

Unity2D 3分間プログラミング

ボタンのつくりかた



今回やりたいことは これ！

- ①画面にボタンをおく
- ②透明のキャラをつくる
- ③ボタンを押すとHPがへる
- ④HPを表示する



画面設定

Unity をひらいて、2Dで新しいプロジェクトをつくります。

画面比を16 : 9に

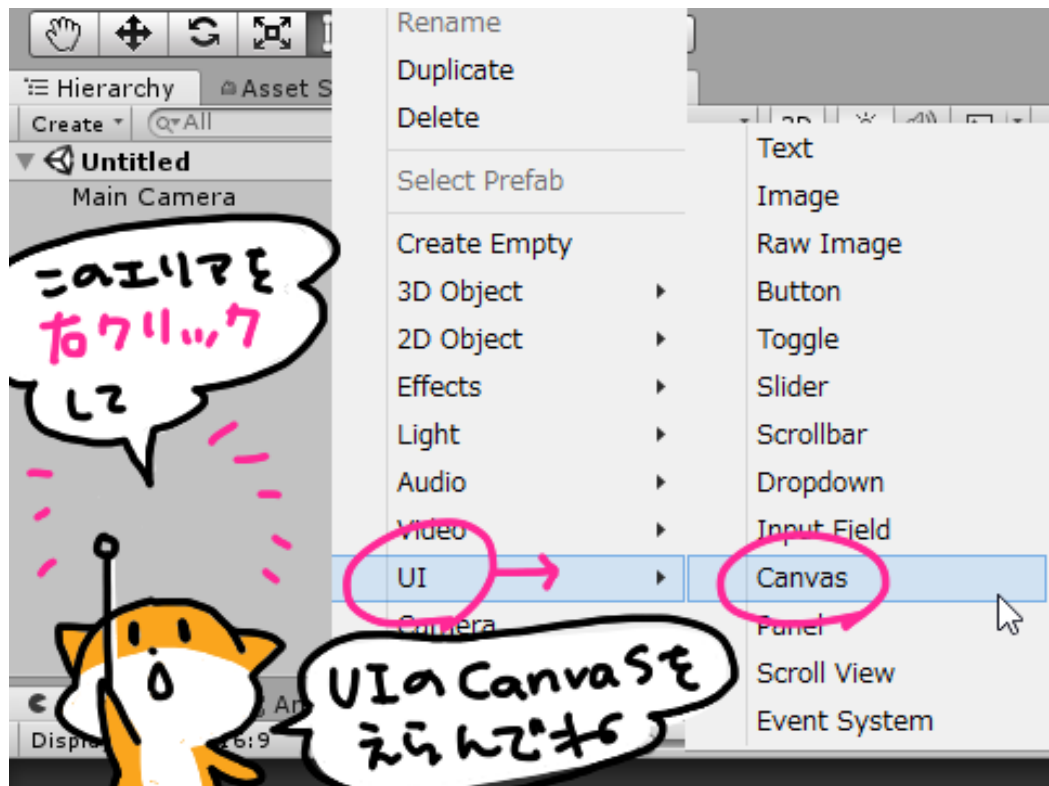
みためにこだわりがなければそのままOKです



キャンバスをおこう

ボタンや文字を表示するのに
キャンバスというのがいります

ヒラエルキーを右クリで
UI⇨Canvasをえらびます



キャンバスの設定



キャンバスを
クリックすると

キャンバスの
インスペクターが
ひらきます

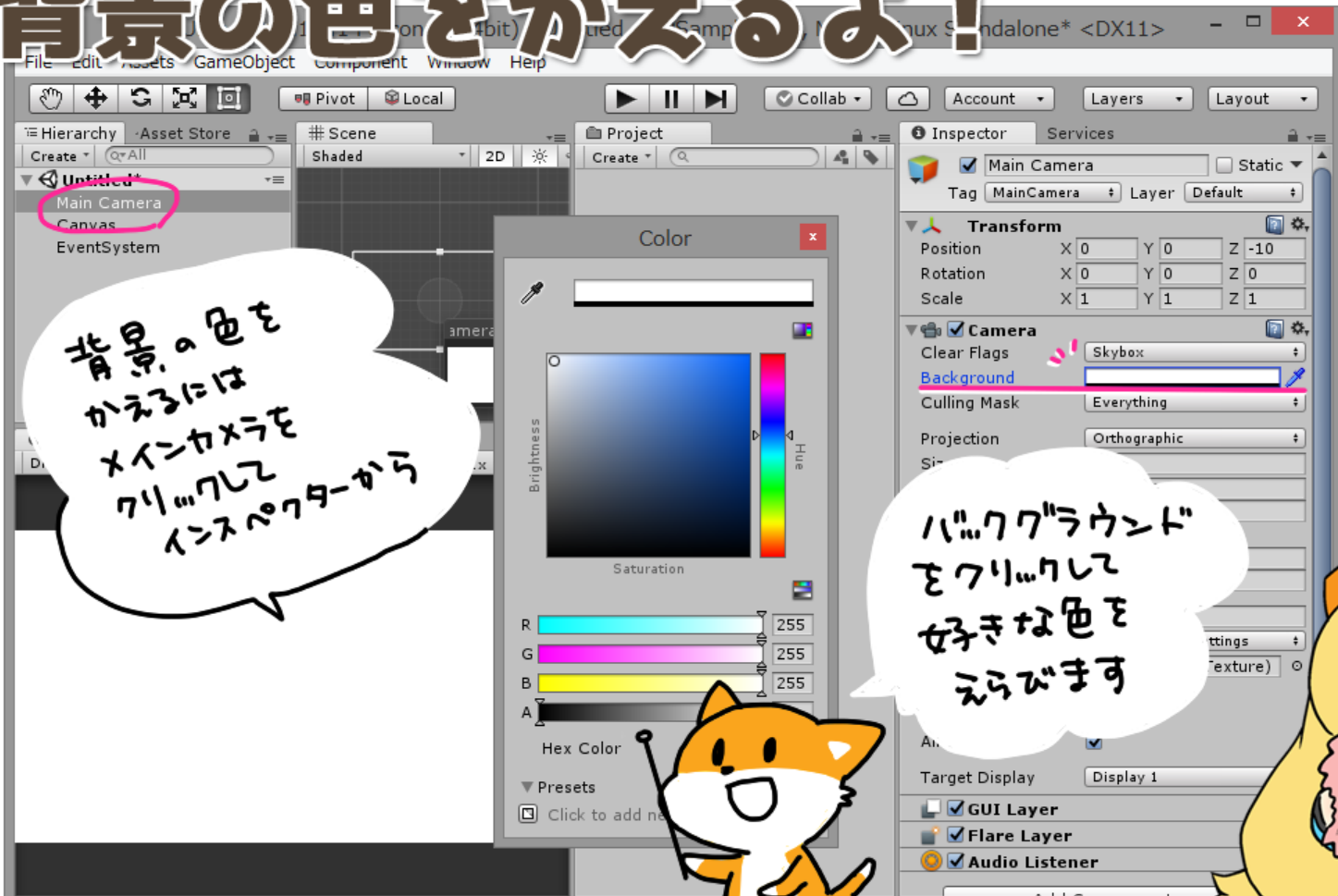
キャンバスを
カメラにあわせて
どのカメラに
あわせるか?を
えらびます。
(今回はメインカメラ)

スケールを画面サイズに
あわせてます。
基準となる画面サイズを
きめます。



こだわりのない人は
とぼしてOK

背景の色をかえるよ!



