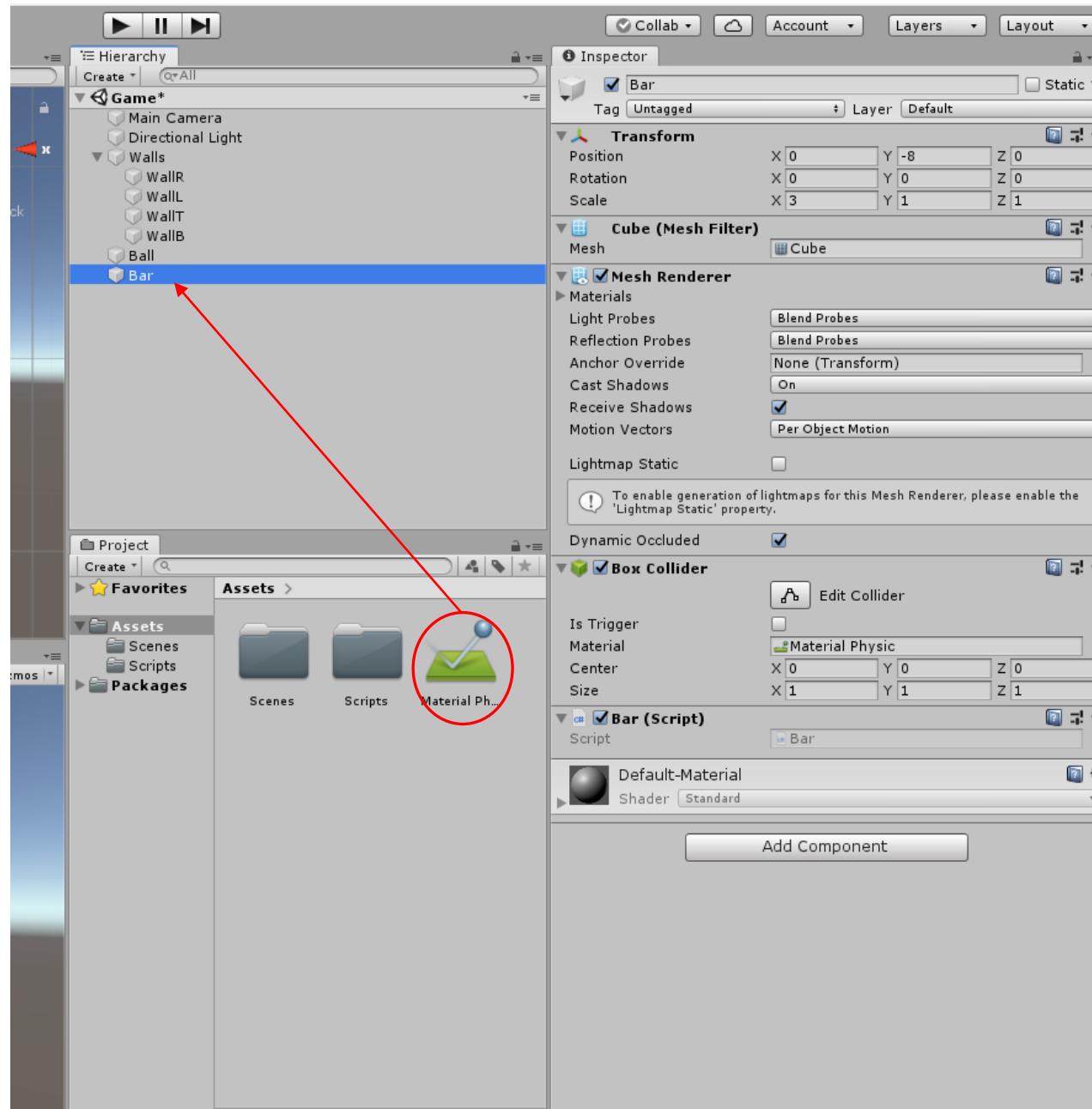
【inspector】

Add Component→Rigidbody

これやらなきや
パドル、ボールぶつ飛ぶ

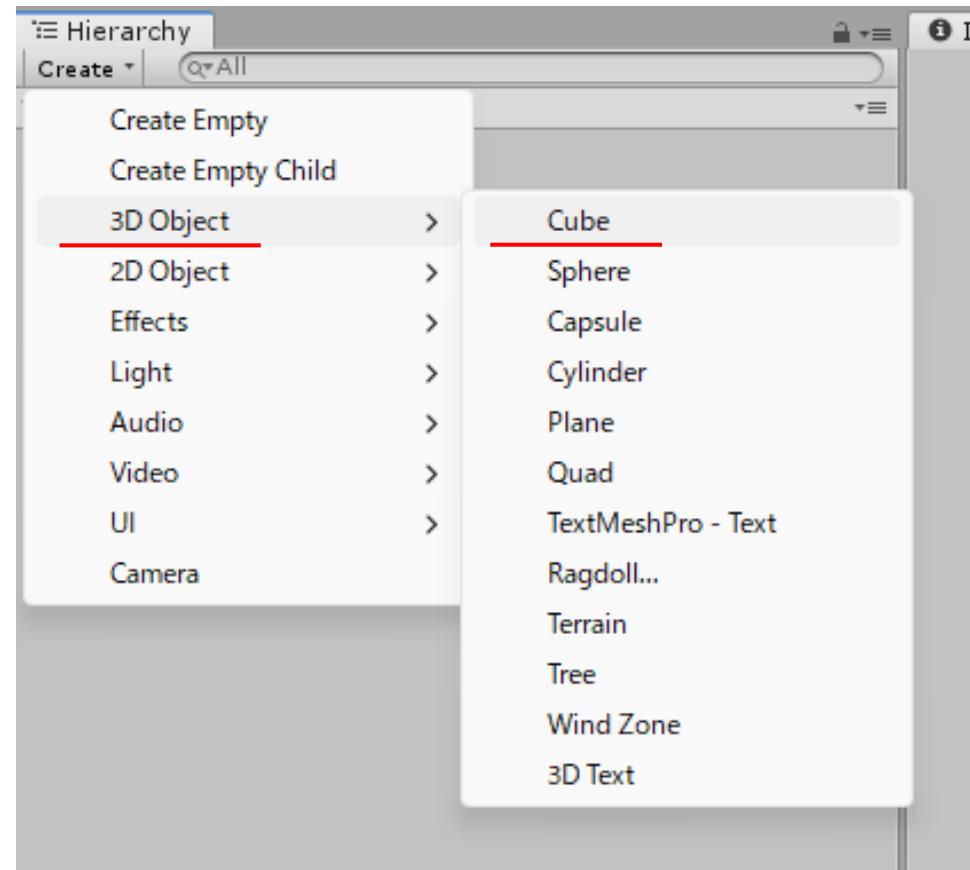
Playerの質量を大きくします。

PlayerにアタッチしているRigidbodyのMassの値を大きく

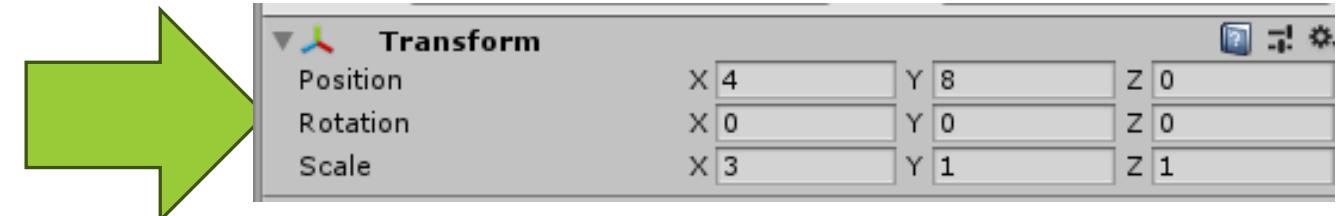


「Material Physics」をBarにアタッチしてあげます。
(衝突でのエネルギー減少を失くすため)

