

2018/6/18(月)

Unityとは…?

以下Wikipediaから(教授ブチギレ~) Unityとは、統合開発環境を内蔵し、複数のプラットフォームに対応する ゲームエンジンである。

iOS, Android, PlayStation 3, PlayStation 4, PS Vita, Xbox 360, Xbox One, Wii U そしてVR/AR向けの開発に対応している

要するに...

ゲーム制作における難しい部分を既にやっておいてくれて
 ユーザーは最低限の技術でゲーム制作を始められる!!(いいのかこれで?)

どうやってゲームを作っていくのか



つまり!

ゲームというのは現実のように「歩いているから進んでいる」ではなく

「<u>キャラクターの歩きモーション</u>」と「<u>オブジェクトを動かすプログラム</u>」を 組み合わせることで「そういう風に錯覚させている」のである!

※あくまで自分が作ってきた時に行きついた考え

プロジェクト作成

まずプロジェクトから作らないと何も始まらない!

1. Unityを起動

- 2. 右上にある「NEW」を押す
- 3. プロジェクト名を決める (今回はブロック崩し)
- 4. 保存先は各自におまかせ
- 5. 「3D」にチェックが入っているか
- 6. 上記ができたら右下 「create project」をクリック



Unityの 画面

プロジェクトを作り終わるとこんな画面が出てくる 英語ばっかりでわけわからね







Hierarchy

そのシーン内にあるオブジェクトを表記 ※シーンに関しては後で説明 初期状態だと「Main Camera」(カメラ)と 「Directional Light」(太陽?)が 入っている Image: Second secon

Console

エラー、警告など Unity によって生成されたメッセージを表示する。 <u>Debug.Log, Debug.LogWarning</u> や <u>Debug.LogError</u> 関数を使用して、 コンソールでメッセージを表示することもできます。

ゲームには表示されない(開発者のみ見れる)。

プログラムが思ったように動かない時はこれを活用してプログラムの動きを追おう (超重要)

E Con	isole			*=
Clear	Collapse	Clear on Play	Error Pause	①2 △0 ❶0
	ove iityEngine.	.Debug:Log(O	bject)	(429)
	ove iityEngine.	Debug:Log(O	bject)	453

Project

プロジェクトの保存先にある「Assets」フォルダの中身を表示する画面 スクリプト(プログラム)データはここに保存される 他にも様々なデータを保存する場所

右の画像だと「New Material」(マテリアル)と 「New Behabiour」(スクリプト)が入ってる

データフォルダって考えるのが楽かな?



Game

現在作っているゲームを実際にプレイした時に映る画面 初期状態(左下)だと何もないので空しか映らないが オブジェクトを追加するとこんな感じ(右下)



Scene

ゲームオブジェクトを配置する画面 ワールド内を自由に見渡せる Gameとの違い:

パワポで例えると 作ってるときの画面が「Scene」で スライドショーにしてる時が「Game」



Inspector

選択したゲームオブジェクトのステータスを表示する画面 オブジェクトの「座標」、「向き」、「大きさ」はここから設定する



🖲 Inspector 🛛 🔀 Navigati	on Services 🔒 📲
🍟 🗹 Main Camera	🗌 Static 🔻
Tag MainCamera	+ Layer Default +
🙏 Transform	🔯 🌣,
Position	X 0 Y 1 Z -10
Rotation	X 0 Y 0 Z 0
Scale	X 1 Y 1 Z 1
📹 🗹 Camera	
Clear Flags	Skybox +
Background	<i>H</i>
Culling Mask	Everything \$
Projection	Perspective +
Field of View	
Clipping Planes	Near 0.3
	Far 1000
Viewport Rect	ZAH
	ていて
Depth	-1
Rendering Path	mponent)
Target Texture	Note (Render Yexture) 0
Occlusion Culling	
Allow HDR	
Allow MSAA	
Target Display	Display 1 +
🖵 🗹 GUI Layer	🔯 🌣
👕 🗹 Flare Layer	🔊 🌣
💿 🗹 Audio Listener	n 🔅
	Add Component

