可視化情報学会ワークショップ

チュートリアル1:
Unityハンズオンセミナー お手軽ビジュアリゼーション講座

ユニティ・テクノロジーズ・ジャパン
安原 祐二

**本チュートリアルは
Unity2018.3がインストール済みの
PCの持参が必要です**

unity

参考：Unityのインストール・アカウント作成

- Unity の動作要件（いずれも厳しい制約ではなく、2016年以降に入手した機器であれば問題ないはずです）
  - Windowの場合は Windows 7 SP1+ または Windows 8, Windows 10, いずれも 64-bit であること
  - Mac の場合は macOS 10.11以降（OS X El Capitan以降）
- Unity 2018.3 のインストール方法
  - インストールは、Unity Hub というランチャーをインストールしてから Unity Hub 経由でインストールするのが推奨されます。
  - 手順： https://docs.unity3d.com/ja/current/Manual/GettingStartedInstallingHub.html
- Unityアカウント作成方法
  - 初めての利用にはUnity起動時にアカウントが必要です。基本的には画面に従うことで完了します。なお、アカウント作成画面は英語になります。
  - 手順：https://docs.unity3d.com/ja/current/Manual/GettingStarted.html

unity

サンプルプロジェクトのダウンロード

- ブラウザから以下の github にアクセスし、ダウンロードしてください
  - https://github.com/dsedb/VisualizationWorkshopTutorial
  - 容量は１M程度です

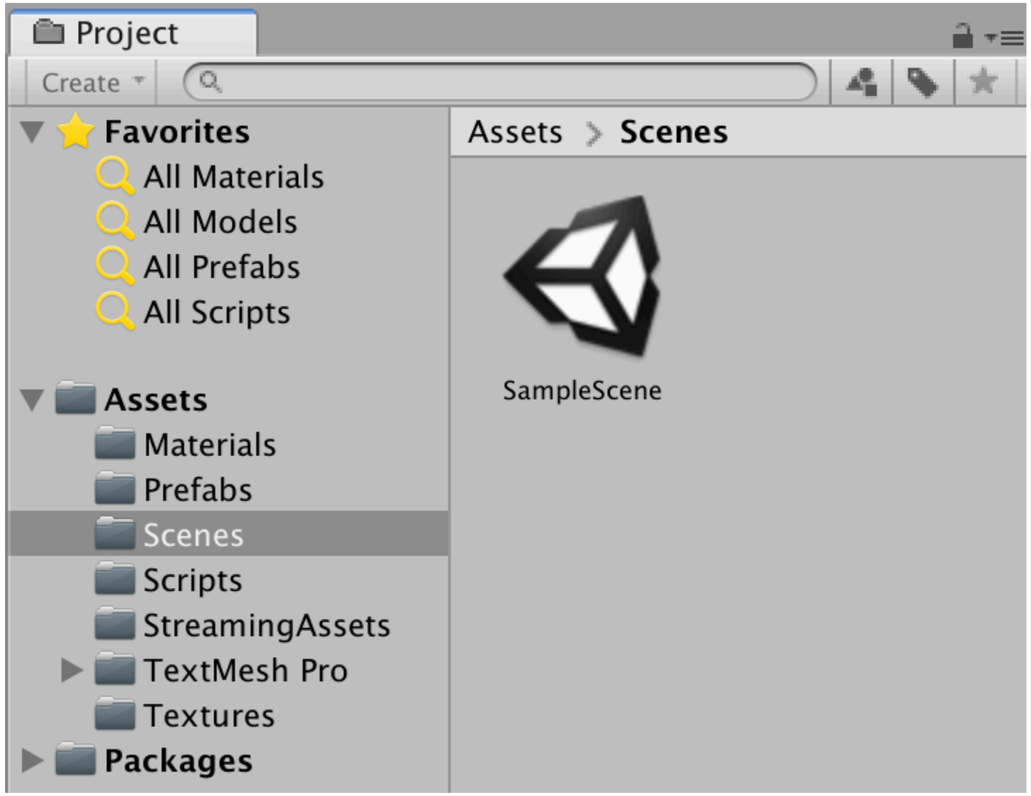
サンプルプロジェクトの開始

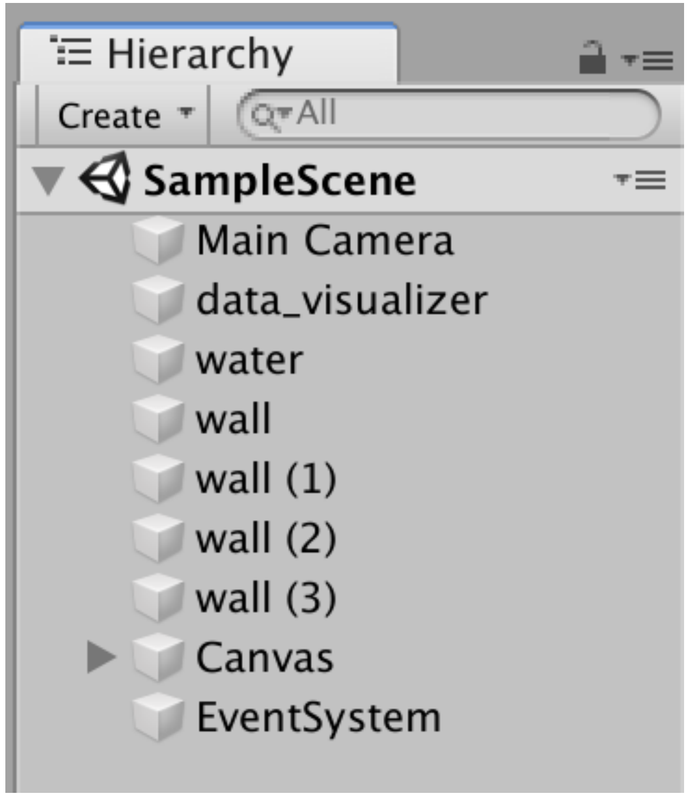
- Unity Hub を起動
- 「開く」から先ほど入手したサンプルプロジェクトを指定

unity

シーンのロード

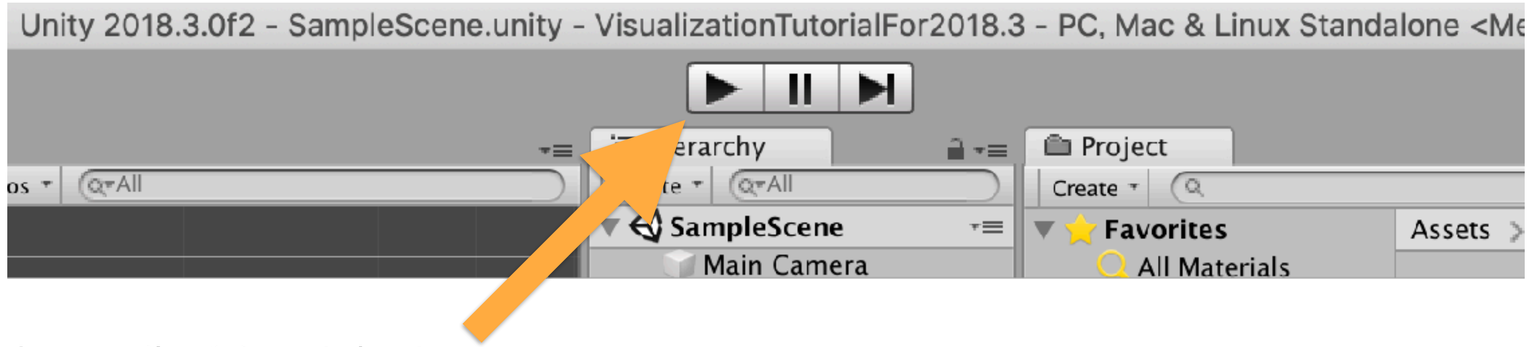
- Project ウインドウから Scenes を選択し、SampleScene をダブルクリック
- Hierarchyウインドウの内容から Scene のロードが正しく行われていることを確認



