演習１：UNITYスクリプトの基礎

(01) 04/13

１A｜Unityとエディタの連携

(02) 04/20

１B｜Transform・キーイベント・マウスイベント

(03) 04/27

１C｜剛体特性・カメラの視点

(04) 05/11

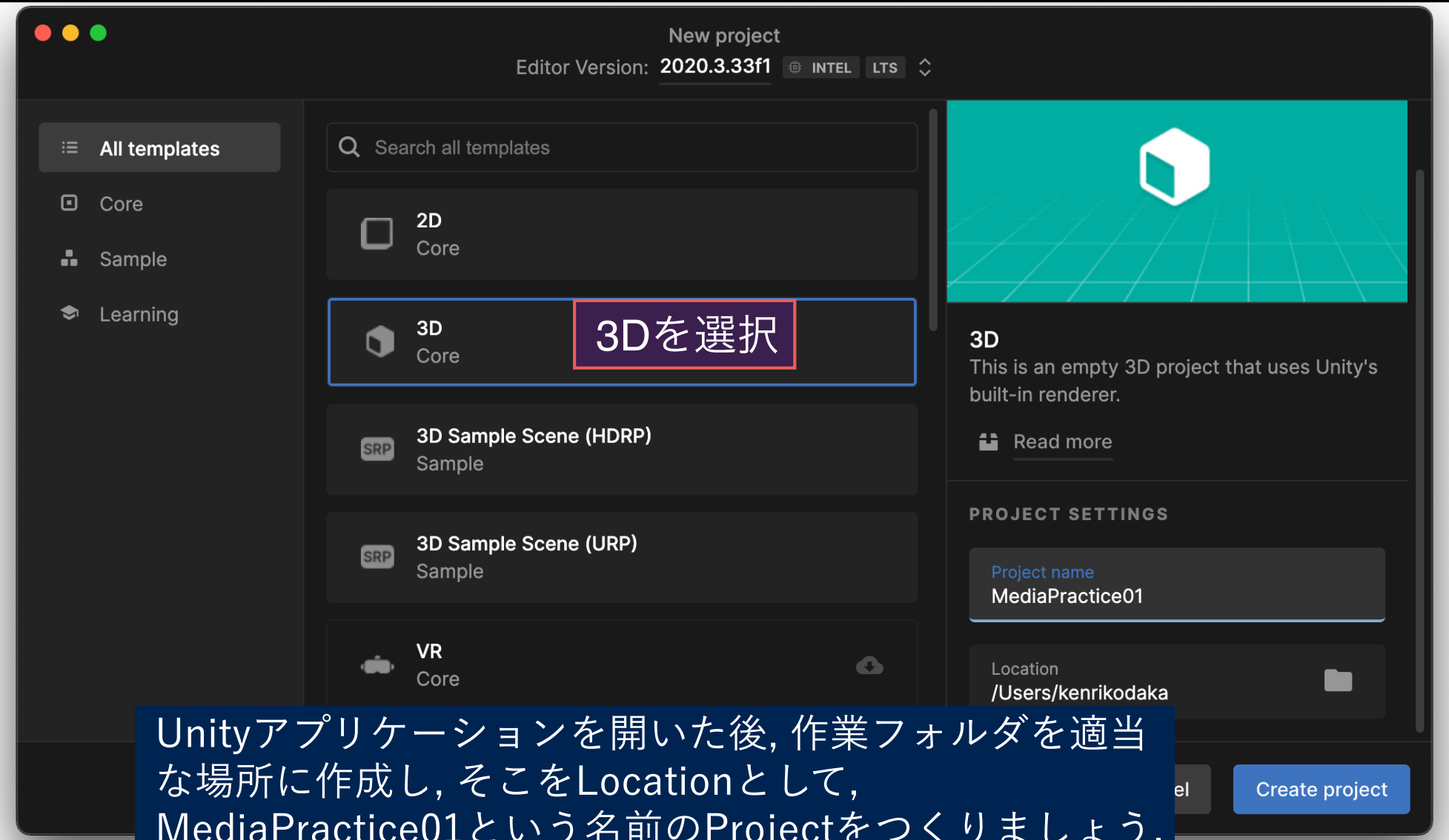
１D｜プレハブ（gameobjectの雛形）, タグ, その他

(05) 05/18

１E｜アバターのアニメーション（間に合えば）

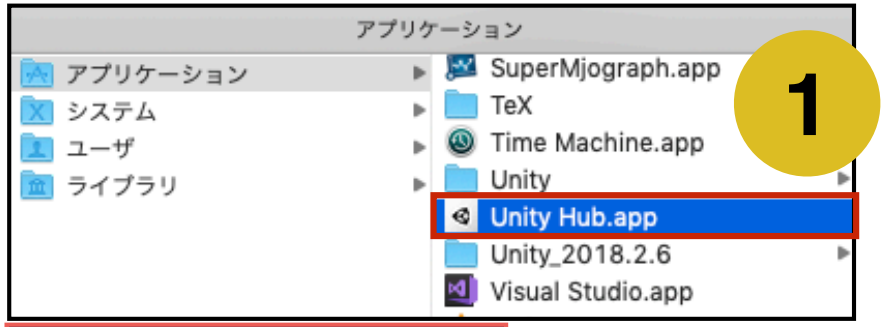
MediaPractice01

Unityとエディタの連携

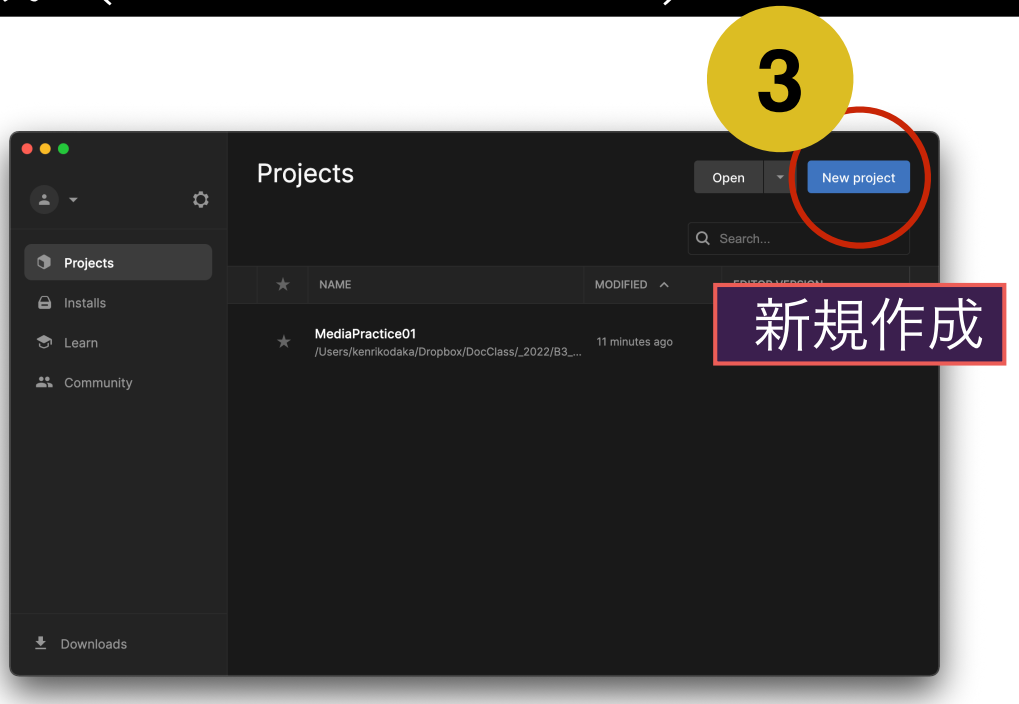


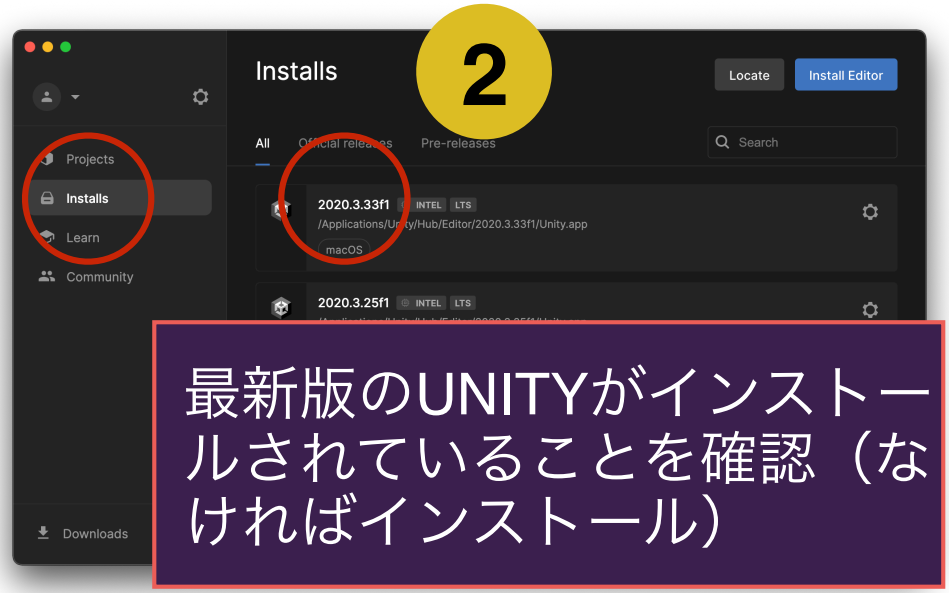
Unityアプリケーションを開いた後, 作業フォルダを適当な場所に作成し, そこをLocationとして, MediaPractice01という名前のProjectをつくりましょう.