座標

向き

大きさ

オブジェクト生成①

 実際にオブジェクトを作ってみよう 上のメニューバーから GameObject>3D Object>Cube を選択

66			
nal (64bit) - Untitled - ブロック崩し - P	PC, Mac & Linux S	Standalone* <dx11></dx11>
Gam	eObject Component Windo	w Help	
	Create Empty	Ctrl+Shift+N	
	Create Empty Child	Alt+Shift+N	Game 🛱 Accet 9
	3D Object	>	Cube
	2D Object	>	Sphere
	Light	>	Capsule
	Audio	>	Cylinder
	Video	>	Plane
	UI	>	Quad
	Particle System		Ragdoll
	Camera		Terrain
	Center On Children		Tree
	Make Parent		Wind Zone
	Clear Parent		3D Text
	Apply Changes To Prefab		50 101
	Break Prefab Instance		
	Set as first sibling	Ctrl+=	
	Set as last sibling	Ctrl+-	
	Move To View	Ctrl+Alt+F	
	Align With View	Ctrl+Shift+F	
	Align View to Selected		
	Toggle Active State	Alt+Shift+A	

オブジェクト生成2

なんか出た!!



オブジェクトを動かす

オブジェクトの動かし方(初期位置の決め方)は何種類かある

- 1. 左上のアイコンをクリックして操作
- 2. Inspectorから座標入力
- 3. Inspectorから(part2)



(b)

S

10 M

0

オブジェクトを動かす



Unityの左上のメニューバー下にこんなのがあるはず



 ・・・このアイコンをクリックすると選択したオブジェクトを動かせるようになる (Wキーでショートカット)

矢印をドラッグして 動かせる (選ばなかったら軸に関係なく自由に 動かせる)



その他機能



ついでにその他のアイコンの機能も紹介(ショートカットキー)

 ${\mathbb C}$





・・・選択したオブジェクトを回転させる(Eキー)



・・・選択したオブジェクトの大きさを変える(Rキー)



・・・上と同じで大きさを変えるが少しやり方が違う(Tキー)

オブジェクトを動かす

Inspectorから座標入力

座標を入力してその位置に移動させる 細かい調整や、計算して設置するときはこっち

Inspector	🔀 Navigation	Services				
👕 🗹 Cube						🔲 Static 🔻
📕 Tag Untag	ged	ŧ Lay	er	Default		+
▼, Transfo	rm					i *,
Position	Х	-0.2396889	Y	0.9687672	Ζ	-0.6129658
Rotation	Х	0	Υ	0	Ζ	0
Scale	Х	1	Y [1	Ζ	1

オブジェクトを動かす(おまけ)

実は「X」、「Y」、「Z」の文字をクリックすると 🕂 みたいに操作できる



ゲームの実行



ゲームの実行はUnityの画面真ん中上にある再生ボタンを押すとゲームが開始され Gameタブでプレイすることができる



今回のテーマ

テーマ

初心者向け講座ということで今回は **ブロック崩し** を作っていきます



「は?アーマードコアみたいなゲームじゃねーのかよ」と思ったあなた! <u>ブロック崩しにはUnityの基礎がたくさん詰ま</u>ってます!



ブロック崩しを改造すると「こがねこのエアライド」になるんです!



ARMORED COMBAT



こがねこのエアライドを改造すると「ARMOREDCOMBAT」になるんです!

ステージ作成

の前に...

カメラの位置を設定しよう これが基準になっていくので後から変えると全部変えることになる Hierarchyの「Main Camera」をクリック(選択)し、Inspectorを下のようにする

O Inspector	🗙 Navigation	Services		<u></u> +≡			
👕 🗹 Main Ca	mera			🗌 Static 🔻			
Tag MainCan	nera		tayer Default	+)		
▼→ Transform	▼ 👃 Transform 🗊 🌣						
Position		X 0	Y 0	Z -10			
Rotation		X 0	Y O	Z 0			
Scale		X 1	Y 1	Z 1			