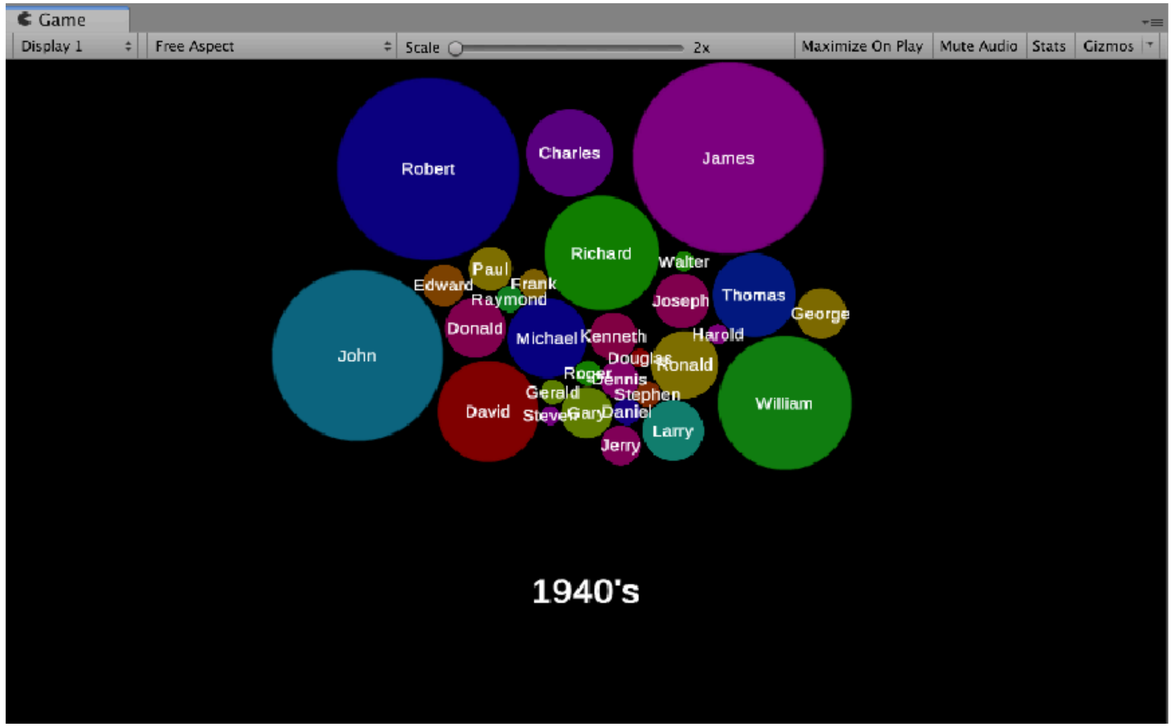


unity

実行



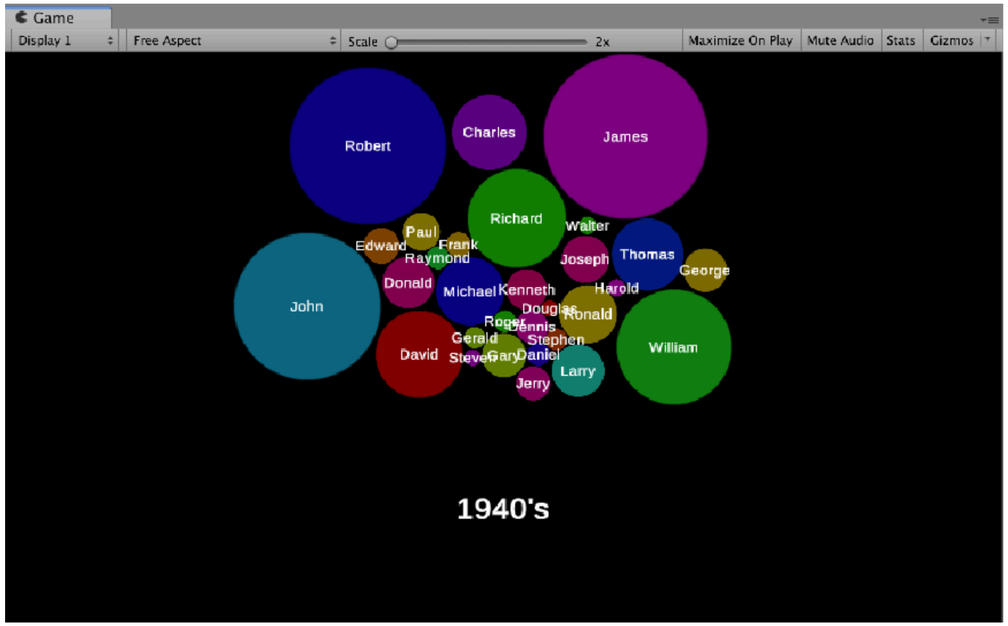
- 画面上部の再生ボタンを押す
- Gameウインドウにて動作確認

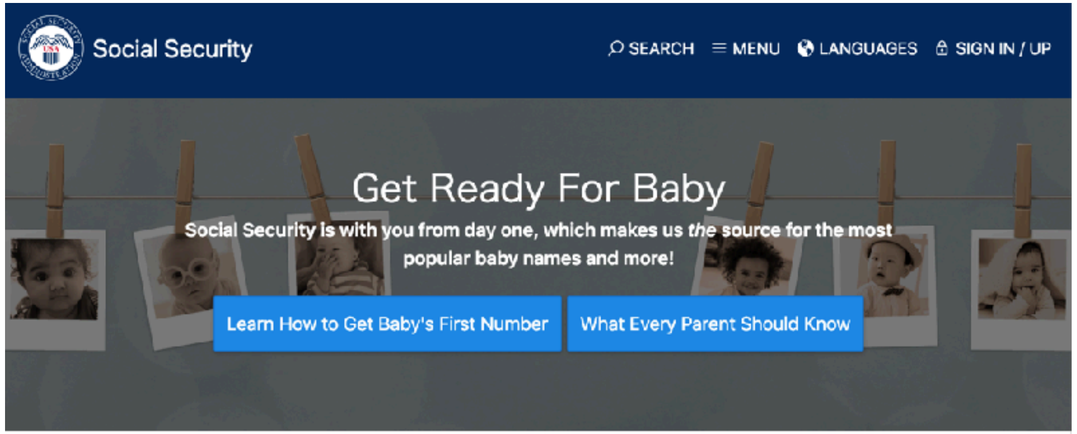


unity

サンプルプロジェクトの内容

- 北米の男の子のファーストネームの記録の年代ごとの推移を可視化
- 北米のSocialSecurityのサイト https://www.ssa.gov/oact/babynames/index.html から引用
- CSVに変換したものを読み込んでいる





Top 10 Baby Names of 2017

Rank	Male name	Female name
1	Liam	Emma
2	Noah	Olivia
3	William	Ava
4	James	Isabella
5	Logan	Sophia
6	Benjamin	Mia
7	Mason	Charlotte
8	Elijah	Amelia
9	Oliver	Evelyn
10	Jacob	Abigail

Watch our video countdown of 2017's Top 10 Most Popular

可視化対象データの内容

- 10年ごとにファイルを分離
- 上位の名前と総数を列挙

- 時系列にスカラ値を持つデータならなんでも可視化できる
  - 近代の国民総生産の推移
  - 世界史の国別人口推移など

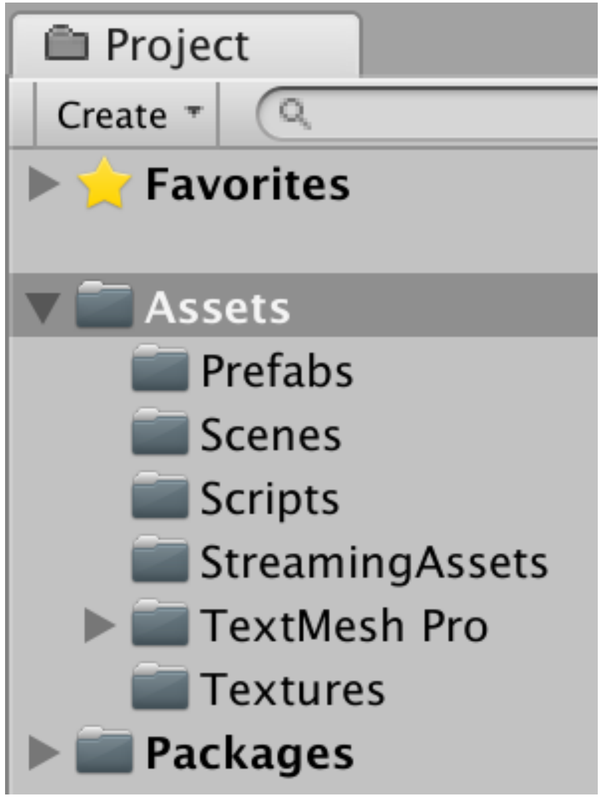
John	89950
William	84881
James	54056
George	47651
Charles	46656
Frank	30967
Joseph	26292
Henry	24139
Robert	24074
Thomas	23750
Edward	23133
Harry	22649
Walter	18185
Arthur	16180
Fred	15602
Albert	14375
Samuel	9129
Clarence	8760
Louis	8275
David	7569

フォルダー構成

フォルダー名	SCMに登録	備考
Assets	yes	ユーザが構成する内容のエントリーポイント
Library	no	各種中間ファイルのキャッシュなど
Logs	no	動作ログ
Packages	yes	インポートするパッケージの記述
ProjectSettings	yes	設定などを保持
Temp	no	一時ファイル。Unityを終了すると自動で消滅する
UnityPackageManager	yes	Unityが使用

※SCMはgitなどを指し、登録しないものは .gitignoreなどで除外する

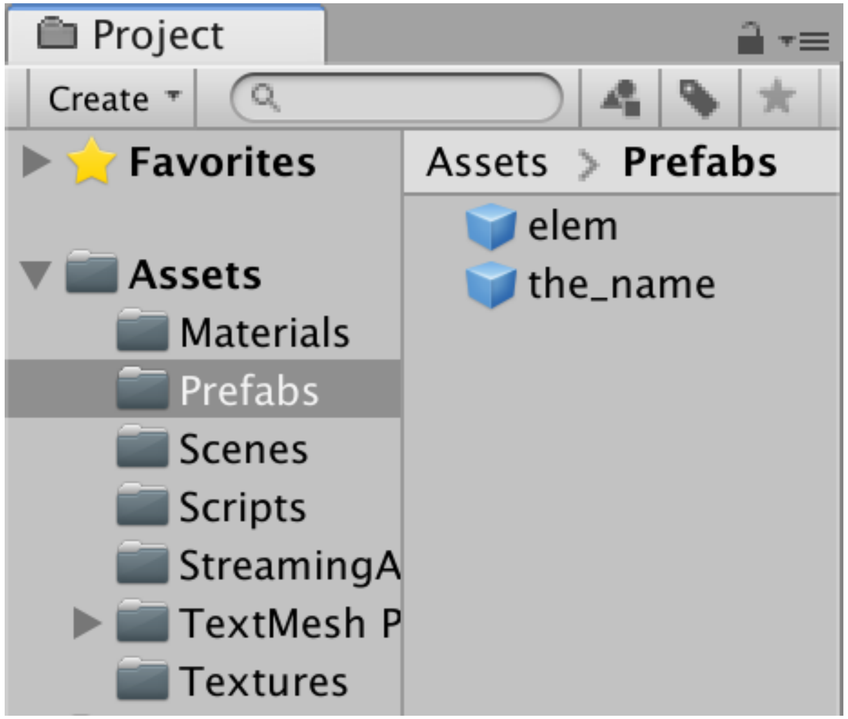
Assetsフォルダ以下の構成



フォルダー名	予約名か	備考
Prefabs	no	プレハブを格納
Scenes	no	シーンを格納
Scripts	no	C#スクリプトを格納
StreamingAssets	yes	ランタイムにファイルアクセスが可能
TextMesh Pro	no	TextMesh Proにより作られるリソース
Textures	no	テクスチャを格納

※予約名でないものは名前を変えても動作する

Prefabs 以下のファイル



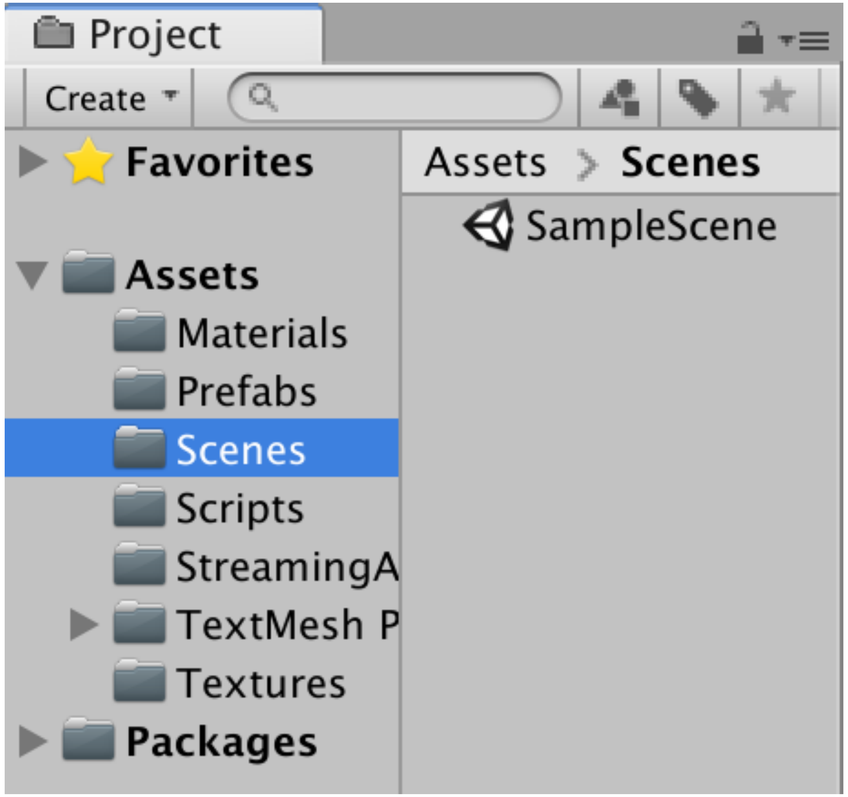
プレハブが格納されている

elem : 円のプレハブ

the_name : 名前のためのテキストのプレハブ

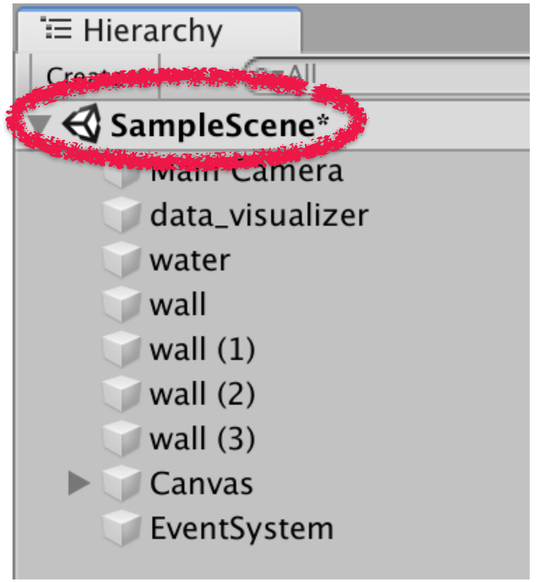
プレハブはUnityの用語で、似た構成の物体を複数生成する際の元になるもの。オブジェクト指向っぽく言えば「クラスオブジェクト」に相当する。ここを編集することは複製元の編集の意味になる。