新規プロジェクトの作成（MediaPractice01）

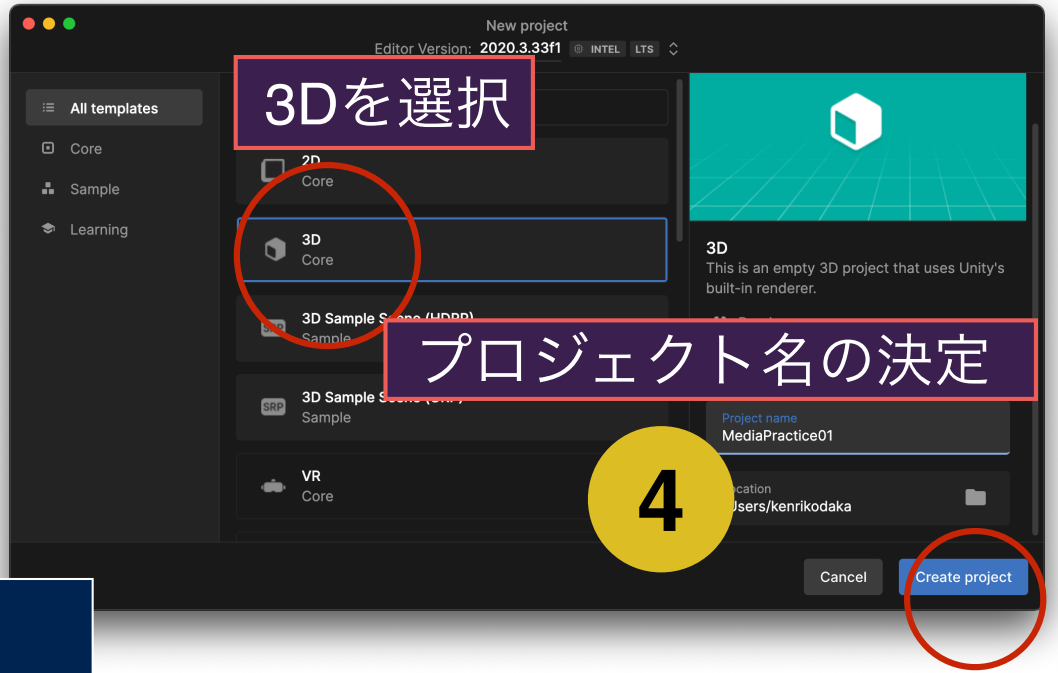


Unity Hubを起動



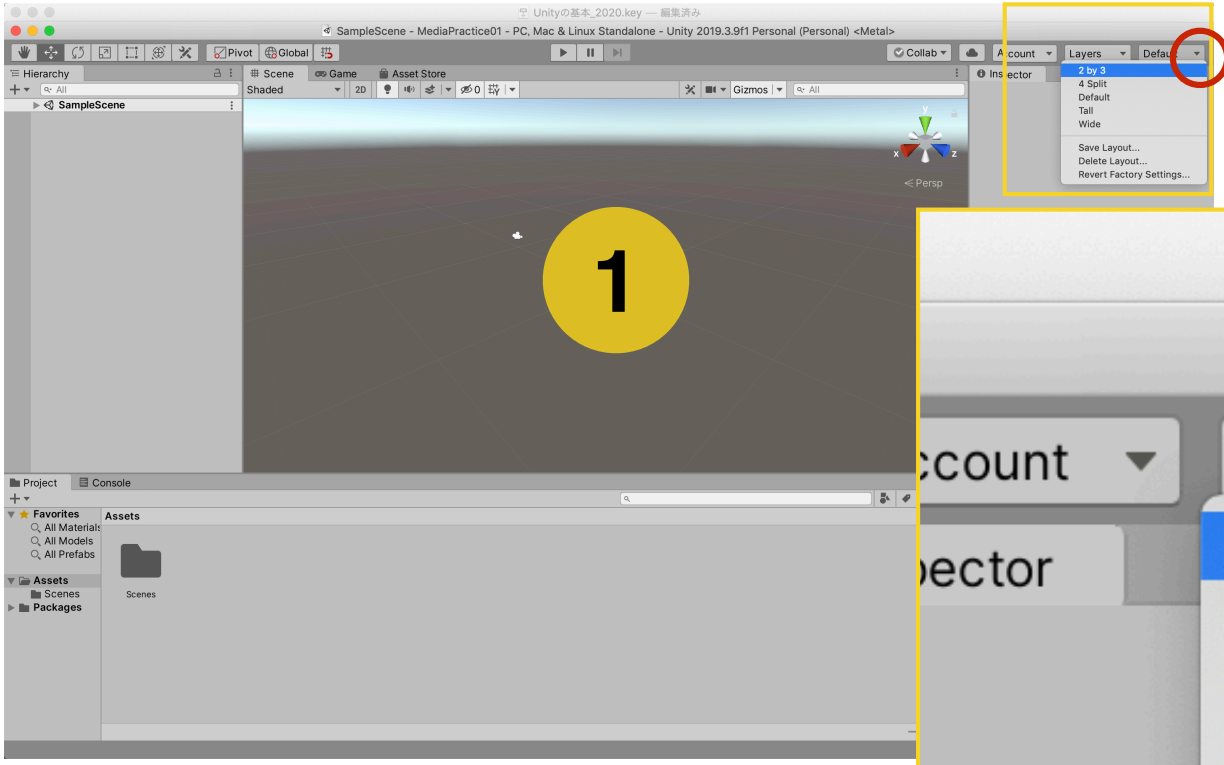


最新版のUNITYがインストールされていることを確認（なければインストール）