先ほどと同じように GameObject>3D Object>Cube でCubeオブジェクトを4つ作る それぞれ名前を 「wall_top」,「wall_bottom」,「wall_left」,「wall_right」 にする

🛛 Inspector 🛛 🔀 Naviga	tion Services	<u> </u>
👕 🗹 wall_top		tatic 🖌
Tag Untagged	‡ Layer Default	+
▼人 Transform		🔯 🌣,
Position	X -0.6854014 Y 1.172457 Z -0.09	343171
Rotation	X 0 Y 0 Z 0	
Scale	X 1 Y 1 Z 1	
🔻 🧾 🛛 Cube (Mesh Filter)	💽 🌣,
Mesh	III Cube	•
🔻 🤪 🗹 Вох Collider		🔯 🌣,
	🚡 Edit Collider	
Is Trigger		
Material	None (Physic Material)	0
Center	X 0 Y 0 Z 0	_
Size	X 1 Y 1 Z 1	
V 🔜 🗹 Mesh Renderer		💿 🌣,
▶ Lighting		
▶ Materials		
		•
Derault-Material		1
▶ Shader Standard		*
	Add Company and	
	Add Component	

ここで名前を変更できる	
または オブジェクトを選択して 「F2キー」でも行ける	

≔ Hierarchy	<u></u>
Create * (Q*All	
🖉 🛇 Untitled*	*≡
Main Camera	
Directional Light	
wall_top	
wall_bottom	
wall_left	
wall_right	

<u>制作を楽にするワンポイント</u>: オブジェクトを選択して 「Ctrl+c」でコピー 「Ctrl+v」で貼り付け

各オブジェクトの座標、大きさを設定する

wall_top

0 Inspecto	or	🔀 Navigatio	in	Services	;			<u></u>
🧊 🗹 🛛	vall_to	ор						🔲 Static 🔻
Tag 🗌	Untago	ged		🕴 Laye	er	Default		+
💌 🙏 🛛 Tra	nsfo	rm						
Position		×	0	١	r [4	Ζ	0
Rotation		×	0	١	٢[0	Ζ	0
Scale		X	18	1	٢Ī	0.1	Ζ	1

wall_bottom

6	Inspector	🔀 Navigation	n Services		∂ -≡
1	🍞 🗹 🛛 🖌	ottom			🔲 Static 🔻
1	Tag Untag	ged	‡ Layer	Default	+
▼	▼人 Transform				
	Position	X	0 Y -	-5 Z	0
	Rotation	X	0 Y (0 Z	0
	Scale	X	18 Y (0.1 Z	1

wall_left

Inspector	🔀 Navigation	Services			<u> -</u> ≡
👕 🗹 wall_l	eft				🗌 Static 🔻
Tag Untag	ged	+ Layer	Default		+
▼人 Transfo	rm				[] \$,
Position	X 9	Y	-0.5	Ζ	0
Rotation	X 0	Y	0] Z	0
Scale	X 0.	1 Y	9	Z	1

wall_ri	ght				
Inspector	🔀 Navigation	Services			<u></u> =
👕 🗹 📷	_right				🗌 🗌 Static 🔻
Tag Unta	gged	‡ Layer	Default		+
▼ <u>↓</u> Transf	orm				[] \$,
Position	X -9	Y	-0.5	Ζ	0
Rotation	X 0	Y	0	Z	0
Scale	X 0.1	Y	9	Z	1

こんな感じの壁ができたら完成

🕻 Game		節 Asset Store					
Display 1	\$ F	Free Aspect	‡ Scale 🔾	1×	Maximize On Play	Mute Audio Stats	Gizmos 🔻
							000000
							1000000
							200000
							0000003
							1000000
							0000005
							200000
							CORRECT OF
							9000000
							000000
							0000000
							1000000
							2000000
							and a second second

ボールの作成



球体は GameObject>3D Object>Sphere で出てくる 名前は「ball」、座標と大きさは下のようにする

🛛 Inspector 🛛 🔀 Navigation	Services	a .=
👕 🗹 ball		🗌 Static 🔻
Tag Untagged	‡ Layer Default	*
▼人 Transform		🔯 🌣,
Position	X 0 Y -1	Z 0
Rotation	X 0 Y 0	Z 0
Scale	X 0.5 Y 0.5	Z 0.5

