演 習 １ ： U N I T Y の 基 礎

(01)　04/12

１Ａ｜Unityとエディタの連携

(02) 04/19

１Ｂ｜Transform・キーイベント・マウスイベント

(03) 04/26

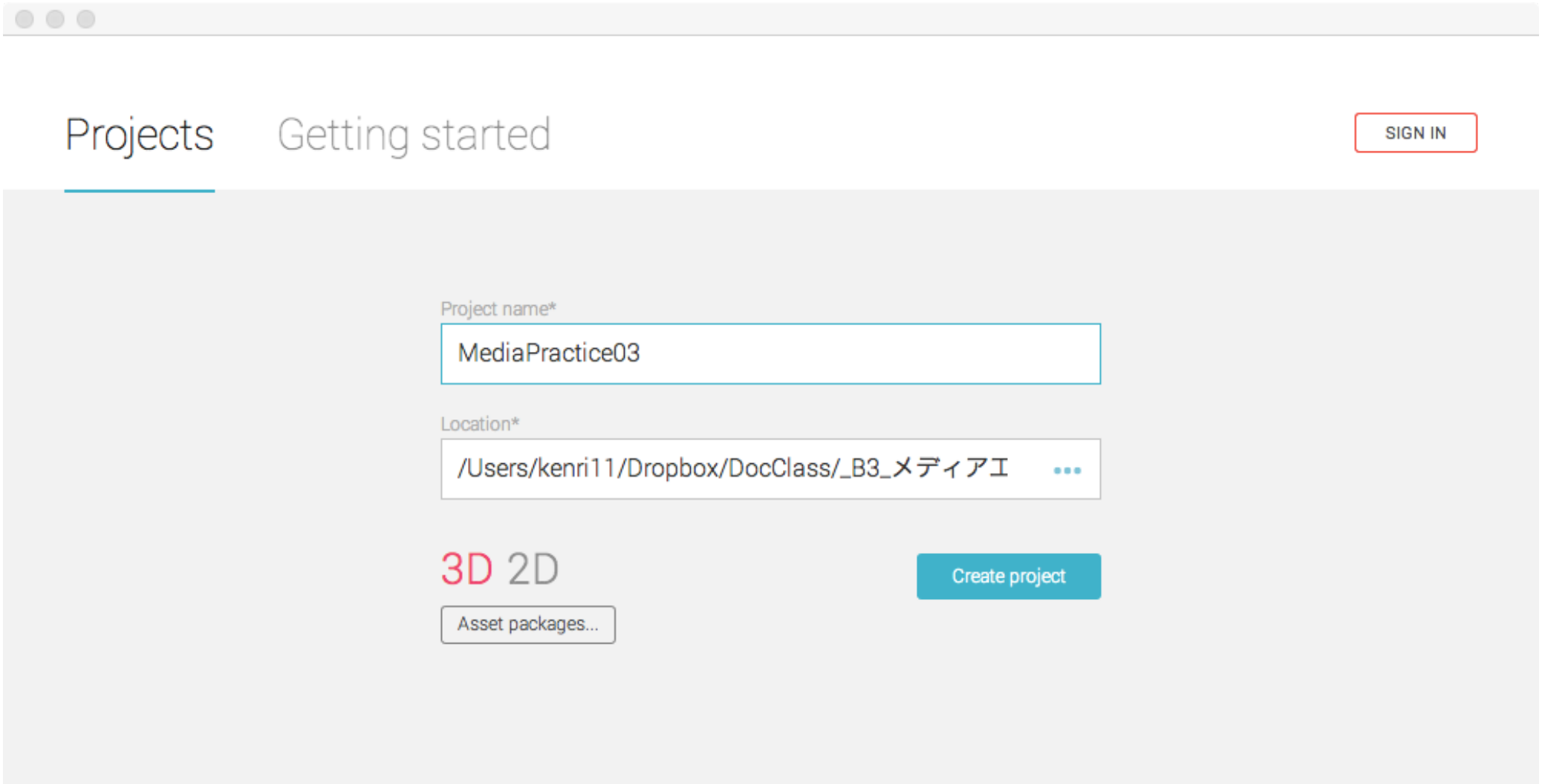
１Ｃ｜剛体特性・カメラの視点

(04)　05/10

１Ｄ｜プレハブ（gameobjectの雛形）, タグ, その他

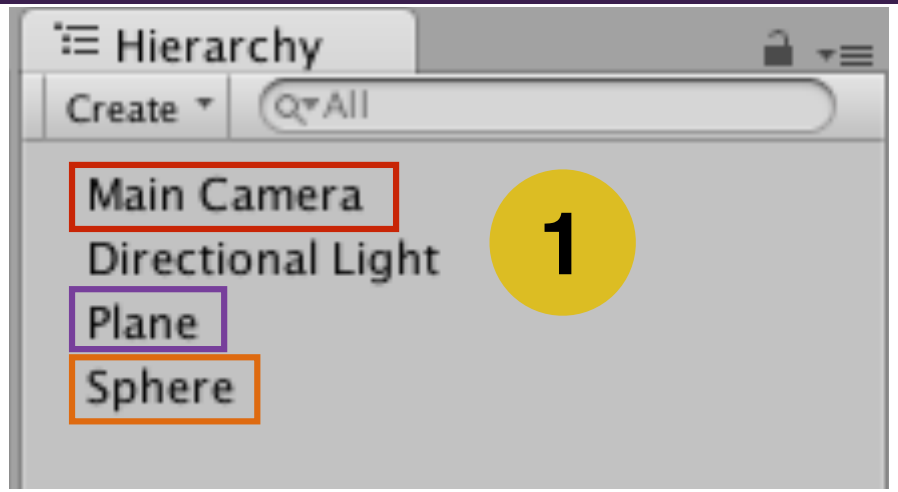
MediaPractice03

剛体特性（ゲームエンジン）・カメラの視点

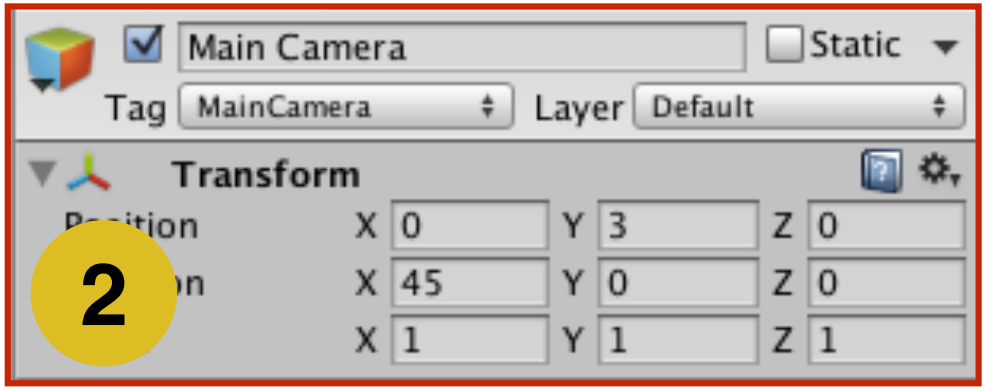


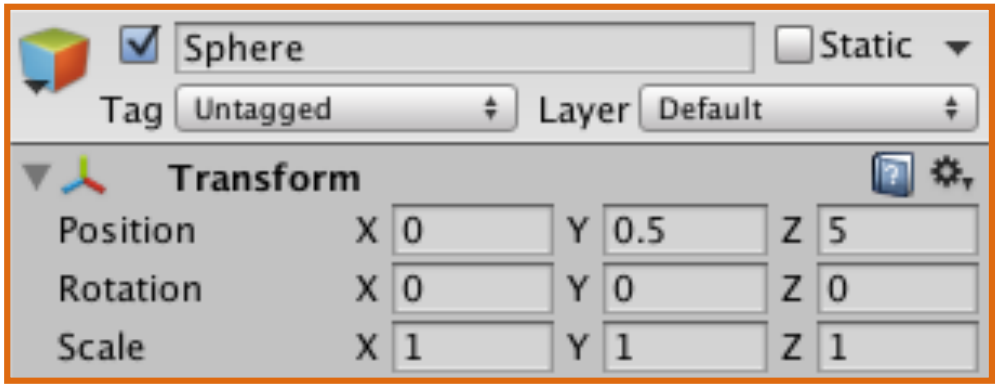
準備

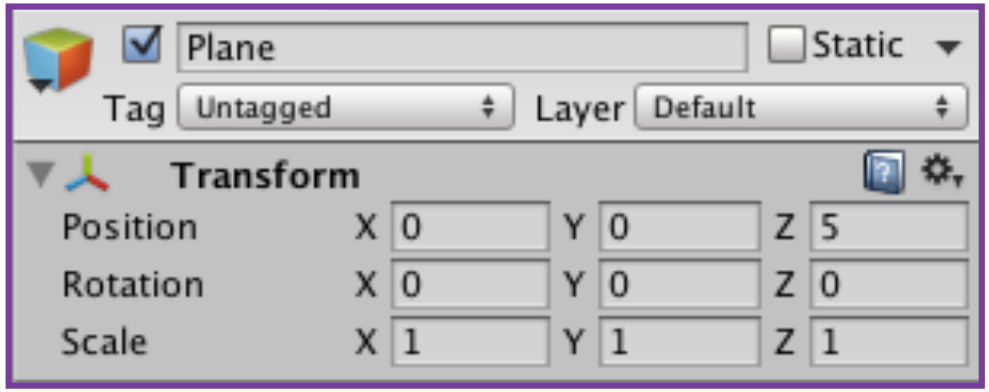
デフォルトで用意されているメインカメラとライトに加えて, 新たにPlane（地面）とSphere（球）をHierarchyビューに追加します（適当に名前を変えてもらって結構です）.