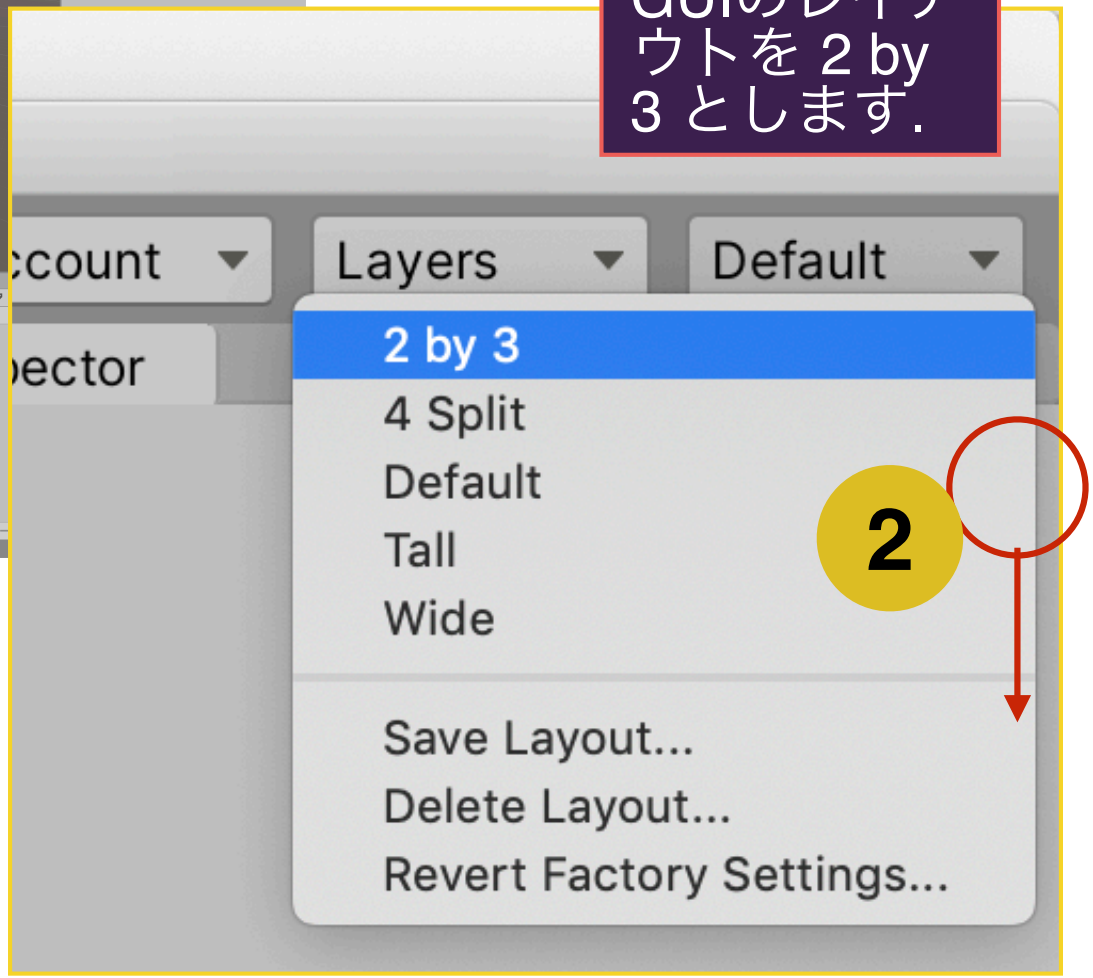


2022年4月時点の、最新の安定版は「2022.3.33f1」です。

GUIレイアウトの変更



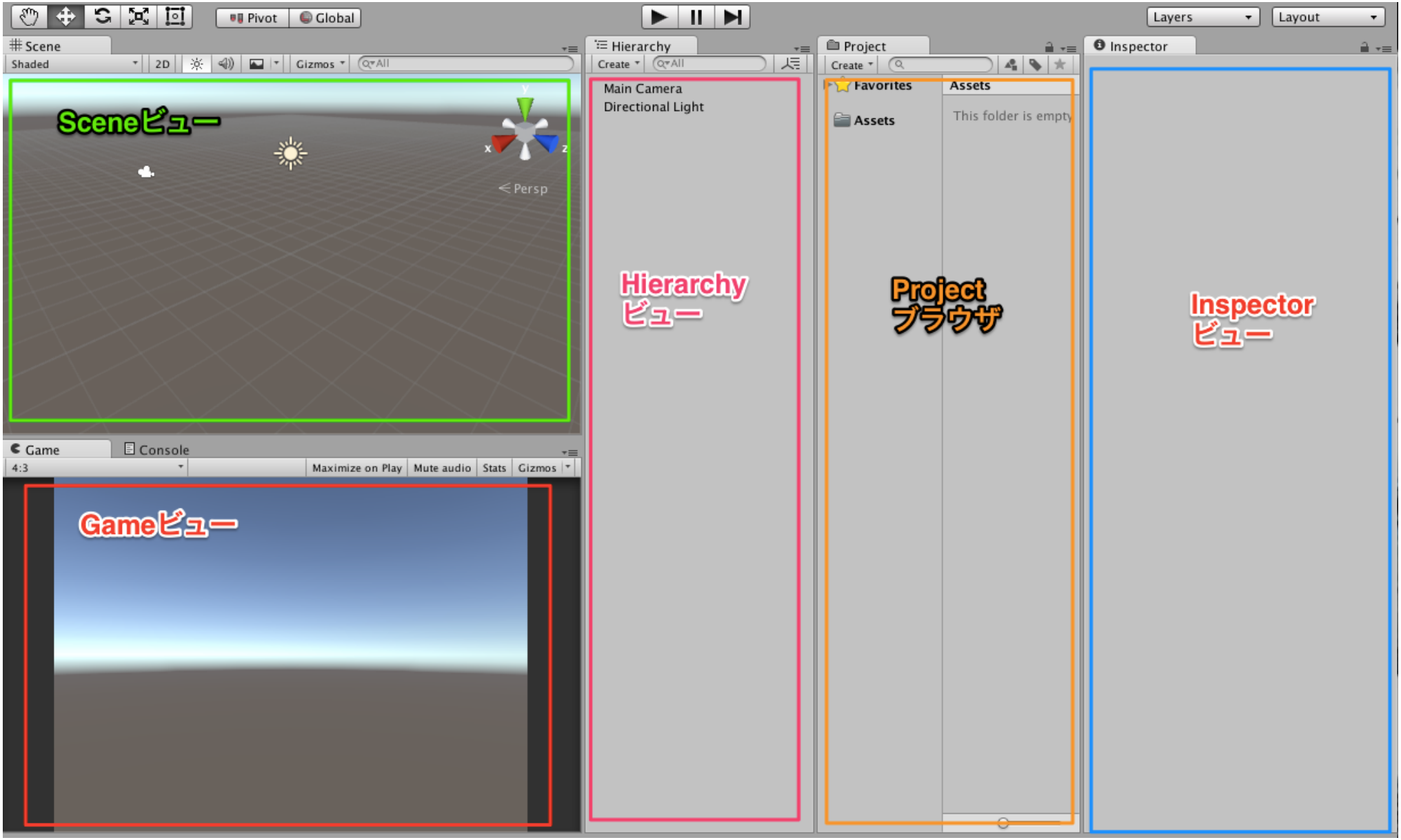
GUIのレイアウトを 2 by 3 とします.