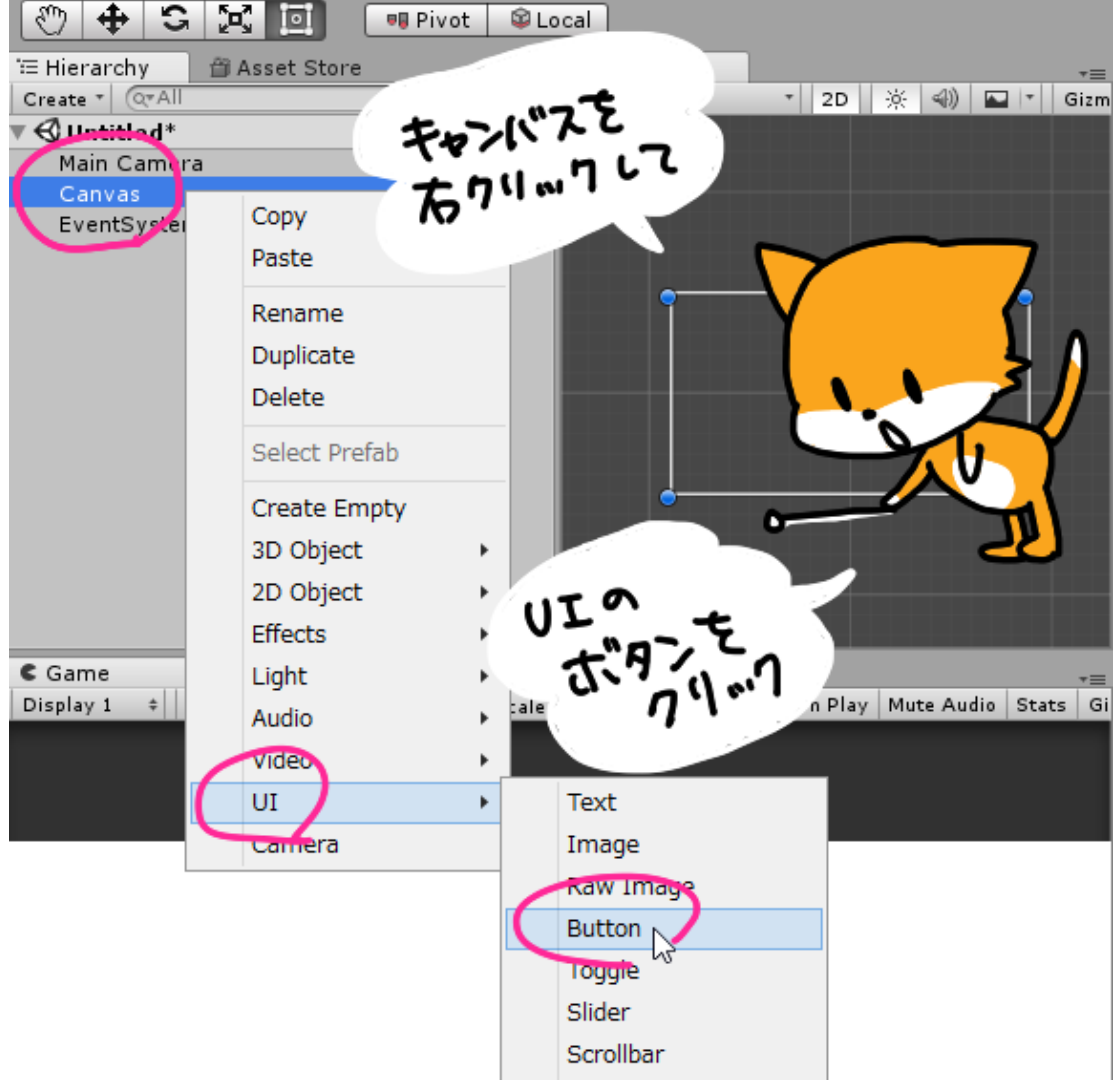
背景の色をかえるには
メインカメラを
クリックして
インスペクターから

ハムククラウド
をクリックして
好きな色を
えらびます



ボタンをおこう

キャンバスの子にしたいので
キャンバスを右クリから
UI⇒ボタンをえらびます



ボタンの大きさを調整するよ

The screenshot shows the Unity 2021.3.1f1 interface. In the Hierarchy panel, a **Button** is selected under a **Canvas**. The Inspector panel shows the **Rect Transform** component with **Width** set to 600 and **Height** set to 100, both values circled in pink. A white speech bubble points to these values with the text: **インスペクタからボタンの幅と高さをいれよう**. Another white speech bubble points to the button in the Hierarchy with the text: **ボタンをクリックして**. A pink speech bubble points to the button in the Game view with the text: **このボタンをドラッグしても大丈夫だよ**. The Inspector also shows the **Canvas Renderer** and **Image (Script)** components. The **Image (Script)** component shows **Source Image** set to **UISprite** and **Image Size** as **32x32**.

インスペクタからボタンの幅と高さをいれよう

ボタンをクリックして

このボタンをドラッグしても大丈夫だよ

Button Image Size: 32x32



ボタンの文字が小さい!

ボタンから
テキストをクリック
してインスペクターを
ひらきます

ダメージ

まず、スケールを
0.5くらいにします

スケールを小さくしたぶん
はばと高さを大きく
します。

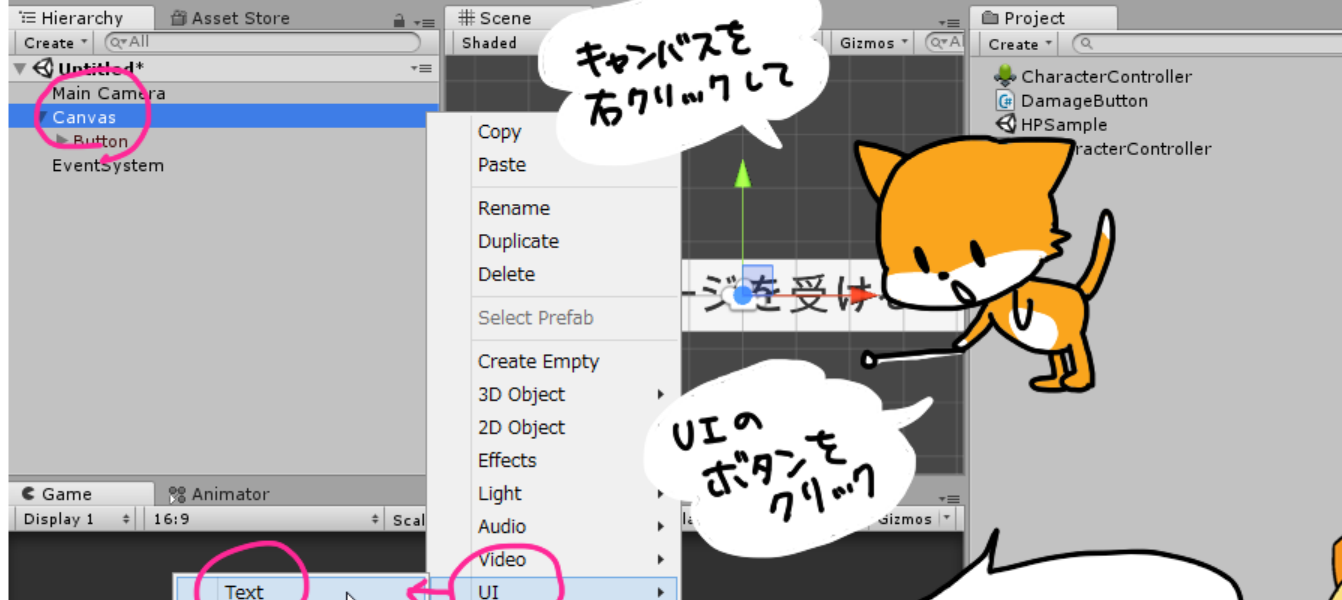
フォントサイズも
スケールを小さくしたぶん
大きめにしておきます

スケールの設定は
文字のにじみ対策

フォントサイズのみ
の変更でもいいよ



文字を表示させてみよう



キャンバスを
右クリックして

UIの
ボタンを
クリック

文字を表示させたはいいぞ
テキストをクリック
しよう

メッセージを受ける



文字を入力しよう



このスケールの設定も文字ののにじみ対策なのでなくてもOKだよ



HPの表示をつくらう

Inspector Panel:

- Text (1)
- Tag: Untagged, Layer: UI
- Rect Transform: center, Pos X: 200, Pos Y: 250, Pos Z: 0, Width: 800, Height: 150
- Canvas Renderer: Text (Script) selected, Text: あなたのHP