各ゲームオブジェクトのtransformを以下のように変更してください.

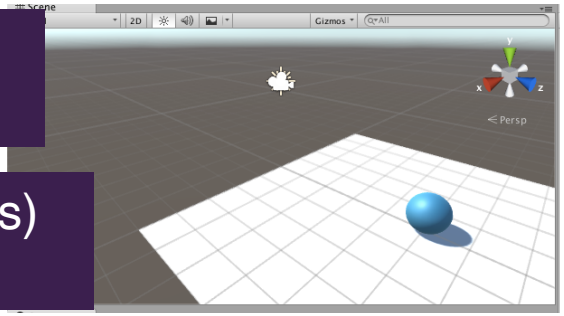






3 Sphereに適当な色のmaterialを関連付けてください（p162）.

4 Sphereにスクリプト (rigidscript.cs) を関連付けてください（p165）.

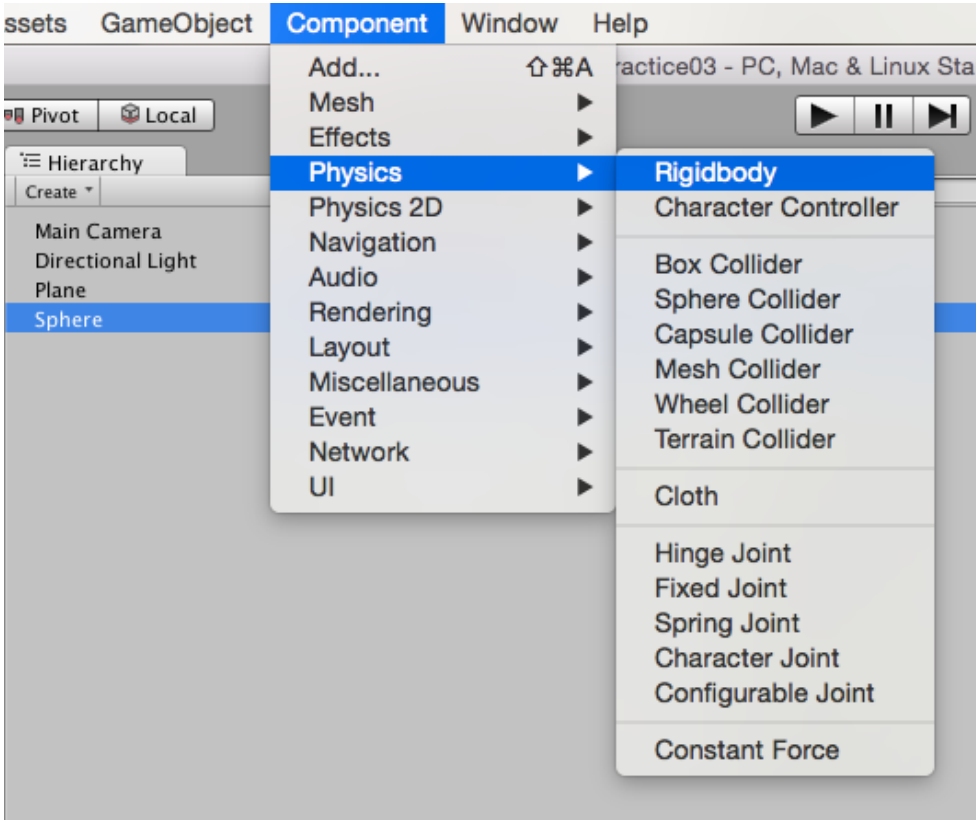


剛体特性の組み込み

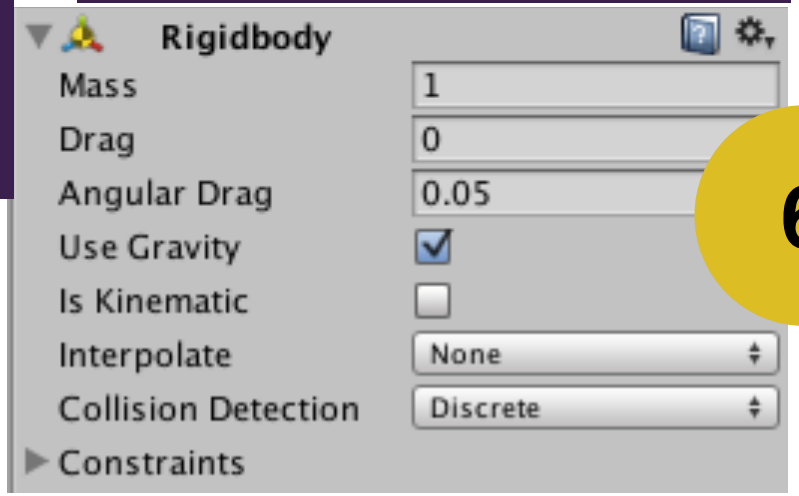
- 特定のゲームオブジェクトに剛体特性を組み込むと, そのオブジェクトは, 外部から物理的な力（衝突・摩擦）を受けるようになります.

5

Hierarchyビューで, Sphereを選択した状態で, メニューより, <Component>→<Physics>→<Rigidbody>を選びます.



Rigidbodyのパラメータは初期状態のままでOKです.



6

Rigidbody	
Mass	質量
Drag	空気抵抗
Use Gravity	重力のON/OFF
Constraints	特定の軸の移動・回転を凍結します.

剛体に外力を加える

- gameobjectの（事後に加えられた）コンポーネントに関するobjectを取り出す場合, 以下の書式を使います.

GameObject

Class GetComponent<classname>()

Rigidbody GetComponent<Rigidbody>()

ゲームオブジェクトのコンポーネントのうち, classnameに対応するクラスに対応するオブジェクトを取り出すためのメソッド.

Rigidbody

void AddForce (float x, float y, float z)

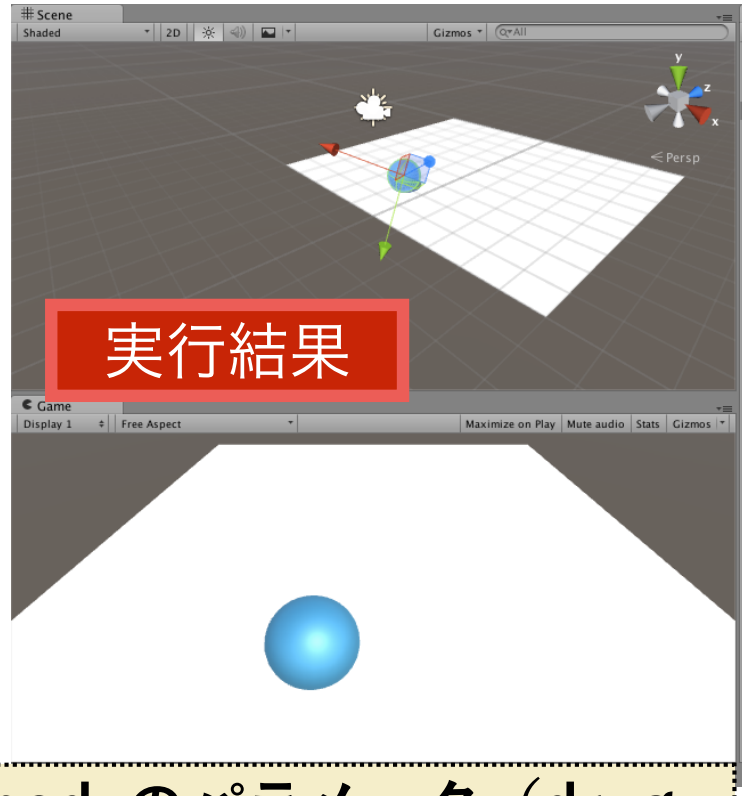
剛体に対して, 3つの軸に対応する外力を加える.