最新版のUNITYでは、DefaultのGUIレイアウトが①となっているので、まずはレイアウトを「2 by 3」に変更してください（②）。

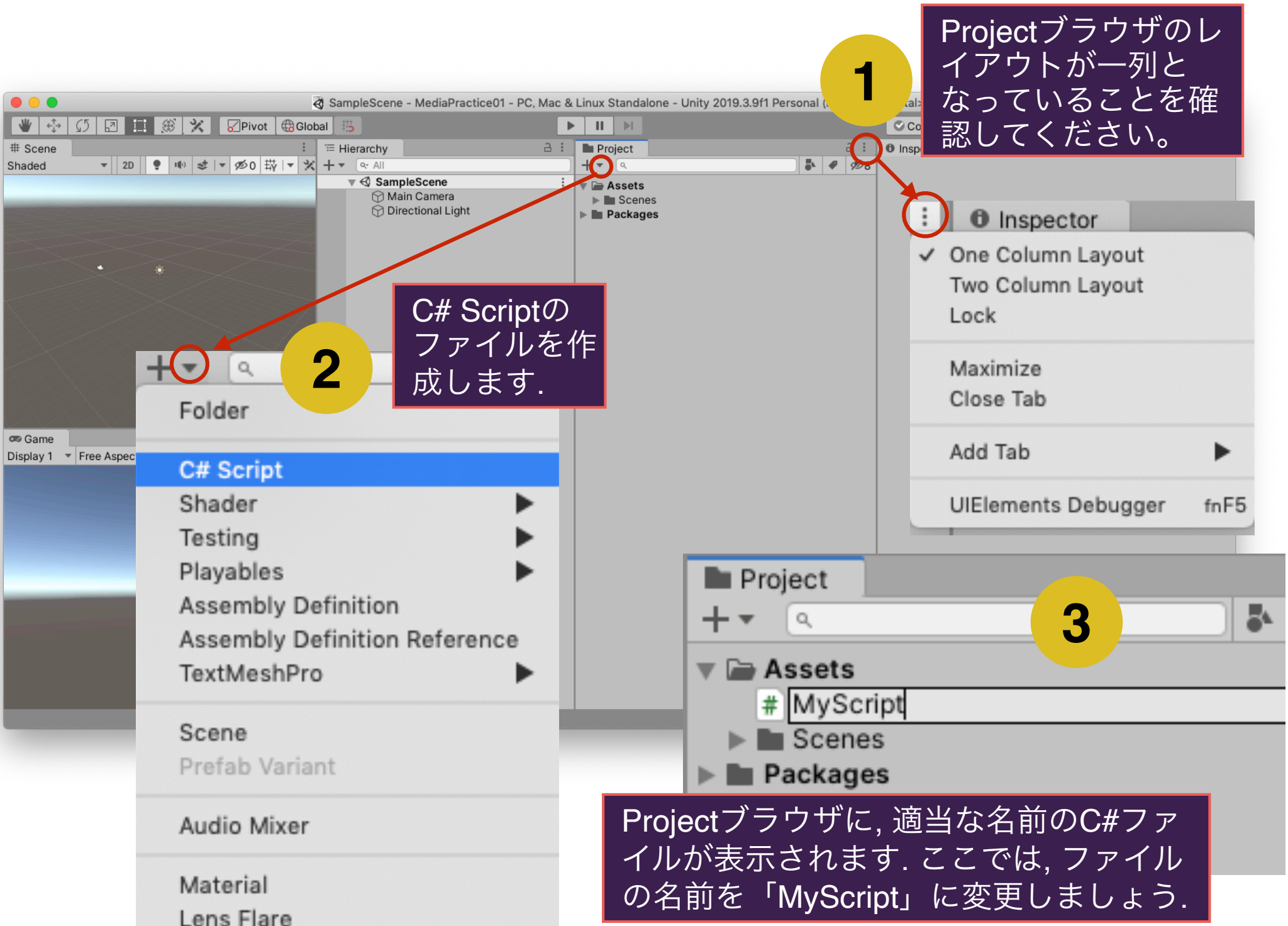
授業資料は、「2 by 3」のレイアウトを前提に進めていきます。

GUIの構造・各パネルの呼び名（2 by 3 レイアウト）

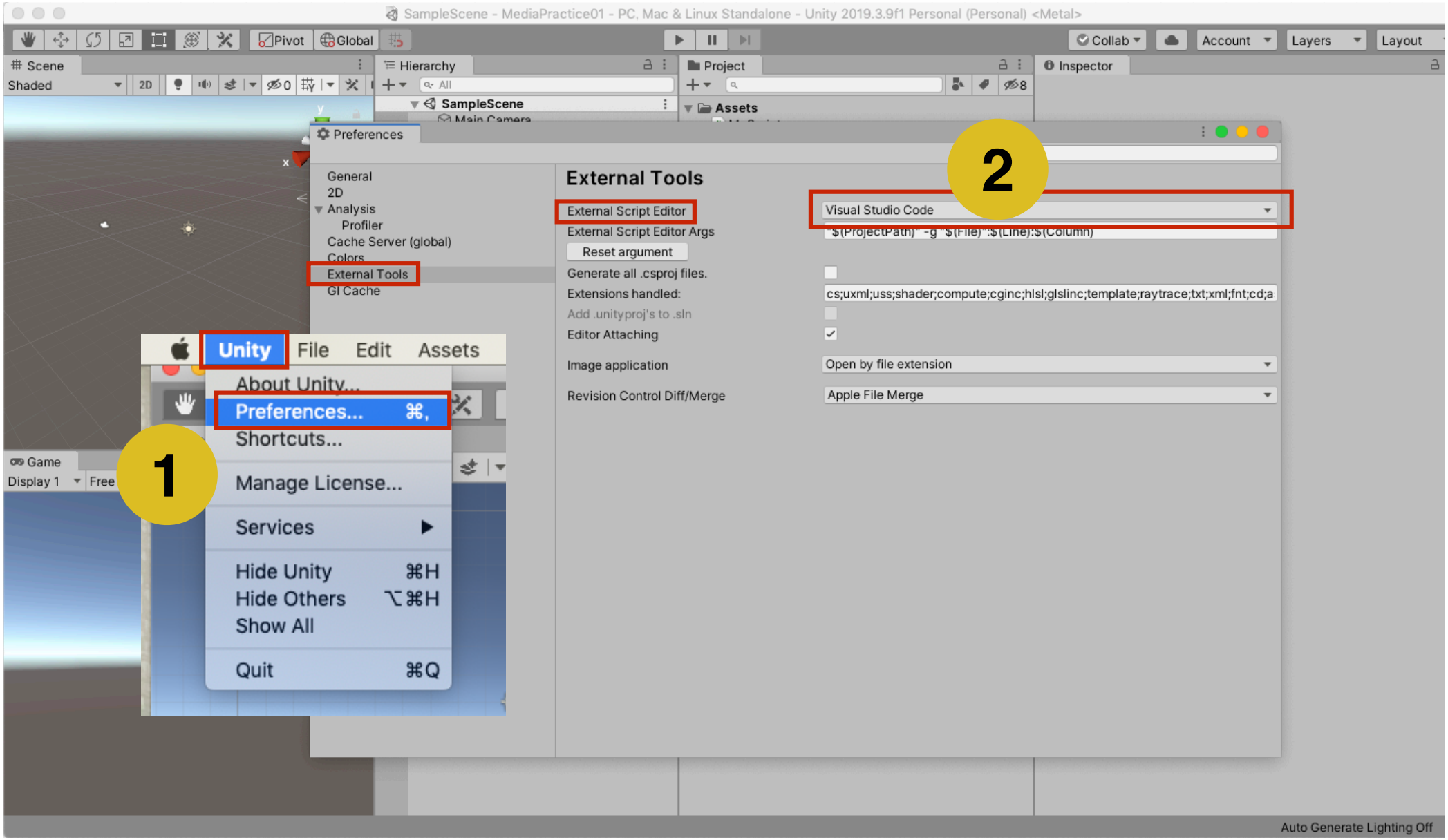


https://github.com/unity3d-jp/FirstTutorial/wiki/0.プロジェクトの作成

スクリプト作成の準備


1
Projectブラウザのレイアウトが一列となっていることを確認してください。
Inspector
One Column Layout
Two Column Layout
Lock
Maximize
Close Tab
Add Tab
UIElements Debugger fnF5
2
C# Scriptのファイルを作成します.
Folder
C# Script
Shader
Testing
Playables
Assembly Definition
Assembly Definition Reference
TextMeshPro
Scene
Prefab Variant
Audio Mixer
Material
Lens Flare
3
Project
Assets
MyScript
Scenes
Packages
Projectブラウザに, 適当な名前のC#ファイルが表示されます. ここでは, ファイルの名前を「MyScript」に変更しましょう.

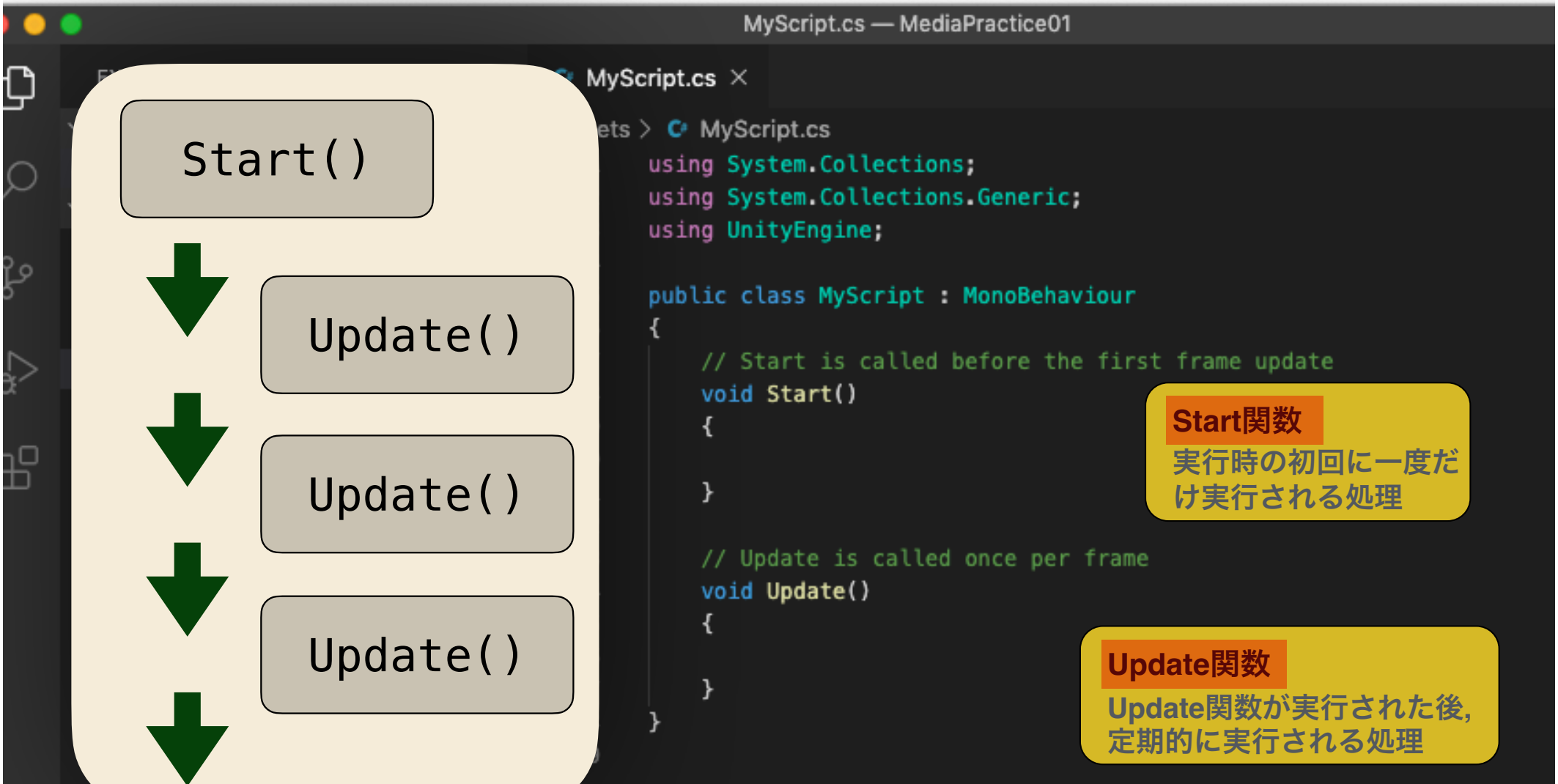
外部スクリプトエディタの設定



はじめに、外部スクリプトのエディタとして「Visual Studio Code」を設定しておきます。これをしないと、プログラム環境として別のテキストエディタが開きます。