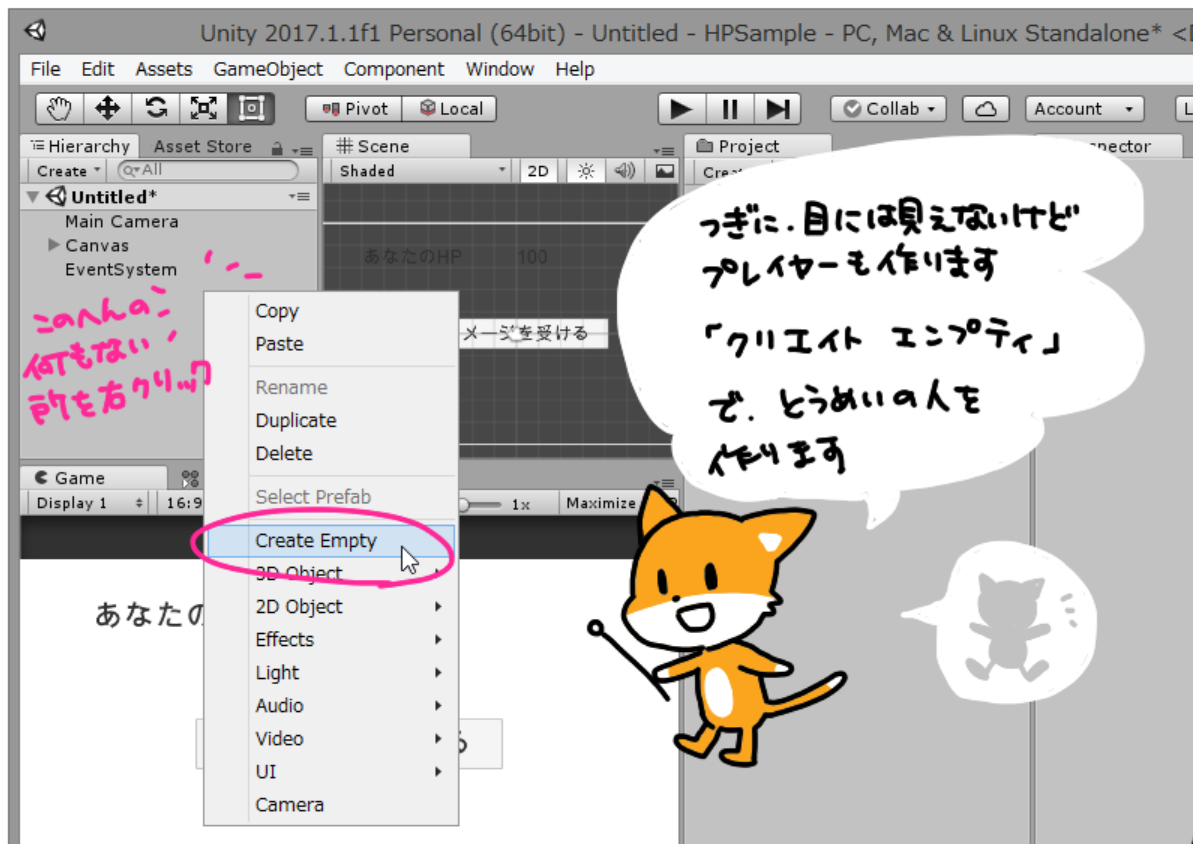
テキストをえらんで"
Ctrl+C コピー
Ctrl+V ペースト
すると
テキストが"
コピーされる

ここに入力

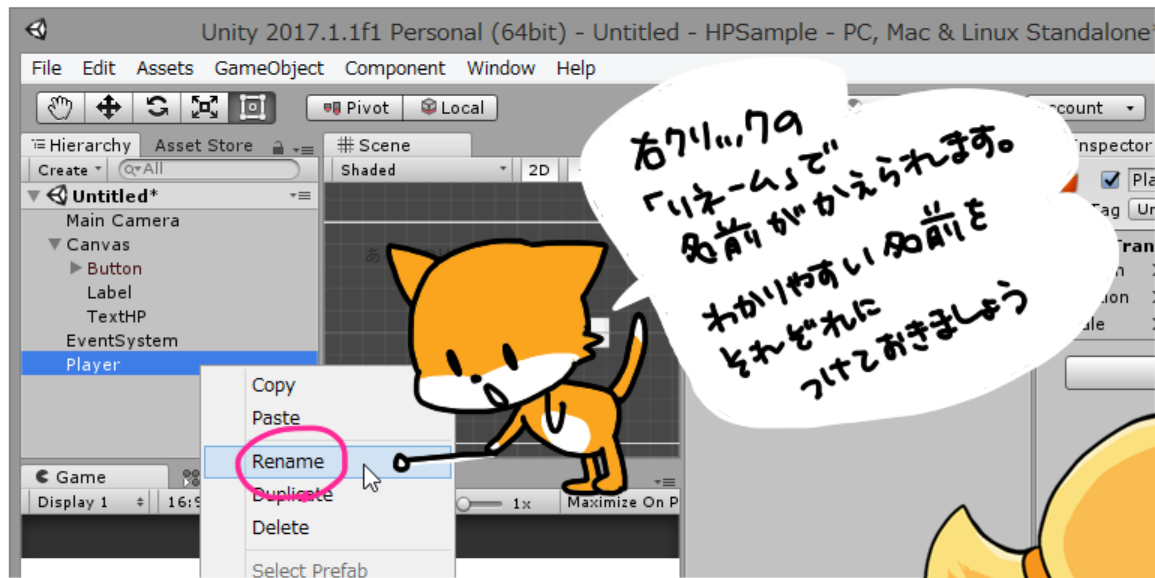
好きな所に
移動させて、
HPを入力します
(100とか)

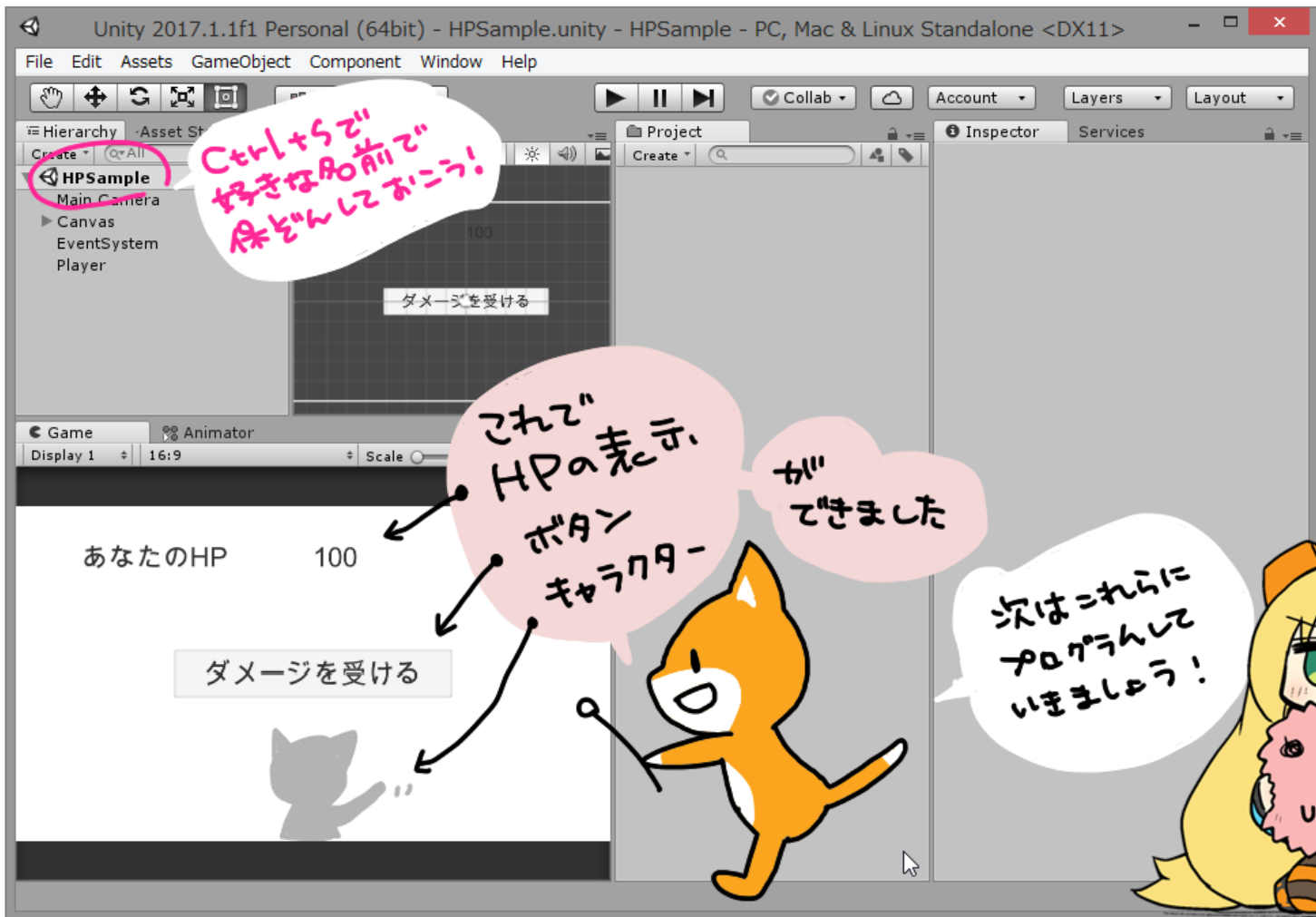


プレイヤーを作ろう



プレイヤーやボタンに名前を付けよう





Ctrl+Sで好きな名前を保存しよう!