Scenes 以下のファイル



シーンが格納されている。

SampleScene : 今回使用するシーン

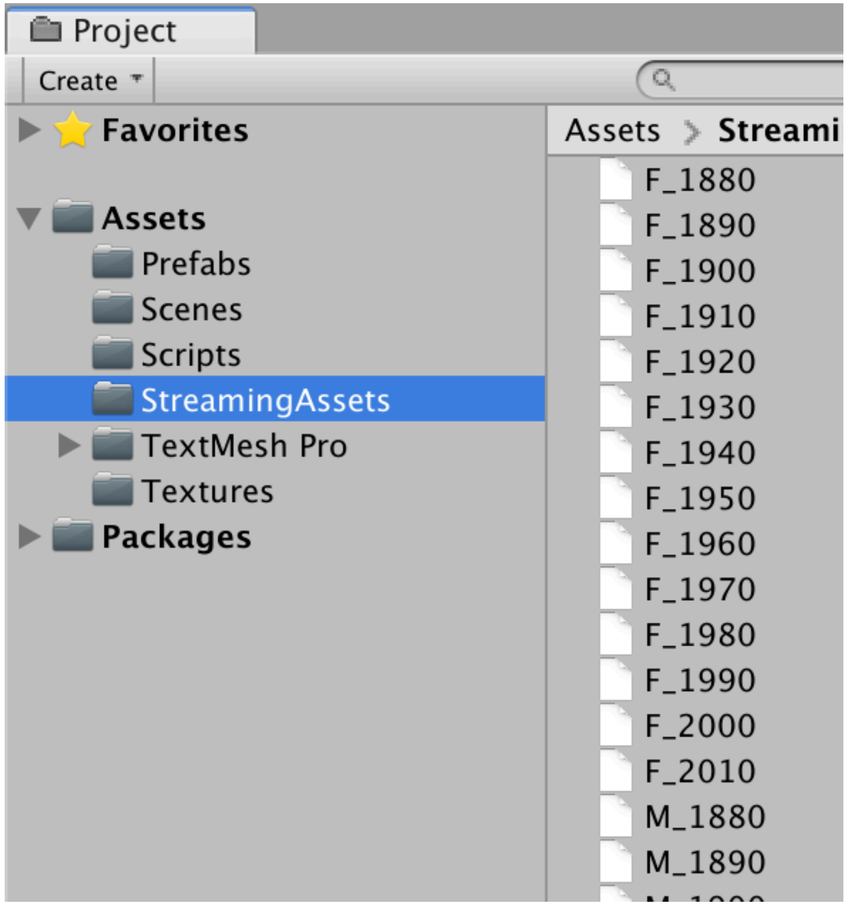


ヒエラルキーのタイトルに一致している。
登場しているオブジェクトのリストがヒエラルキー。
ヒエラルキーを保存したものがシーン。

シーンをダブルクリックすることで、シーンをロードできる。
何か実験したいときは New Scene から新しいシーンを作り、保存してから SampleScene をロードすると元に戻る。

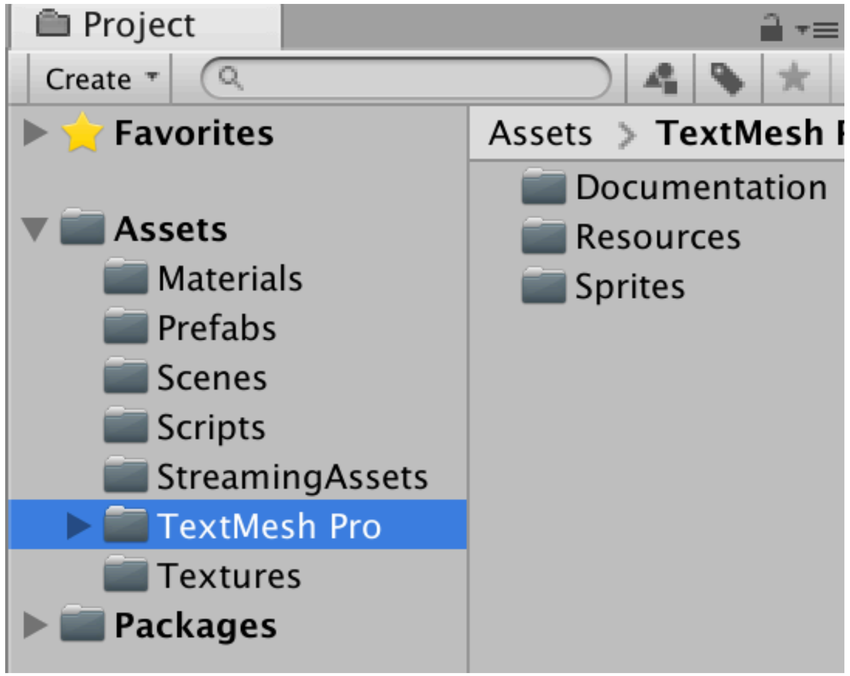
unity

StreamingAssets 以下のファイル



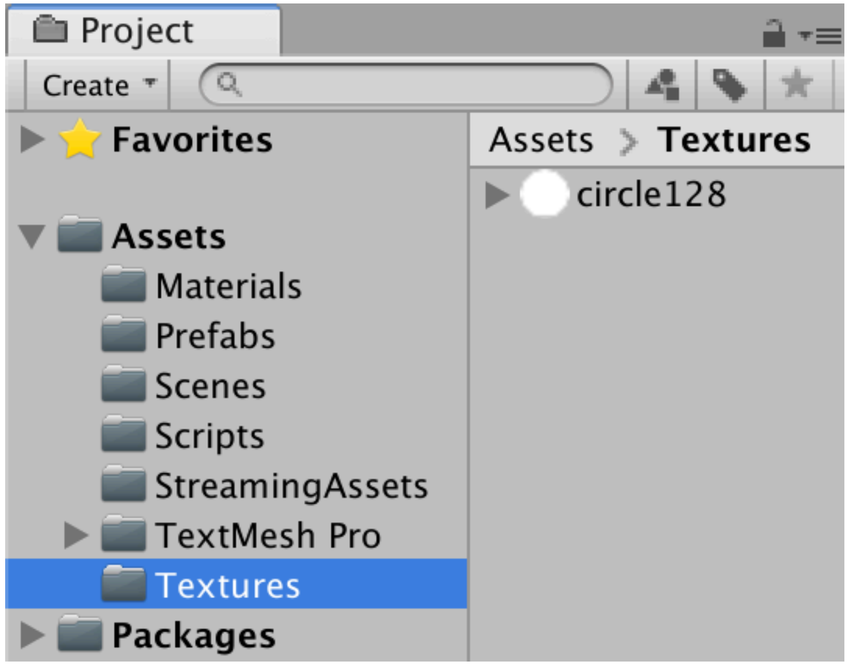
CSVファイルが格納されている。UnityはビルドしてWindowsアプリやMacアプリを作成でき、その際に自由に読み込めるファイルをバンドル（同梱）することができる。
Streaming Assets フォルダの下にあるファイルはバンドルされる。

Text Mesh Pro以下のファイル



TextMesh Pro というテキストレンダリングモジュールを
インポートした際に生成される。
内容を知る必要にせまられることはない。

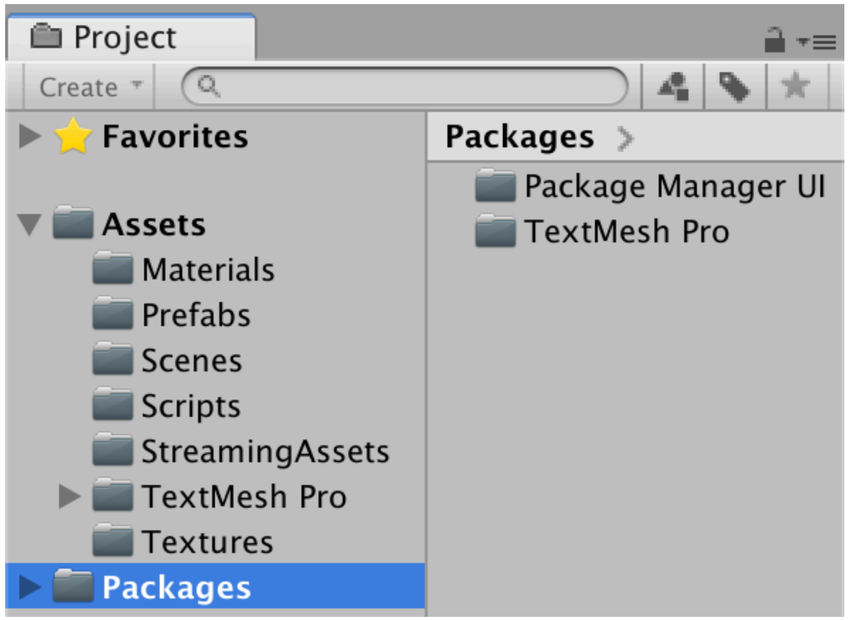
Textures以下のファイル



使用するテクスチャ（画像）が格納されている。
Prefabs/elem の Sprite Renderer で指定されている。

circle128 : 白く塗りつぶされた円

Packages以下のファイル



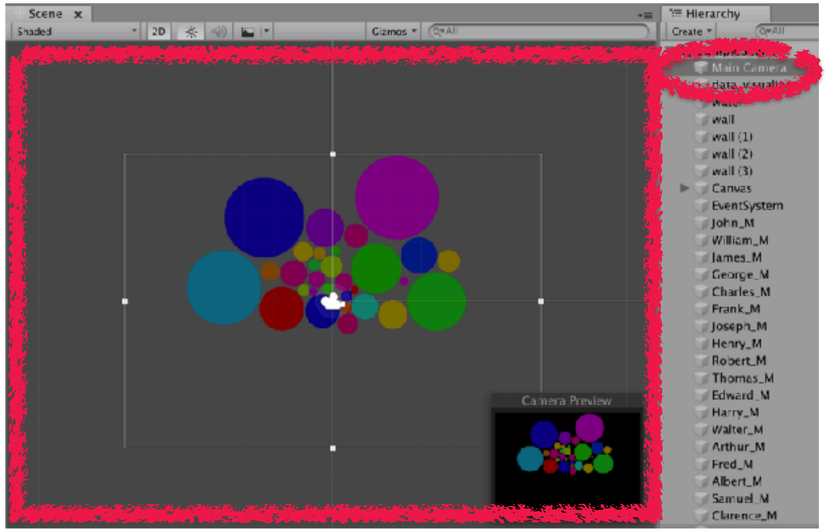
Unityのパッケージマネージャでパッケージをインポートすると
生成されるファイル群が格納されている。
Package Manager UI
TextMesh Pro
がインポートされていることが確認できる。
内容を知る必要にせまられることはない。