エディタを開く

- Project ブラウザの ***MyScript.cs*** をダブルクリックすると, 別アプリケーションのエディタが開き, csファイルの編集が可能となります.
- スクリプトに対応するクラスには, あらかじめ **Start** 関数 と **Update** 関数がインクルードされています.



ゲームオブジェクトとスクリプトの関連付け

MyScript.cs

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

1

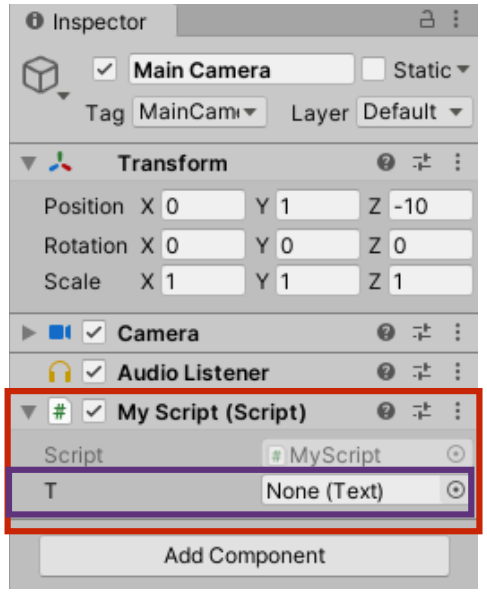
新たに二つの文を挿入してください.

2

Hierarchy ビューの Main Camera を 選択します（Inspector ビューに Main Camera のコンポーネントが表示されます）。

Projectブラウザの中にあるMyScript を Inspector ビューにドラッグ&ドロップします.

3

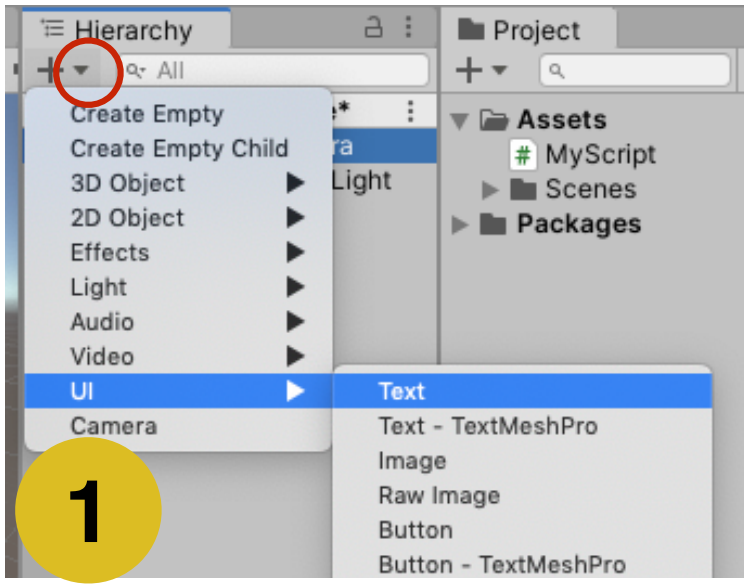


inspectorビュー内の「T」は, スクリプト内の変数「t」に対応しています.

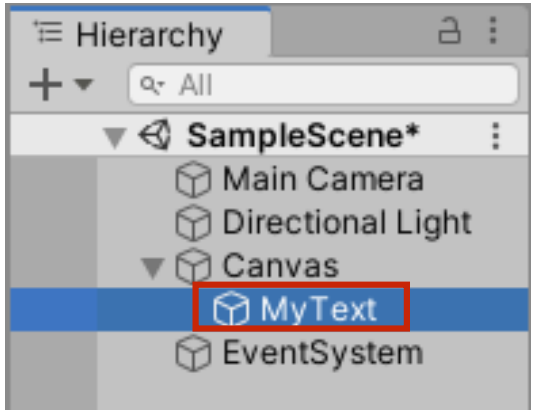
4

結果, Inspectorビューにスクリプトの内容が表示されます（Main Cameraのコンポーネントとして登録されたということです）.

Text（ゲームオブジェクト）の追加

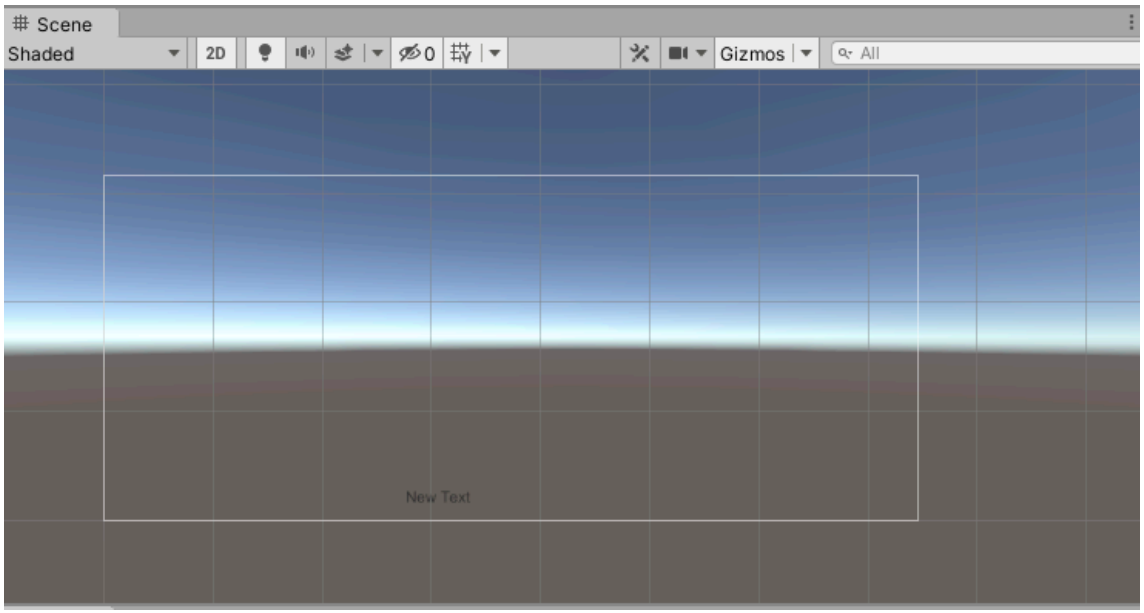


Hierarchy ビューの ▽ を押して, UIよりText を選択してください.



2

ここでは, Text を MyTextにリネームしましょう.