rigidscript.cs

```
public class rigidscript : MonoBehaviour {

    void Start () {
    }

    void Update () {
        if (Input.GetKey (KeyCode.UpArrow)) {
            this.GetComponent<Rigidbody> ().AddForce (0, 0, 1);
        }
        if (Input.GetKey (KeyCode.DownArrow)) {
            this.GetComponent<Rigidbody> ().AddForce (0, 0, -1);
        }
        if (Input.GetKey (KeyCode.RightArrow)) {
            this.GetComponent<Rigidbody> ().AddForce (1, 0, 0);
        }
        if (Input.GetKey (KeyCode.LeftArrow)) {
            this.GetComponent<Rigidbody> ().AddForce (-1, 0, 0);
        }
    }
}
```



上下左右で, 球に力が加わります. rigidbodyのパラメータ（drug, useGravity）を変化させたときの挙動を確認してください.

カメラの追従１

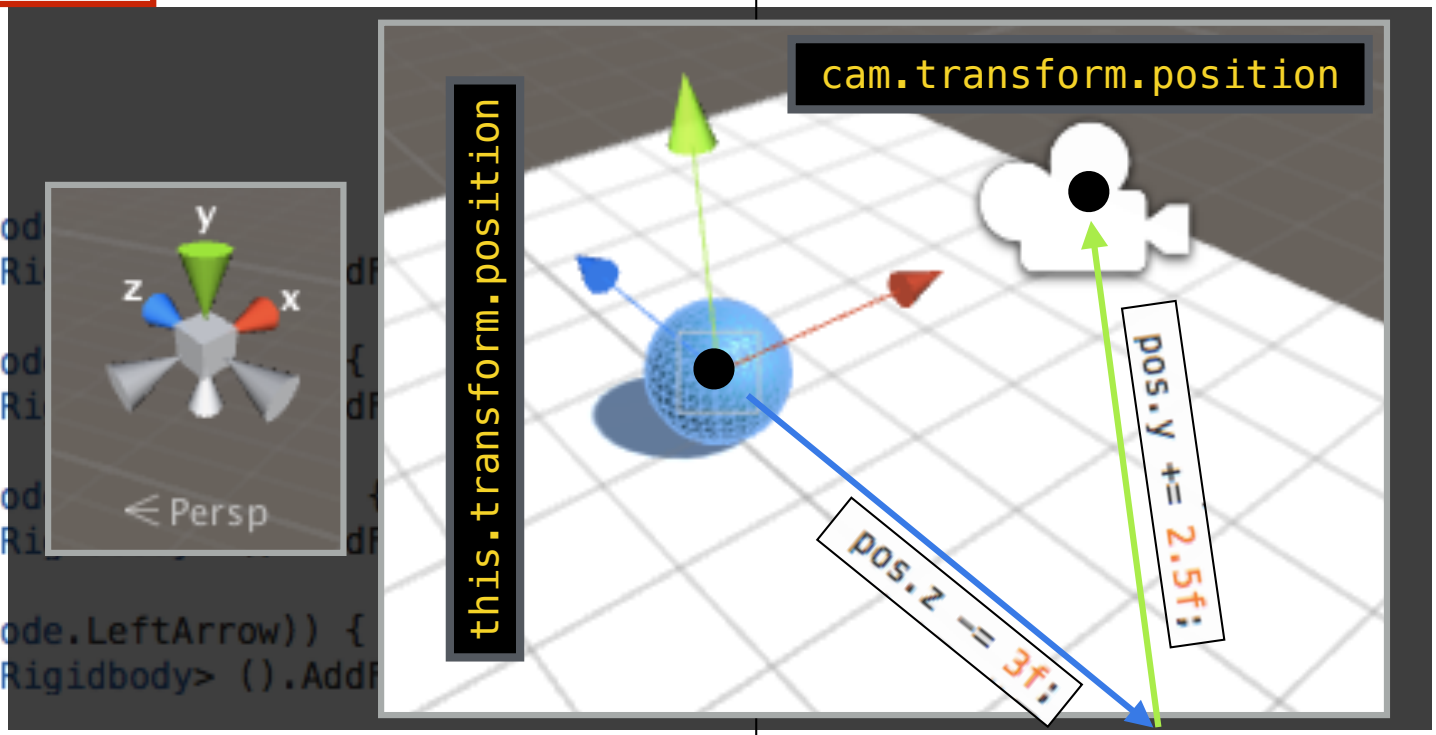
- 赤枠の部分をrigidscript.csに追記してください.

```
public class rigidscript : MonoBehaviour {
    //メインカメラ（GUI上で直接関連付ける）
    public GameObject cam = null;

    void Start () {
    }

    void Update () {
        if (Input.GetKey (KeyCod
            this.GetComponent<Ri
        }
        if (Input.GetKey (KeyCod
            this.GetComponent<Ri
        }
        if (Input.GetKey (KeyCod
            this.GetComponent<Ri
        }
        if (Input.GetKey (KeyCode.LeftArrow)) {
            this.GetComponent<Rigidbody> ().AddF
        }
        //Sphereの現在の位置
        Vector3 pos = this.transform.position;
        pos.y += 2.5f; //2.5上方
        pos.z -= 3f;   //3.0後方
        //カメラの位置をSphereの斜め上後方とする.
        cam.transform.position = pos;
    }
}
```

メインカメラとSphereの位置関係

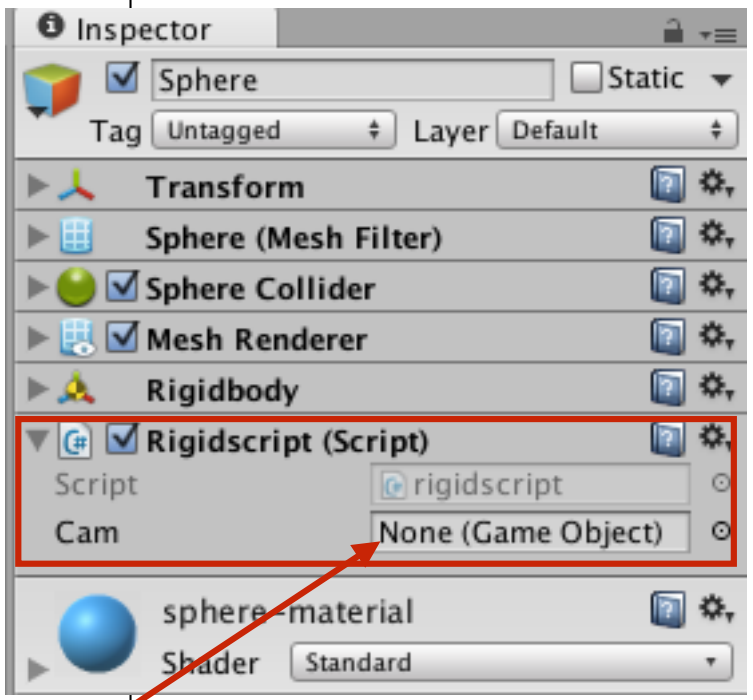


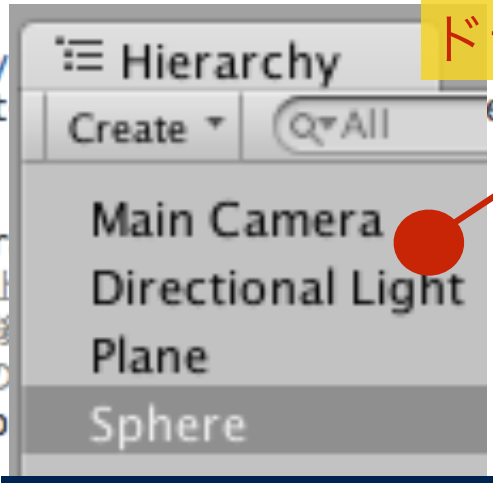
rigidscript.cs

カメラの追従１

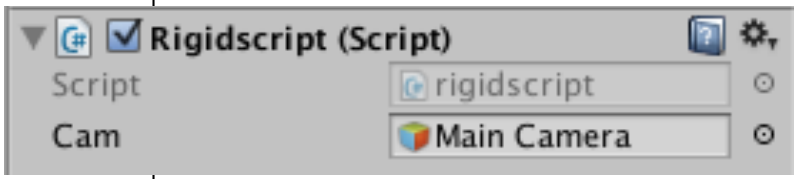
```
public class rigidscript : MonoBehaviour {
    //メインカメラ（GUI上で直接関連付ける）
    public GameObject cam = null;
        }
        if (Input.GetKey (KeyCode.DownArrow)) {
            this.GetComponent<Rigidbody> ().AddForce (0, 0,
        }
        if (Input.GetKey (KeyCode.RightArrow)) {
            this.GetComponent<Rigidbody> ().AddForce (1, 0,
        }
        if (Input.GetKey (Key
            this.GetComponent
        }
        //Sphereの現在の位置
        Vector3 pos = this.tr
        pos.y += 2.5f; //2.5上
        pos.z -= 3f;   //3.0後
        //カメラの位置をSphereの
        cam.transform.positio
    }
}
```