球体は GameObject>3D Object>Sphere で出てくる 名前は「ball」、座標と大きさは下のようにする

🛛 Inspector 🛛 🔀 Na	vigation Services			∂ =
🁕 🗹 ball				🗌 🗌 Static 🔻
Tag Untagged		‡ Layer Default		\$
▼人 Transform				🔯 🌣,
Position	X 0	Y -1	Z	0
Rotation	X 0	Y 0	Z	0
Scale	X 0.5	Y 0.5	Z	0.5



この状態だと宙に浮くだけで動かないので物理法則を与えます(すごい(小並感))

ballを選択してInspector内の「Add Component」を クリック>Physics>Rigidbodyを選択

O Inspector 🔀 Navig	ation S	ervices				â -=
🍞 🗹 ball						Static 🔻
Tag Untagged		,	Layer	Default		•
▼人 Transform						2
Position		X 0		Y -1	Z 0	
Rotation		X 0		Y 0	Z 0	
Scale		X 0.5		Y 0.5	Z 0.5	
🔻 🧾 🛛 Sphere (Mesh Fi	lter)					2
Mesh		📗 Sphere				G
🔻 ڬ 🗹 Sphere Collider						2
		🔥 Edi	t Collider			
Is Trigger						
Material		None (Phy	sic Mater	rial)		c
Center		X 0		Y 0	Z 0	
Radius		0.5				
▼ 🛃 🗹 Mesh Renderer ▶ Lighting ▶ Materials						1
Default-Material						0
▶ J Shader Standard						
		Add Com	ponent			
	Q					
		Compo	nent			
	Mesh			> A		
	Effects					
	Physic					
	Physic	s 2D		•		
	Naviga	tion		- F		
	Audio			•		
	Video			•		
	Rende	rina				
	Layout					
	Miscell	aneous		•		
	Analyt	ice.				
	Sorieta					
	Event					





ちなみに…



するとInspector内に「Rigidbody」が追加されます

🛛 Inspector 🛛 🔀 Navigation	Services		≜ •≡
👕 🗹 ball		Sta	atic 🔻
Tag Untagged	+ Layer Default		;
▼ 🙏 Transform			(
Position	X 0 Y -1	Z 0	
Rotation	X 0 Y 0	Z 0	
Scale	X 0.5 Y 0.5	Z 0.5	
🔻 🧾 Sphere (Mesh Filter)			💽 🌣,
Mesh	🏭 Sphere		0
🔻 😂 🗹 Sphere Collider			💽 \$,
	🔥 Edit Collider		
Is Trigger			
Material	None (Physic Material)		0
Center	X 0 Y 0	Z 0	_
Radius	0.5		
🔻 🛃 🗹 Mesh Renderer			(
▶ Lighting			
▶ Materials			
🔻 🙏 Rigidbody			💽 \$,
Mass	1		
Drag	0		
Angular Drag	0.05		
Use Gravity	\checkmark		
Is Kinematic			
Interpolate	None		+
Collision Detection	Discrete		+
▶ Constraints			

クリック

「Use Gravity」のチェックを外し 「Freeze Position」の「Z」 「Freeze Rotation」の「X」「Y」「Z」 にチェックを入れる

🔻 🙏 Rigidbody	🔯 🗢,
Mass	1
Drag	0
Angular Drag	0.05
Use Gravity	
Is Kinematic	
Interpolate	None +
Collision Detection	Discrete +
▼ Constraints	
Freeze Position	🗆 X 🗌 Y 🗹 Z
Freeze Rotation	✔X ✔Y ✔Z

🔻 🙏 🛛 Rigidbody		🔯 🌣,
Mass	1	
Drag	0	
Angular Drag	0.05	
Use Gravity		
Is Kinematic		
Interpolate	None	;
Collision Detection	Discrete	\$
🔻 Constraints		
Freeze Position	🗆 X 🗆 Y 🗹 Z	
Freeze Rotation	🗹 X 🗹 Y 🗹 Z	

Mass・・・質量 Drag・・・空気抵抗 Use Gravity・・・重力の有無 Constrains・・・それぞれposition, rotationを固定するかしないか

などなどいろいろあるが詳しくは下記のサイトに説明が載ってる <u>https://docs.unity3d.com/ja/current/Manual/class-Rigidbody.html</u>

今度はボールに「摩擦力」や「反発力」といった物理演算を 追加します(そんなこともできるのかっ!)!