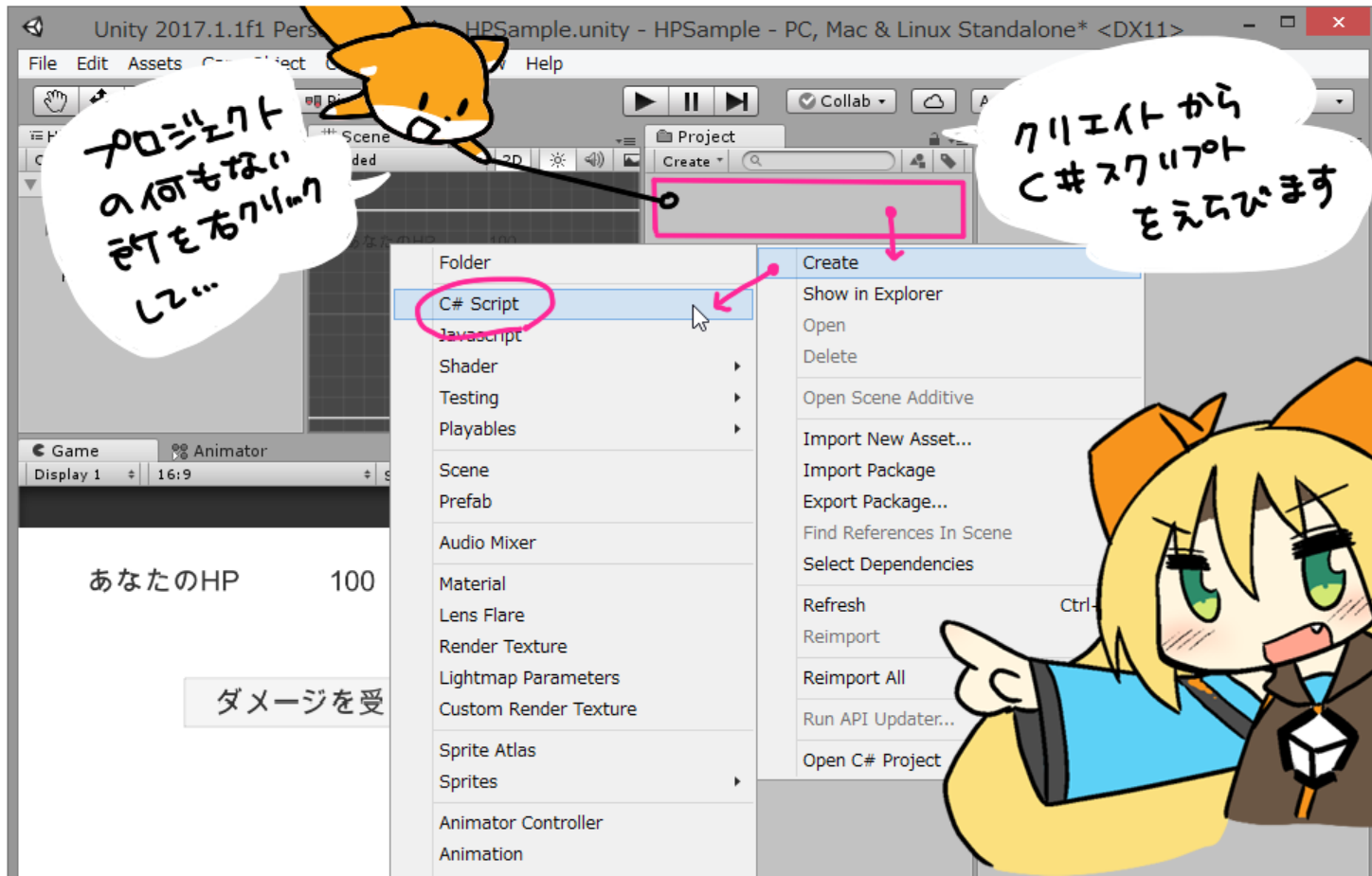
これでHPの表示ボタンキャラクター

かできました

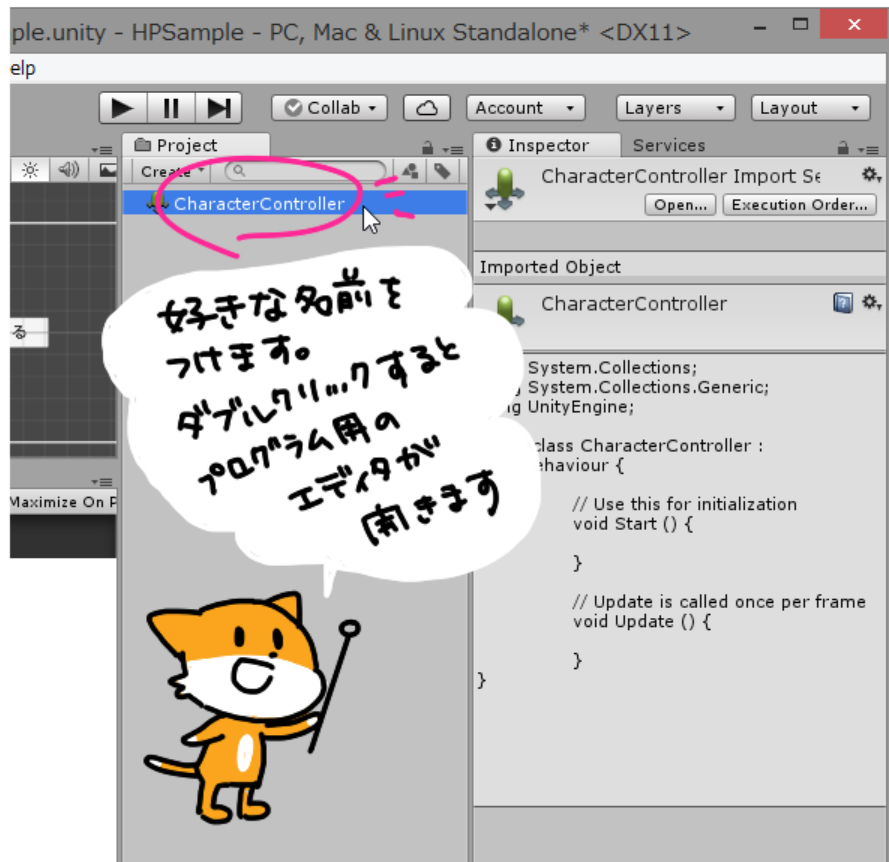
次はこれらにプログラムしていきましょう!