グローバル変数に，内容が未定のゲームオブジェクトを定義しておきます．この時点ではまだ，RigidscriptのCamには何も関連付けられていないことに注意してください「None（Game Object）」．





ドラッグ&ドロップ

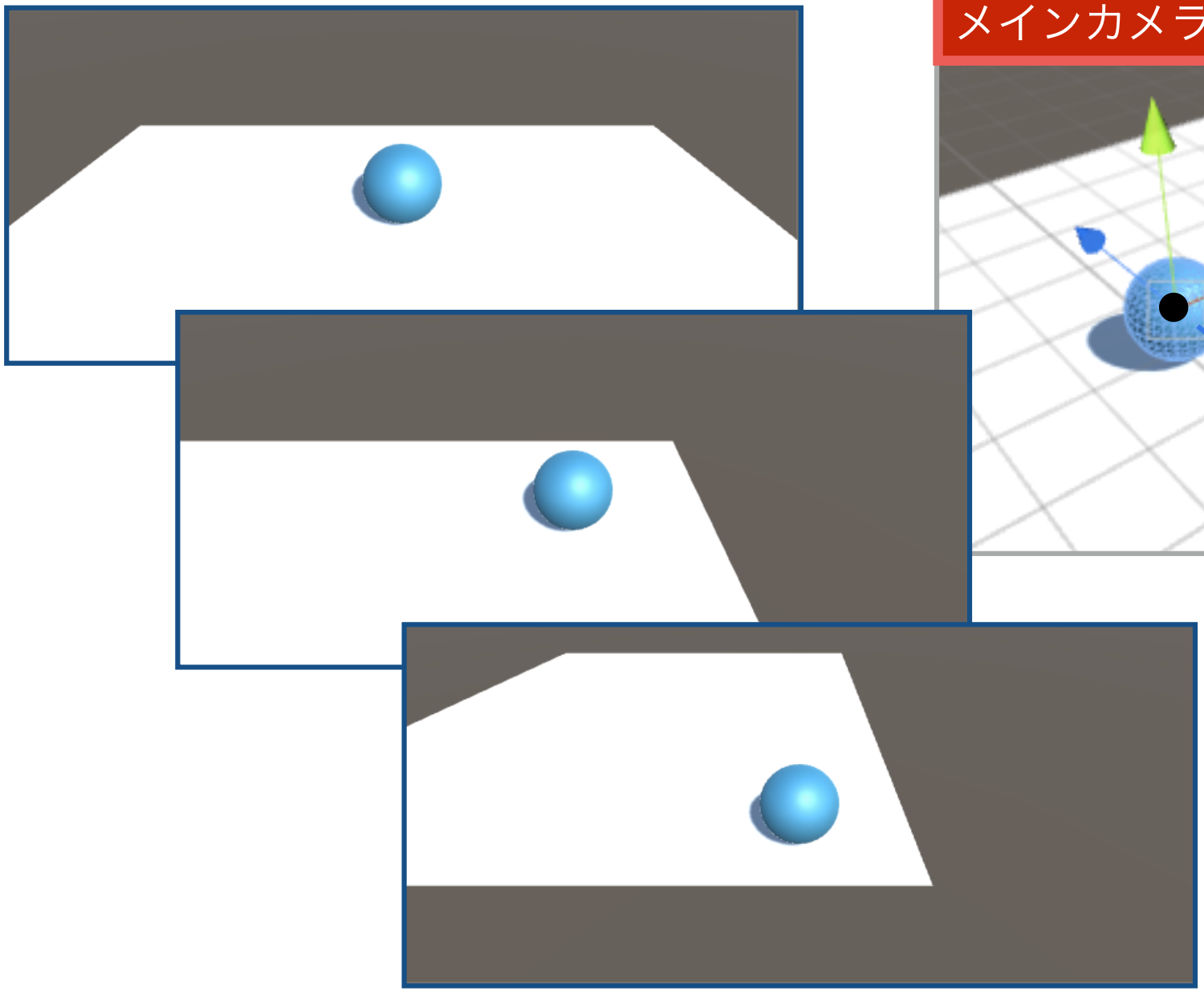


Sphereを選択した状態で，Camの領域に，Main Cameraを直接ドラッグ&ドロップすることによって，rigidscript.cs内のcamとメインカメラが正しく対応付けられます．スクリプトから名前で関連づけて読み込む場合は，Findを使います．

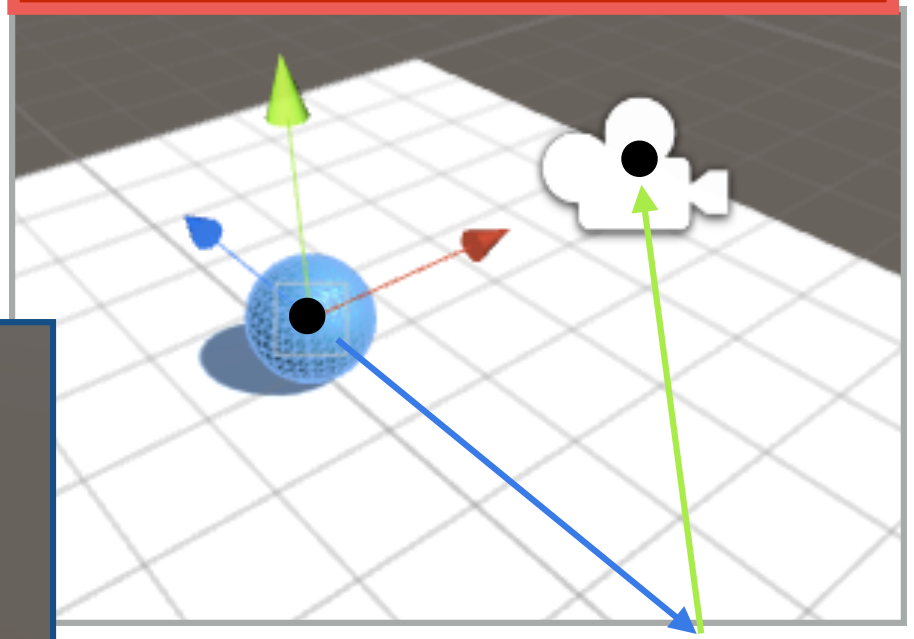
rigidscript.cs

カメラの追従１

実行結果

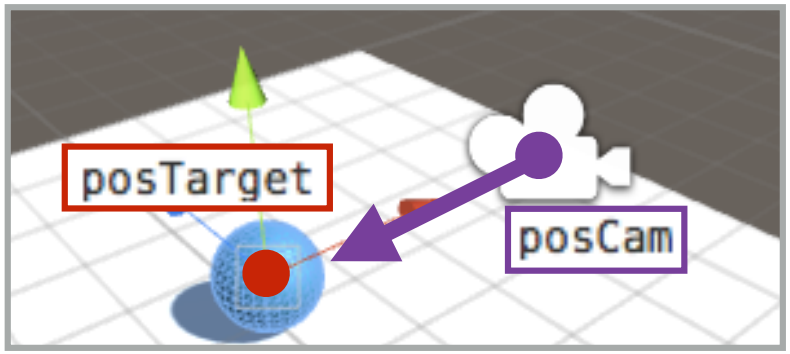


メインカメラとSphereの位置関係



カメラの追従２（LookRotation）

- rigidscript.cs の Update関数の後半部分を以下のように修正すると, カメラの位置は保ったまま, 常にボールの中心へとカメラの視点を向けるようになります.

