

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

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



ユニティ・テクノロジーズ・ジャパン 安原祐二



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
 - が推奨されます。
 - 手順: <u>https://docs.unity3d.com/ja/current/Manual/GettingStartedInstallingHub.html</u>
- Unityアカウント作成方法
 - カウント作成画面は英語になります。
 - 手順:<u>https://docs.unity3d.com/ja/current/Manual/GettingStarted.html</u>



• 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

• 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

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



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





cene をダブルクリック Eしく行われていることを確認



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 2018.3.0f2 - SampleScene.unity - VisualizationTutorialFor2018.3 - PC, Mac & Linux Standalone <Me

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 <u>https://www.ssa.gov/oact/babynames/index.html</u>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



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	7560

フォルダー構成

フォルダー名	SCMに登録	
Assets	yes	ユーザ
Library	no	各種中
Logs	no	動作口
Packages	yes	インポ
ProjectSettings	yes	設定な
Temp	no	一時フ
UnityPackageManager	yes	Unity

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



備考

が構成する内容のエントリーポイント

間ファイルのキャッシュなど

グ

ートするパッケージの記述

どを保持

ァイル。Unityを終了すると自動で消滅する

が使用



フォルダー Prefabs Scenes Scripts StreamingAss TextMesh Pro Textures



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

※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



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。







StreamingAssets以下のファイル

💼 Project	
Create *	Q
Favorites	Assets > Stream
	F_1880
V Assets	F_1890
📰 Prefabs	F_1900
Scenes	F_1910
Scripts	F_1920
StreamingAssets	F_1930
🕨 📰 TextMesh Pro	F_1940
Textures	F_1950
🕨 📰 Packages	F_1960
	F_1970
	F_1980
	F_1990
	F_2000
	F_2010
	M_1880
	M_1890
	LL 1000



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



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。



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

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









カメラをクリックした場合

シーンの構成

Main Camera

'≡ Hierarchy 🔒 📲	O Inspector	<u> </u>
Create * Q*All	Main Camer	a Static 🔻
🔻 🚭 SampleScene 🛛 📲 🔺	Tag	
👘 Main Camera 👘	Tag Maincamera	Layer Default
🕥 data_visualizer	▼ 🙏 Transform	🛄 다 호.
water	Position X 0	Y 0 Z -10
🗊 wall	Rotation X 0	Y 0 Z 0
🔰 wall (1)	Scale X 1	Y 1 Z 1
wall (2)	T Camera	□ -! ඊ.
wall (3)	Clear Flags	Solid Color +
Canvas 🖉	Background	
EventSystem	Gulling Mask	
John_M	Culling Mask	Everything
William_M	Projection	Orthographic +
James_M	Size	5
George_M Charles M	Clipping Planes	Near 0.3
Erank M		Far 1000
loseph M	Viewport Rect	
Henry M	X 0	Y 0
Robert_M	W 1	H 1
Thomas_M		
Edward_M	Depth	-1
Harry_M	Rendering Path	Use Graphics Settings 🕴
Walter_M	Target Texture	None (Render Texture 🛛 🛛
Arthur_M	Occlusion Culling	
🗇 Fred_M	Allow HDR	
Albert_M	Allow MSAA	
Samuel_M	Allow Dynamic Resol	u
Clarence_M	-	
Louis_M	larget Display	Display 1 +
David_M	Add Co	mnanant
Joe_M	Add Co	Simponent



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

シーンの構成

data_visualizer

🔚 Hierarchy 🗙 🛁 📲	Inspector	2
Create • Q•All	data visualizer	Statio
🔻 🍕 SampleScene 🛛 📼 🔺		
Main Camera	Tag Untagged + Layer	Default
💗 data_visualizer	▼ 人 Transform	💽 🗐 🗐
water	Position X 0 Y 0	Z 0
🕥 wall	Rotation X 0 Y 0	Z 0
🕥 wall (1)	Scale X 1 Y 1	Z 1
wall (2)	- Dete Mieuelinen (Cenint)	
🕥 wall (3)	▼ C# ✓ Data Visualizer (Script)	
Canvas	Script DataVisi	ualizer
EventSystem	Elem Prefab 🛛 🗊 elem	
John_M	Name Prefab 🛛 🗊 the_nan	ne
🧊 William_M	Canvas	(Canvas)
James_M	Text Time Text Time	_time (Text
George_M		
Charles_M	Add Component	