プログラムファイルを作ろう



ファイルに名前を付けよう



あとで
プレイヤーにアタッチする
プログラムだから
プレイヤー hogehoge
とかがいいかも



プログラムしよう

```
Assembly-Char... - MonoDevelop-Unity
File Edit View Search Project Build Run Version Control Tools Window Help
Debug Unity Editor MonoDev... Press 'Control+', to search
Solution HPSample Assembly-Char... CharacterController.cs
No selection
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class CharacterController : MonoBehaviour {
6
7     // HPの実装
8     public int hp; // INT(整数)
9
10    // 死亡フラグ
11    private bool isDead; // BOOL(真偽値)
12
13    // 開始時に呼ばれる
14    void Start () {
15
16        this.isDead = false;
17
18    }
19
20    // 1フレームごとに呼ばれる
21    void Update () {
22
23    }
24 }
25
```

HPなどの
変数を作って
みます。

これは
生きてるぞぞ
フラグは
falseに
してみます。



変数をパブリックに
しておくと
外からなかみを
かえられるよ



プログラムしよう

```
CharacterController.cs
CharacterController ▶ Start ()
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class CharacterController : MonoBehaviour {
6
7     // HPの実装
8     public int hp; // INT(整数)
9
10    // 死亡フラグ
11    private bool isDead; // BOOL(真偽値)
12    ついか
13    public bool IsDead() {
14        return this.isDead;
15    }
16
17    // 開始時に呼ばれる
18    void Start () {
19        this.isDead = false;
20    }
21
22    // 1フレームごとに呼ばれる
23    void Update () {
24    }
25
26 }
```

変数も作ります
「死んでる？」と
聞かれたら、今の状態を
おしえてくれます。

プライベートな
変数にアクセス
するので
「アクセス」とも
いいます

フツーはこうやって
外からカッテに変数を
さわれないように
しとくのだ



プログラムしよう

```
CharacterController.cs
CharacterController ▶ No selection

7 // HPの実装
8 public int hp; // INT(整数)
9
10 // 死亡フラグ
11 private bool isDead; // BOOL(真偽値)
12
13 public bool IsDead() {
14     return this.isDead;
15 }
16
17 public void Damage( int value ) {
18     // ダメージ値を減算
19     this.hp -= value;
20
21     // 死んだ?
22     if( this.hp <= 0 ) {
23         // 0以下にはしない
24         this.hp = 0;
25
26         // 死亡フラグを立てる
27         this.isDead = true;
28     }
29 }
30
31
32
```

ダメージを受ける用の
関数も作ります。

ダメージ値を引くと
HPを減らします。

死んだ時のフラグも
こゝにします。



HPも外でカッテに
へらされたらヤダから
こんなかんじにするよ



プログラムをアタッチしよう

プレイヤーの
インスプレターを
ひらいて

さきの
プログラムを
ドラッグして
みましょう

パブリックで
作った変数が
表示されています

ここでHPの値を
自由にかえられる
ようになりました

CharacterController

Inspector

Player

Tag Untagged Layer Default

Transform

Position X 0 Y 0 Z 0

Rotation X 0 Y 0 Z 0

Scale X 1 Y 1 Z 1

Character Controller (Script)

Script CharacterControll

Hp 0

Add Component

Hierarchy

Asset Store

Scene

Shaded

2D

HPSample*

Main Camera

Canvas

Button

Label

TextHP

EventSystem

Player

ダメージを受ける

Game

Animator

Display 1

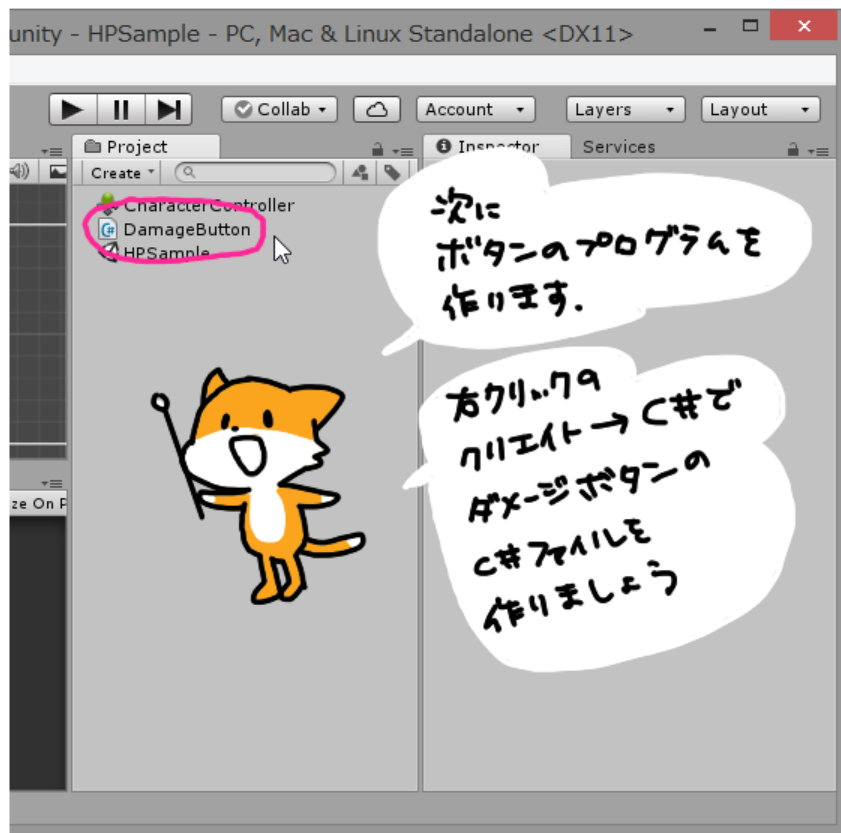
16:9

あなたのHP 100

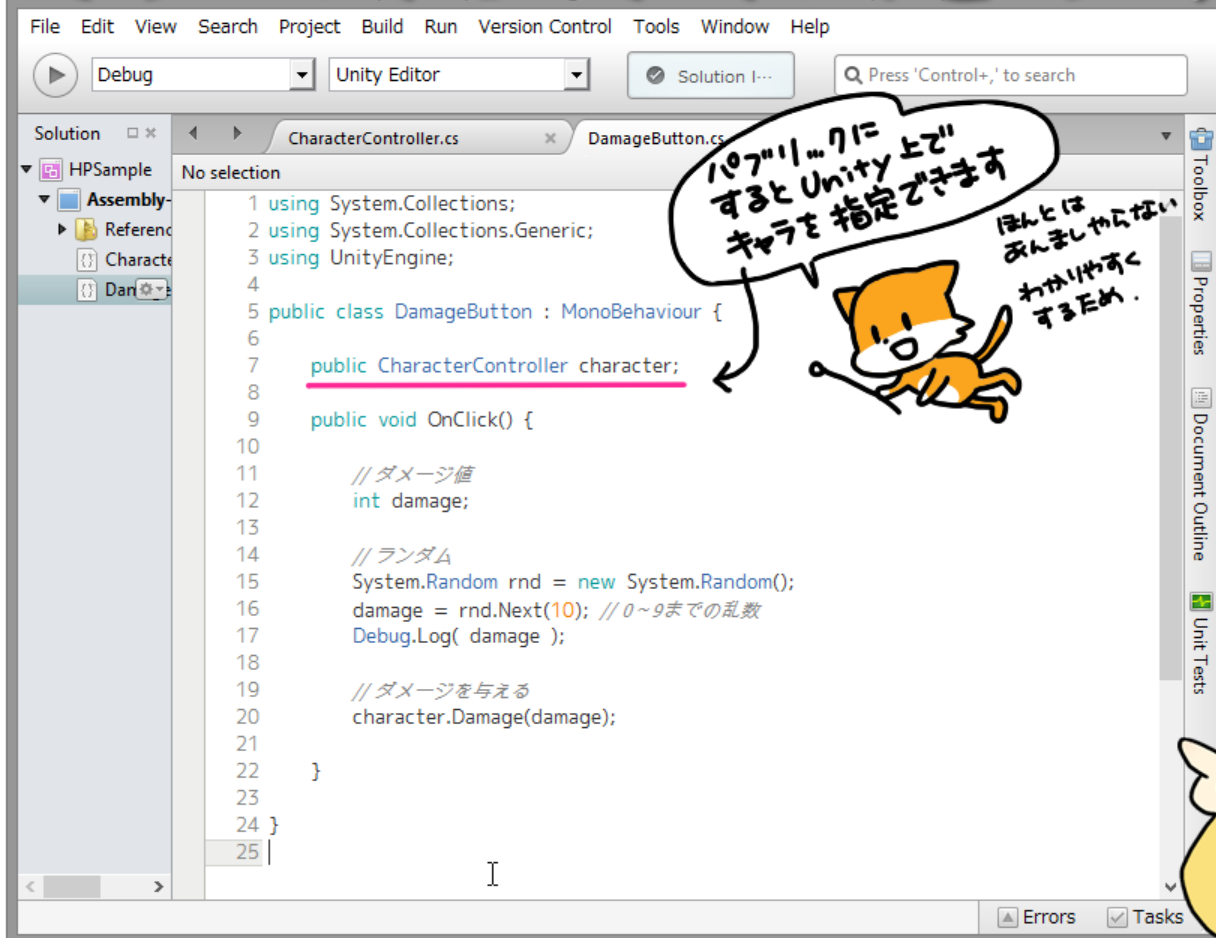
ダメージを受ける



ボタンのプログラムをつくろう



ボタンのプログラムをつくろう



```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class DamageButton : MonoBehaviour {
6
7     public CharacterController character;
8
9     public void OnClick() {
10
11         // ダメージ値
12         int damage;
13
14         // ランダム
15         System.Random rnd = new System.Random();
16         damage = rnd.Next(10); // 0~9までの乱数
17         Debug.Log( damage );
18
19         // ダメージを与える
20         character.Damage(damage);
21
22     }
23
24 }
25
```

1/07/11...クに
すると Unity 上で
キャラを指定できます

ほんとは
あんまりやらない
わかんなく
するだけ。

あとから
Unity 上で指定するよ



ボタンのプログラムをつくろう

The screenshot shows the Unity Editor interface with a C# script named `DamageButton.cs` open. The code defines a `DamageButton` class that inherits from `MonoBehaviour`. It includes a `CharacterController` reference and an `OnClick()` method that generates a random damage value and applies it to the character.