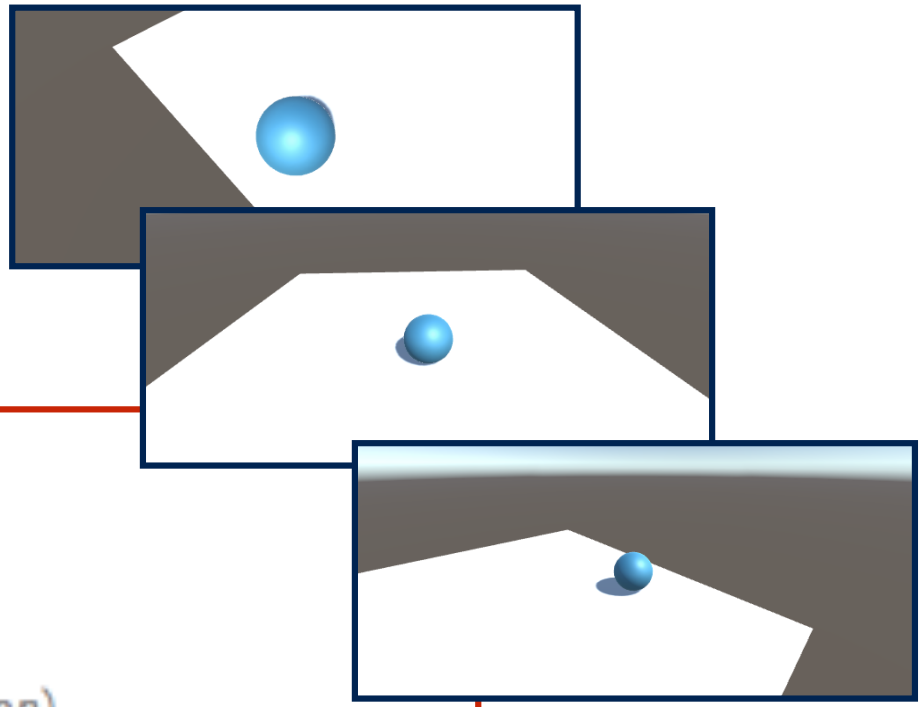



```
//Sphereの現在の位置
Vector3 pos = this.transform.position;
pos.y += 2.5f; //2.5上方
pos.z -= 3f;   //3.0後方
//カメラの位置をSphereの斜め上後方とする.
cam.transform.position = pos;
```

```
Quaternion.LookRotation
(posTarget - posCam);
```

Quaternion ＊**LookRotation**(Vector3 posTarget – Vector3 pos)

pos から posTargetを向いた時の角度を（Quaternion型）で取得.



```
//Sphereの現在の位置
Vector3 posTarget = this.transform.position;
//Cameraの位置
Vector3 posCam = cam.transform.position;

//posCamからposTargetを向いた時の角度（Quaternion）
Quaternion rot = Quaternion.LookRotation (posTarget - posCam);
//カメラの角度（rotation）をrotにセットする.
cam.transform.rotation = rot;
```

rigidscript.cs

実行結果

小課題

- rigidscript.cs をさらに修正し（あるいは新たにスクリプトをつくって），カメラが球の周りを回転しながら（回転の仕方は自由），球への視点を維持するようにしてください. さらに, 上下の矢印キーで, 球への距離を操作できるようにしましょう.

MediaPractice03

剛体特性（ゲームエンジン）・カメラの視点

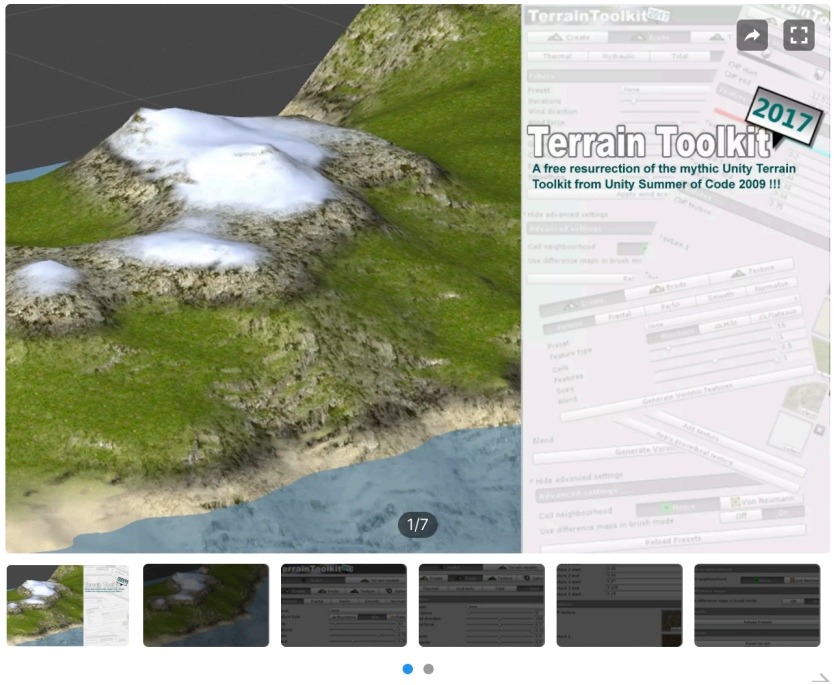
https://lab.kenrikodaka.com/univclass/unity-memo/#terrain

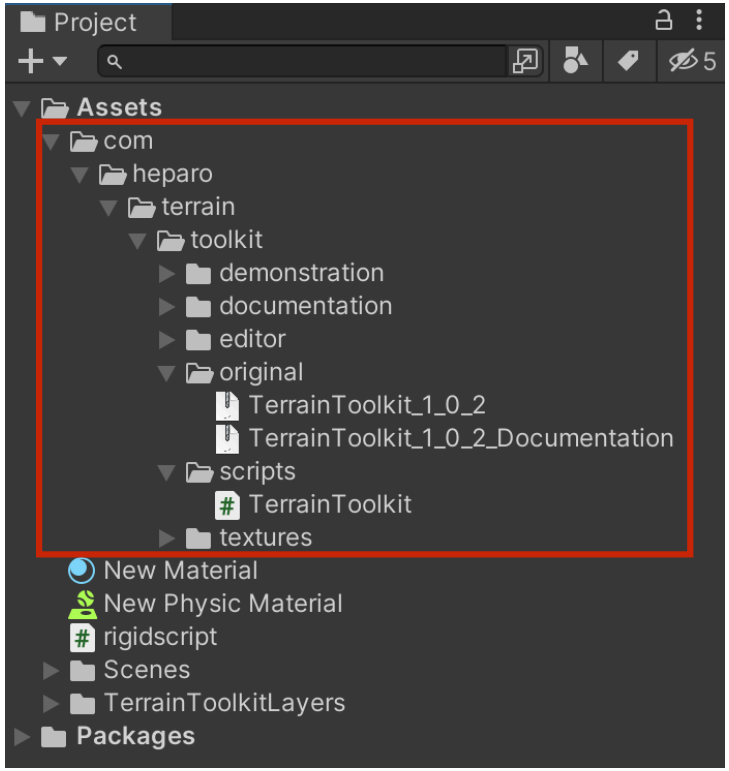
補講１：Terrainの利用

Assetのインポート

- Asset StoreからTerrain Toolkit 2017のダウンロードを行います。Asset Storeを利用するには、UNITY IDが必要ですので、IDを持っていない人は、まずアカウントを作成しましょう。