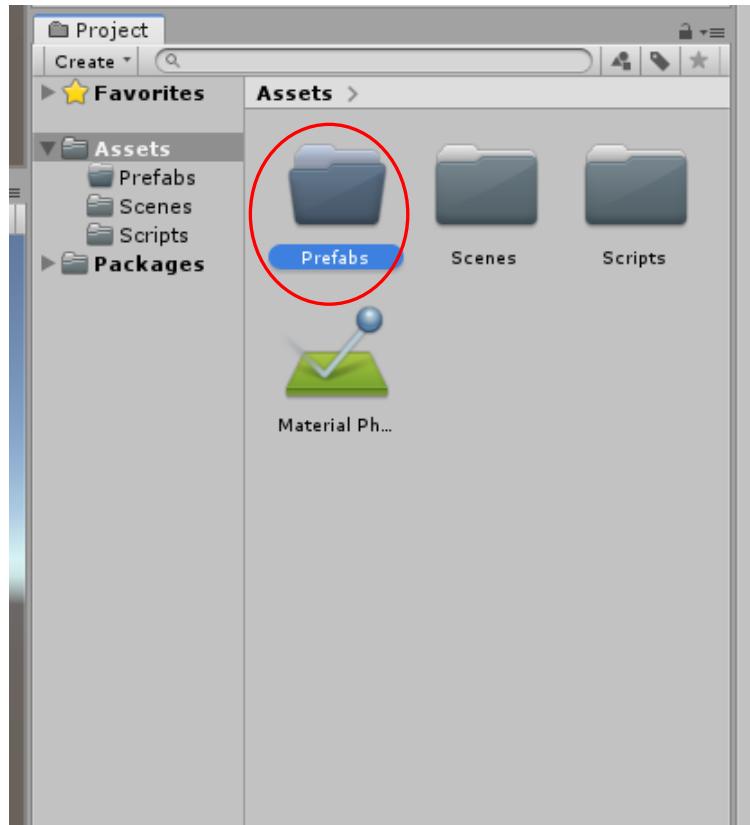
ブロックの作成



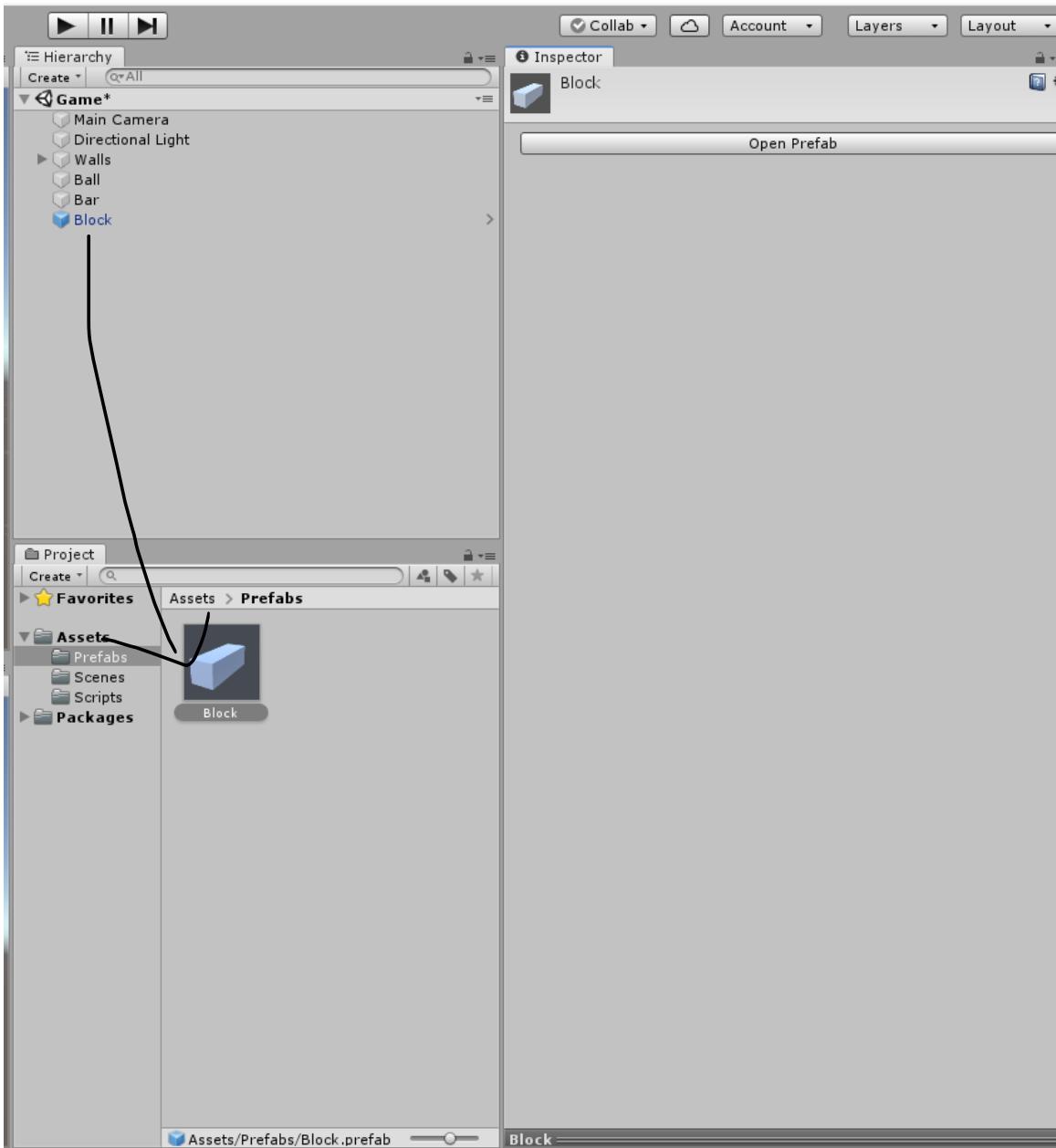
名前 : Block



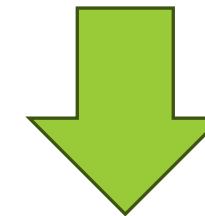
Prefabsファイル作成



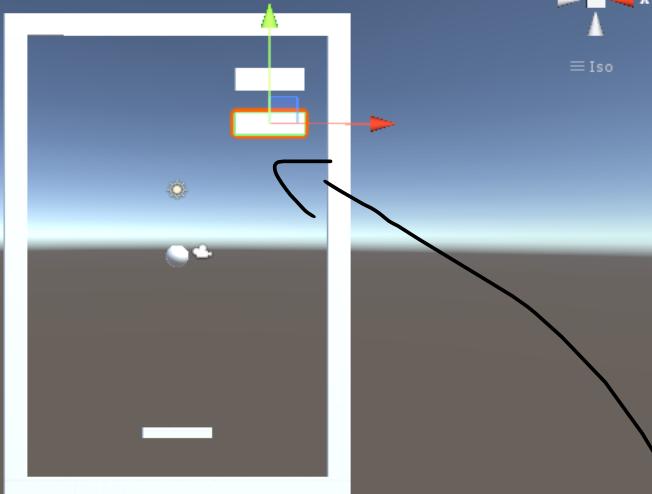
【Project】
Create → folder → 名前 : Prefabs



HierarchyビューからProjectビューに
ドラッグ&ドロップすれば作成できる



ProjectビューからSceneビューに
ドラッグアンドドロップ

Scene
Shaded
2DGizmos
All

Hierarchy

- Game*
- Main Camera
- Block (1)
- Directional Light
- Walls
 - WallR
 - WallL
 - WallIT
 - WallB
- Bar
- Ball
- Block

Collab Account Layers Layout

Inspector Tile Palette

Block (1)
Tag Untagged Layer Default
Prefab Open Select Overrides

Transform
Position X 4 Y 6 Z 0
Rotation X 0 Y 0 Z 0
Scale X 3 Y 1 Z 1

Cube (Mesh Filter)
Mesh Cube

Mesh Renderer
Materials
Light Probes
Reflection Probes
Anchor Override
Cast Shadows
Receive Shadows
Motion Vectors
Lightmap Static

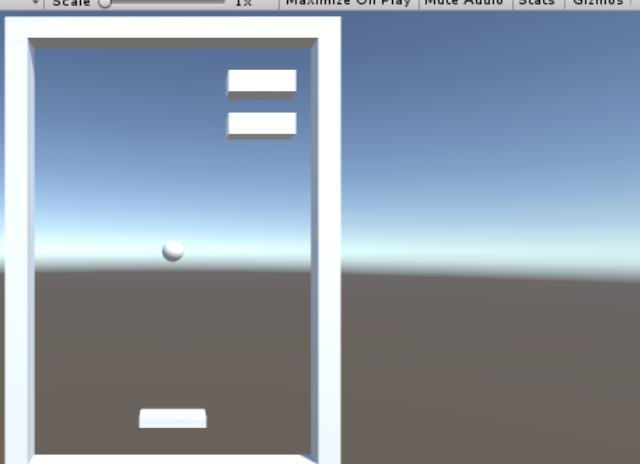
To enable generation of lightmaps for this Mesh Renderer, please enable the 'Lightmap Static' property.

Dynamic Occluded

Box Collider
Edit Collider
Is Trigger
Material None (Physic Material)
Center X 0 Y 0 Z 0
Size X 1 Y 1 Z 1