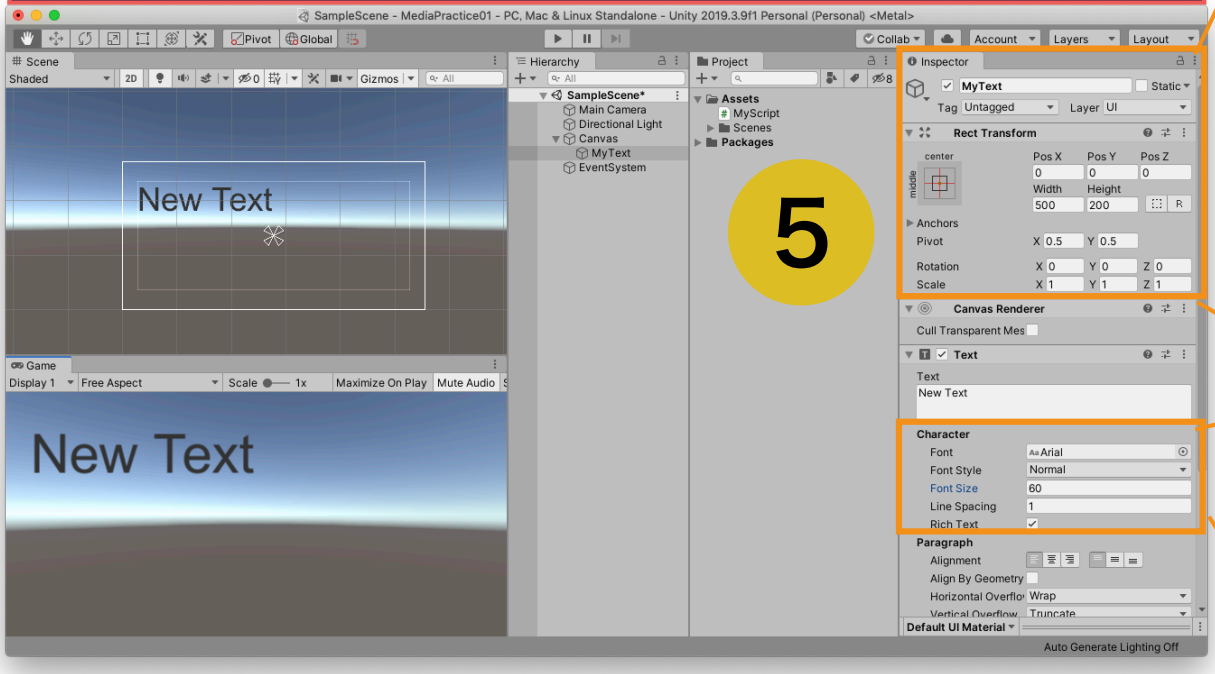
4

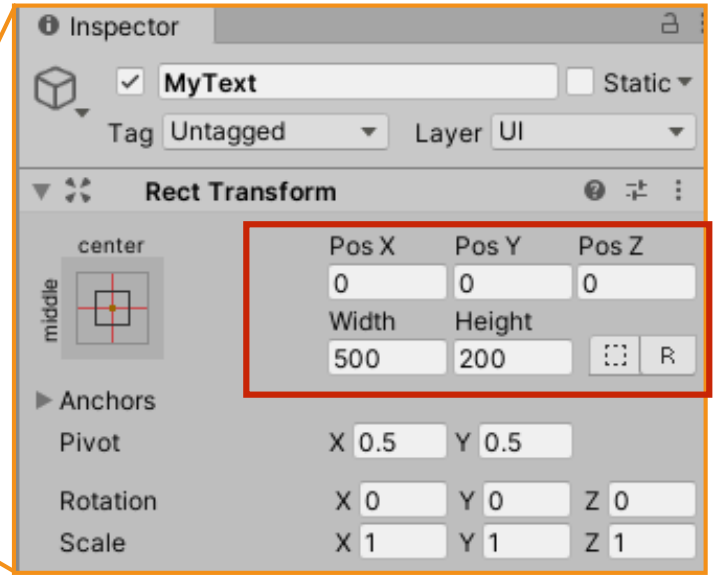
白の長方形の枠がすべてSceneビューにおさまるように拡大・縮小（マウスホイールの回転）、移動（option＋ドラッグ）をして調整します.

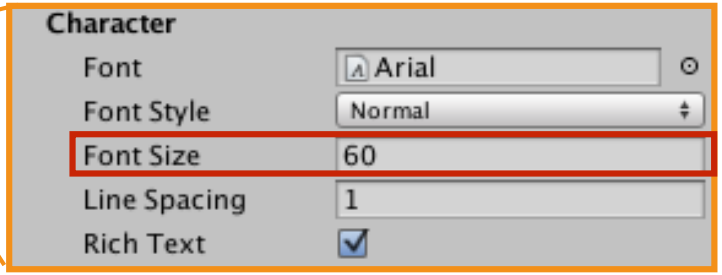
Scene ビューの操作方法については例えば以下を参考にしてください。

https://miyagame.net/scene-view-method/

MyTextのインスペクタを以下のように変更してください. 白の枠内に「New Text」という文字が表示されるはずです.

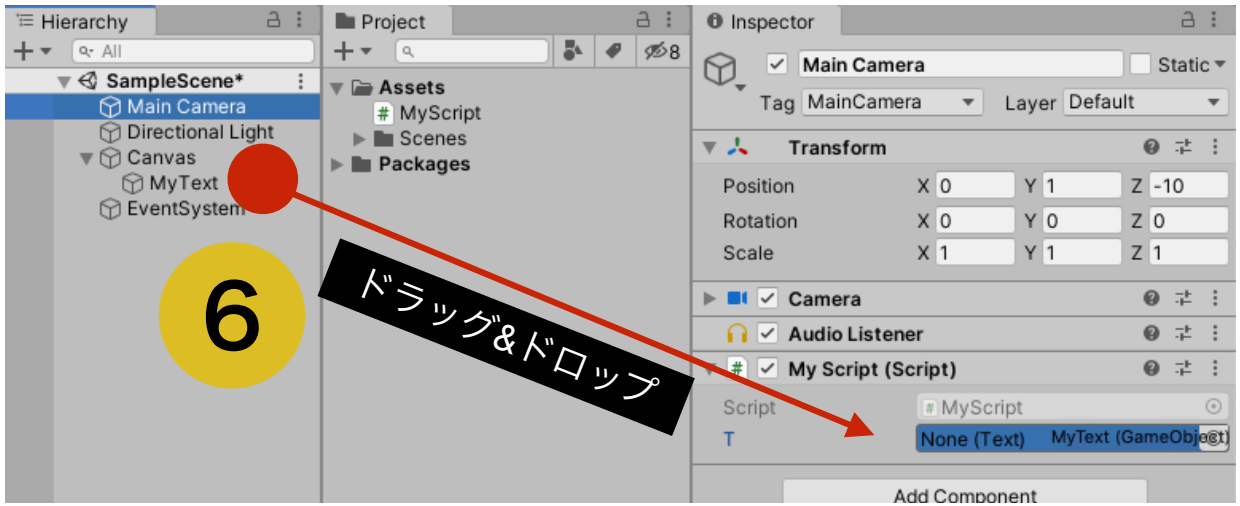




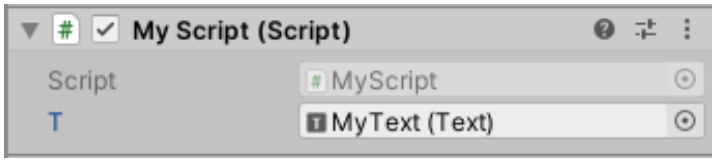


Main Cameraを選択した状態で, My Scriptのpublic変数「T」の項目に, HierarchyビューのMyTextをドラッグ&ドロップします.

7



結果, 「T」の項目がNoneからMyTextへと変更されます.