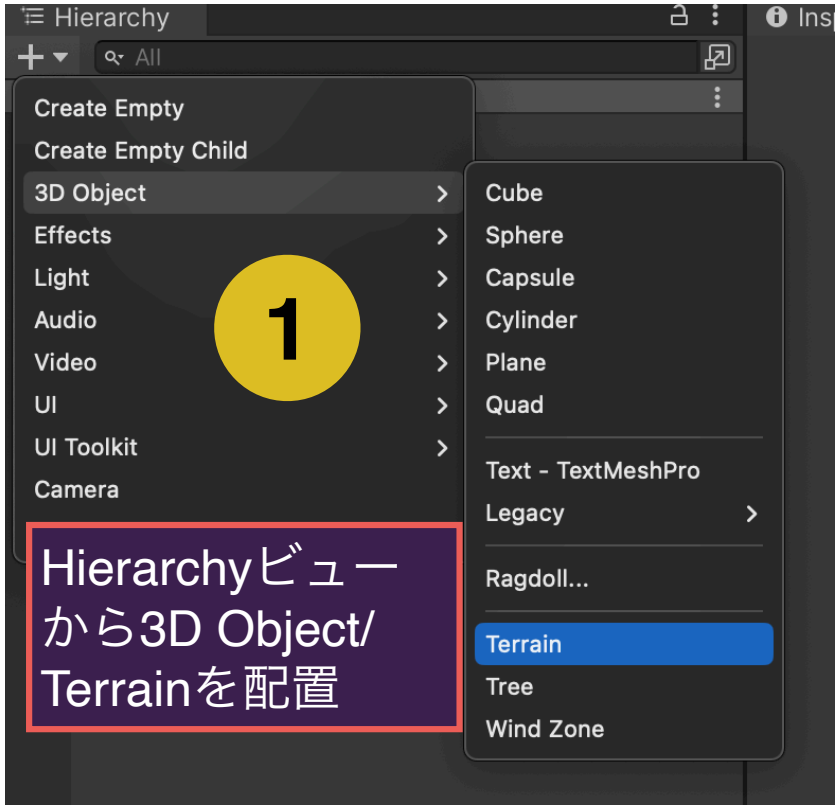
UNITY ASSET STOREからダウンロードします。

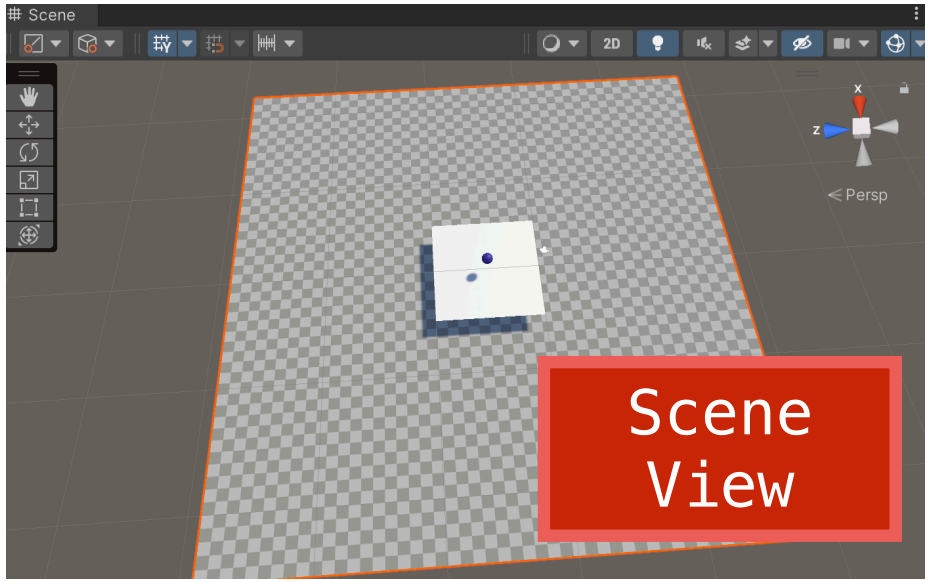




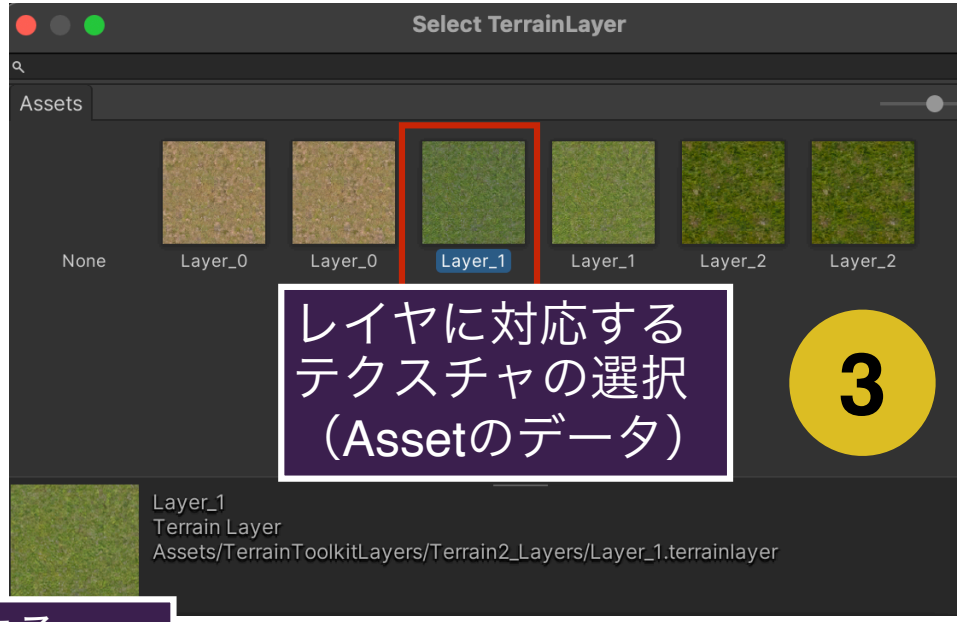
インポートが完了すると、Projectビューに関連のファイル群が配置されます。

Terrainの位置・スケール

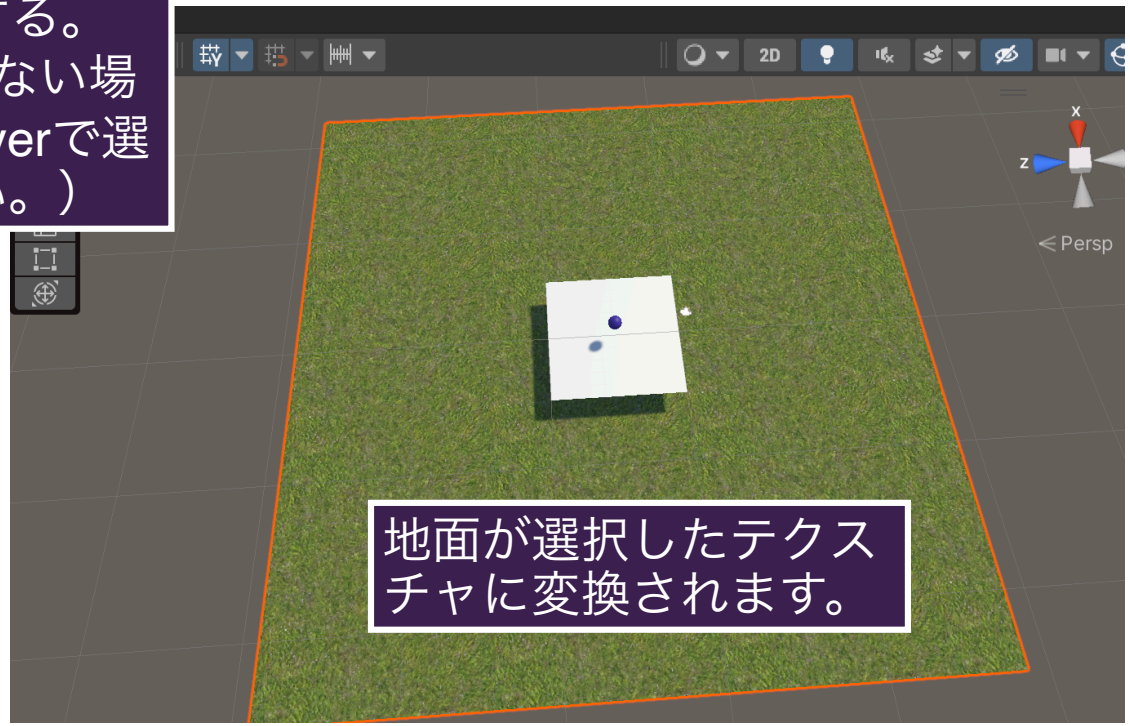




Terrainのテクスチャ

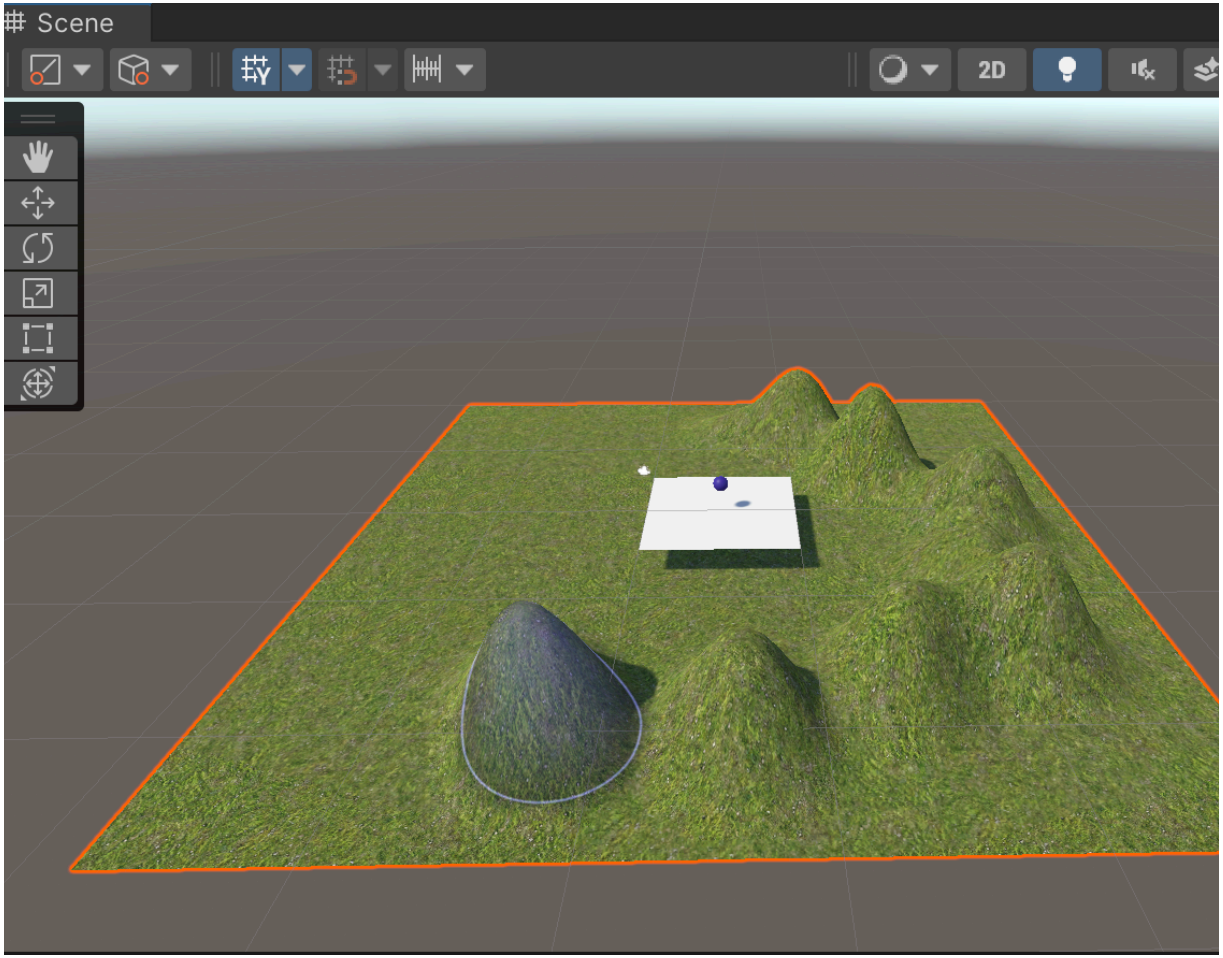


レイヤを追加する。（! うまくいかない場合、Create Layerで選択してください。）