Default-Material
Shader Standard

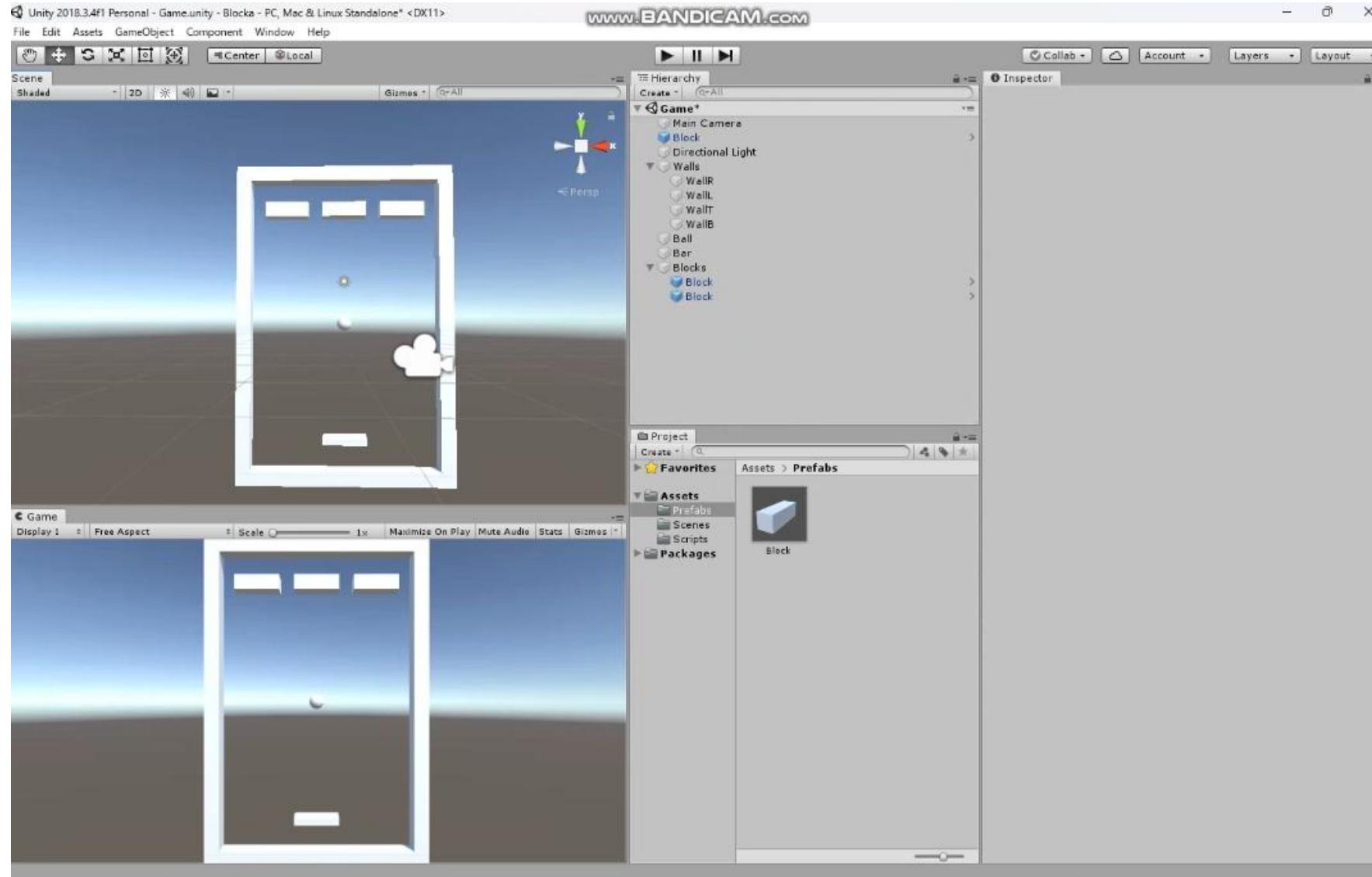
Add Component



Project Create Favorites Assets > Prefabs

Prefab Scenes Scripts Packages





* 裏技 *
ブロックを選択する
↓
コントロールキー + D

緑の矢印持ちながら下に

Blockのスクリプト作成

【Project】

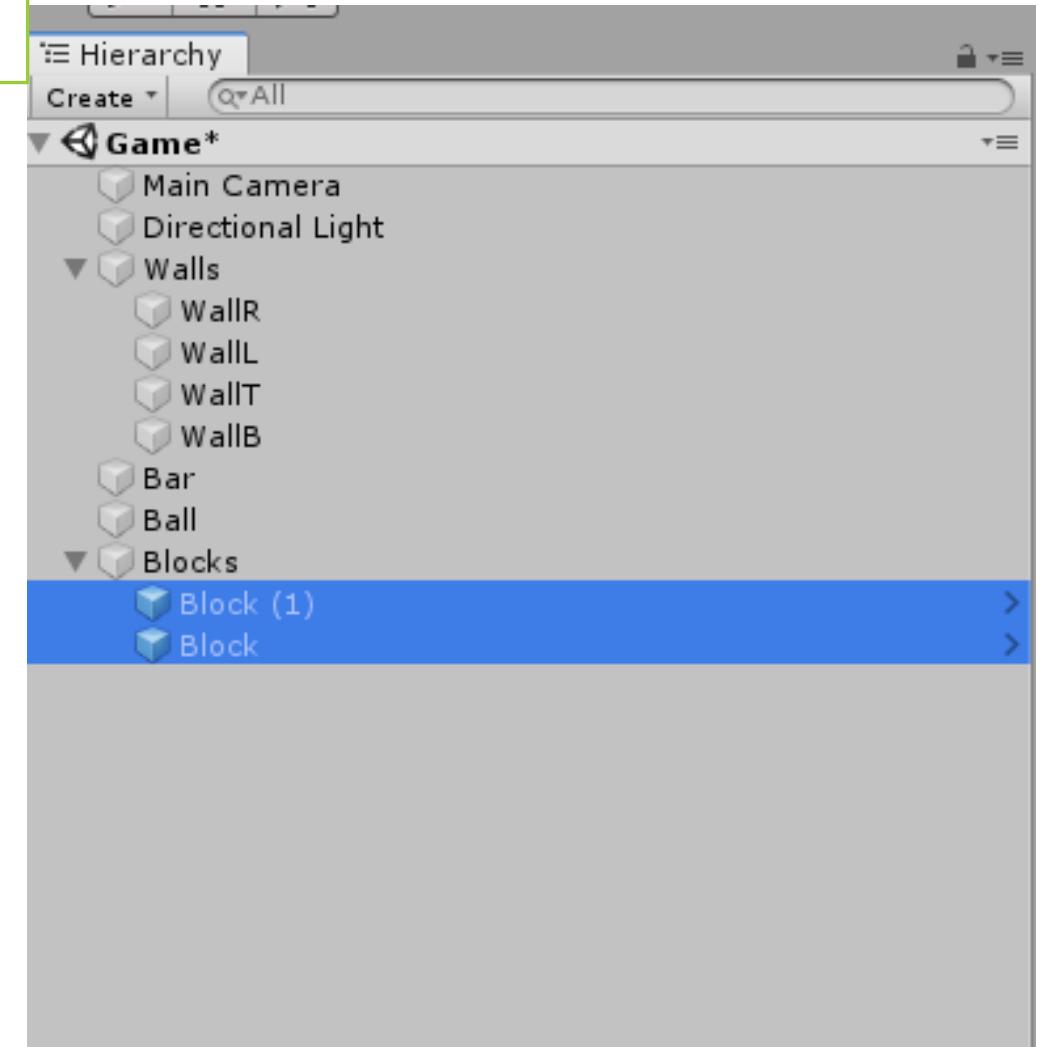
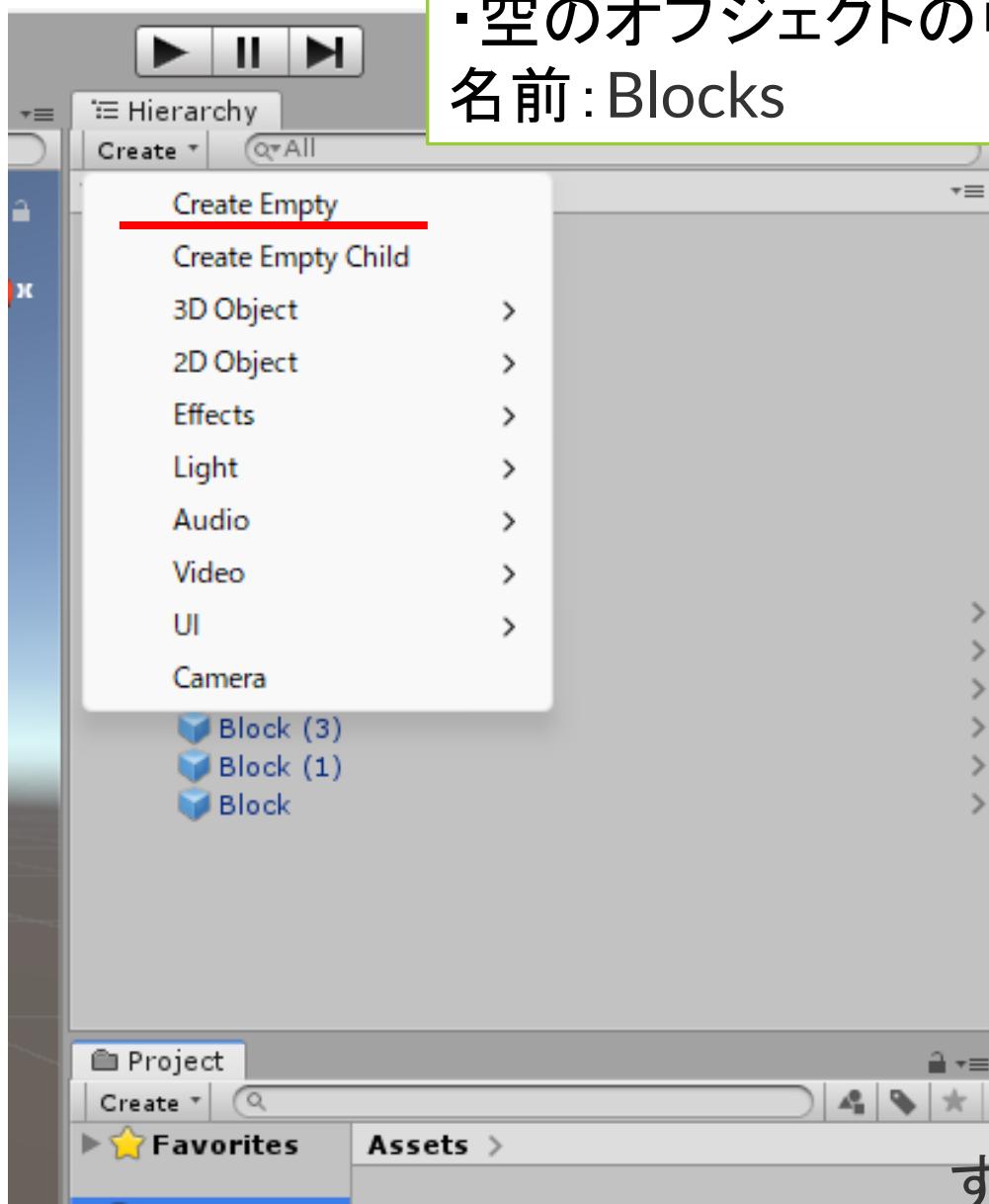
scriptsフォルダ選択