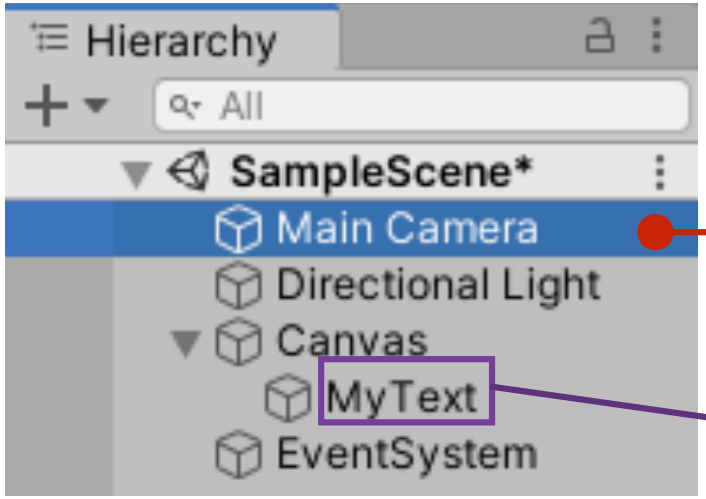


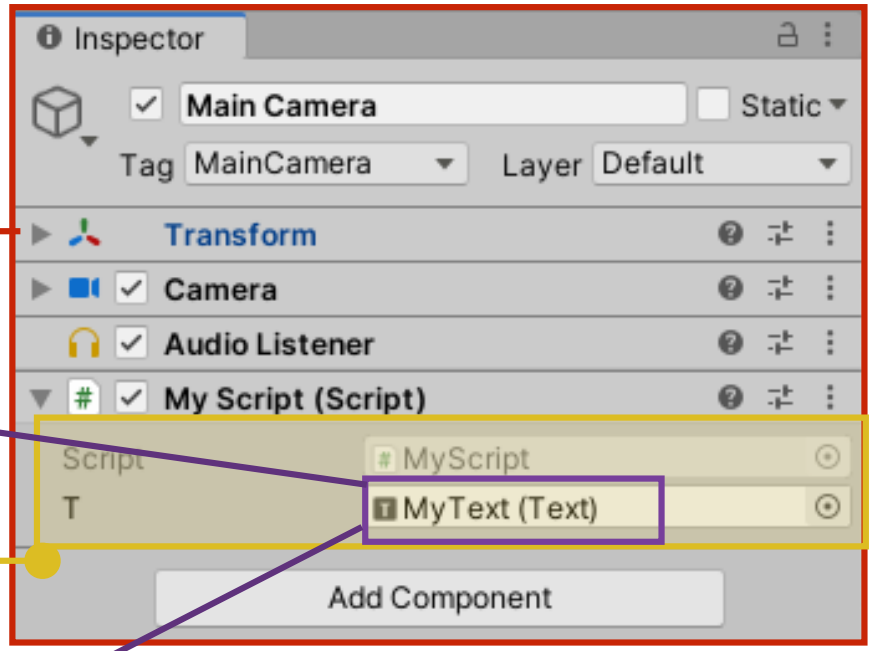
ゲームオブジェクトとスクリプト内変数の関係

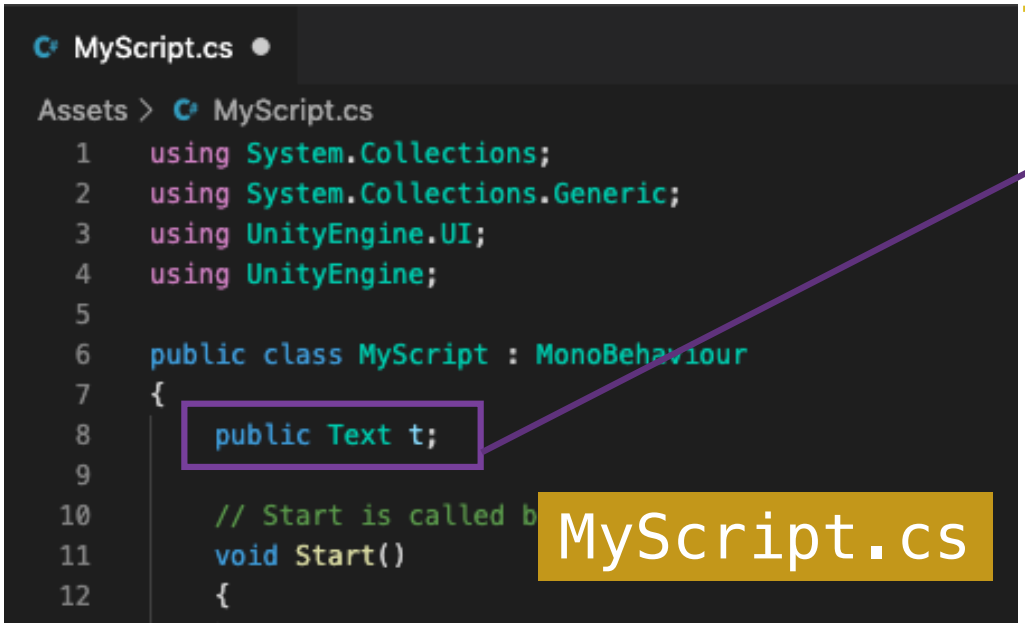
シーンの実行

Hierarchyビューの中にある全てのゲームオブジェクトが起動

起動したゲームオブジェクトのコンポーネントに登録されている全てのスクリプトが起動







cs内の変数 t は, MyTextで表示される文字列と結びついている.

そのため, 変数 t に対する操作が, シーン内のMyTextの変化として反映される.

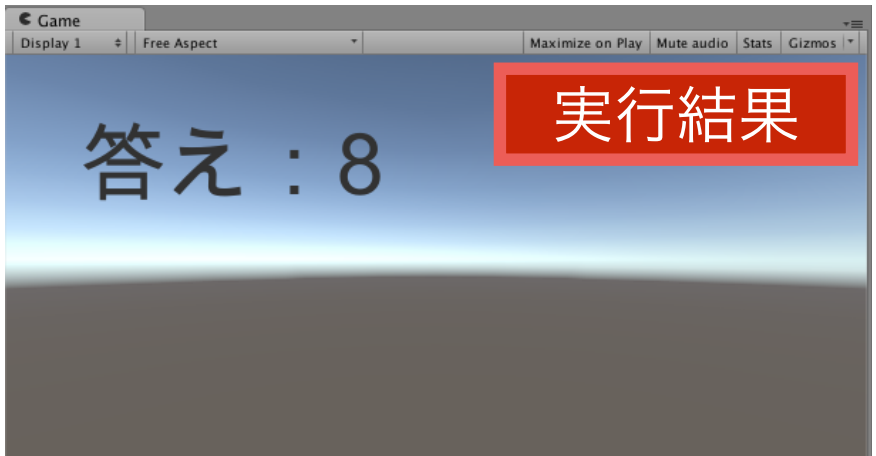
C#における変数の型

```
public class MyScript : MonoBehaviour {
    public Text t;

    /* 整数はint型 */
    int x1 = 3;
    public int x2 = 5; //インスペクタから操作可能
    /* 実数はfloat型が慣例 */
    float y1 = 10.3f; //fをつけるとfloat型
    double y2 = 10.3; //fをつけないとdouble型
//  float y2 = 10.3; //エラー
    /* その他の型 */
    string s1 = "abc";  //文字列はstring
    bool b1 = true;     //真偽値はbool

    void Start () {
        int z1 = x1 + x2;
//      int z2 = x1 + y1; //エラー：(int)+(float)
        float z2 = x1 + y1;
        int z3 = x1 + (int)y1; //キャスト

        t.text = "答え：" + z1;
//      t.text = "答え：" + z2;
//      t.text = "答え：" + z3;
    }

    void Update () {
    }
}
```

MyScript.cs

int	100　-3　98765
double	100.0　-3.0　98765.2
float	100.0f　-3.0f　98765.2f
char	‘a’　‘Z’　‘あ’
string	“abc”　“あいうえお”　“a”
bool	true　false

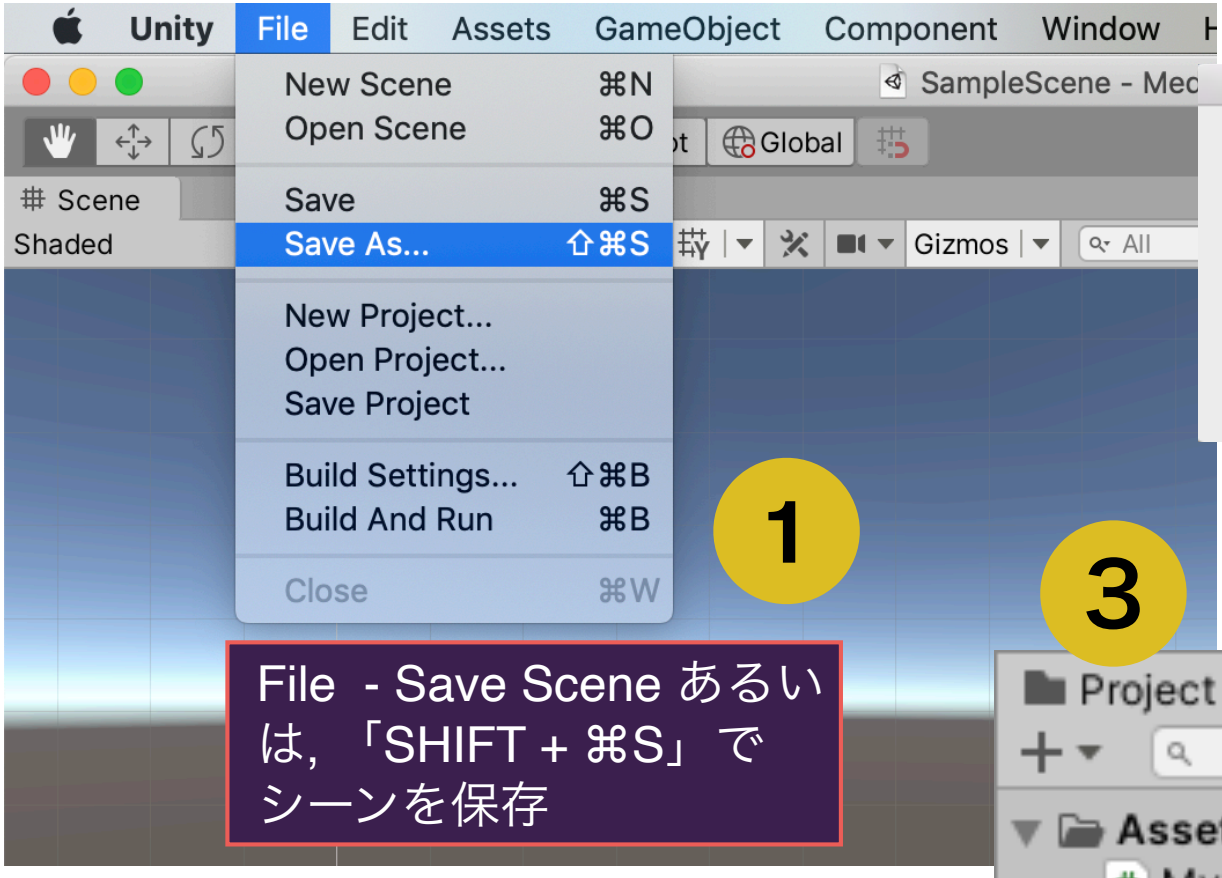
整数と実数には様々な型がありますが, Unityの計算では, 特別な理由がない限り, 「整数は **int** , 実数は **float** を使う」と覚えておきましょう.