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

シーンの構成

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





シーンの構成

Canvas

C unity

'≔ Hierarchy 🗙 🔒 📼	Inspector	r 🔒 📲
Create * (Q*All	Ca	anvas 📃 Static 🔻 🎽
SampleScene 📲 🔺	Tao Ur	Intagged + Laver UI +
Main Camera		
data_visualizer	Rect	t Transform
water	Some values	driven by Canvas.
wall		Pos X Pos Y Pos Z
wall (1)		302 213.5 0
wall (2)		Width Height
Wall (5)		800 565.5629 🔛 R
EventSystem	Anchors	
lohn M	Min	X 0 Y 0
William M	Max	X 0 Y 0
lames M	Pivot	X 0.5 Y 0.5
George M	Deterior.	
Charles M	Rotation	
Frank_M	Scale	X 0.755 Y 0.755 Z 0.755
Joseph_M	🖉 🔳 🗹 Canv	ivas 🔟 🗐 🗐 👘
Henry_M	Render Mo	ode Screen Space - Overlay +
🕥 Robert_M	Pixel Per	erfect 🗌
🗇 Thomas_M	Sort Ord	der 0
Edward_M	Target D	Display Display 1 +
Harry_M	Additional	Shader Cha Mixed +
Walter_M	Shade	ler chappels Normal and Tancent are most
Arthur_M	A often	n used with lighting, which an Overlay canvas
Fred_M	does neede	a not support. Its likely these channels are not led
Albert_M		,
Samuel_M	🖉 🐨 🔛 🗹 Canv	ivas Scaler (Script) 🛛 🔯 🗐 🗐 🗐
Clarence_M	UI Scale Mo	ode Scale With Screen Size +
Louis_M David_M	Reference F	Resolution
	X 800	Y 600
Charlie M	Screen Mat	tch Mode Match Width Or Height +
Richard M	Match	
Ernest M	Materi	Width Height
Roy M	Reference F	Pixels Per U 100
Will_M		- hie Devee teer (Caultat)
Andrew_M	Grap	phic Raycaster (Script)
×	Script	GraphicRaycaster 🛛 🔍 🤜

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

シーンの構成

EventSystem

🗄 🗃 🖬 🖬 🖬	=	Inspector			2	
Create * Q*All	\sum	EventSy	stem		Static	-
▼ 🚭 SampleScene 🛛 -=	*		4	t Laver Default		•
Main Camera		Tag Cintagget		• Layer Delaut		-
🔰 data_visualizer		Transforn	n		I	\$,
🔰 water		Position	X 0	Y 0 Z	0	
vall 🔰 🗤		Rotation	X 0	Y 0 Z	0	
🔰 wall (1)		Scale	X 1	Y 1 Z	1	
wall (2)		The sys	tem (Script)		φ.
wall (3)		Script	tem (E EventSystem		0
Canvas		First Solocted		None (Came Object)		0
EventSystem		First Selected	F		·	<u> </u>
John_M		Send Navigation	Event			_
William_M		Drag Threshold		10		
James_M		🔻 🚣 🗹 Standalor	ie Inp	ut Module (Scrip	1	\$.,
George_M		Script		LandaloneInput M	1odule	\odot
Charles_M		Horizontal Axis		Horizontal		
Flank_M		Vertical Axis		Vertical		
Henry M		Submit Button		Submit		-
Robert M		Cancel Button		Cancel		-1
Thomas_M		Input Actions Pe	r Seco	10		-
Edward_M		Repeat Delay		0.5		-
Harry_M		Force Module Ac	tive			
🕥 Walter_M				<u> </u>		-
🕥 Arthur_M		🔻 🕼 🗹 Base Inpu	t (Scr	ript)	10 1	\$,
🜍 Fred_M		Script		BaseInput		\odot
Albert_M						
Samuel_M			Add C	Lomponent		
Clarence M						