Create→C#Scripts 名前:Block

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

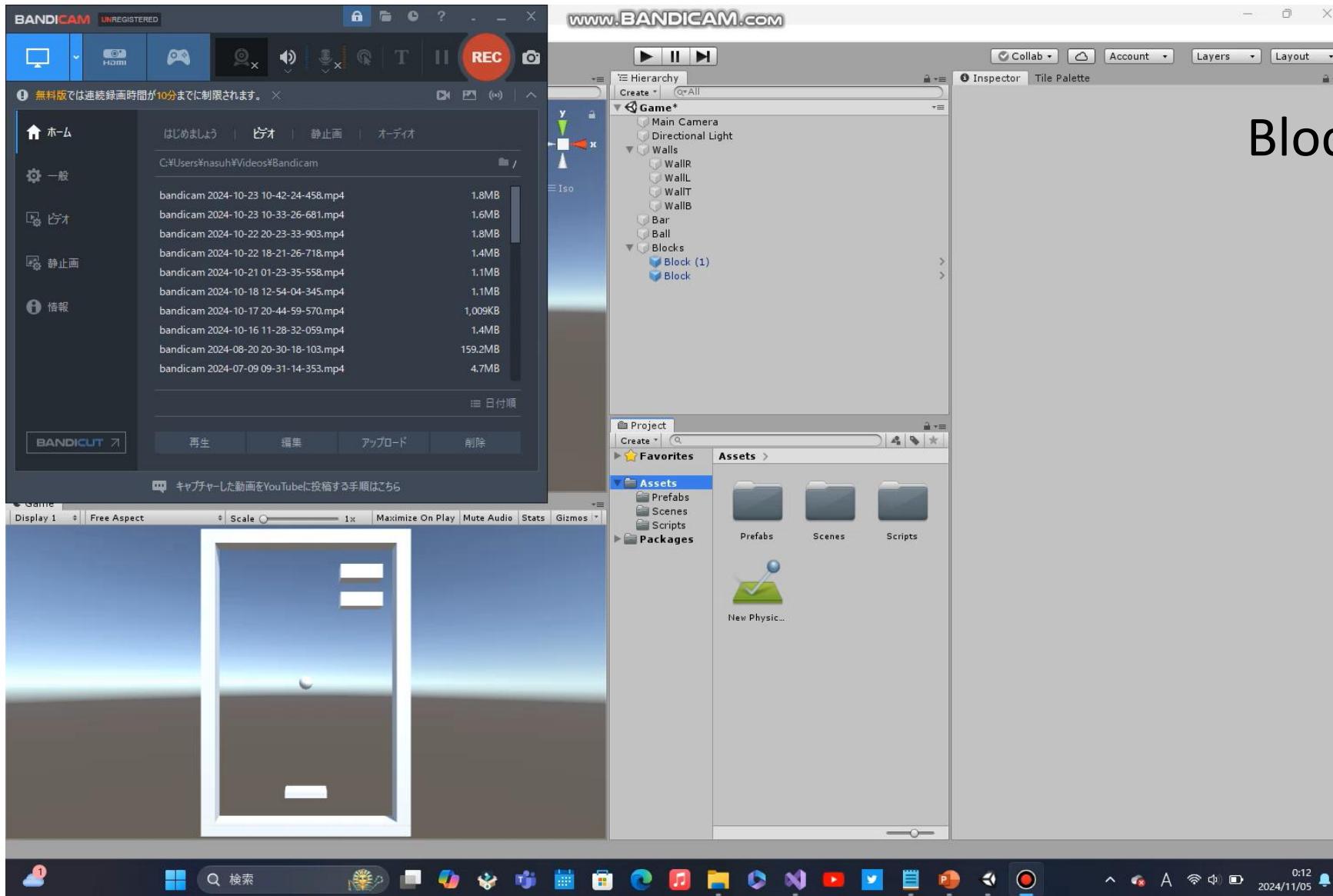
public class Block : MonoBehaviour
{
    private void OnCollisionEnter(Collision collision)
    {
        Destroy(gameObject);
    }
}
```

・空のオブジェクトの中に入れる
名前: Blocks



すべてのBlockを選択し、Blocksにドラッグアンドドロップ

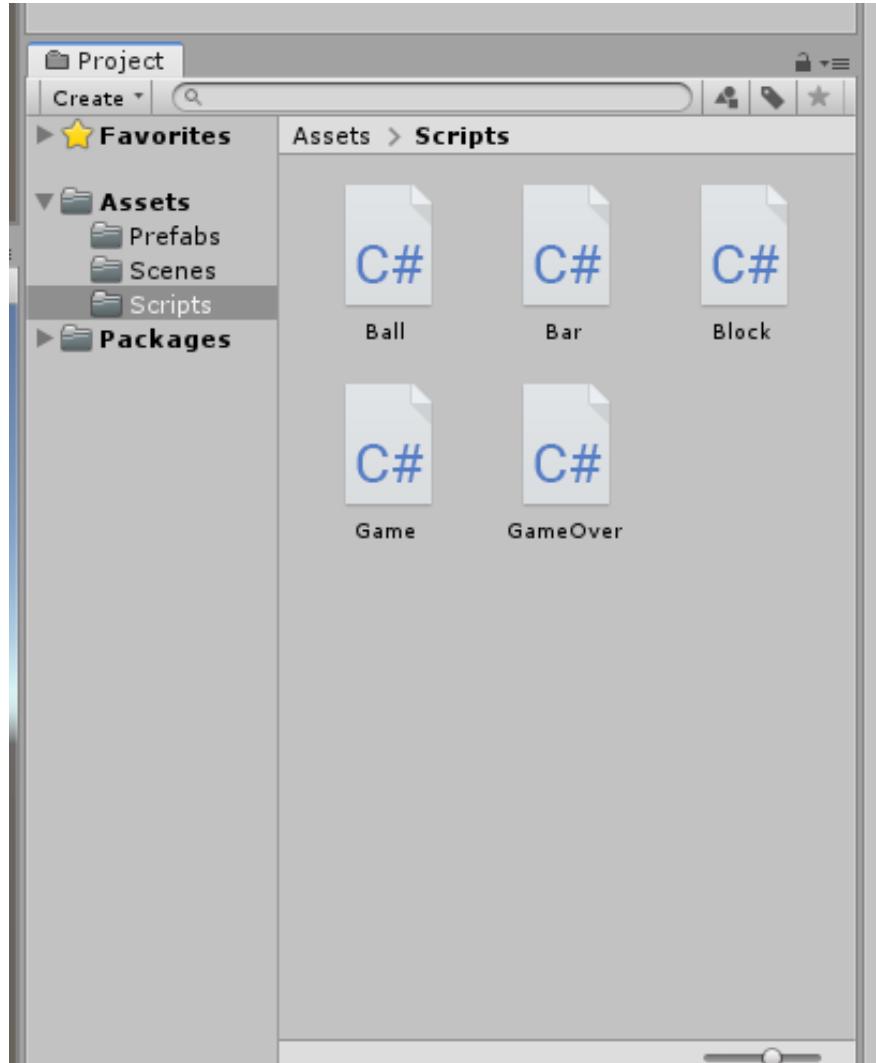
Blockにスクリプトつける



ゲームオーバーの判定

ブロック崩しはボールが下に行ってしまったらミス
下の壁に当たったときにボールを削除する

GameOverスクリプト作成

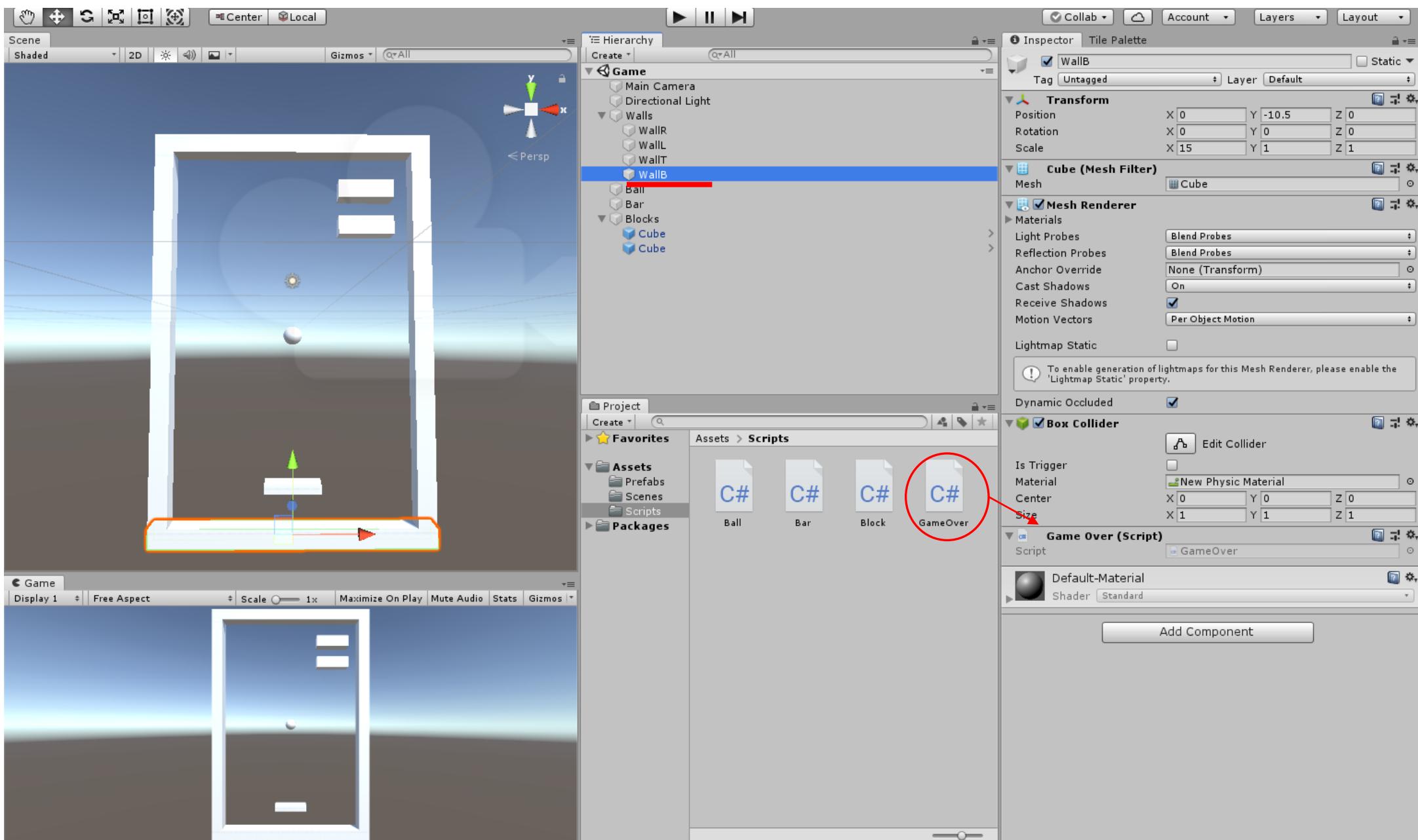


scriptsフォルダの中にGameOverスクリプト作成

Create→C#スクリプト 名前:GameOver

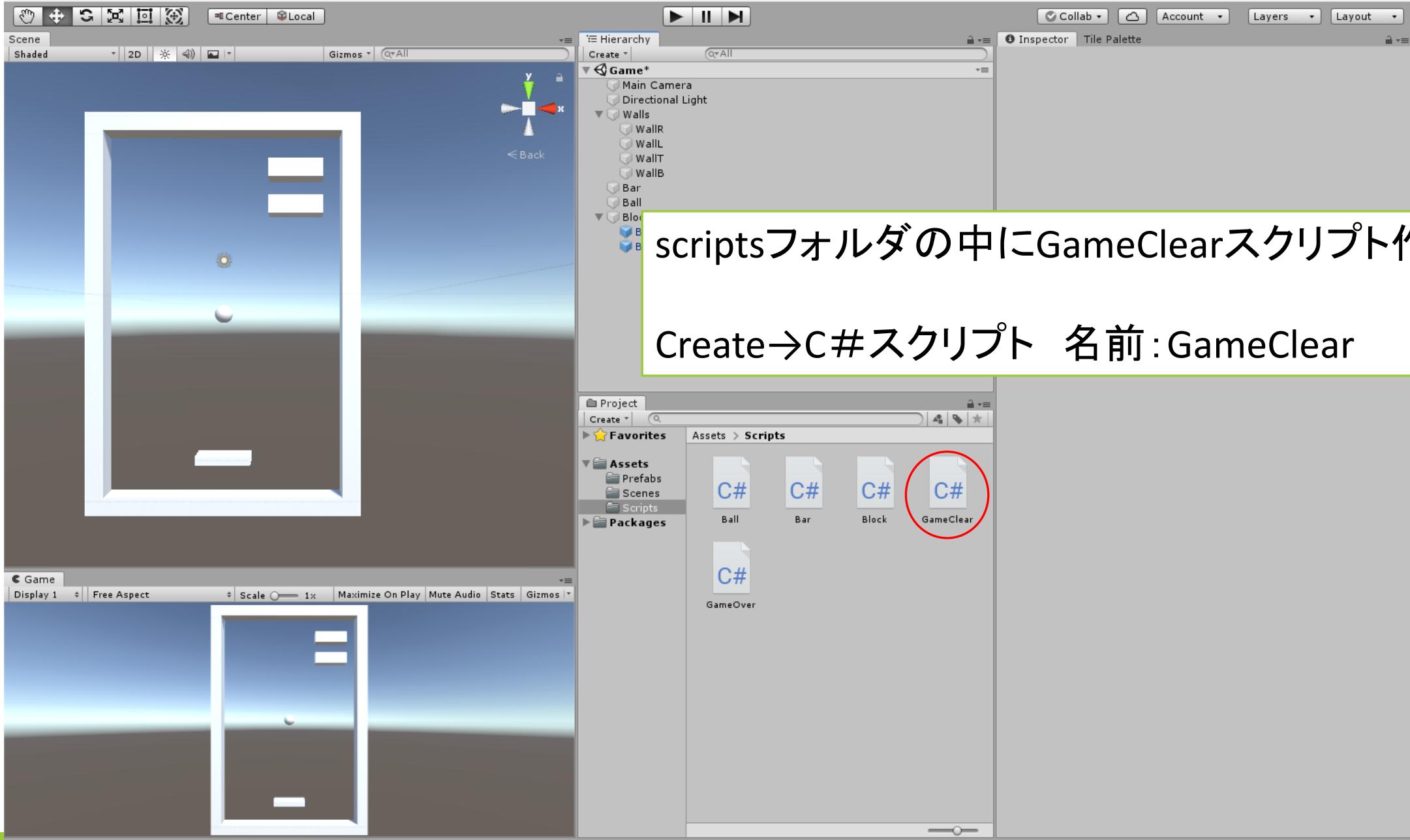
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

class GameOver : MonoBehaviour
{
    // 衝突時に呼ばれる
    void OnCollisionEnter(Collision collision)
    {
        // 当たったゲームオブジェクトを削除する
        Destroy(collision.gameObject);
    }
}
```