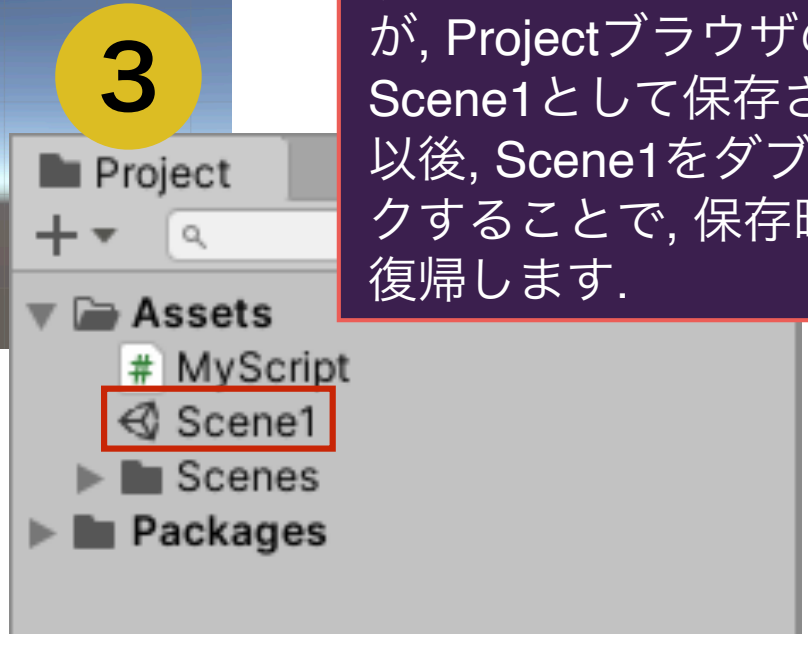
float型には数字の後に「f」をつけることを忘れないように！！

シーンの保存

- シーンの現在の状態（Hierarchyビューにどのゲームオブジェクトが配置され，それらがそのようなパラメータを有しているか）を保存することで，特定の状態へとすぐに復帰することができます．



File - Save Scene あるいは，「SHIFT + ⌘S」でシーンを保存



現在のパラメータの状態などが，Projectブラウザの中にScene1として保存されます．以後，Scene1をダブルクリックすることで，保存時の状態が復帰します．

最新版では、初期状態のシーンには「SampleScene」という名前が付けられています。

配列の利用（c#）

```
    public int id = 0;

//  int[] x; //配列の宣言（A）
//  x = new int[3]; //配列のサイズを3とする（B）
//  x[0]=1; x[1]=3; x[2]=5; //値の設定（C）
//  int[] x = new int[3]; //(A)(B)
    int[] x = {1,3,5}; //(A)(B)(C)

    string[] s = {"Hello","Welcome","Bye","Afternoon"};

    void Start () {
    }

    void Update () {
        if (id == 0) {
            //x.Length:配列xのサイズ（3）
            //Random.value：0.0~1.0の乱数
            //Mathf.Floor：少数切り捨ての関数です
            int ri = (int)Mathf.Floor (x.Length * Random.value);
            t.text = x [ri].ToString ();
        } else {
            int ri = (int)Mathf.Floor (s.Length * Random.value);
            t.text = s [ri].ToString ();
        }
    }

}
```

表示の切り替えフラグ

配列の宣言と値の代入

MyScript1.cs

int 配列.Length

配列のサイズ

float Mathf.Floor(実数)

引数の値の小数点を切り捨て

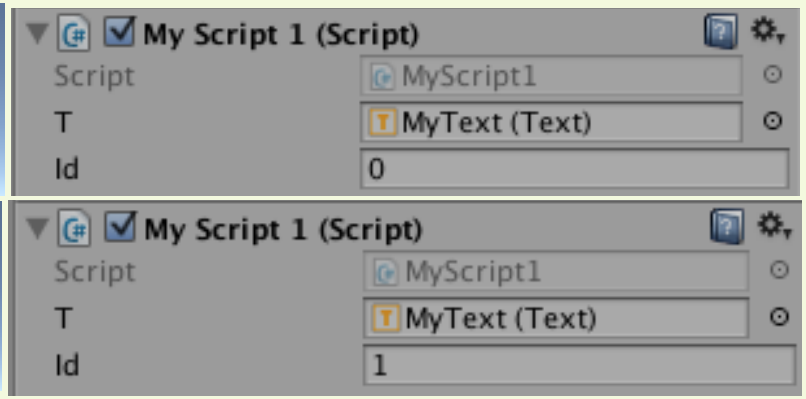
float Random.value

0～1のランダムな実数

実行結果

5

Welcome



inspectorビューにおいて，idの値を直接変更すると表示する配列が切り替わります．ただ，このままだと，要素の表示の切り替わりが速すぎますね．

ストップウォッチ（時間処理）の実装

```
 5 public class MyScript2 : MonoBehaviour {
 6     public Text t;
 7     public int id = 0;
 8     int[] x = {2,3,5,7,11,13,17,19,23,29,31};
 9     string[] s = {"Hello","Welcome","Bye","Afternoon"};
10
11     //drawRandom関数処理の時間間隔（秒）
12     public float timeOut = 1.0f;
13     //前回のdrawRandom関数処理からの累積経過時間（秒）
14     private float timeElapsed = 0.0f;
15
16     void Start () {
17     }
18
19     void Update () {
20         //Time.deltaTime：前回のフレームからの経過時間（秒）
21         timeElapsed += Time.deltaTime;
22
23         if (timeElapsed > timeOut) {
24             drawRandom ();
25             timeElapsed = 0.0f; //累積時間をリセット
26         }
27     }
28
29     //配列の中の値をランダムに表示する関数
30     void drawRandom(){
31         if (id == 0) {
32             int ri = (int)Mathf.Floor (x.Length * Random.value);
33             t.text = x [ri].ToString ();
34         } else {
35             int ri = (int)Mathf.Floor (s.Length * Random.value);
36             t.text = s [ri].ToString ();
37         }
38     }
39
40 }
```

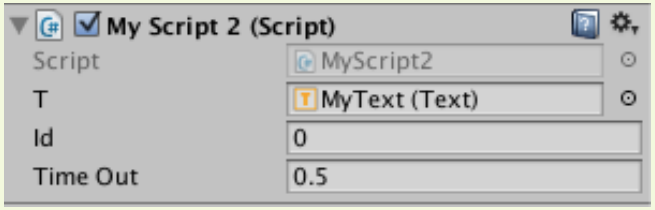
MyScript2.cs

timeElapsed は, ストップウォッチの値を保持する変数であり, 指定時間を過ぎたらdrawRandom関数を実行するようにしています.

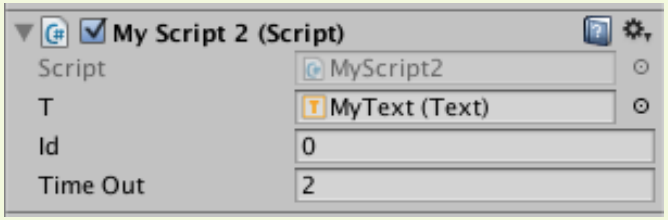
`float Time.deltaTime`

Update関数実行間のインターバル（秒）

実行結果



0.5秒おきに表示が切り替わります.



2.0秒おきに表示が切り替わります.

ストップウォッチ（時間処理）の実装