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

シーンの構成

John_M

🚭 unity

🔚 Hierarchy 🗙 🔒 📲	Inspector	🗎 📲
Create * Q*All	🔽 🔽 lohn M	Static 🔻 🔺
\blacksquare SampleScene $\neg \equiv \blacksquare$		
🧊 Main Camera 🦳	Tag untaggeo	- Layer Default
🧊 data_visualizer	🔻 🦶 🛛 Transform	🛄 🎞 🏝
🤍 water	Position X -	3.705 Y 0.4647 Z 0
🧊 wall	Rotation X C) Y O Z O
🧊 wall (1)	Scale X 1	3.284 Y 13.284 Z 1
🧊 wall (2)		
🧊 wall (3)	Sprite Kender	
🕨 🧊 Canvas	Spine	Picificierza 0
EventSystem	Color	
john_M	Flip	
🤍 William_M	Material	Sprites-Default O
James_M	Draw Mode	Simple +
George_M	Continer Lawer	Default
Charles_M	Sorting Layer	Derault
Frank_M	Order in Layer	0
Joseph_M	Mask Interaction	None +
Henry_M	Sprite Sort Point	Center :
Robert_M	🔻 🖲 🗹 Circle Collide	r 2D 🔝 🗟 🕸
Thomas_M		
Edward_M		Lait Collider
Harry_M	Material	None (Physics Material 💈 💿
Walter_M	ls Trigger	
Arthur_M	Used By Effector	
Fred_M	Offset	
Albert_M	x 0	Y O
Samuel_M	Radius	0.1
Clarence_M	Naurus	0.1
David M		
	🔻 🔶 🛛 Rigidbody 2D) 🗊 🗟 🔅
JOE_M Charlie M	Body Type	Dynamic +
Richard M	Material	None (Physics Material 2 💿
Ernost M	Simulated	
Roy M	Use Auto Mass	
Will M	Mass	0.04
Andrew M	Linear Dreg	0
	Linear Drag	•

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。



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



elem プレハブのコンポーネント群

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

Inspector	→ =
Scale	X 1.6020 Y 1.6020 Z 1
🔻 💽 🗹 Sprite Re	nderer
Sprite	⊡circle128 O
Color	
Flip	
Material	Sprites-Default
Draw Mode	Simple 🕈
Sorting Layer	Default \$
Order in Layer	0
Mask Interactior	None 🕈
Sprite Sort Point	Center +
	llider 2D

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

1 Inspector		= 円形の
Mask Interaction	None \$	
Sprite Sort Point	Center \$	壁や浮
Circle Collider	2D 🚺 📑 🛱,	
	Latit Collider	
Material	None (Physics Material 2 💿	
ls Trigger		
Used By Effector		
Offset		
X 0	Y 0	
Radius	0.1	
▶ Info		
Rigidbody 2D	[] ;! \$,	
Body Type	Dynamic \$	
Material	None (Physics Material 2 0	

D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。

孚力に反応する計算に使用される。

Rigidbody 2D

Inspector	
▶ Into	
Rigidbody 2D	in the second
Body Type	Dynamic +
Material	None (Physics Material 2 0
Simulated	
Use Auto Mass	
Mass	0.04
Linear Drag	8
Angular Drag	0.05
Gravity Scale	1
Collision Detection	Discrete +
Sleeping Mode	Start Awake +
Interpolate	None +
Constraints	
Info	en se
Elomont (Scrin	+) – – 1 (4

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)

Inspector		
Material	None (Physics Material 2 💿 🔎	
Simulated		
Use Auto Mass		
Mass	0.04	
Linear Drag	8	
Angular Drag	0.05	
Gravity Scale	1	
Collision Detection	Discrete 🗘	
Sleeping Mode	Start Awake 🗘	
Interpolate	None 🗘	
Constraints		
▶ Info		
🔻 🛃 Element (Script		
Script	G# Element ⊙	
Carrow a straight for the star star and the star and the star and the star and the star star and the star and		

nity 🖓

Element.cs を保持することを示す。

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。

MonoBehaviourを継承

オブジェクトに追加するにはMonoBehaviour である必要がある。

public TextMeshProUGUI m_TextTime; // インスペクタから current_time オブジェクトを設定

🗄 Hierarchy 📄 🚽	Inspector	<u></u>	=	
Create * Q*All SampleScene *=	Tag Untagged	er 🗌 Statio	-	
Main Camera data_visualizer	Transform		\$,	
wall wall (1)	Rotation X 0 Scale X 1	Y 0 Z 0 Y 1 Z 1	=	
wall (2) wall (3)	▼ 🗗 🗹 Data Visualize Script	r (Script)	æ	
EventSystem	Elem Prefab	Velem	0	
	Canvas	Canvas (Canvas)		
	Text Time	Tcurrent_time (Text	0	
	Add Component			