ゲームクリアの判定

すべてのブロックを壊すことができれば、ゲームクリア

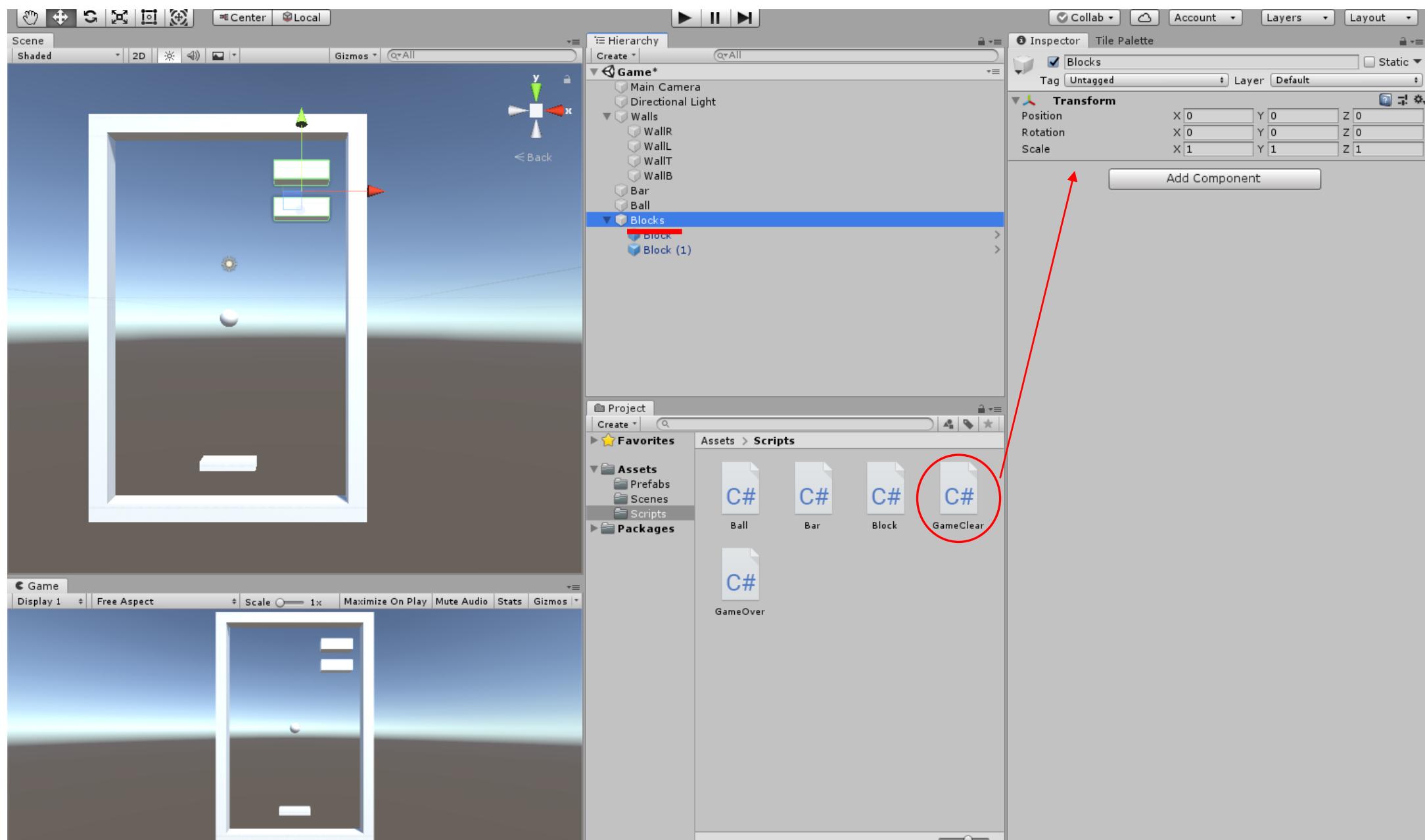


```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

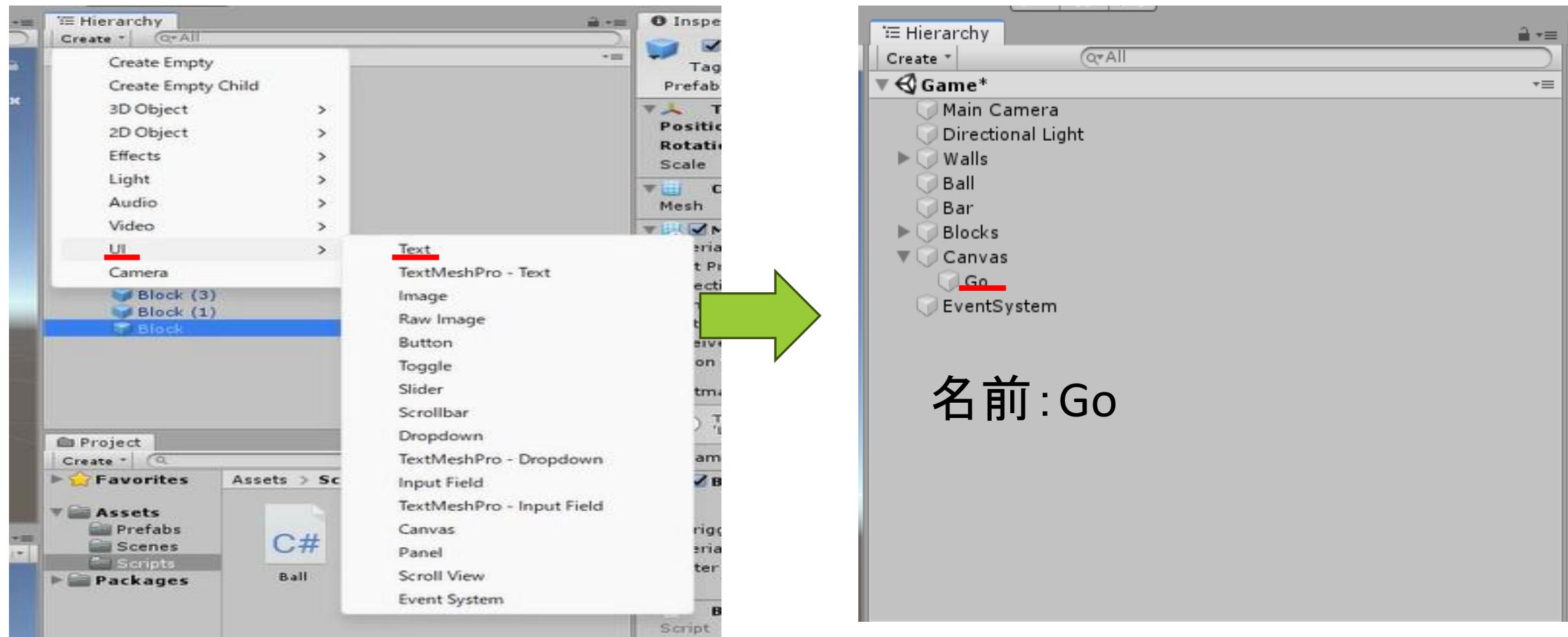
class GameClear : MonoBehaviour
{
    Transform myTransform;

    void Start()
    {
        // Transformコンポーネントを保持しておく
        myTransform = transform;
    }

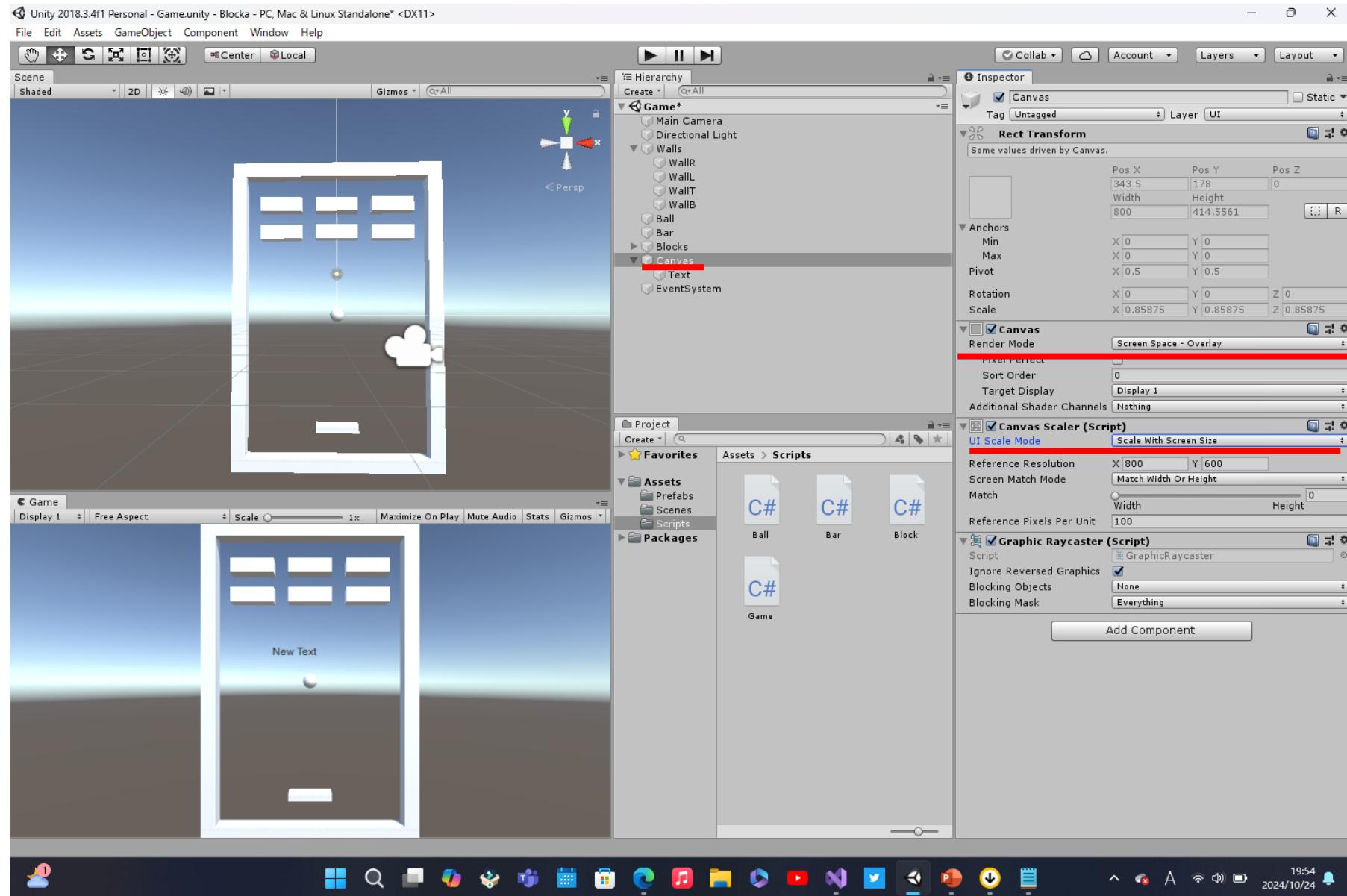
    void Update()
    {
        // 子供がいなくなったらゲームを停止する
        if (myTransform.childCount == 0)
        {
            Time.timeScale = 0f;
        }
    }
}
```