シーンの構成

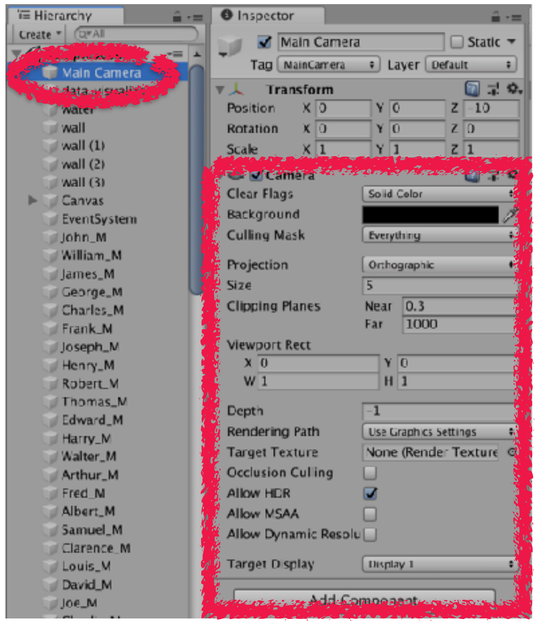
ポーズボタンで再生を止める

ヒエラルキーのオブジェクトをクリックすると

シーンビュー上で
ハイライトされる



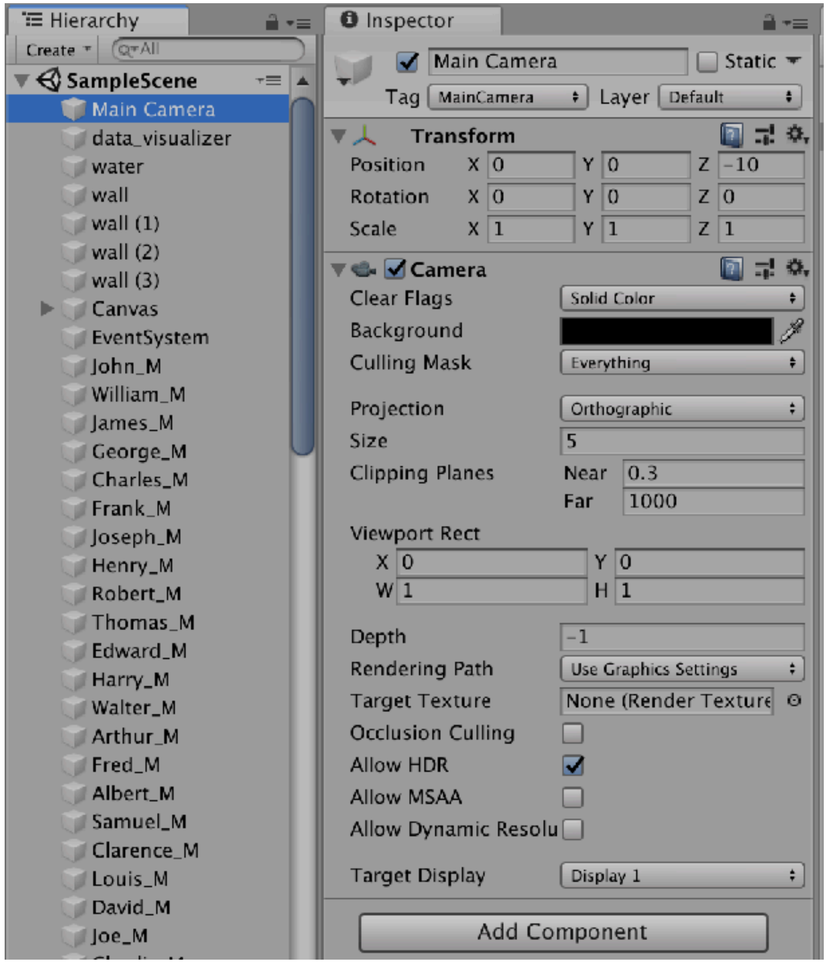
インスペクタで
情報を確認できる



カメラをクリックした場合

unity

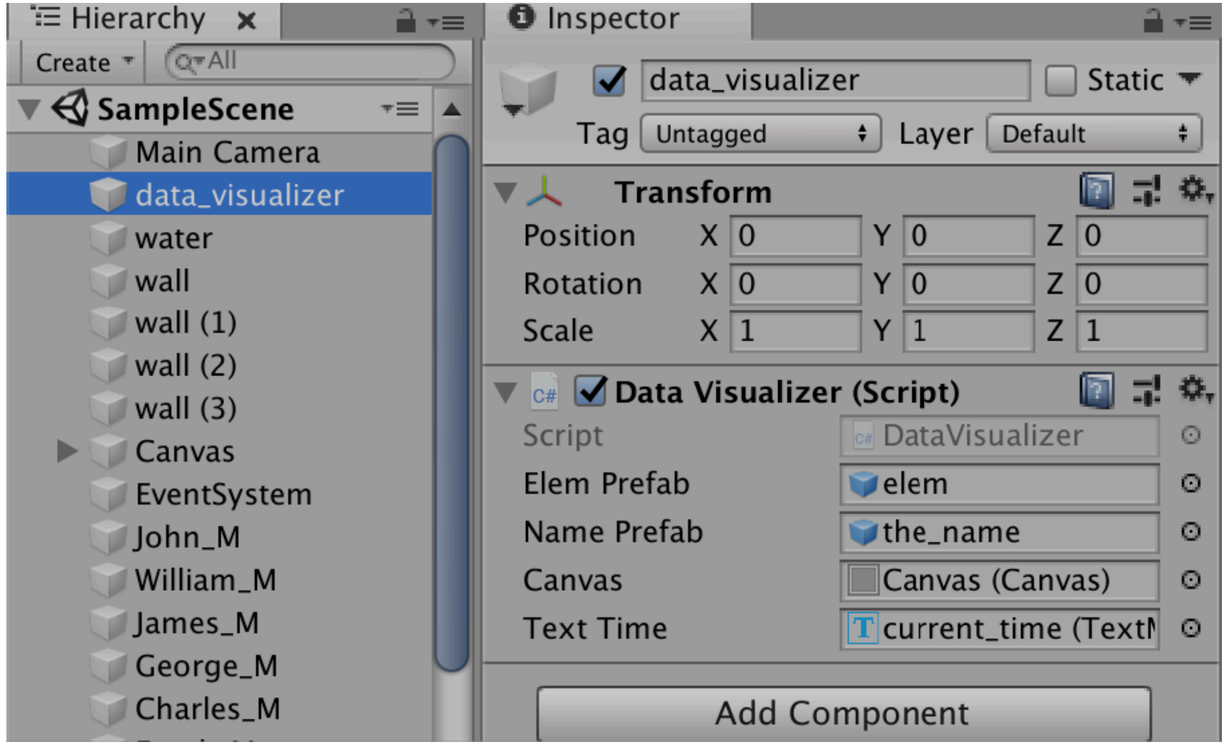
Main Camera



画面に表示するための仕組み
背景色などを変えられる
UI（テキストなど）を含まない設定になっている

unity

data_visualizer

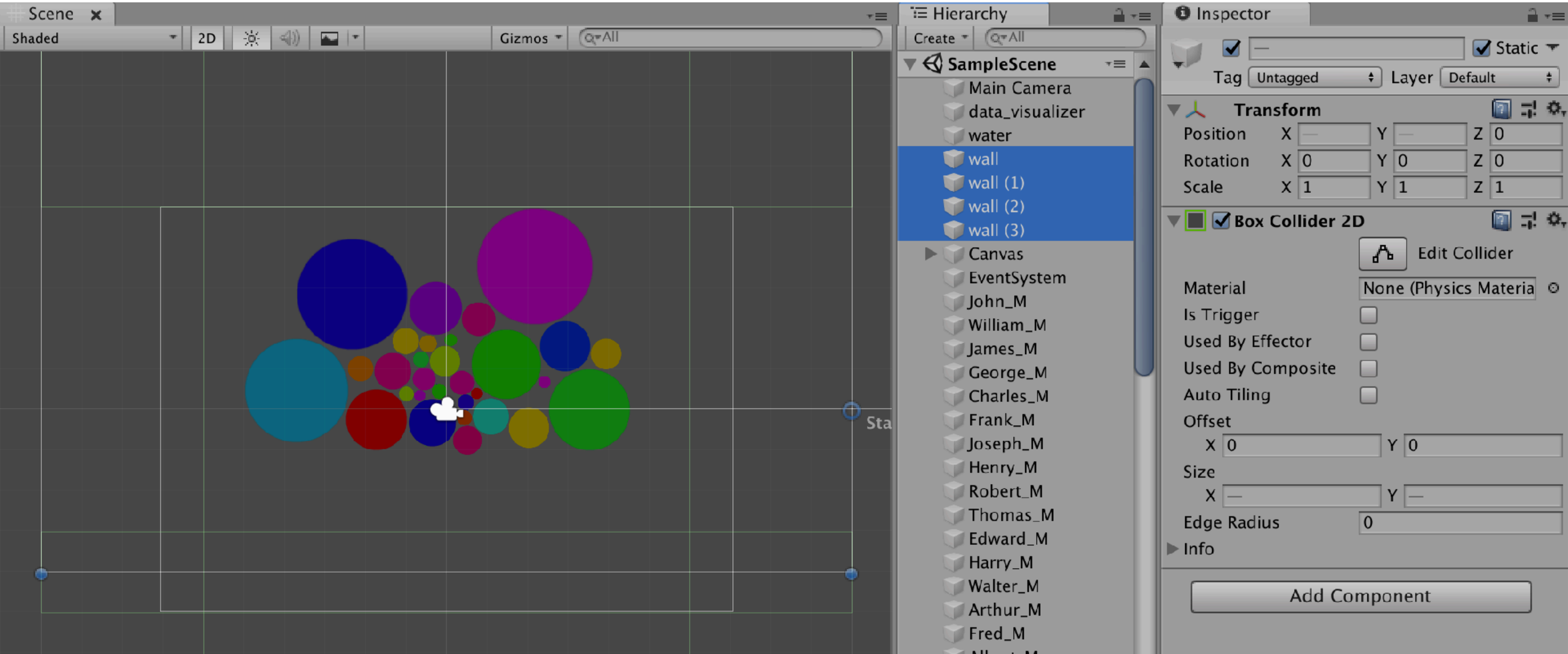


マネージャ的な存在
DataVisualizer.cs に記述されたC#スクリプトを持つ
概念上のオブジェクトなので
Transform の Position, Rotationなどに意味はない

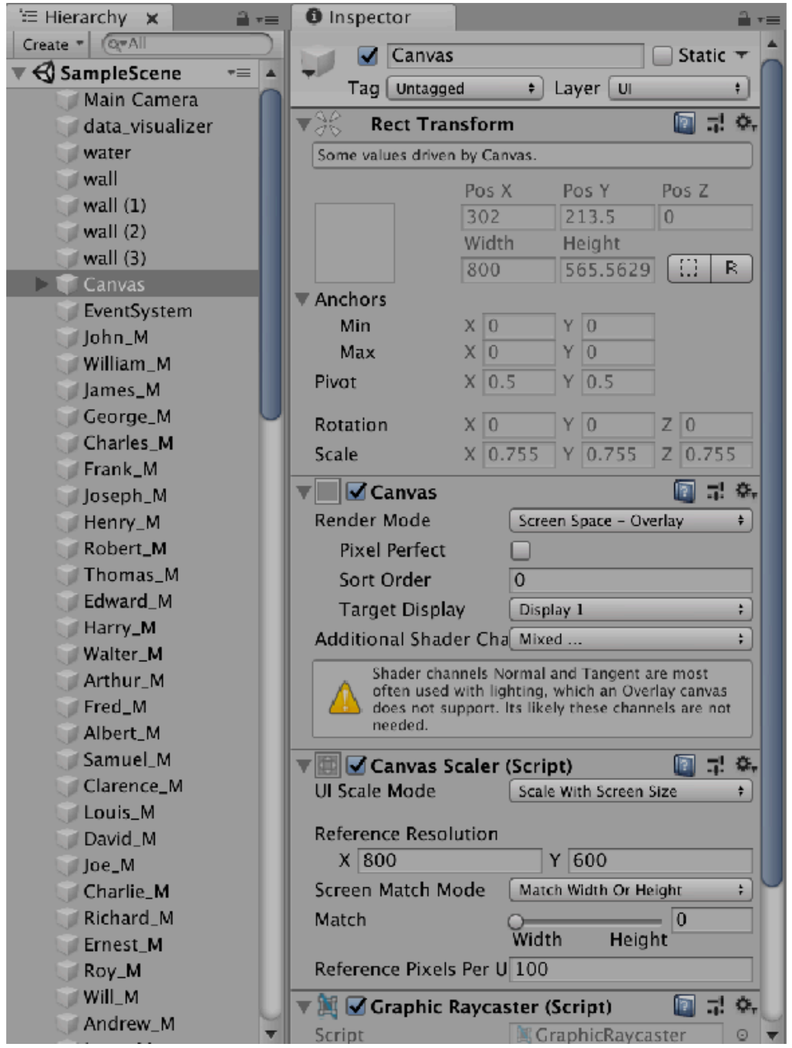
unity

wall, wall (1), wall (2), wall (3)