オブジェクトを参照

8 public class DataVisualizer : MonoBehaviour 9 { 10 11 12 13

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。

public GameObject m_ElemPrefab; // インスペクタから elem プレハブを設定 public GameObject m_NamePrefab; // インスペクタから the name プレハブを設定 public Canvas m_Canvas; // インスペクタから Canvas オブジェクトを設定 public TextMeshProUGUI m_TextTime; // インスペクタから current_time オブジェクトを設定

🔚 Hierarchy 📄 📲	O Inspector	≡
Create ▼ Q▼All SampleScene = Main Camera	✓ ✓ data_visualizer □ Static ■ Tag Untagged + Layer Default +	
<pre>data_visualizer water wall wall (1) wall (2) wall (3) Canvas EventSystem</pre>	TransformImage: Constraint of the second secon	*,
	C# Data Visualizer (Script) Script DataVisualizer Elem Prefab elem Name Prefab the_name Canvas Canvas (Canvas)	
	Text Time Current_time (Text) Add Component	and the second of the second sec

特殊関数:Awake

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

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。

```
1 using System.Collections.Generic;
 2 using UnityEngine.Assertions;
 3 using UnityEngine;
 5 namespace VIS {
 6
 7 // CSVファイルの行に対応
 8 public class InfoUnit
 9 {
                                // 名前
10
      public string m_Name;
      public int m_Value;
                               // 数値
11
                                // 0:Male, 1:Female
      public int m_Type;
12
13 }
14
15 // CSVファイルひとつに対応
16 public class DataUnit
17 {
                                 // 年代
      public int m_Time;
18
      public List<InfoUnit> m_InfoUnitList;
19
20
      public Dictionary<string, InfoUnit> m_InfoUnitDictionary; // 検索用
21 }
22
23 // 全てのデータを保持
24 public class Data
25 {
26
      public List<DataUnit> m_DataUnitList;
      public HashSet<string> m_NameSet; // 重複のない名前の集合
27
      public float m_MaxValue;
                               // 正規化のため全体の最大値を算出しておく
28
29
30
      // 年代を列挙
      public int[] GetKeyTimeList()
31
32
33
          var res = new int[m_DataUnitList.Count];
          for (var i = 0; i < m_DataUnitList.Count; ++i) {</pre>
34
35
36
              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
      float m_CurrentScale = 0f; // 現在のスケール
10
      float m_TargetScale = 0f; // ターゲットスケール
11
      GameObject m_NameObject; // 名前表示用のオブジェクト
12
13
      // 名前設定
14
      public void SetName(string name) { m_Name = name; }
15
16
      // 名前オブジェクト設定
      public void SetNameObject(GameObject name_object) { m_Nam
17
18
      // 正規化用の拡大率を与えてターゲットスケールを更新
19
      public void SetTargetData(DataUnit data_unit, float ratio)
20
21
22
          if (data_unit.m_InfoUnitDictionary.ContainsKey(m_Name)
             var info_unit = data_unit.m_InfoUnitDictionary[m_
23
             m_TargetScale = ((float)info_unit.m_Value)*ratio;
24
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
      {
          if (m_Name == null) // 念の為nullアクセス回避
33
34
             return;
          // 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
35
36
          m_CurrentScale = Mathf.Lerp(m_CurrentScale, m_TargetS
          // フケニルを設定
27
```


特殊関数:Update

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


Unityにより毎フレーム(1/60秒周期などで)呼ばれる。

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)

改造してみよう

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。

			2	-=
		Sta	atic	Ŧ
r Def	faul	lt		+
	Z Z Z	0 0 1	;; 	*,
			7:	\$,
g				+

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。

改造してみよう

ボールの重力を無効にするには

'≡ Hierarchy 🗙 🔒 -=	Inspector		∂ •≡
Create * (Q*All	🔻 🔶 🛛 Rigidbody 2D	に に 回 二	*,
< 🜍 elem	Body Type	Dynamic	\$
👕 elem	Material	None (Physics Mate	0
	Simulated		
	Use Auto Mass		
	Mass	0.1	
	Linear Drag	8	
	Angular Drag 🍡	.	
	Gravity Scale	1	
	Collision Detection	Jiscrete	•
	Sleeping Mode	Start Awake	÷
	Interpolate	None	•
	Constraints		
	▶ Info		
	🔻 🖬 🗹 Element (Script	t) 🔝 🖬	φ,
	Script	Element	0
	Sprites-Defau	ult 🔯	¢, 🗸

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。

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