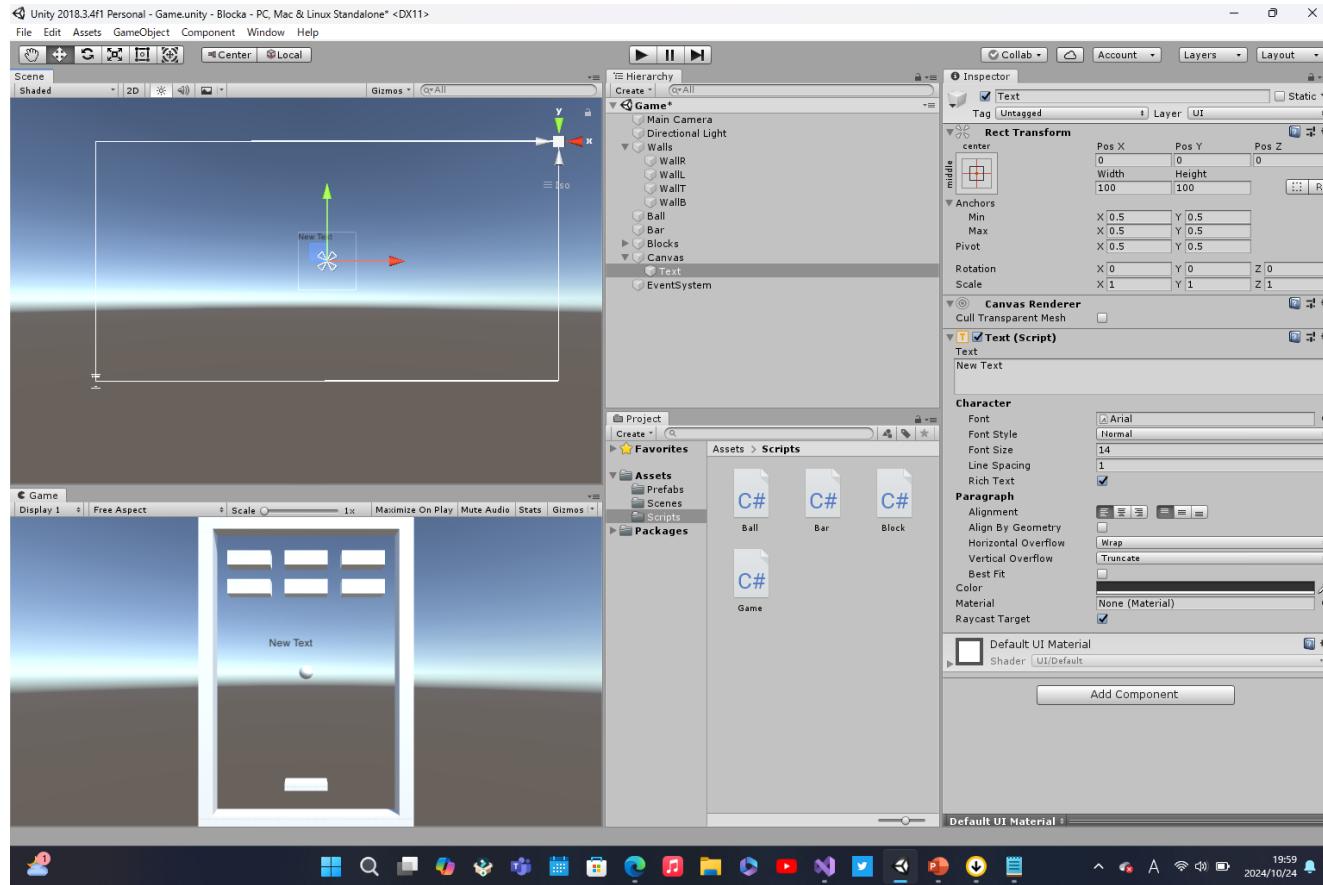
text作成



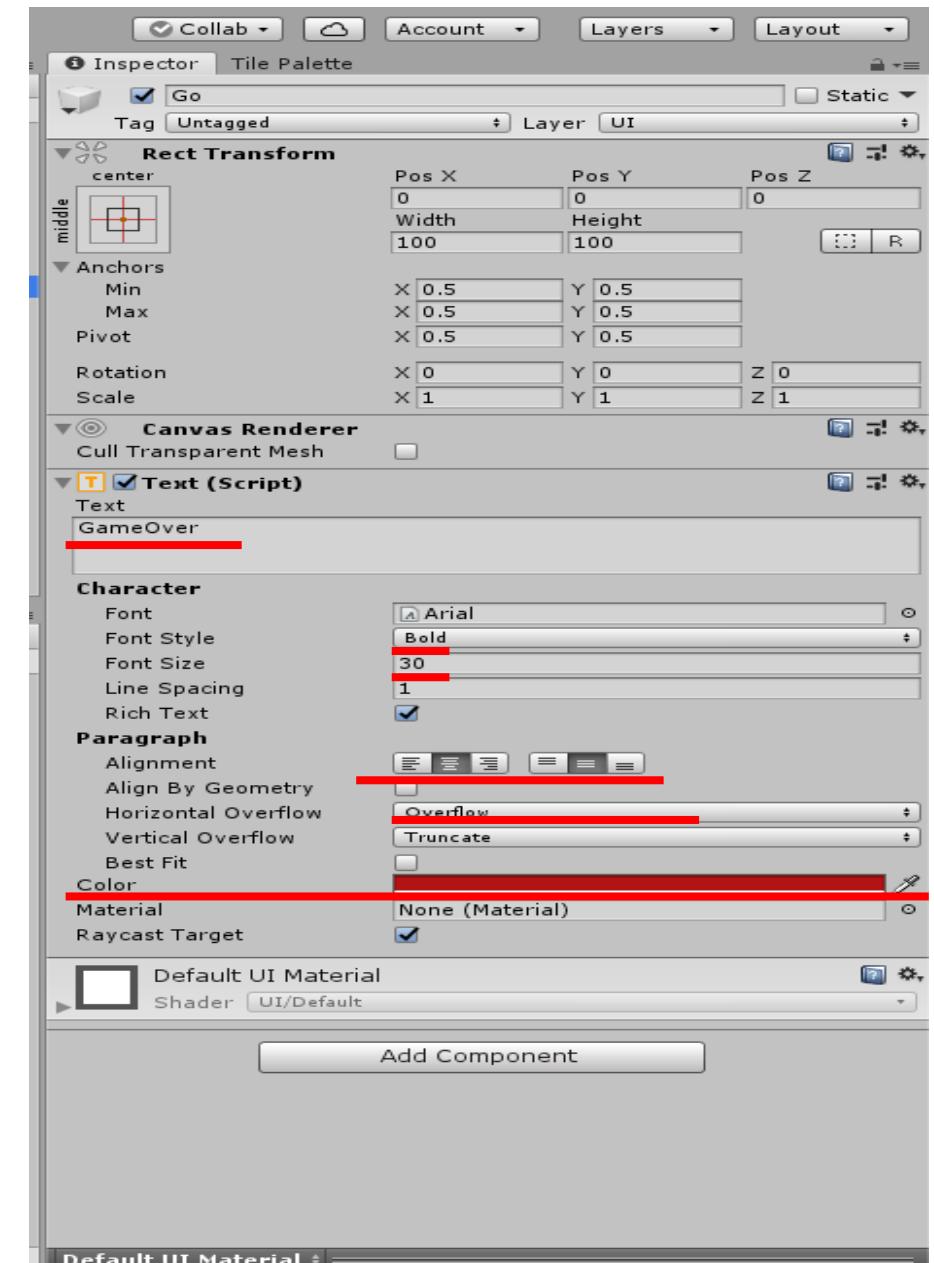
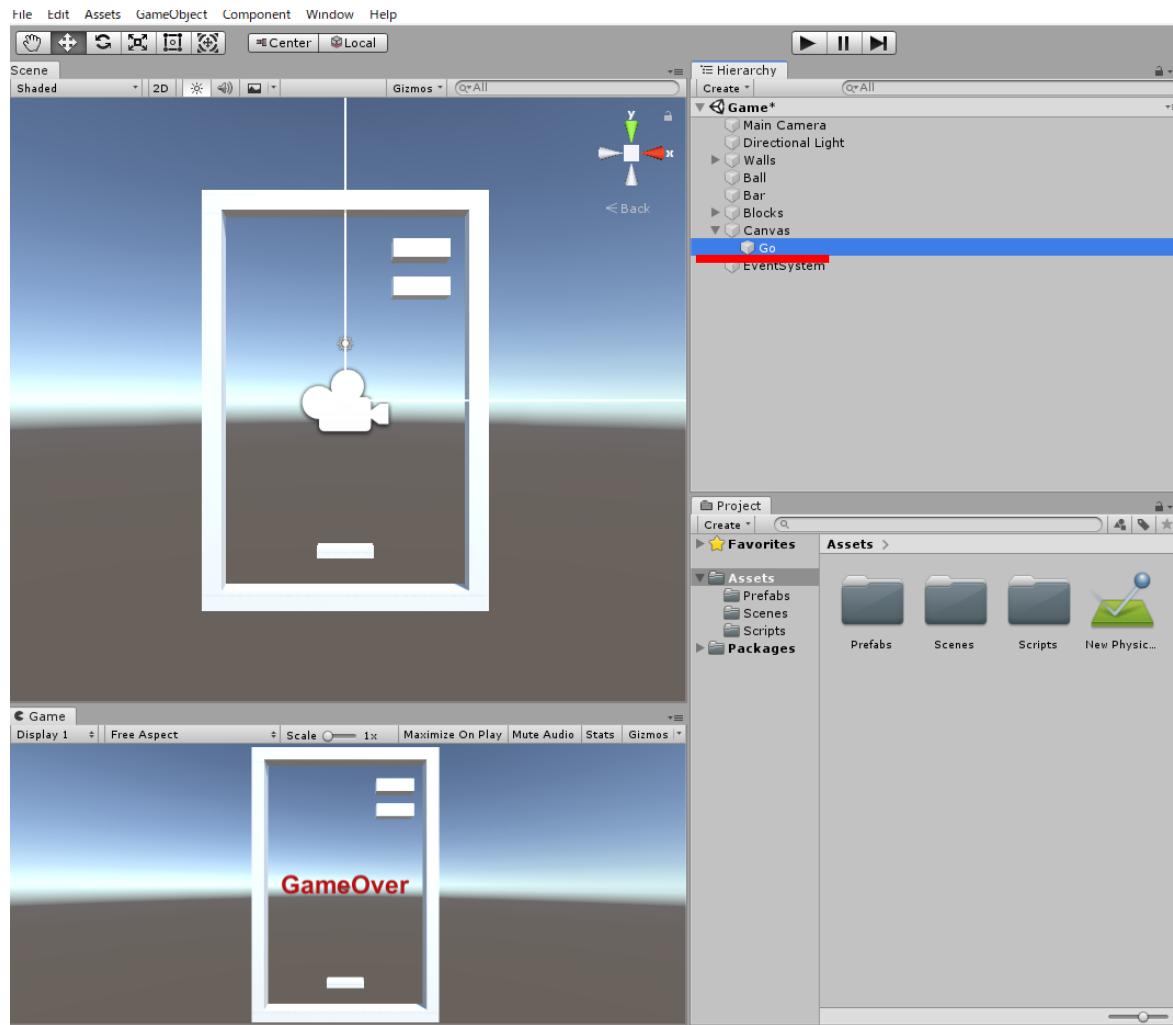
Canvasの設定



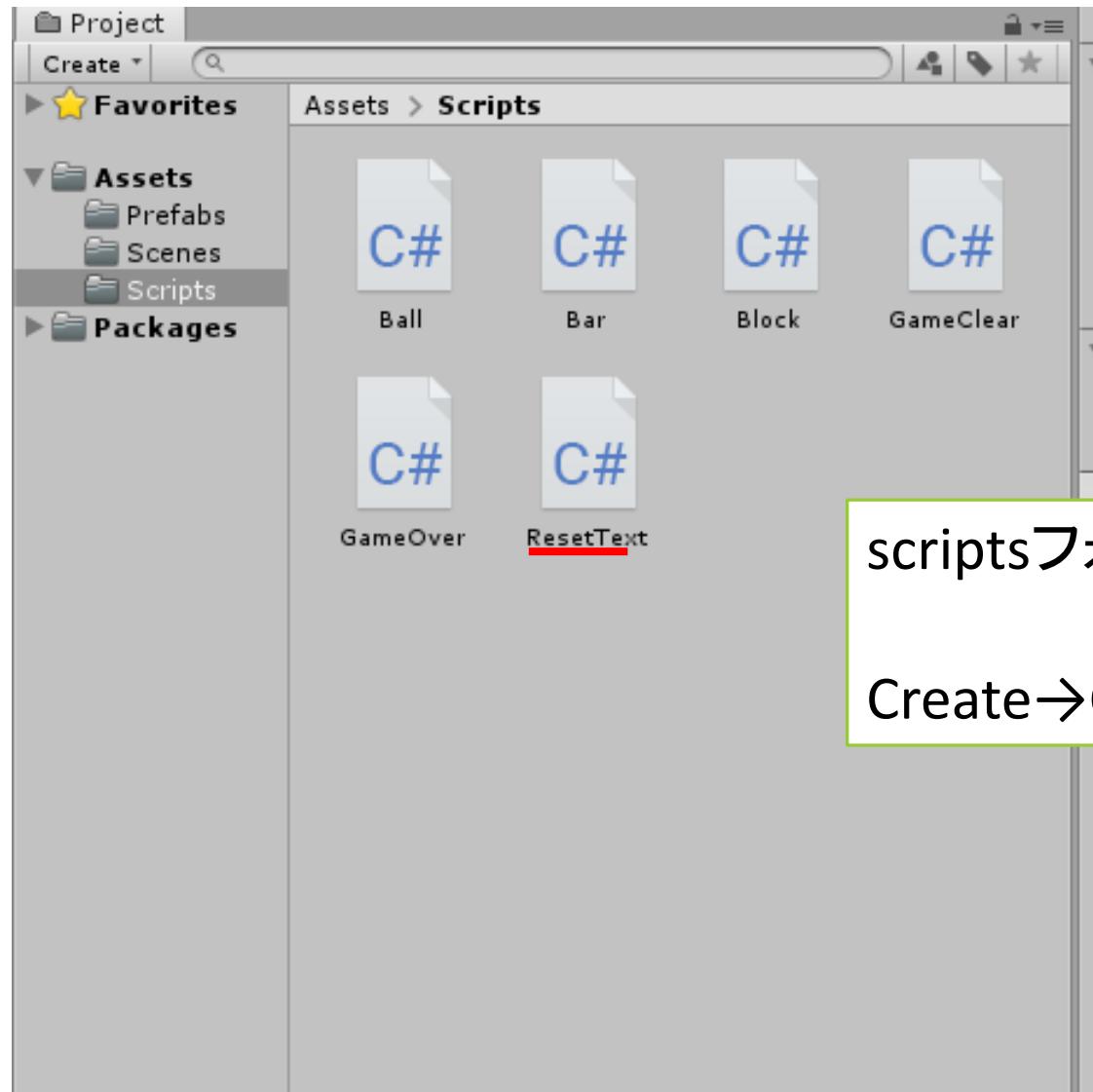
Textの設定



TextダブルクリックするとBlockやPlayerが置いてある位置とは別のところにあることがわかります。Canvasを表す枠線がゲーム画面に対応するので、画面中央に文字が来るように位置を調整しましょう。