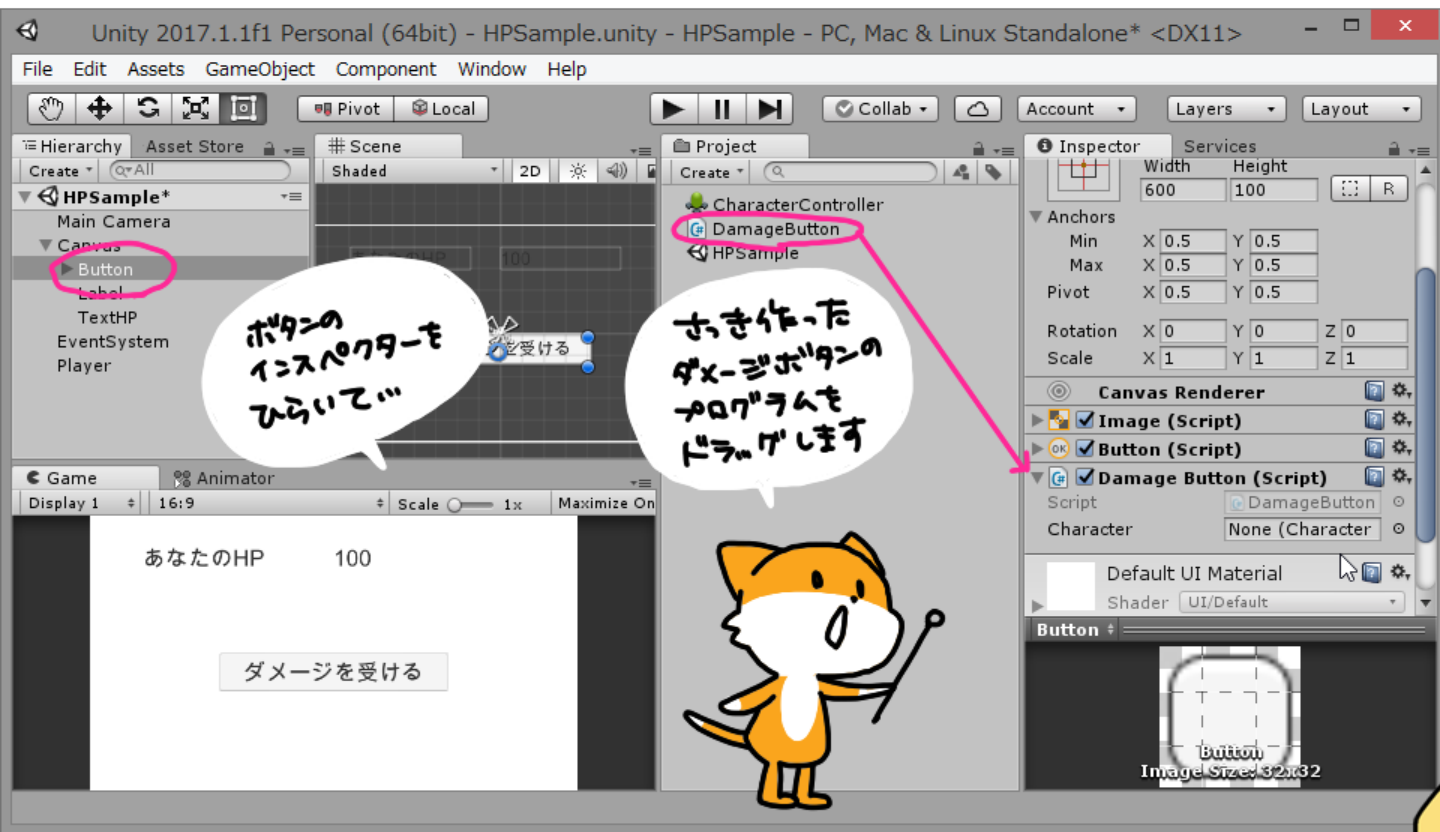
```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class DamageButton : MonoBehaviour {
6
7     public CharacterController character;
8
9     public void OnClick() {
10
11         // ダメージ値
12         int damage;
13
14         // ランダム
15         System.Random rnd = new System.Random();
16         damage = rnd.Next(10); // 0~9までの乱数
17         Debug.Log( damage );
18
19         // ダメージを与える
20         character.Damage(damage);
21     }
22 }
23
24 }
25
```

Handwritten annotations in Japanese explain the code:

- Top speech bubble: オンクリックでボタンが押された時の処理をかきます (I will write the processing when the button is clicked with OnClick).
- Arrow pointing to line 12: 変数をつくって (I create a variable).
- Arrow pointing to lines 15-17: 変数にランダム値を入れます (I put a random value into the variable).
- Arrow pointing to line 20: おきキャラ用に作ったダメージ関数でかき、キャラにダメージをあてます。 (I use the damage function I made for the character to apply damage to the character).



プログラムをアタッチしよう



ボタンの
インスペクターを
ひらいて...

さき作った
ダメージボタンの
プログラムを
ドラッグします



こんどはボタンに
アタッチするよ



プログラムをアタッチしよう

The screenshot shows the Unity 2017.1.1f1 Personal interface. The Hierarchy panel on the left shows the scene structure: HPSample* > Main Camera > Canvas > EventSystem > Player. A pink circle highlights the 'Player' object. A pink arrow points from this circle to the 'Inspector' panel on the right. In the Inspector, the 'Character' section is expanded, and a pink circle highlights the 'Player (Character Controller)' component. A pink arrow also points from the 'Player' object in the Hierarchy to this component. The Inspector also shows other components like 'Image (Script)', 'Button (Script)', and 'Damage Button (Script)'. The Game view at the bottom left shows a character with 'あなたのHP' (Your HP) and a 'ダメージ' (Damage) button. A speech bubble from the character says: 'キャラクターのオブジェクトにプレイヤーをドラッグします' (I drag the player to the character object). The Project panel shows 'CharacterController', 'DamageButton', and 'HPSample'.

Unity 2017.1.1f1 Personal (64bit) - HPSample.unity - HPSample - PC, Mac & Linux Standalone* <DX11>

File Edit Assets GameObject Component Window Help

Hierarchy Asset Store # Scene

Shaded 2D

Project

Inspector Services

Max X 0.5 Y 0.5

Pivot X 0.5 Y 0.5

Rotation X 0 Y 0 Z 0

Scale X 1 Y 1 Z 1

Canvas Renderer

Image (Script)

Button (Script)

Damage Button (Script)

Script DamageButton

Character Player (Character Controller)

Default UI Material

Shader UI/Default

Add Component

Button Image Size: 32x32

Game Animator

あなたのHP

ダメージ

キャラクターのオブジェクトにプレイヤーをドラッグします



ボタンを押されたときの設定をしよう

The screenshot shows the Unity 2017.1.1f1 Personal interface. The Hierarchy panel on the left shows a 'Button' object under a 'Canvas' under 'HPSample*'. The Inspector panel on the right shows the 'Button (Script)' component selected, with the 'On Click ()' event list highlighted in cyan. The Game View at the bottom shows a button labeled 'ダメージを受ける' (Receive Damage) and a text display showing 'あなたのHP 100' (Your HP 100).