Box Collider 2D を保持することで壁を実現する。
ボールが外に逃げないように壁になっている

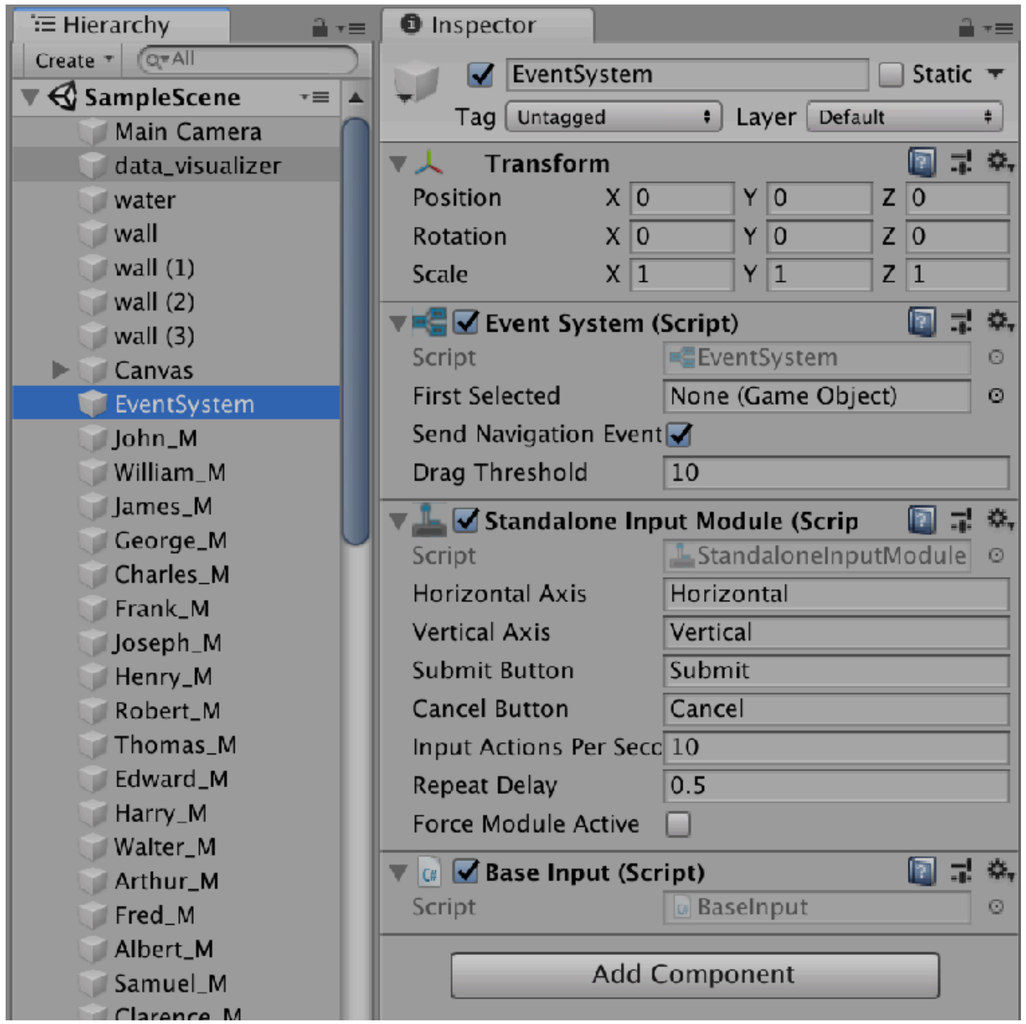


Canvas



UIを管理する。UI要素を作成すると自動的に作成され、
UI要素はこの下に格納される。
今回は文字表示がUI要素になっている

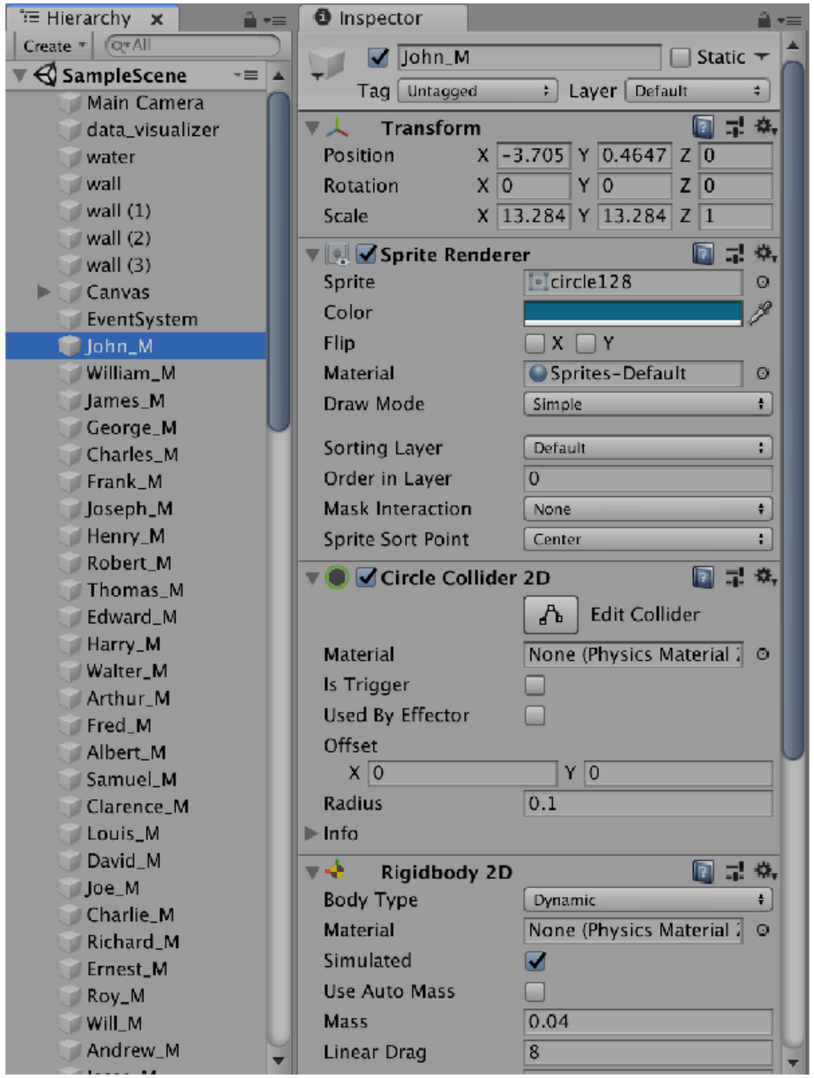
EventSystem



UI要素を追加するとCanvasと同時に自動的に作成される。
機能としてはキー押し下げなどを扱うが、通常意識する必要はない

unity

John_M



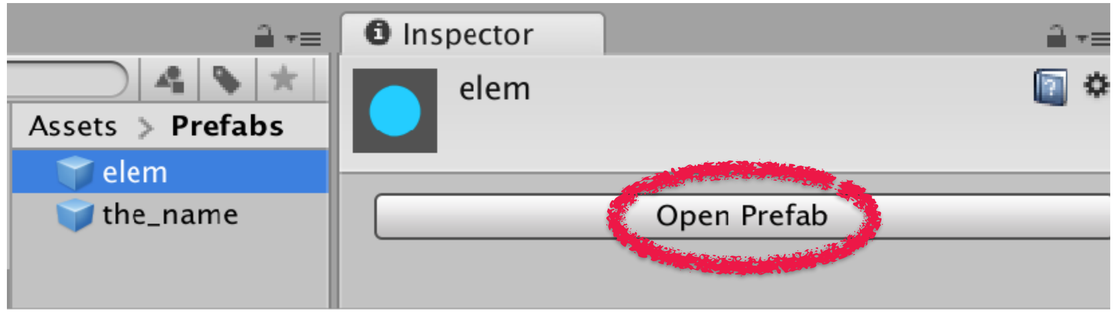
プレハブの elem から生成されたオブジェクト。
インスペクタの中を elem と比較すると同じ構成になっているのがわかる。