Scriptからテキストの内容を変更する

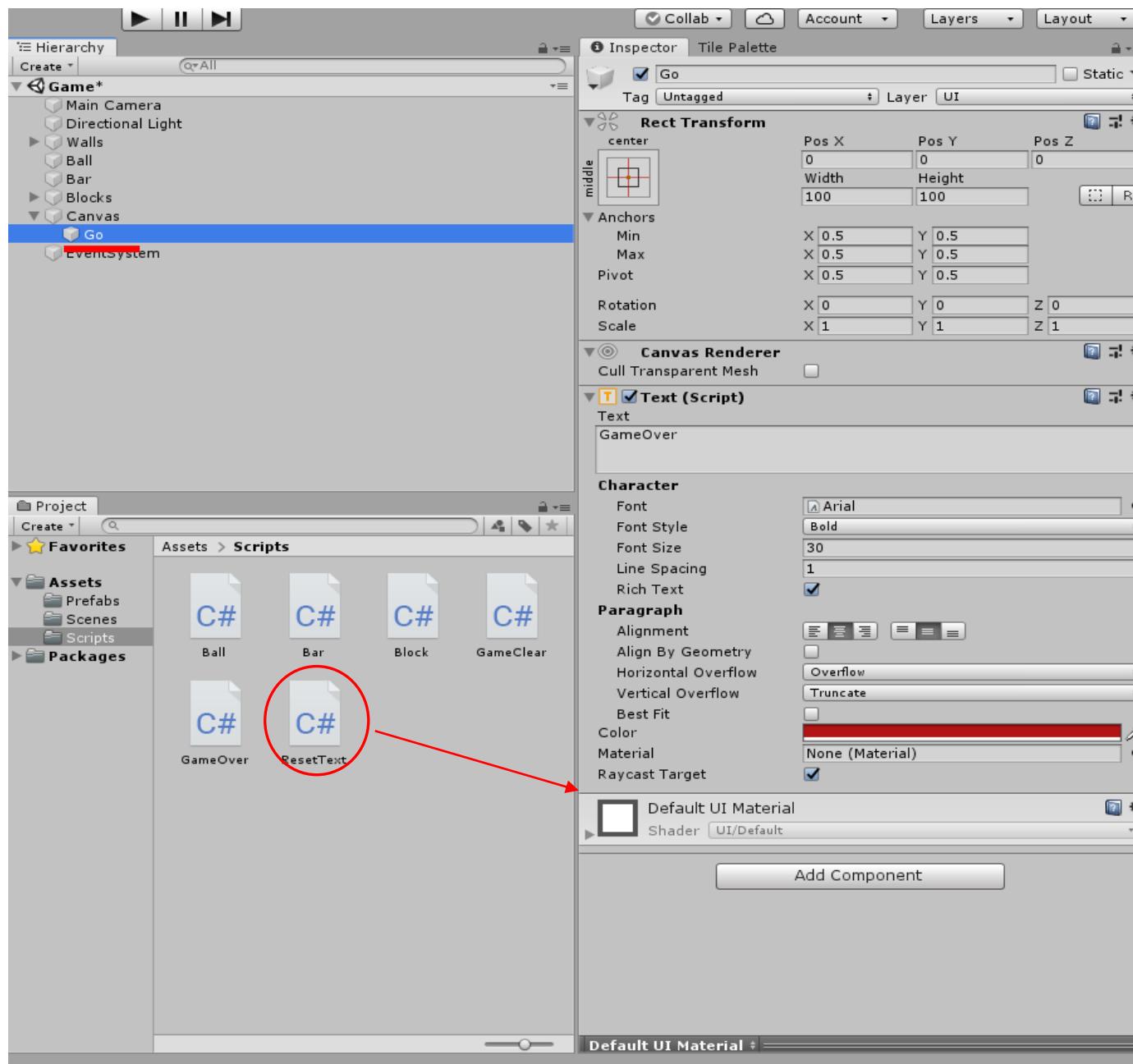


scriptsフォルダの中にResetTextスクリプト作成

Create→C#スクリプト 名前: ResetText

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

class ResetText : MonoBehaviour
{
    void Start()
    {
        // アクセスは1回きりなので、フィールド変数を用意しなくてもいい
        Text myText = GetComponent<Text>();
        // textに空の文字列を設定する
        myText.text = "";
    }
}
```



ゲームオーバー時には「Game Over」に
変更する

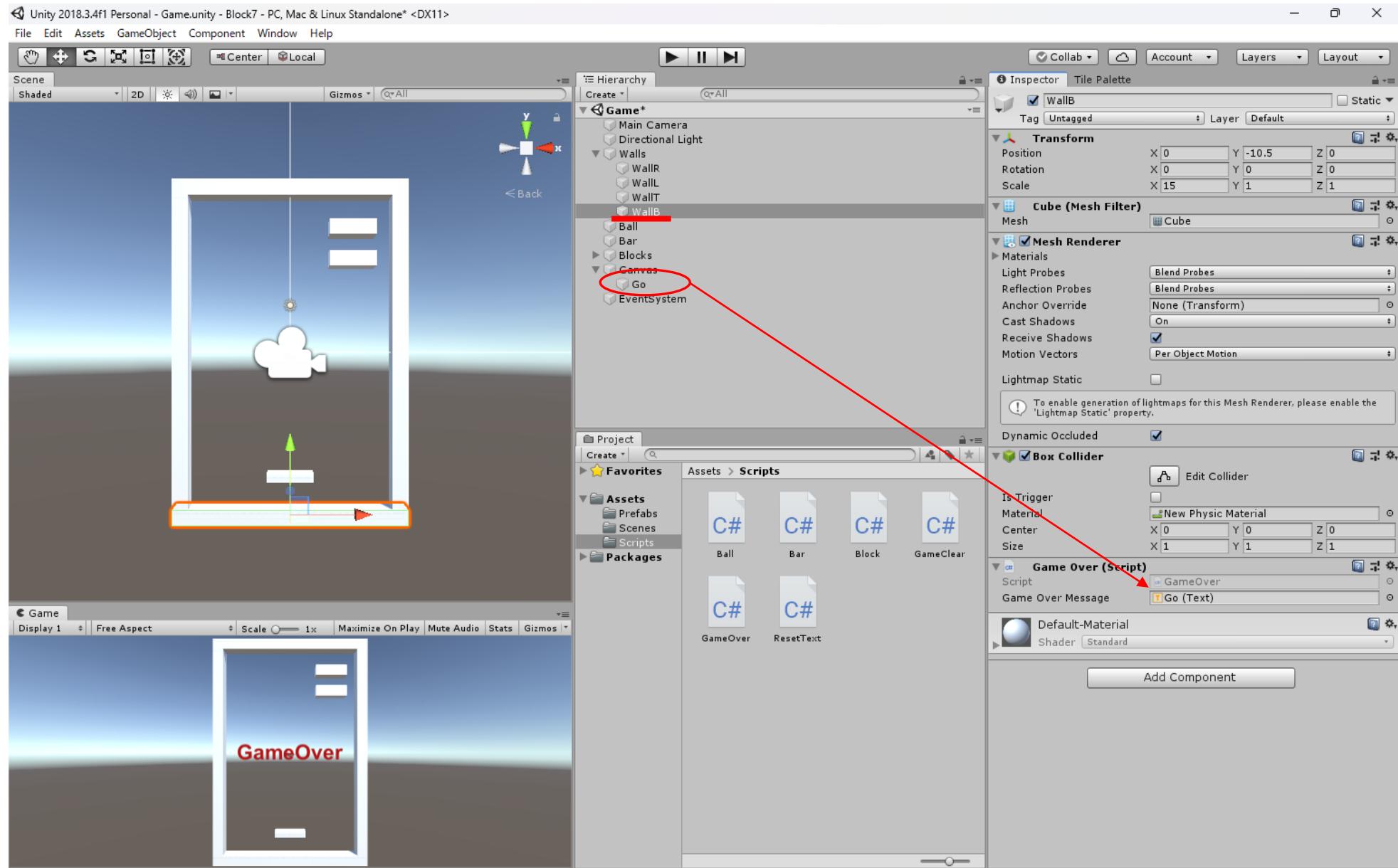
GAMEOVERスクリプト追加

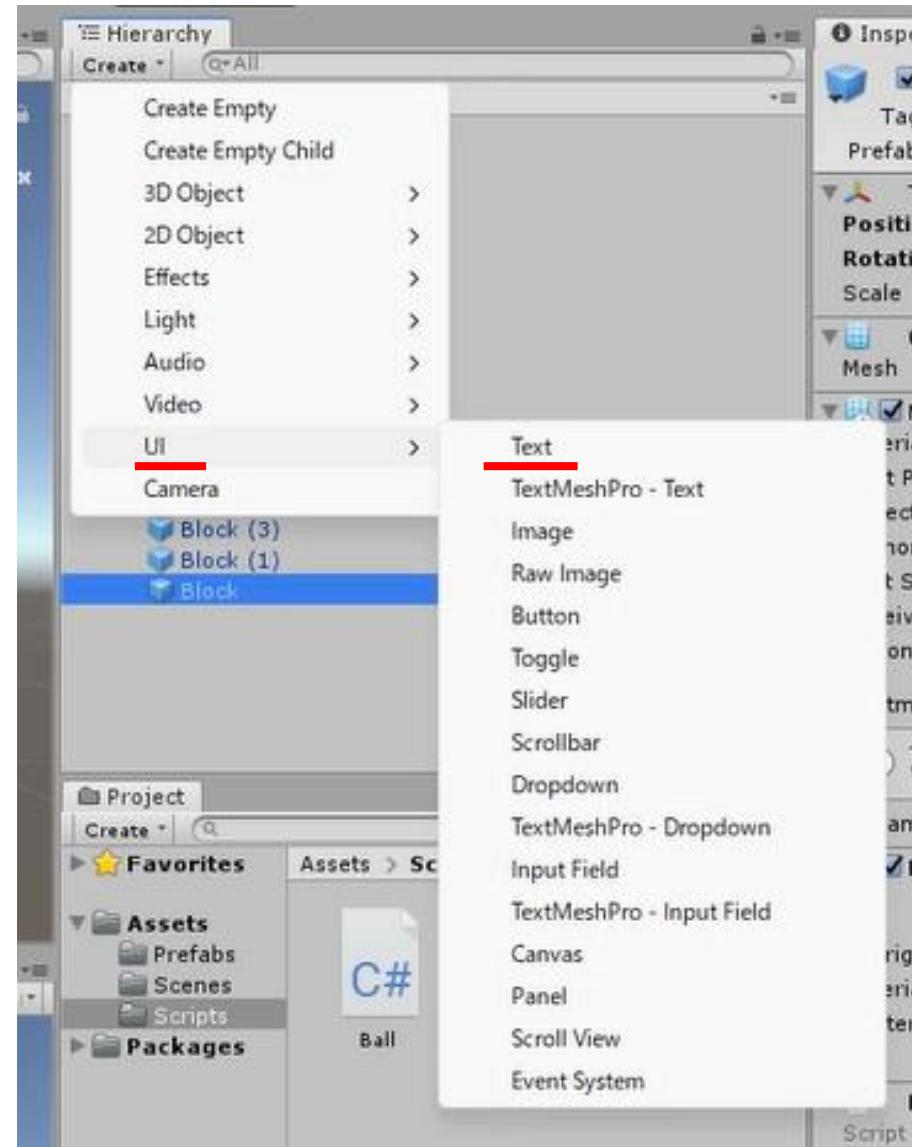
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

class GameOver : MonoBehaviour
{
    // publicにしてInspectorから設定できるようにする
    public Text gameOverMessage;

    // 衝突時に呼ばれる
    void OnCollisionEnter(Collision collision)
    {
        // Game Overと表示する
        gameOverMessage.text = "Game Over";

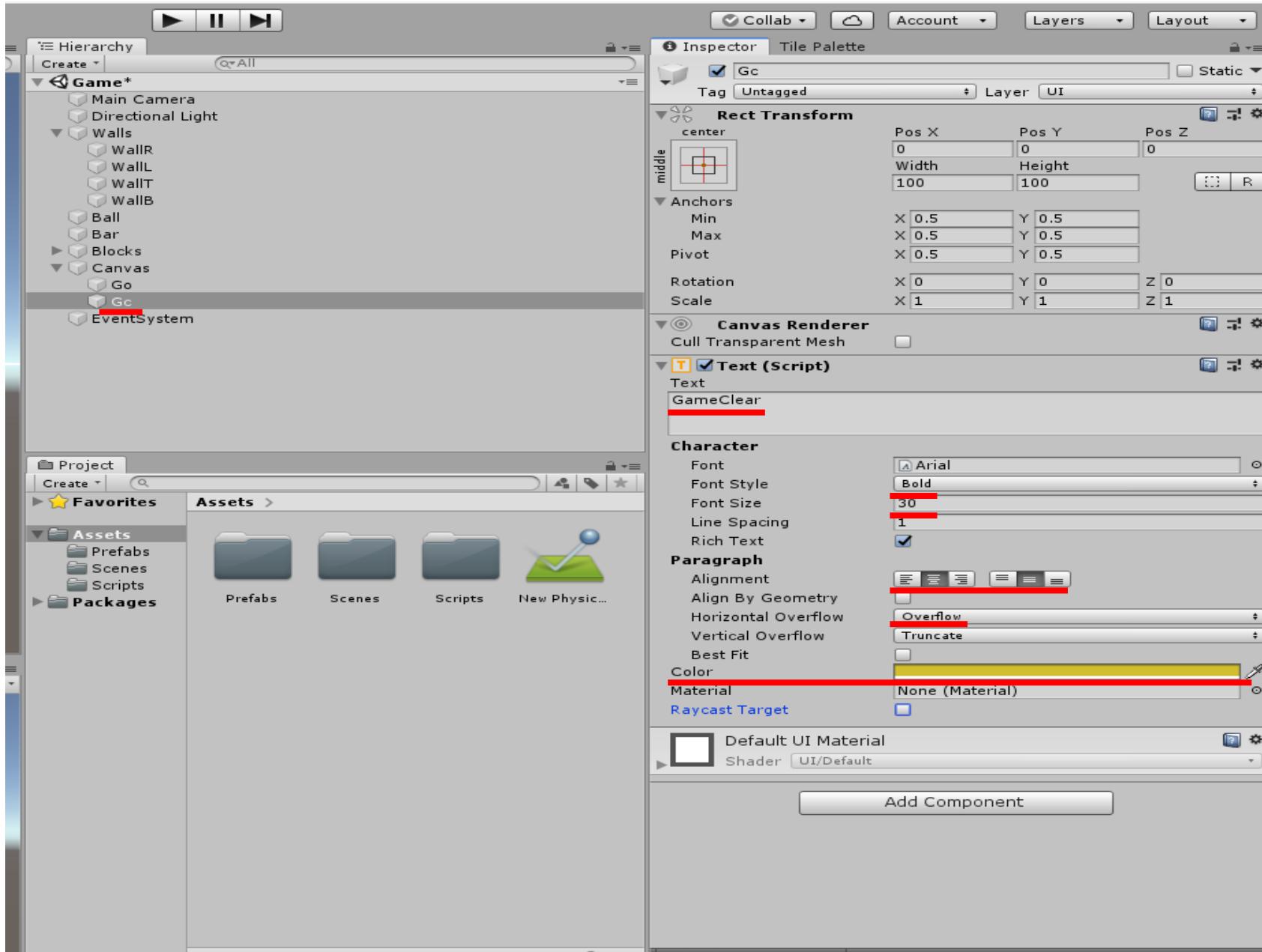
        // 当たったゲームオブジェクトを削除する
        Destroy(collision.gameObject);
    }
}
```