Terrainの地形

シーンビュー



地形に凹凸をつける。

ブラシの形状の設定

ブラシのサイズの設定

シーンビュー上で、特定のTerrainについて、ドラッグすると、ドラッグの距離に応じて、地形が盛り上がります。SHIFTを押しながらドラッグすると、逆に凹みます。

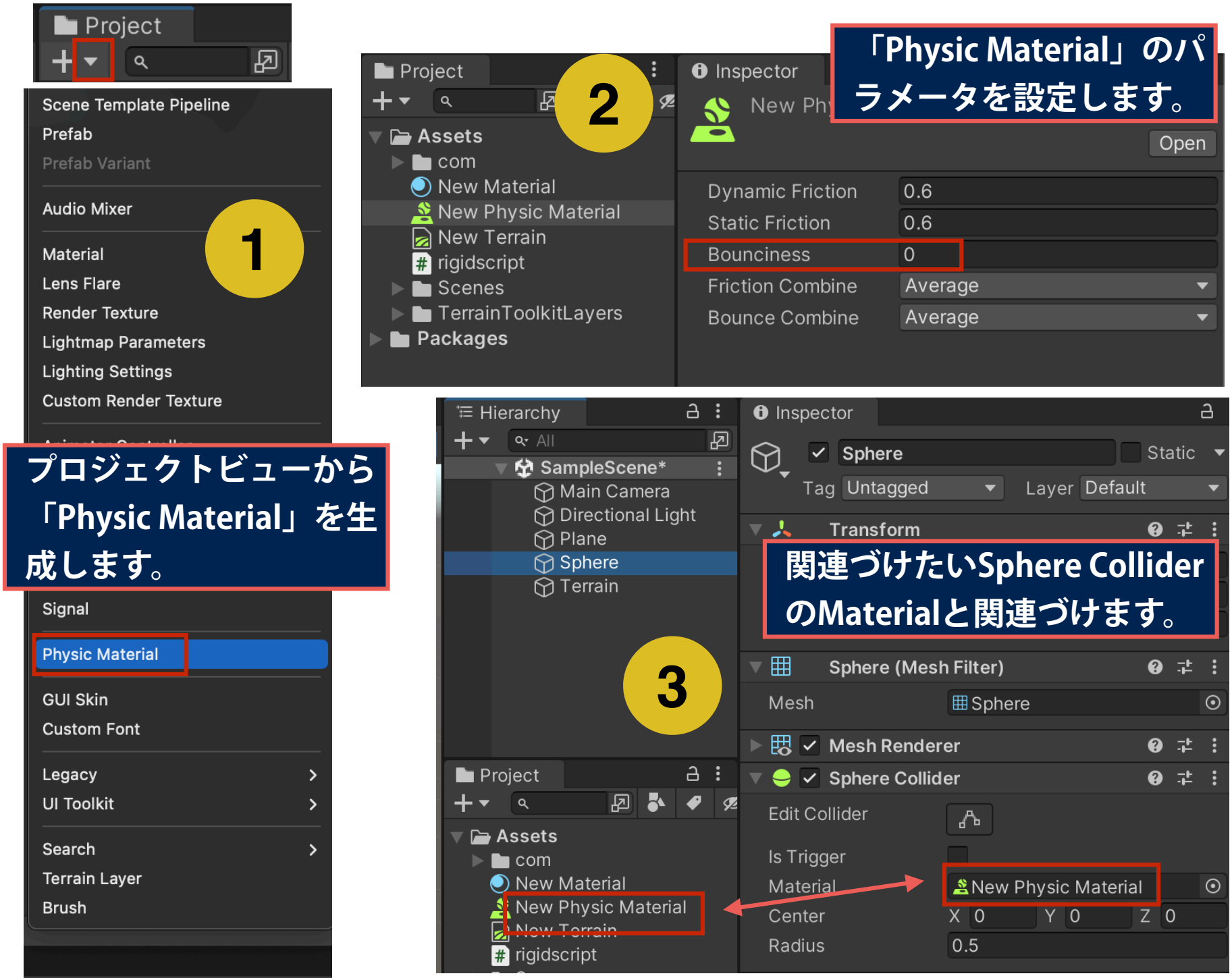
MediaPractice03

剛体特性（ゲームエンジン）・カメラの視点

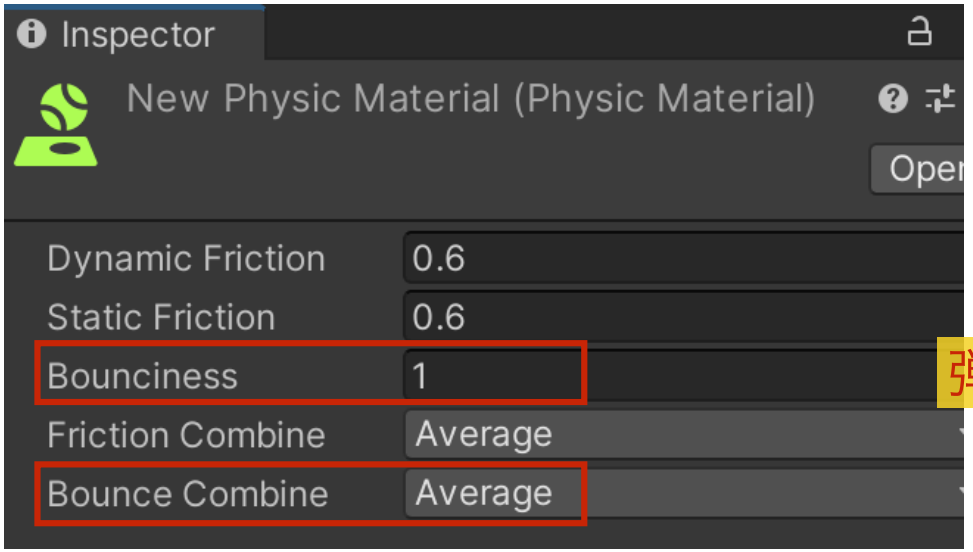
https://lab.kenrikodaka.com/univclass/unity-memo/#pmaterial