John_M以下のオブジェクトはすべて、プレハブelemから生成されたもの。

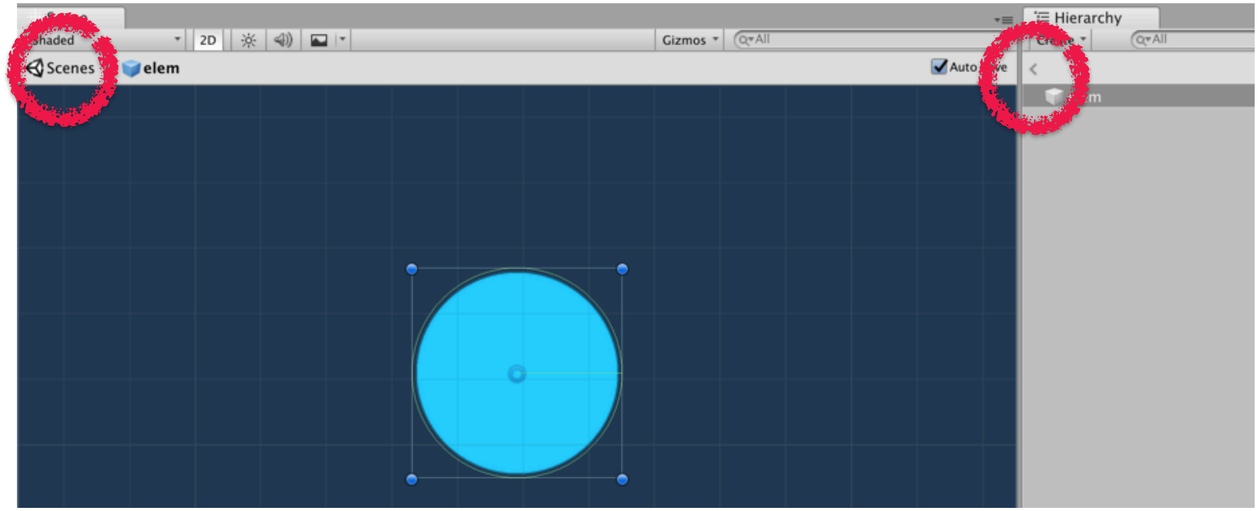
unity

elem プレハブのコンポーネント群

Inspector ウインドウの Open Prefab を押すと
Scene ウインドウおよび Hierarchy ウインドウが編集モードに入る

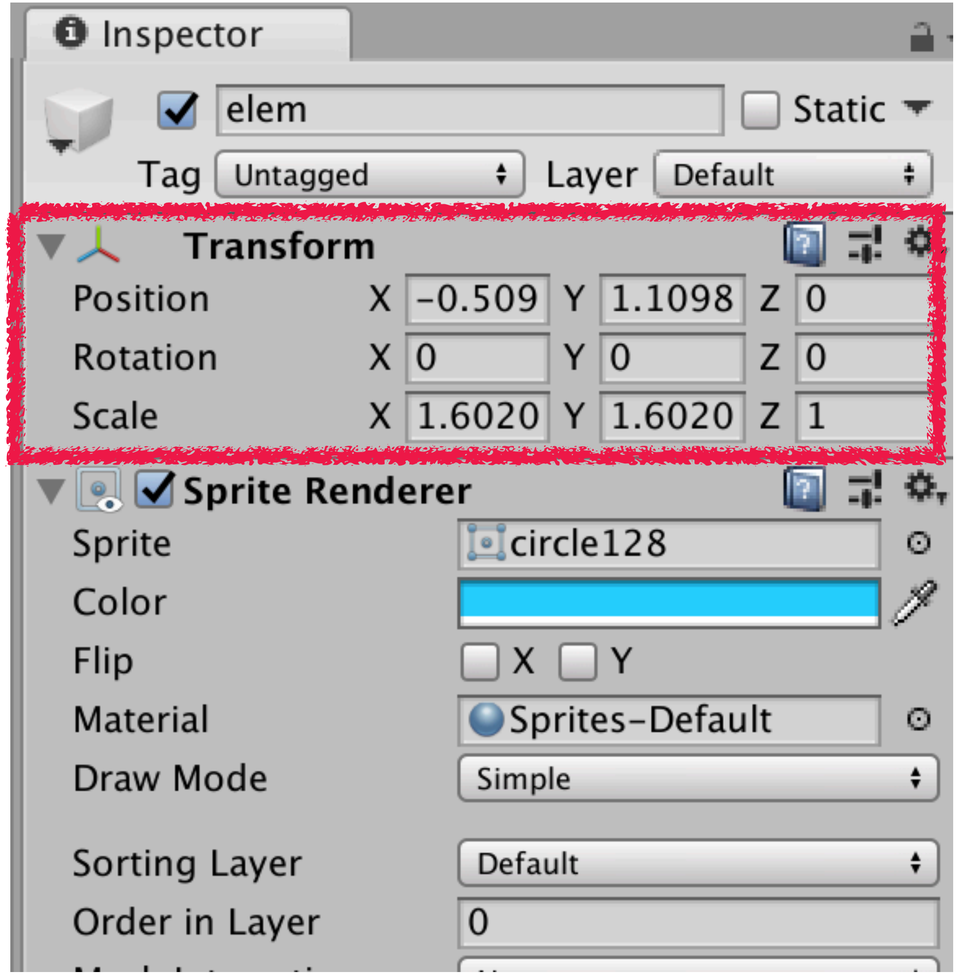


編集モードから復帰するにはこれらのボタン



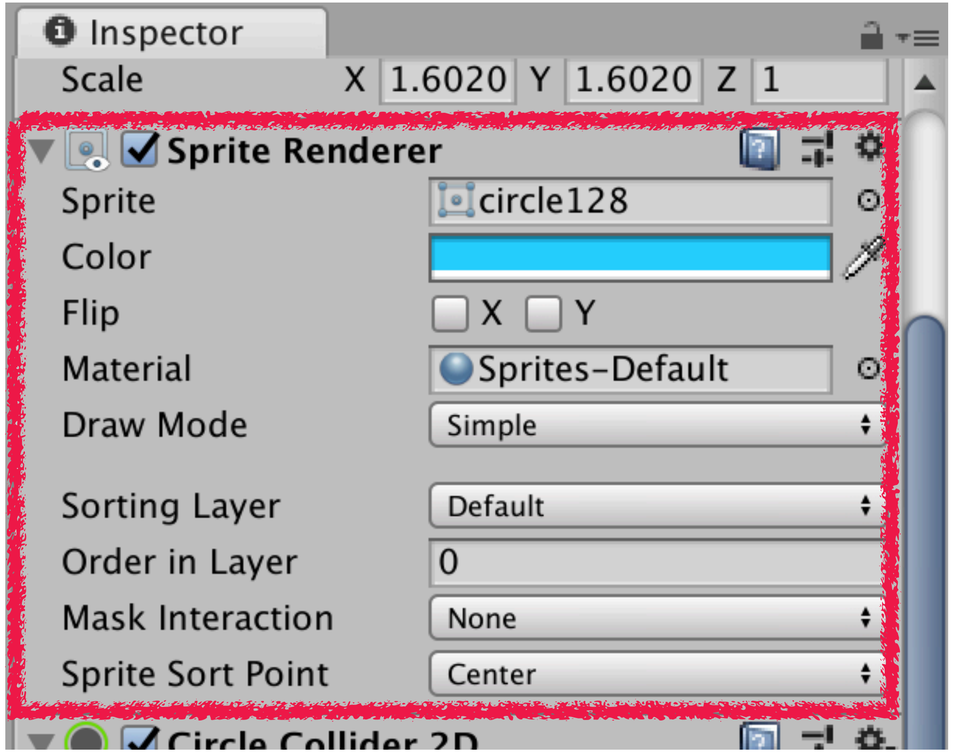
unity

Transform



すべてのオブジェクトが持つ、オブジェクトの位置・回転を保持する

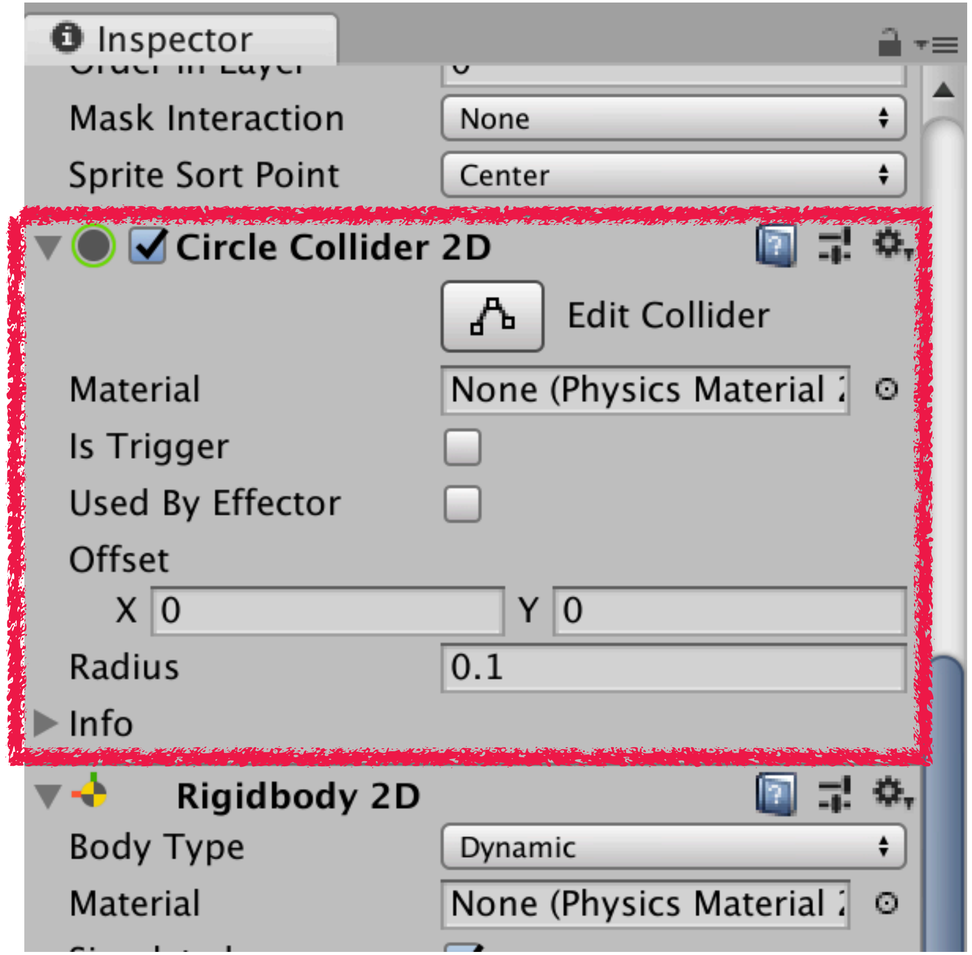
Sprite Renderer



二次元のオブジェクトを描画するコンポーネント。

Sprite : テクスチャ
Color : 色
Filp : 反転
Material : 素材。シェーダを自作する場合はマテリアルの作成が必要
などなど

Circle Collider 2D

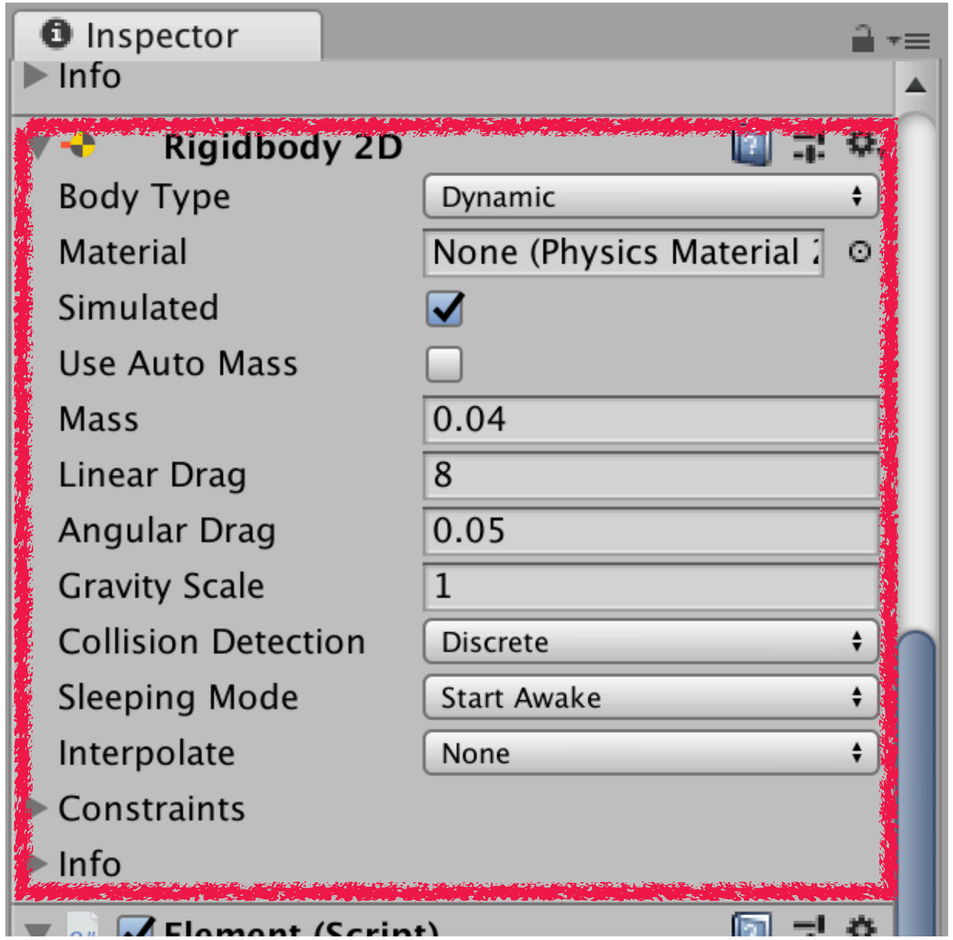


円形のコライダーコンポーネント。衝突判定用のエリア指定となる。

壁や浮力に反応する計算に使用される。

unity

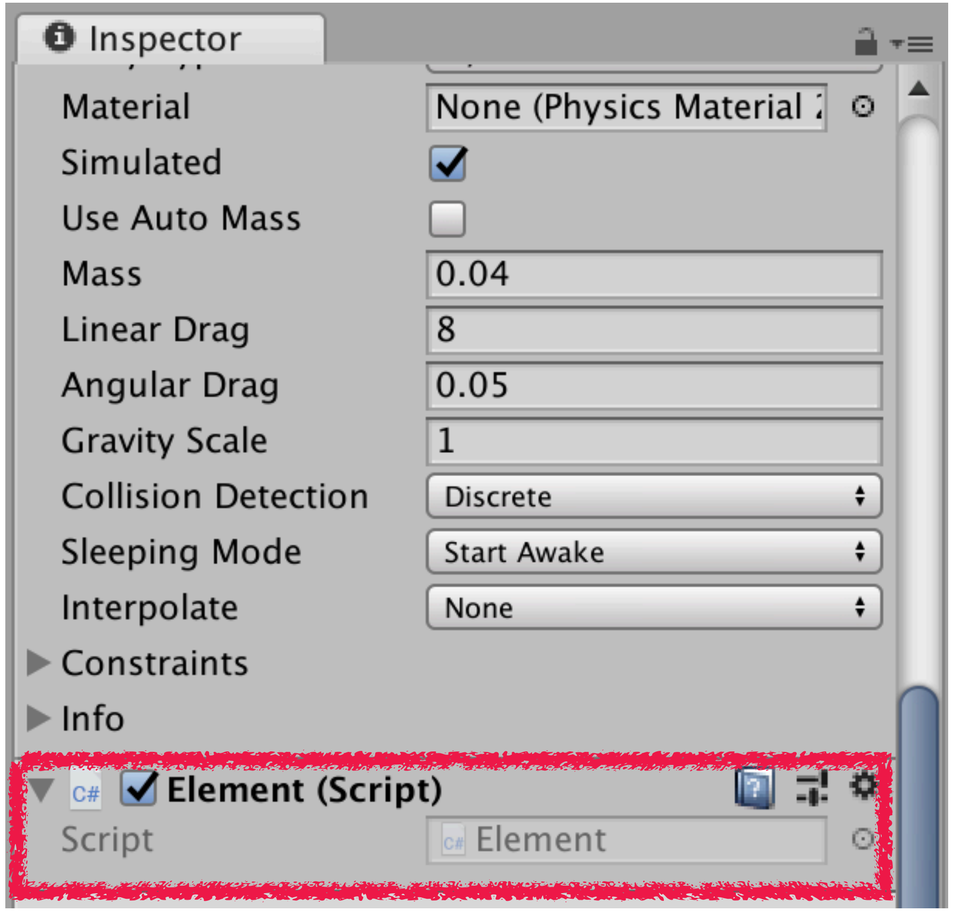
Rigidbody 2D



剛体シミュレーションのコンポーネント。

質量などが定義される。動きに影響のあるDragなどの値を保持。

Element (Script)

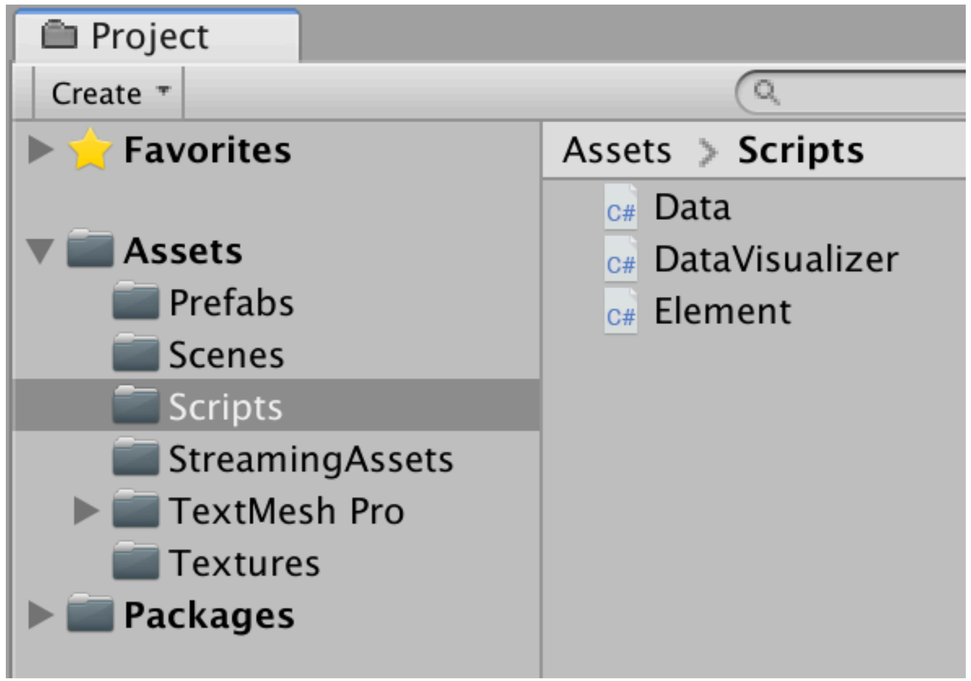


Element.cs を保持することを示す。

unity

Unityのプログラム

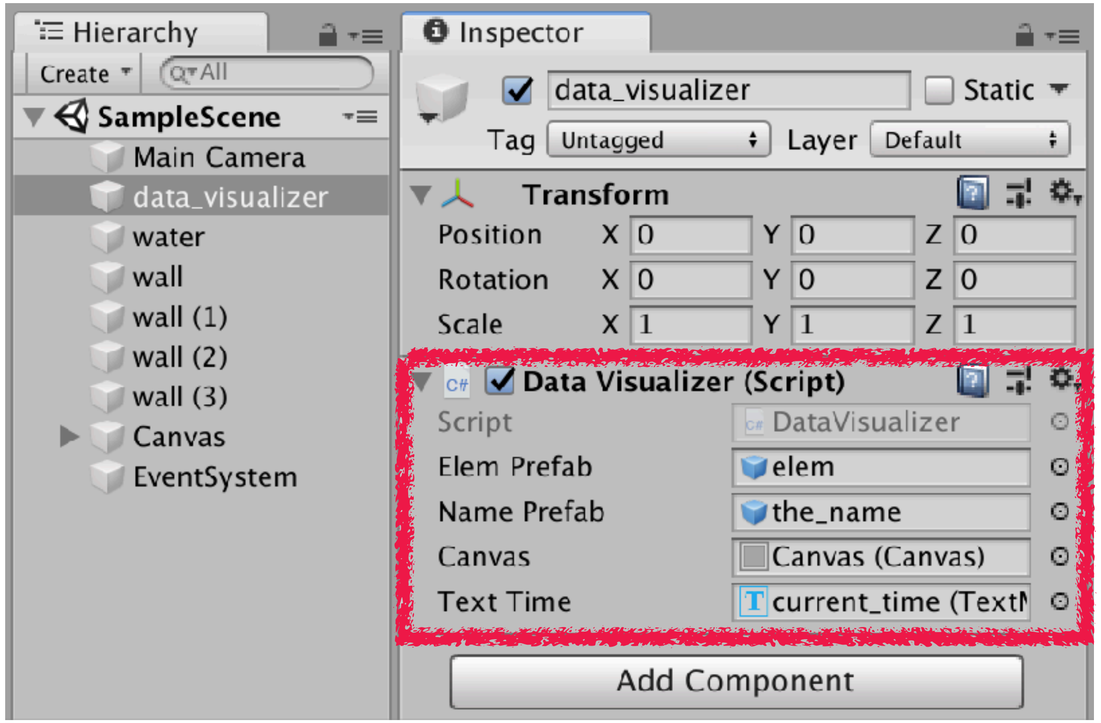
詳細な解説はソースコードのコメントを参照のこと。



unity

MonoBehaviourを継承

```
8  public class DataVisualizer : MonoBehaviour
9  {
10     public GameObject m_ElemPrefab; // インスペクタから elem プレハブを設定