Unity 2017.1.1f1 Personal (64bit) - HPSample.unity - HPSample - PC, Mac & Linux Standalone* <DX11>

File Edit Assets **GameObject** Component Window Help

Hierarchy Asset Store # Scene Project Inspector Services

CharacterController
DamageButton
HPSample

Image (Script)
Button (Script)

Interactable

Transition Color Tint

Target Graphic Button (Image)

Normal Color
Highlighted Color
Pressed Color
Disabled Color
Color Multiplier 1
Fade Duration 0.1

Navigation Automatic Visualize

On Click ()

Runtime C# No Function
None (O) +

DamageButton (Scr. Add to list)

Game Animator

Display 1 16:9 Scale 1x Maximize On

あなたのHP 100

ダメージを受ける

これでボタン
完成!?
と思わせておいて
まだ設定があります

OnClick
イベントに
どれを割りあてるか
設定します



「+」を
クリック

これがメンドイ...



ボタンを押されたときの設定をしよう

Unity 2017.1.1f1 Personal (64bit) - HPSample.unity - HPSample - PC, Mac & Linux Standalone* <DX11>

File Edit Assets **GameObject** Component Window Help

Hierarchy: HPSample* > Canvas > Button

Inspector: Image (Script), Button (Script)

Button (Script) Properties:

- Interactable:
- Transition: Color Tint
- Target Graphic: Button (Image)
- Normal Color: [Color Field]
- Highlighted Color: [Color Field]
- Color: 1
- Flash: 0.1
- Navigation: [Dropdown]
- On Click: [Event List]
- Runtime Clickable: No Function
- Button (Asset)
- DamageButton (Script)
- Shader: Default

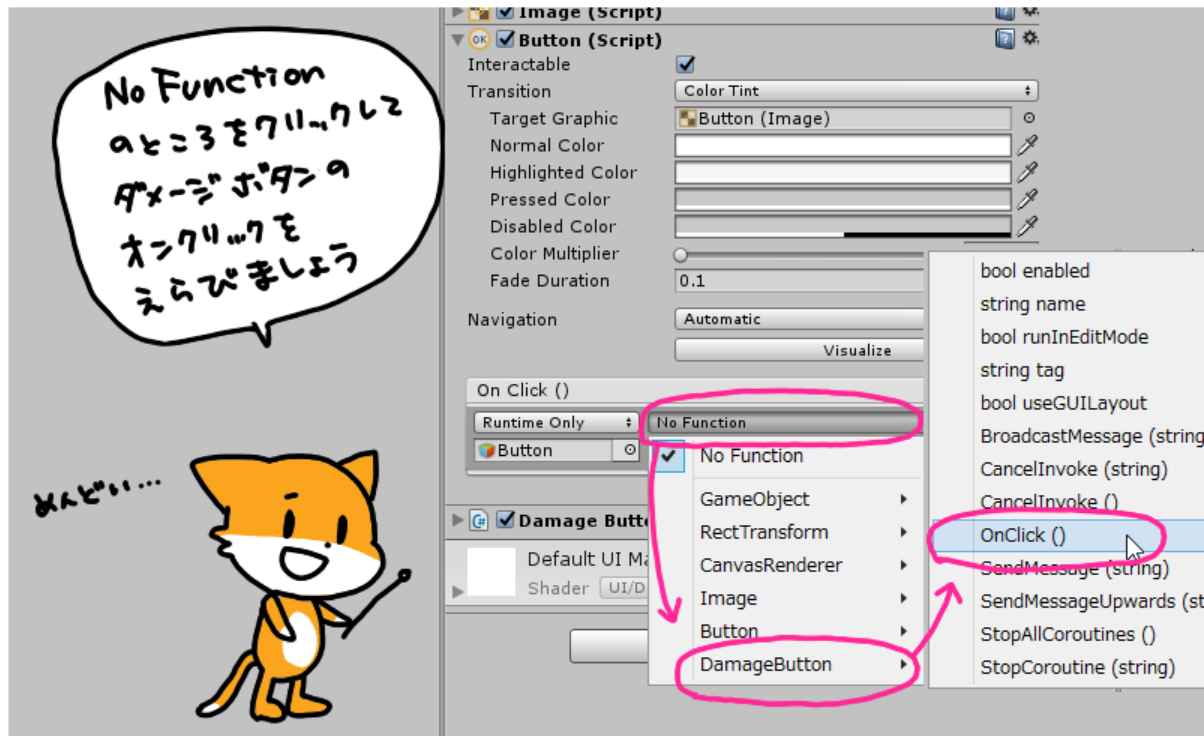
Select Object Dialog:

- Assets: Button (Selected)
- Scene: [None]
- Canvas
- EventSystem
- Button Game Object

Game View: あなたのHP 100, ダメージを受ける



ボタンを押されたときの設定をしよう



No Function
あと3を711712
7x-ジ ボタンの
オンクリックを
えらびましょう

なんだい...



これで
アタッチされた
プログラムの
オンクリックが
関連付けされたよ



プレイしてみよう

Unity 2017.1.1f1 Personal (64bit) - HPSample.unity - HPSample - PC, Mac & Linux Standalone <DX11>

File Edit Assets GameObject Component Window Help

Start (Play) button circled in pink.

Inspector: Player (Tag: Untagged, Layer: Default), Transform (Position: X 0, Y 0, Z 0; Rotation: X 0, Y 0, Z 0; Scale: X 1, Y 1, Z 1), Character Controller (Script) (Hp: 95).

Hierarchy: HPSample > Main Camera > Canvas > Button (Label: TextHP, EventSystem, Player).

Game View: あなたのHP 100, ダメージを受ける (button).

Annotations:

- Speech bubble: スタートをお試ししよう
- Speech bubble: ダメージを受ける ボタンを押すと...
- Speech bubble: プレイヤーのインスペクター上でのHPと、ステージ上のHPが、連動していません。
- Handwritten note: こちらのHPが、正しい!!
- Handwritten note: こちらのHPは、ちゃんと入る



表示用のプログラムをつくろう

Unity 2017.1.1f1 Personal (64bit) - HPSample.unity - HPSample - PC, Mac & Linux Standalone <DX11>

File Edit Assets GameObject Co

もう一回おして
編集モードに
もどしてあげる

右クリック
クエイトから
もう一つ
C#ファイル
追加します。

Inspector Services

Player Static

Tag Untagged Layer Default

Transform

Position X 0 Y 0 Z 0

Rotation X 0 Y 0 Z 0

Scale X 1 Y 1 Z 1

Character Controller (Script)