補講２：Physics Material

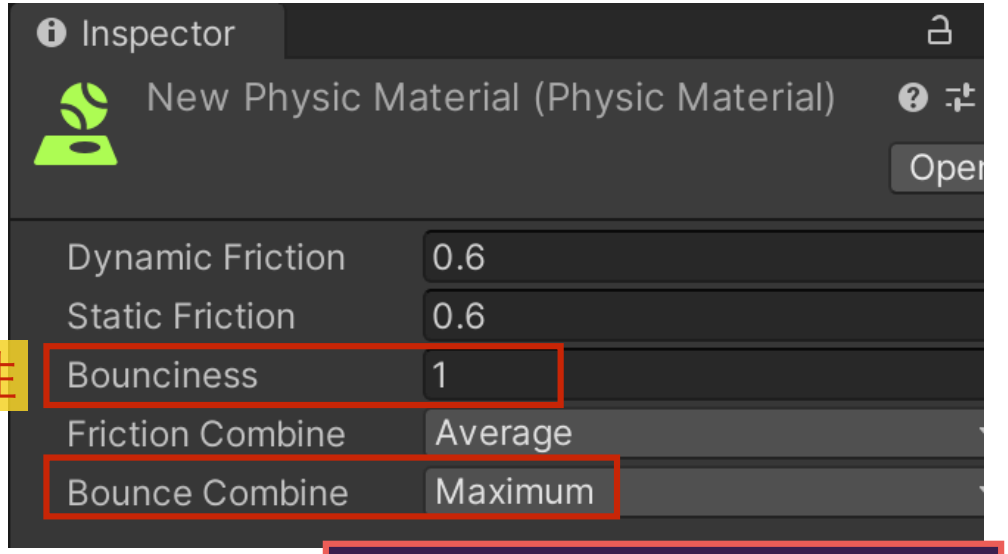
Terrainの地形


Project
Scene Template Pipeline
Prefab
Prefab Variant
Audio Mixer
Material
Lens Flare
Render Texture
Lightmap Parameters
Lighting Settings
Custom Render Texture
Signal
Physic Material
GUI Skin
Custom Font
Legacy
UI Toolkit
Search
Terrain Layer
Brush
1
プロジェクトビューから「Physic Material」を生成します。
Project
Assets
com
New Material
New Physic Material
New Terrain
rigidscript
Scenes
TerrainToolkitLayers
Packages
2
Inspector
New Phy
Open
Dynamic Friction 0.6
Static Friction 0.6
Bounciness 0
Friction Combine Average
Bounce Combine Average
「Physic Material」のパラメータを設定します。
Hierarchy
All
SampleScene*
Main Camera
Directional Light
Plane
Sphere
Terrain
3
Project
Assets
com
New Material
New Physic Material
New Terrain
rigidscript
Inspector
Sphere
Static
Tag Untagged
Layer Default
Transform
関連づけたいSphere ColliderのMaterialと関連づけます。
Sphere (Mesh Filter)
Mesh Sphere
Mesh Renderer
Sphere Collider
Edit Collider
Is Trigger
Material New Physic Material
Center X 0 Y 0 Z 0
Radius 0.5

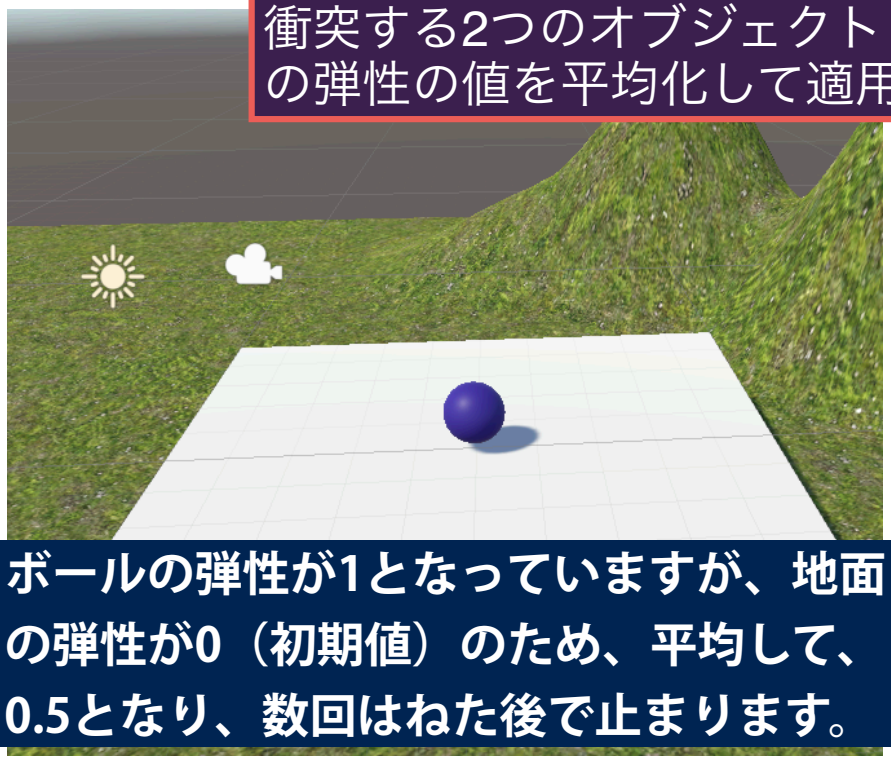
Physic Materialのパラメータ



弾性

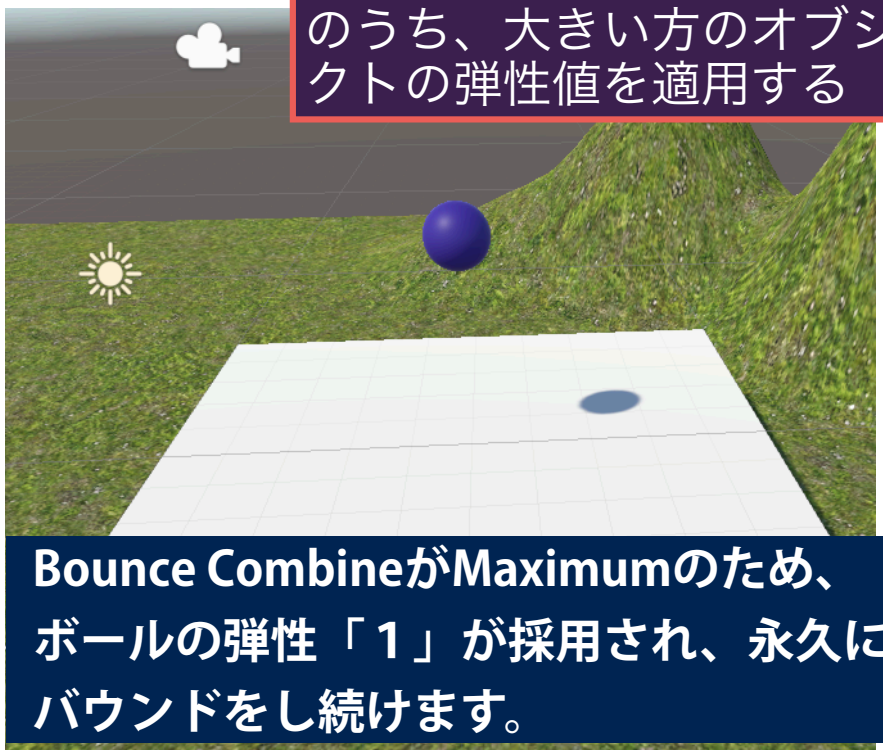


衝突する2つのオブジェクトの弾性の値を平均化して適用



ボールの弾性が1となっていますが、地面の弾性が0（初期値）のため、平均して、0.5となり、数回はねた後で止まります。

衝突する2つのオブジェクトのうち、大きい方のオブジェクトの弾性値を適用する



Bounce CombineがMaximumのため、ボールの弾性「１」が採用され、永久にバウンドをし続けます。