🚭 Unity	5.6.0f3 Pe	rsonal (64bit) - U	ntitled - ブロック	7崩し - PC, N	Mac & Linux Standalone* <dx11></dx11>
File Ed	it Assets	GameObject	Component	Window	Help
۳.	+ (Create		>	Folder
'≔ Hiera Create ' ▼	ircl (r (tit (ecti (Show in Explorer Open Delete Open Scene Addi	tive		C# Script Javascript Shader Testing
wall wall wall wall	_t« _b _l« _ri	mport New Asset mport Package Export Package	t	>	Scene Prefab
Dali	5	Find References In Select Dependence	n Scene c ies		Audio Mixer Material Lens Flare
		Refresh Reimport	(Ctrl+R	Render Texture Lightmap Parameters
	- F	Reimport All			Sprites
	(Run API Updater Open C# Project			Animator Controller Animation
					Animator Override Controller Avatar Mask
Econo	ala d				Physic Material
Clear	OIE Collapse	Clear on Play E	rror Pause	0	Physics Material 2D
					GUI Skin Custom Font
					legacy

左上のメニューバーから Assets>Create>Physic Material 名前を「pm_ball」に



pm_ballを選択するとInspectorにこんなのが 表示される

Inspector	🔀 Navigation	Services	<u> </u>
🔎 pm_ball			💽 🗢,
			Open
Dynamic Fricti	on	0.6	
Static Friction		0.6	
Bounciness		0	
Friction Combi	ne	Average	+
Bounce Combi	ine	Average	+

> Dynamic Friction → 動摩擦力
> Static Friction → 静止摩擦力
> Bounciness → 弾性力
> Friction Combine → 衝突時の物体間の摩擦力
> Bounce Combine → 衝突時の物体間の弾性力

こんな感じに値を変更

🛛 Inspector 🛛 🔀 Navigation	Services	a ,≖≡
pm_ball		🔯 🌣,
		Open
Dynamic Friction	0	
Static Friction	0	
Bounciness	1	
Friction Combine	Minimum	+
Bounce Combine	Maximum	+

Assets内の「pm_ball」をHierarchy内の「ball」に ドラッグ&ドロップ



いよいよ玉を動かすプログラムを書いてみよう

メニューバーから Assets>Create>C# Script を選択

または Project内で 右クリック>Create>C# Script

名前を「Ball」にする

※先ほどのpm_ballのようにBallにドロップ&ドロップ

📲 Animator

_#

Ball

New Mater

demoworld

Assets

4 💊 ★

-

Project

🛚 🏫 Favorites 🛛

Assets

All Materia All Models All Scripts

All Prefat

Create

ballスクリプトをクリックすると Visual Studioが立ち上がる(はず)



こんな感じにコードを追加

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

pub<u>lic class ball MonoBehaviour (</u>

Rigidbody rb; // Use this for initialization void Start()

> rb = this.GetComponent<Rigidbody>(); float force_x = Random.Range(-10, 10); Vector3 force = new Vector3(force_x, 10, 0); rb.velocity = force;

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

実際に動かしてみよう!

先ほど教えた再生ボタンを押してゲームを開始してみよう!