Script CharacterControll

Hp 100

Add Component

Game Animator

Display 1 16:9 Scale 1x Maximize On

あなたのHP 100

ダメージを受ける

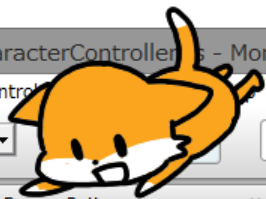
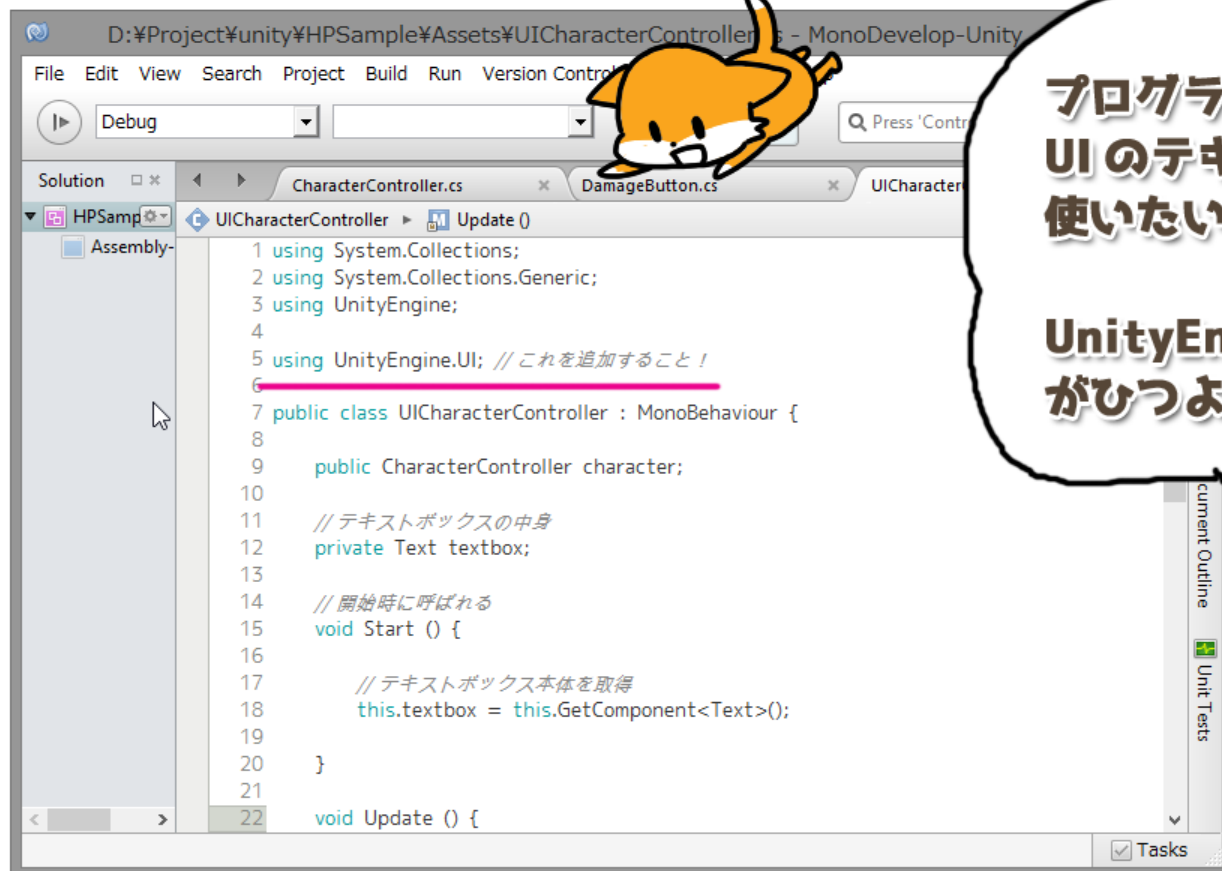
ダメージを受ける

5

また
プログラム...



表示用のプログラムをつくろう



プログラムで
UIのテキストを
使いたい時は

UnityEngine.UI
がひつようだよ



表示用のプログラムをつくろう

```
D:\Project\unity\HPSample\Assets\UICharacterController.cs - MonoDevelop-Unity
File Edit View Search Project Build Run Version Control Tools Window Help
Debug
MonoDev...
Press 'Control+', to search
Solution
CharacterController.cs
DamageButton.cs
UICharacterController.cs
HPSamp
Assembly-
UICharacterController
Update ()
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 using UnityEngine.UI; // これを追加すること!
6
7 public class UICharacterController : MonoBehaviour {
8
9     public CharacterController character;
10
11     // テキストボックスの中身
12     private Text textbox;
13
14     // 開始時に呼ばれる
15     void Start () {
16
17         // テキストボックス本体を取得
18         this.textbox = this.GetComponent<Text>();
19
20     }
21
22     void Update () {
```

テキストを使うための
準備をします



表示用のプログラムをつくろう

```
19 }
20 }
21 }
22 void Update () {
23     // 死んでいる
24     if( character.IsDead() ) {
25         // 死んでいる
26         this.textbox.text = "こいつはもう死んでいる";
27     }
28 }
29 // 死んでいない
30 else {
31     // 文字列に変換して代入
32     this.textbox.text = character.hp.ToString();
33 }
34 }
35 }
36 }
37 }
38 }
39 }
40 }
```

アップデート内は
毎フレームごとに
実行されるよ

つま、**「ずっ」と**
みたいなも!

キャラクターが
死んでたら Xメッセージを
生きてたら キャラのHPを
テキストに入れて
表示させるよ!

テキストにプログラムをアタッチ

Unity 2017.1.1f1 Personal (64bit) - HPSample.unity - HPSample - PC, Mac & Linux Standalone* <DX11>

File Edit Assets GameObject Component Window Help

Hierarchy Asset Store # Scene Project Inspector Services

CharacterController
DamageButton
HP Sample
UICharacterController ①

TextHP
Tag Untagged Layer UI

Rect Transform
Canvas Renderer
Text (Script)
UI Character Controller (Script)
Script UICharacterContr
Character Player (Character) ②

Default UI Material
Shader UI/Default

Add Component

あなたのHP 100
ダメージを受ける

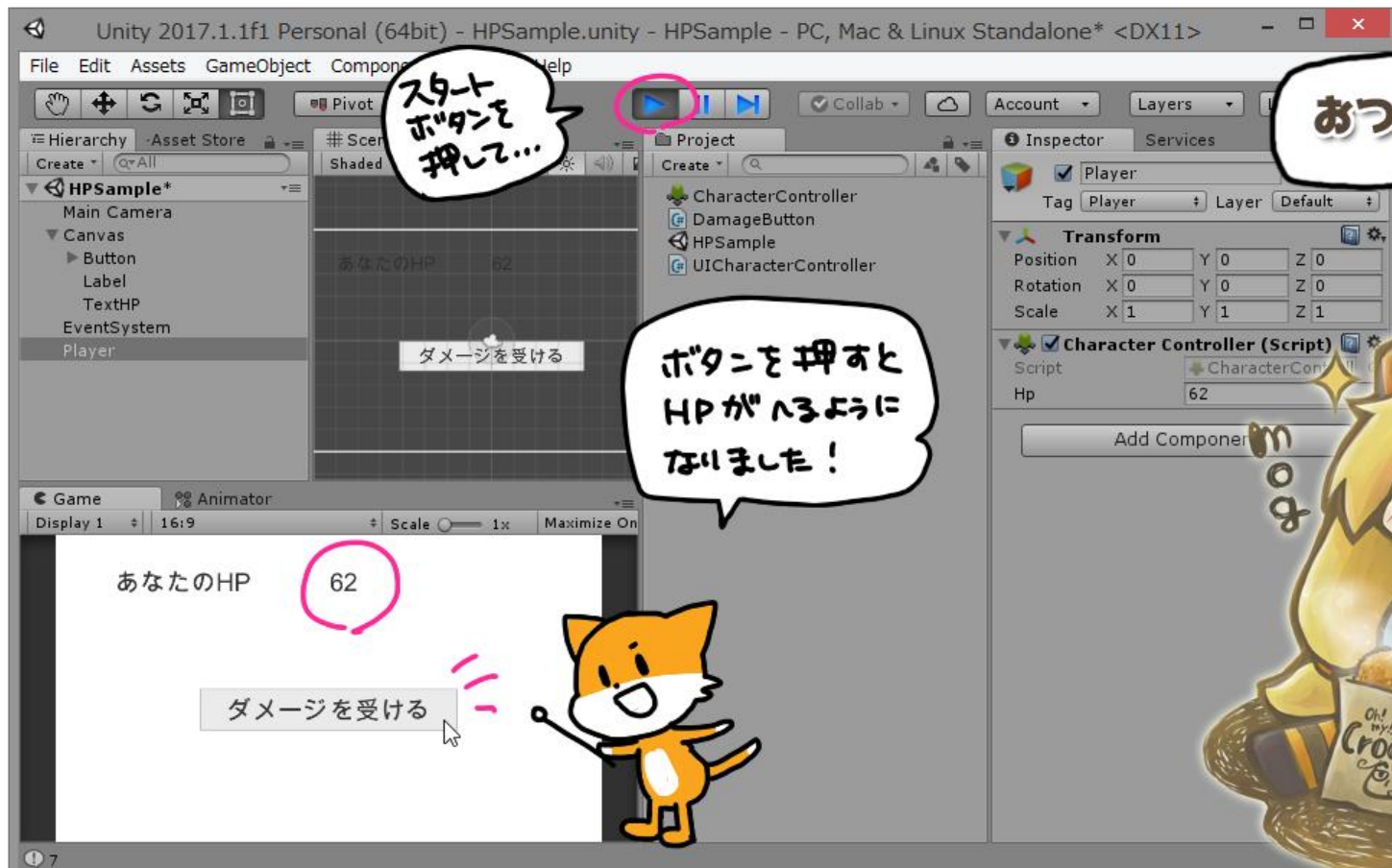
① プログラムをテキストのインスペクターにアタッチ

② キャラクターのオブジェクトにプレイヤーをアタッチする

あなたのHP
ダメージを受ける



かんせい！



おつかれさまー！

ボタンを押すとHPが62になりました！