11     public GameObject m_NamePrefab; // インスペクタから the name プレハブを設定
12     public Canvas m_Canvas;         // インスペクタから Canvas オブジェクトを設定
13     public TextMeshProUGUI m_TextTime; // インスペクタから current_time オブジェクトを設定
```

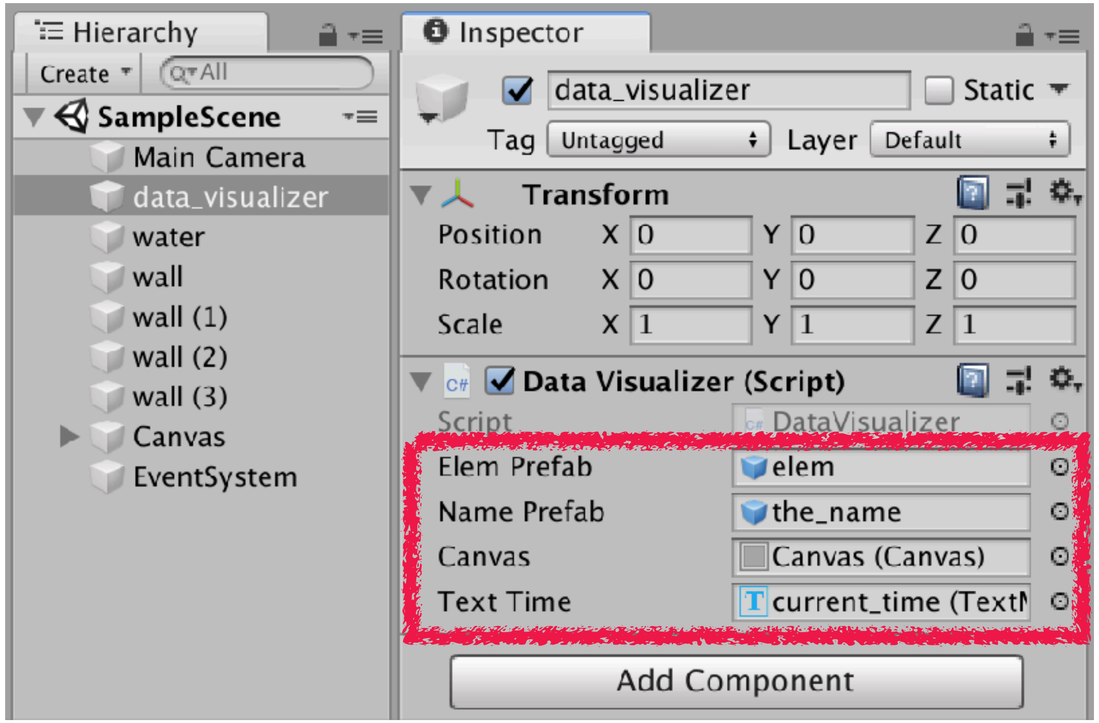


オブジェクトに追加するにはMonoBehaviour
である必要がある。

unity

オブジェクトを参照

```
8  public class DataVisualizer : MonoBehaviour
9  {
10     public GameObject m_ElemPrefab; // インスペクタから elem プレハブを設定
11     public GameObject m_NamePrefab; // インスペクタから the name プレハブを設定
12     public Canvas m_Canvas;         // インスペクタから Canvas オブジェクトを設定
13     public TextMeshProUGUI m_TextTime; // インスペクタから current_time オブジェクトを設定
```



public で宣言することで、インスペクタから
他のオブジェクトを参照することができる。

unity

特殊関数：Awake

```
 8 public class DataVisualizer : MonoB
 9 {
10     public GameObject m_ElemPrefab;
11     public GameObject m_NamePrefab;
12     public Canvas m_Canvas;
13     public TextMeshProUGUI m_TextTi
14     Data m_Data;
15     List<Element> m_ElementList;
16
17     void Awake()
18     {
19         Random.InitState(12345);
20     }
21
```