ボールが勝手に動き出して 壁に当たって跳ね返っている

バーの作成





バーはカプセルタイプのオブジェクトを変形させて使う カプセルは GameObject>3D Object>Capsule で出てくる 名前は「bar」、座標と大きさは下のようにする

O Inspector	🔀 Navigation	Services					<u> </u>
🁕 🗹 bar							🗌 Static 🔻
Tag Untagge	d		🕴 Layer 🚺	Default			+
▼人 Transform	n						🔯 🌣,
Position		X 0	Y	-4.5	Z	0	
Rotation		X 0	Y	0	Z	90	
Scale		X 0.5	Y	1	Z	0.5	



バーを動かすプログラム

ボールの時と同じように C# Scriptを作る 名前を「Bar」にする

同じくBarスクリプトをbarオブジェク トにドロップ&ドロップする using System.Collections; using System.Collections.Generic; using UnityEngine;

コードを追加

public class Ball : MonoBehaviour {

// Use this for initialization
void Start()

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

if (Input.GetKey(KeyCode.D) || Input.GetKey(KeyCode.RightArrow))

transform.position += new Vector3(10, 0, 0) * Time.deltaTime;

else if (Input.GetKey(KeyCode.A) || Input.GetKey(KeyCode.LeftArrow))

transform.position += new Vector3(-10, 0, 0) * Time.deltaTime;

ちゃんと動作するか実際にプレイして確認 バーがちゃんと移動してればOK!

= 1×

Maximize On Play Mute Audio Stats Gizmos

C Game

Display 1 + Free Aspect

Asset Store

+ Scale O

このままだと壁を貫通してしまうのでまずい



壁を貫通しないための関数(limit_area)を追加①

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

```
public class Bar : MonoBehaviour {
```

```
float[] limit_area = new float[2];
float bar_width;
void limit_area_over()
```

```
Vector3 pos = transform.position;
if (this.transform.position.x - bar_width < limit_area[0])</pre>
```

```
pos.x = limit_area[0] + bar_width;
transform.position = pos;
```

```
}
else if (this.transform.position.x + bar_width > limit_area[1])
{
    pos.x = limit_area[1] - bar_width;
    transform.position = pos;
}
```



壁を貫通しないための関数(limit_area)を追加2)

// Use this for initialization
 void Start()

```
limit_area[0] = GameObject.Find("wall_left").transform.position.x;
limit_area[1] = GameObject.Find("wall_right").transform.position.x;
```

bar_width = this.gameObject.transform.localScale.y;

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

```
if (Input. GetKey(KeyCode. D) || Input. GetKey(KeyCode. RightArrow))
```

transform.position += new Vector3(10, 0, 0) * Time.deltaTime;

```
else if (Input. GetKey(KeyCode. A) || Input. GetKey(KeyCode. LeftArrow))
```

```
transform.position += new Vector3(-10, 0, 0) * Time.deltaTime;
```

limit_area_over();



これで壁を貫通する問題は解決!



Barと壁の色が同じで見づらいので

次はバーの色を変えていきます Physic MaterialやC#Scriptを作った時と同じように Assets>Create>Material (右クリックも同様) 名前を「Bar_Material」にしてbarオブジェクトにドラッグ&ドロップ



Bar_Materialを選択し、Inspector内のMainMapsの下にある「Albedo」を選択 皆さんのお好みの色に設定してください





色が変わった!

C Game	🛱 Asset Store		_			* ≡
Display 1	Free Aspect	≑∣ Scale ⊖	1×	Maximize On Play	Mute Audio Stats	Gizmos *

ブロックの作成



Cubeオブジェクトを作り、座標と大きさを下のようにする 名前は「block_1」にする 色は先ほどバーとやったようにMaterialを作り、好きな色を付けてください

🟮 Inspector 🛛 🔀 Navigation	Services	∂ •≡
🍞 🗹 block_1		🗌 Static 🔻
Tag Untagged	‡ Layer Default	;
▼ 人 Transform		🛐 🌣,
Position	X -8 Y 3	Z 0
Rotation	X 0 Y 0	Z 0
Scale	X 1.5 Y 0.35	Z 1

今の工程をblock_1とは別で 「block_2」,「block_3」という名前で新しく作ってください (Materialも「Block_2_Material」「Block_3_Material」という名前で好きな色で作って ください)