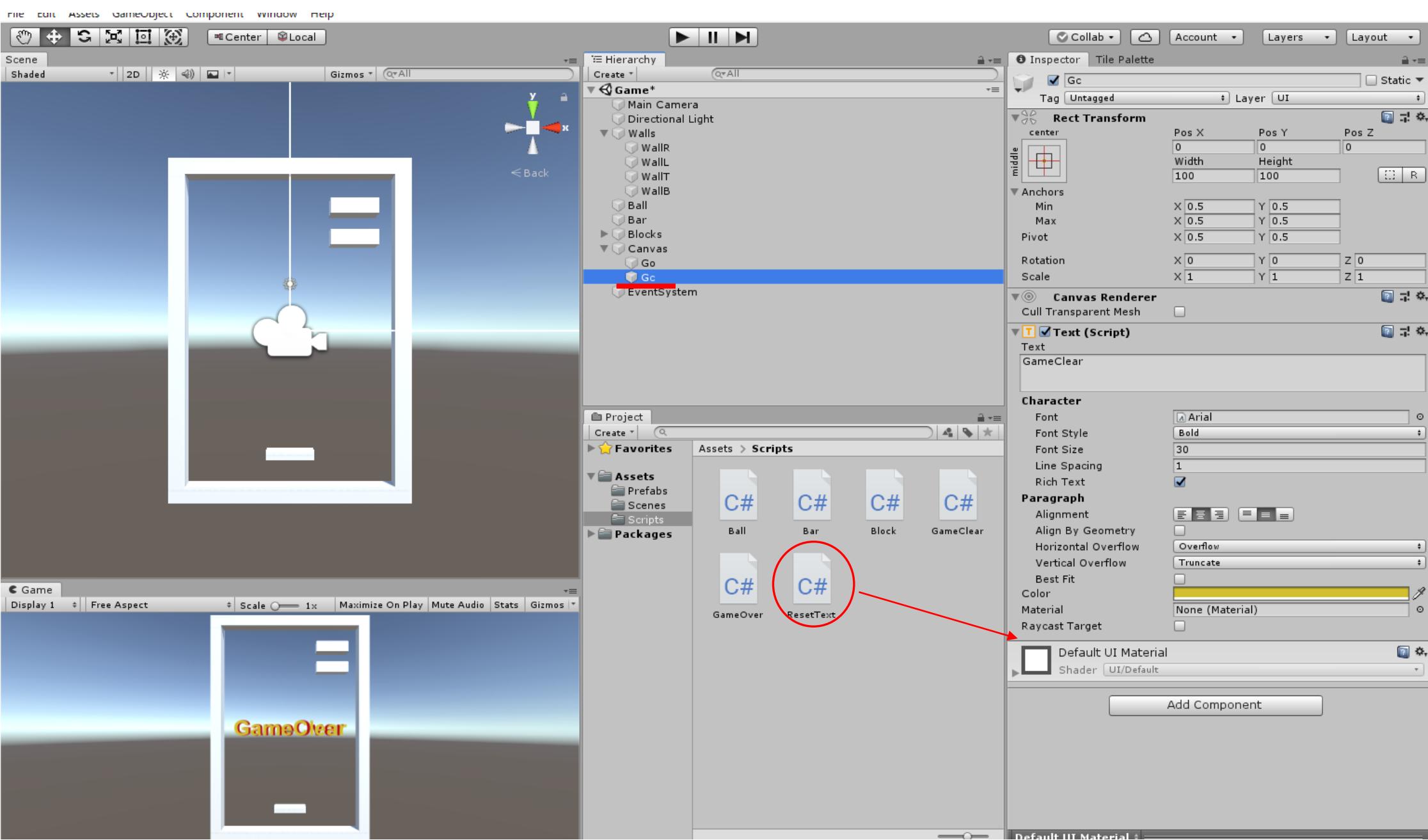




GAMECLEARのテキスト
さつきと同様

名前: Gc





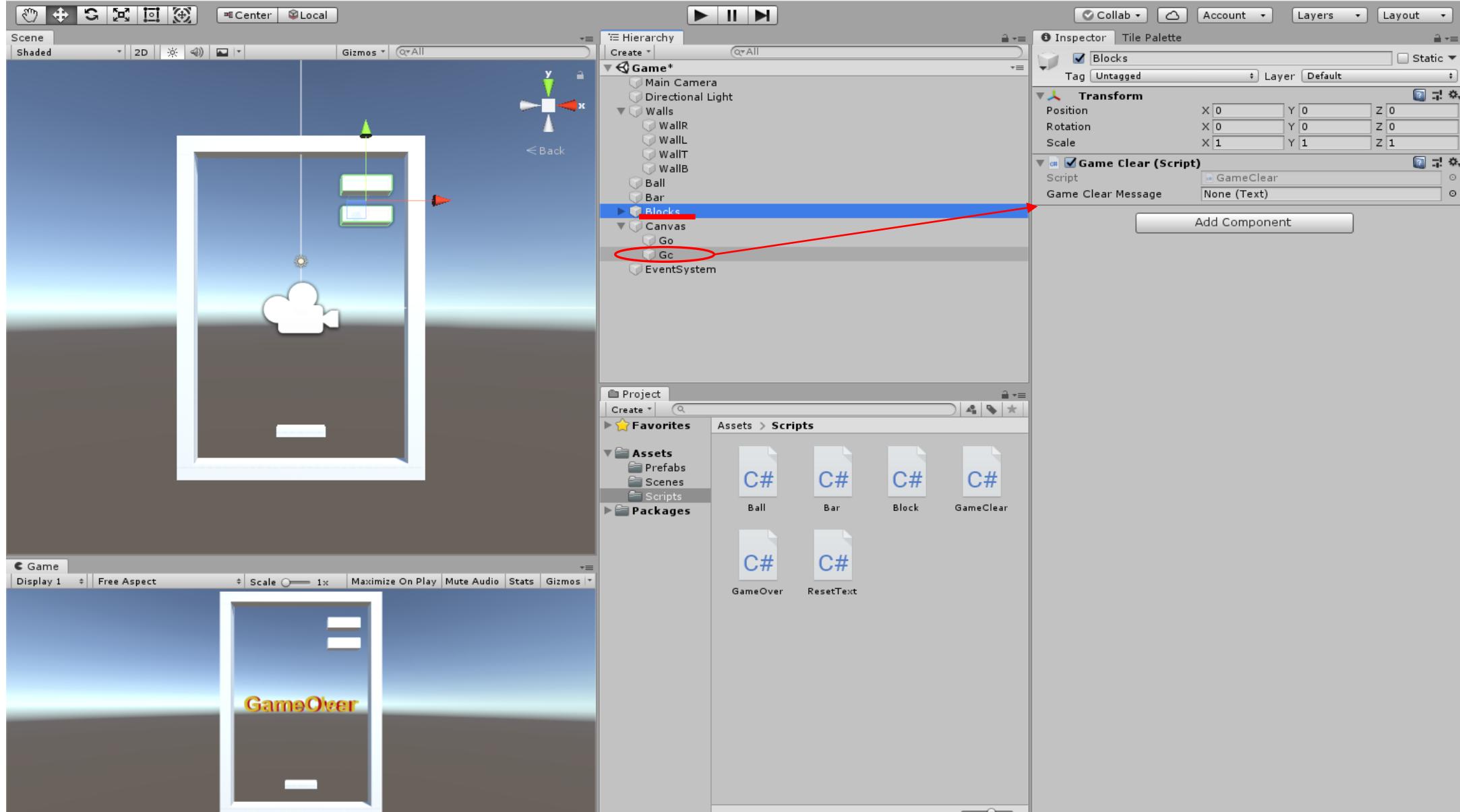
GAMECLEARスクリプト追加

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

class GameClear : MonoBehaviour
{
    public Text gameClearMessage;
    Transform myTransform;

    void Start()
    {
        // Transformコンポーネントを保持しておく
        myTransform = transform;
    }

    void Update()
    {
        // 子供がいなくなったらゲームを停止する
        if (myTransform.childCount == 0)
        {
            gameClearMessage.text = "Game Clear";
            Time.timeScale = 0f;
        }
    }
}
```



自由制作

- ・ブロックの位置・数の変更
- ・Ball・Barのスピード変更
- ・カメラの位置変更
- ・消えないブロックを作る
- ・色を付ける

参考文献

[Unityでできることとは？活用事例や面白い使い方も紹介 | 侍エンジニアブログ](#)

[Unityプログラミング講座 | GSD ONLINE | 広島大学ゲーム制作同好会](#)