AwakeはUnityによりオブジェクト生成直後（すなわち再生直後）に呼ばれる。

ここでは乱数のシードを設定。

特殊関数：Start

```
 8 public class DataVisuali
 9 {
10     public GameObject m_
11     public GameObject m_
12     public Canvas m_Canv
13     public TextMeshProUG
14     Data m_Data;
15     List<Element> m_Elem
16
17     void Awake()
18     {
19         Random.InitState
20     }
21
22     void Start()
23     {
24         m_Data = new Dat
25         m_Data.load();
26         // data.dump();
27
```

StartはUnityによりUpdateループ（後述）の直前に一度だけ呼ばれる。

ここではデータを読み込む。

Data.cs

Unity固有の特徴はなく、一般的なC#プログラム。
CSVをパースしながら読み込む。

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

namespace VIS {

// CSVファイルの行に対応
public class InfoUnit
{
    public string m_Name;       // 名前
    public int m_Value;         // 数値
    public int m_Type;          // 0:Male, 1:Female
}

// CSVファイルひとつに対応
public class DataUnit
{
    public int m_Time;          // 年代
    public List<InfoUnit> m_InfoUnitList;
    public Dictionary<string, InfoUnit> m_InfoUnitDictionary; // 検索用
}

// 全てのデータを保持
public class Data
{
    public List<DataUnit> m_DataUnitList;
    public HashSet<string> m_NameSet; // 重複のない名前の集合
    public float m_MaxValue;          // 正規化のため全体の最大値を算出しておく

    // 年代を列挙
    public int[] GetKeyTimeList()
    {
        var res = new int[m_DataUnitList.Count];
        for (var i = 0; i < m_DataUnitList.Count; ++i) {
            res[i] = m_DataUnitList[i].m_Time;
        }
```

Element.cs

ボールごとの動作を記述。

```
7  public class Element : MonoBehaviour
8  {
9      string m_Name;               // 名前
10     float m_CurrentScale = 0f;   // 現在のスケール
11     float m_TargetScale = 0f;    // ターゲットスケール
12     GameObject m_NameObject;     // 名前表示用のオブジェクト
13
14     // 名前設定
15     public void SetName(string name) { m_Name = name; }
16     // 名前オブジェクト設定
17     public void SetNameObject(GameObject name_object) { m_Nam
18
19     // 正規化用の拡大率を与えてターゲットスケールを更新
20     public void SetTargetData(DataUnit data_unit, float ratio
21     {
22         if (data_unit.m_InfoUnitDictionary.ContainsKey(m_Name
23             var info_unit = data_unit.m_InfoUnitDictionary[m_
24             m_TargetScale = ((float)info_unit.m_Value)*ratio;
25         } else {
26             m_TargetScale = 0f; // スケールゼロ
27         }
28     }
29
30     // Unityにより自動的に呼ばれるUpdate
31     void Update()
32     {
33         if (m_Name == null)      // 念の為nullアクセス回避
34             return;
35         // スケールをターゲットスケールに近づけるアニメーション
36         m_CurrentScale = Mathf.Lerp(m_CurrentScale, m_TargetS
37         // スケールを設定
```

特殊関数：Update

```
7  public class Element : MonoBehaviour
8  {
9      string m_Name;               // 名前
10     float m_CurrentScale = 0f;   // 現在のスケール
11     float m_TargetScale = 0f;    // ターゲットスケール
12     GameObject m_NameObject;     // 名前表示用のオブジェクト
13
14     // 名前設定
15     public void SetName(string name) { m_Name = name; }
16     // 名前オブジェクト設定
17     public void SetNameObject(GameObject name_object) { m_Nam
18
19     // 正規化用の拡大率を与えてターゲットスケールを更新
20     public void SetTargetData(DataUnit data_unit, float ratio
21     {
22         if (data_unit.m_InfoUnitDictionary.ContainsKey(m_Name
23             var info_unit = data_unit.m_InfoUnitDictionary[m_
24             m_TargetScale = ((float)info_unit.m_Value)*ratio;
25         } else {
26             m_TargetScale = 0f; // スケールゼロ
27         }
28     }
29
30     // Unityにより自動的に呼ばれるUpdate
31     void Update()
32     {
33         if (m_Name == null)      // 念の為nullアクセス回避
34             return;
35         // スケールをターゲットスケールに近づけるアニメーション
36         m_CurrentScale = Mathf.Lerp(m_CurrentScale, m_TargetS
37         // スケールを設定
```

Unityにより毎フレーム（1/60秒周期などで）呼ばれる。

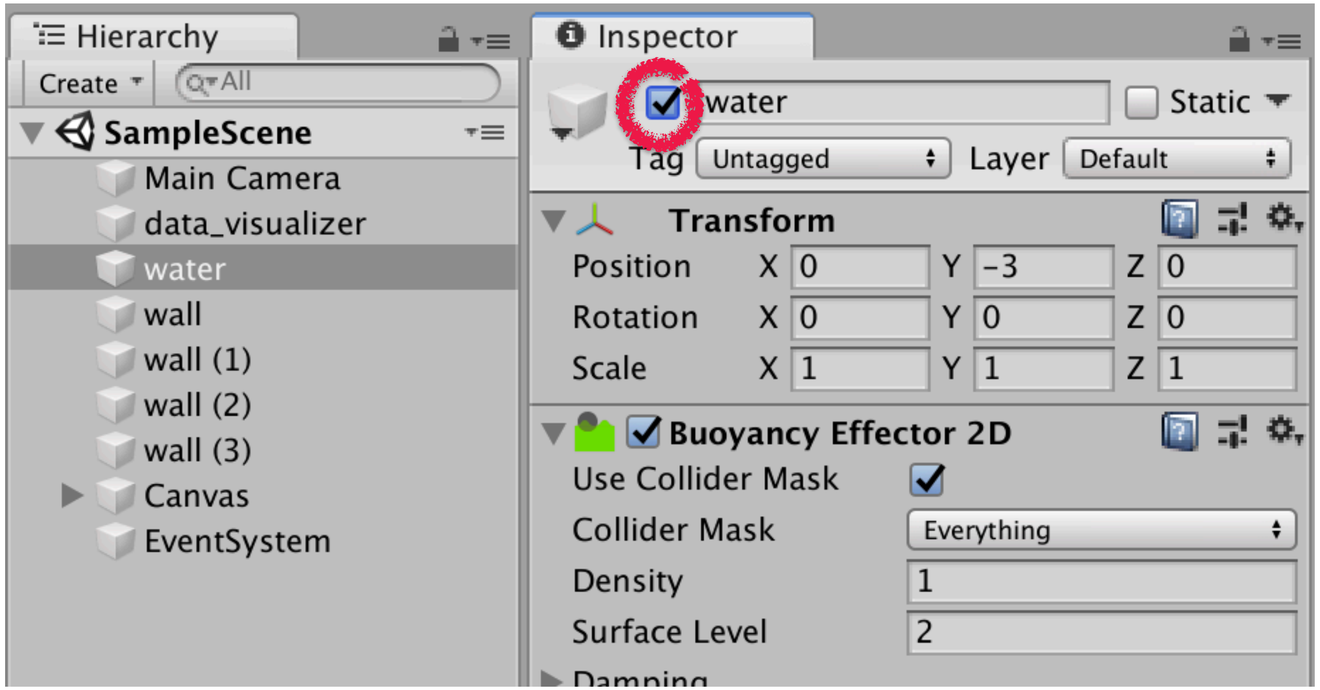
unity

改造してみよう

- 年代ごとに４秒待つところを短くしてみよう→DataVisualizer.csを修正
- girls のデータを表示してみよう→Data.csを修正
- 半径に値を入れているので面積比例に直してみよう→Element.csを修正
- 動きを修正してみよう
  - 浮力をやめたらどうなる？
  - 生成地点を画面中央にするには？
  - 重力を無効にしたらどうなる？
  - AddForceで引力を働かせてるコードを変えてみよう
  - Mathf.Lerpでボールが膨らむ遷移のコードを変えてみよう（遅くするなど）

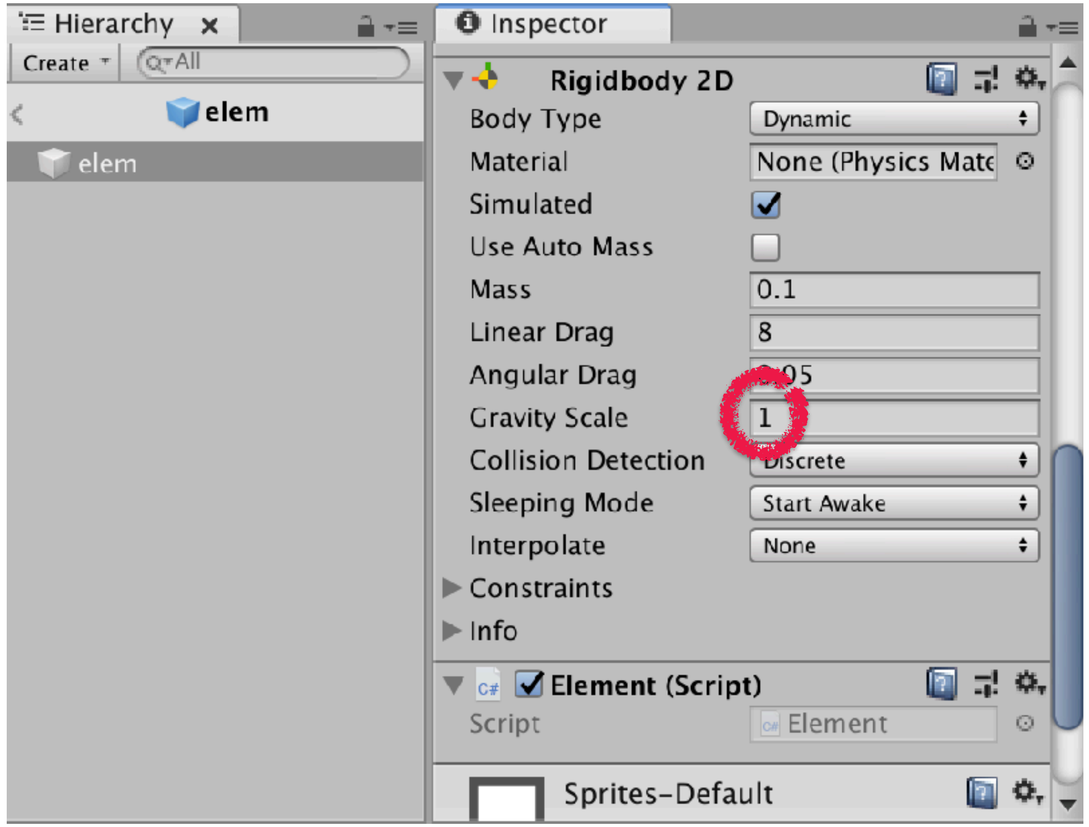
浮力を無効にするには

water オブジェクトを削除すればよい。



削除すると元に戻せなくなると心配な場合
Inspectorでチェックを外すと
そのオブジェクトが「存在しない」と同義になる。

ボールの重力を無効にするには



前述のプレハブの編集にてelemの編集画面にし、
Gravity Scale を 0 にする。

unity

本資料：

http://dsedb.sakura.ne.jp/doc/tutorial-yasuhara.pdf

補足資料：

http://dsedb.sakura.ne.jp/doc/tutorial-supplement-yasuhara.pdf

デモ動画：

https://youtu.be/Bj7sMynJS60

記事：

https://qiita.com/yuji_yasuhara/private/9c9f621d0dc16d8773a8

unity