Services	<u> </u> ,=
	🗌 Static 🔻
‡ Layer Default	+
	🔃 🌣,
X -8 Y 3	Z 0
X 0 Y 0	Z 0
X 1.5 Y 0.35	Z 1
	🔯 🚓
🏭 Cube	0
	🔯 🌣,
🚡 Edit Collider	
None (Physic Material)	0
X 0 Y 0	Z 0
X 1 Y 1	Z 1
	🔯 🌣,
	1000
	🛐 \$,
	•
	Image: Services Image: Image

🛛 Inspector 🛛 🔀 Navigation	Services			<u> </u>
👕 🗹 block_3				Static 🔻
Tag Untagged	ŧ La	ayer Default		;
🔻 🙏 Transform				🛐 🌣,
Position	X -8	Y 3	Z 0	
Rotation	X 0	Y 0	Z 0	
Scale	X 1.5	Y 0.35	Z 1	
🔻 🧾 Cube (Mesh Filter)				🔯 🌣,
Mesh	🏢 Cube			0
🔻 🤪 🗹 Box Collider				[🖉 🔅,
	🔥 Edit Co	llider		
Is Trigger				
Material	None (Physic	Material)		0
Center	X 0	Y 0	Z 0	
Size	X 1	Y 1	Z 1	
🔻 🛃 🗹 Mesh Renderer				🔯 🌣,
▶ Lighting				
Materials				
Block_3_Material				🔯 🔅
▶ Shader Standard				•

Prefab化

Hierarchyにあるblock_1,block_2,block_3をProject内にドラッグ&ドロップ これをPrefab化(プレハブ化)という ロドナブジェクトなナ島に生ました。使る時は便利なので建物的に使っている

同じオブジェクトを大量に生成したり、使う時は便利なので積極的に使っていきま しょう

(これを使いこなせるかで作業効率がかなり変わります)



Prefab化した後は元になったオブジェクトは破棄しましょう

ⁱ ≡ Hierarchy	≟ .=
Create * Q*All	
🔻 🚭 Untitled*	* ≡
Main Camera	
Directional Light	
wall_top	
wall_bottom	Constant of the local division of the local
wall_left	COLUMN A
wall_right	A DESCRIPTION OF A DESC
ball	
bar	
block_1	
block_2	
block_3	

'≔ Hierar	chy	<u> </u>
Create *	Q*All	
🔻 🚭 Unti	itled*	*≡
Main	Camera	
Direc	ctional Light	
wall_	_top	
wall_	_bottom	
wall_	_left	
wall_	_right	
ball		
bar		

プログラムからオブジェクトを生成

手動でブロックを配置するのは面倒なのでプログラムから オブジェクトを生成→設置しましょう 「Create_Blocks」という名前のC#Scriptを作ってください

コードを追加①

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

public class Create_Blocks : MonoBehaviour {

public GameObject[] prefabs; GameObject block; float block_width; float block_height; int create_count_x = 8; int create_count_y = 3; float[] create_range_x = { -9, 9 }; float distance_x; float[] create_range_y = { 3, 0 }; float distance_y; // Use this for initialization

コードを追加2

```
// Use this for initialization
```

```
void Start () {
    block_width = prefabs[0].transform.localScale.x;
    block_height = prefabs[0].transform.localScale.y;
    distance_x = (Mathf.Abs(create_range_x[0] - create_range_x[1]) - create_count_x * block_width) / create_count_x;
    distance_y = (Mathf.Abs(create_range_y[0] - create_range_y[1]) - create_count_y * block_height) / create_count_y;
```

```
int prefabs_number = 0;
Vector3 pos = prefabs[prefabs_number].transform.position;
for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 8; j++) {
        block = GameObject.Instantiate(prefabs[prefabs_number],pos,Quaternion.identity);
        pos.x += block_width + distance_x;
    }
    pos.x = prefabs[prefabs_number].transform.position.x;
    pos.y -= block_height + distance_y;
    prefabs_number += 1;
```

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

Create_Blockを持たせるオブジェクトを生成

Create_Blockはどのオブジェクトに持たせても問題はないが 「誰でも良かった」みたいな犯罪者じみている発想は危険なので しっかり専用のオブジェクトを用意したあげましょう

Cubeなどと同じようにメニューバーから GameObject>Create Empty で「空のGameObject」を生成 名前を「create_blocks」にする

<u>Create Empty</u>とは...

CubeやSphereのような3Dモデルがなく、 初期状態では「Transform」(座標)しかもっていない オブジェクトの生成やユーザーには見えない、 裏でやってるようなプログラムなどをここに付けるといいかも

🛛 Inspector 🛛 🔀 Navigation	Services				<u> -</u> ≡
👕 🗹 create_blocks					🗌 🗌 Static 🔻
Tag Untagged		🔹 Layer 🖸	Default		+
▼ 人 Transform					🔯 🌣,
Position	X 0	Y	0	Z	0
Rotation	X 0	Y	0	Ζ	0
Scale	X 1	Y	1	Z	1
	Add Co	omponent			

先ほど作ったCreate_Blocksスクリプトを Create_blocksオブジェクトにドラッグ&ドロップする

Prefabsをクリックすると... 下にSizeっていうものが出てくる ここに先ほど作ったblock達を入れていくのだが

🛛 Inspector 🛛 🔀 Navigation	Services			≟ +≡
👕 🗹 create_blocks				🗌 🗌 Static 🔫
Tag Untagged	+	Layer Default		÷
▼人 Transform				🔯 🌣,
Position	X 0	Y 0	Z	0
Rotation	X 0	Y 0	Z	0
Scale	X 1	Y 1	Z	1
🔻 📴 🗹 Create_Blocks (Script)				💽 * ,
Script	Create	Blocks		0
▶ Prefabs				

🛛 Inspector 🛛 🔀 Navigation 🛛 S	Services	a +≡
👕 🗹 create_blocks		🗌 🗌 Static 🔻
Tag Untagged	+ Layer Default	+
▼人 Transform		🔯 🌣,
Position	X 0 Y 0 Z 0	
Rotation	X 0 Y 0 Z 0	
Scale	X 1 Y 1 Z 1	
🔻 健 🗹 Create_Blocks (Script)		🔯 🌣,
Script	© Create Blocks	0
🔻 Prefabs		
Size	0	

ここでちょっとした小技を...

通常、ここで入れるオブジェクトの数をSizeに入力して入れる オブジェクトを一つ一つドラッグ&ドロップするのだが 数が多いときはかなりめんどくさい!

ので一度に大量のオブジェクトをリストに入れる方法を紹介

- 1. まずcreate_blocksオブジェクトを選択し、Inspectorの右上 にある「旨」マークをクリックして「旨」にする
- 2. 入れるオブジェクトを複数選択する (Ctrlを押しながら選択していくだけ)
- 3. そのままprefabsにドラッグ&ドロップする

🛛 Inspector 🛛 🔀 Navigation Se	ervices		
👕 🗹 create_blocks			🗌 Stauc 🔻
Tag Untagged	‡ Layer	Default	\$
▼人 Transform			💽 \$,
Position	X 0	Y 0	Z 0
Rotation	X 0	Y 0	Z 0
Scale	X 1	Y 1	Z 1
🔻 健 🗹 Create_Blocks (Script)			💽 \$,
Script	Create_Blocks		0
▼ Prefabs			
Size	0		

L. L.



生成するオブジェクトも入れられたので早速確認



このままではballがblockに 当たっても破壊されないクソゲーなので 「ballに当たったら破壊される」プログラムを実装する 「Block」という名前のC#Scriptを作り blocks_1,2,3にドラッグ&ドロップする

コードを追加

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

public class Block : MonoBehaviour {

void OnCollisionEnter(Collision other) {
 Destroy(this.gameObject);

// Use this for initialization
void Start () {

}

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

プログラム上エラーは吐かれないが 関数名を「<u>OnCollisionEnter()</u>」にしないと ちゃんと衝突時に<u>動作してくれないので注意!</u>

今回はここまで

すいません、ここまでしか資料が間に合いませんでした… 全ては大学と前日にガンダム動物園で遊んだのが原因です 心よりお詫びします