株式会社アウトスタンディング
さいたまげーむす　勉強会資料

勉強会情報
https://atnd.org/users/184556

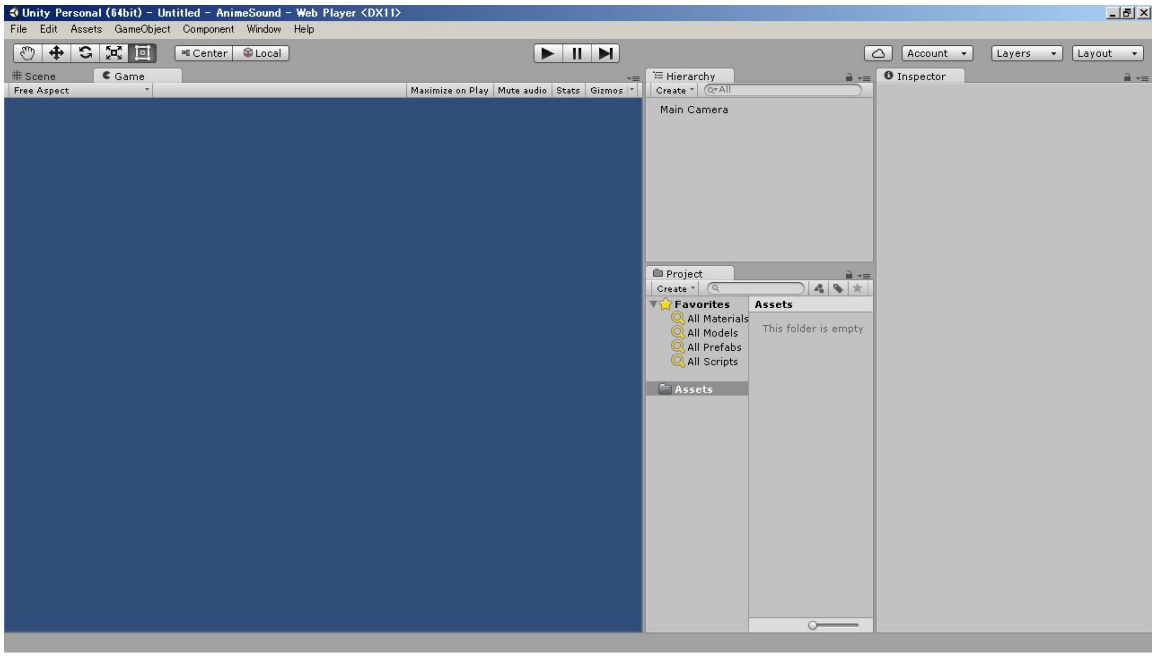
アプリ作ってます！

Android
https://play.google.com/store/apps/developer?id=SAITAMA_GAMES

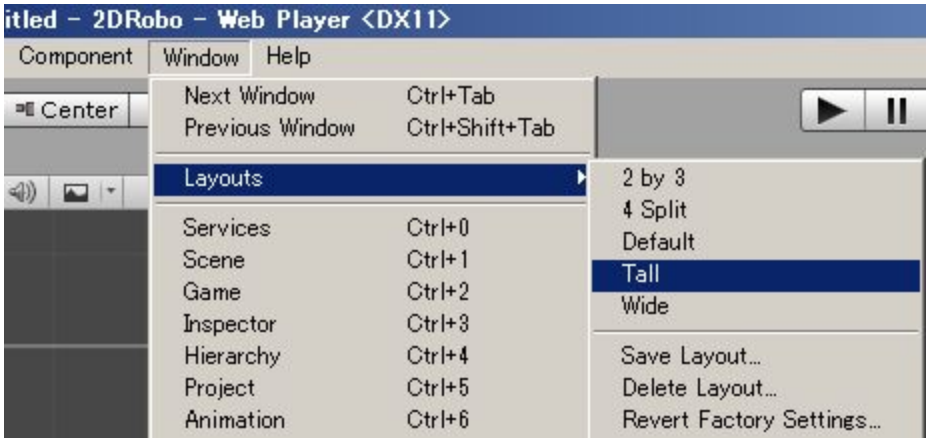
IOS
https://itunes.apple.com/jp/app/pichannobaningu!/id880087045

●ゲームの内容
右から左へ流れてくるアイコンと一致したキーをタイミング良く押す事でスコアを得ると共に、プレイヤーが各ボタンと連動したポーズを取る、「パラッパラッパー」を意識したゲームです。システムとしては、ビートマニア等の基礎となるプログラムを用いています。
※今回はサウンドを使用しない為、音ゲーとは決して呼べないのですが、ご了承下さい。

(0)起動
使用する素材 AnimeSound.zip をダブルクリックして解凍し、「AnimeSound」ファイル、及び「AnimeSound 作成資料」がある事を確認します。
Unity5 を起動し、Project name の欄に適当なプロジェクト名（今回は”AnimeSound”）を入力します。
尚、Location については、データを保存したい場所を指定します。
今回は3Dゲームを作成するので、3D 2D の欄にて”3D”を選択し、最後に”Create project”を押して起動して下さい。



上の画面が出てきたら、Windowメニュー > Layouts > Tall と選択し、作業レイアウトをTallモードにして下さい。

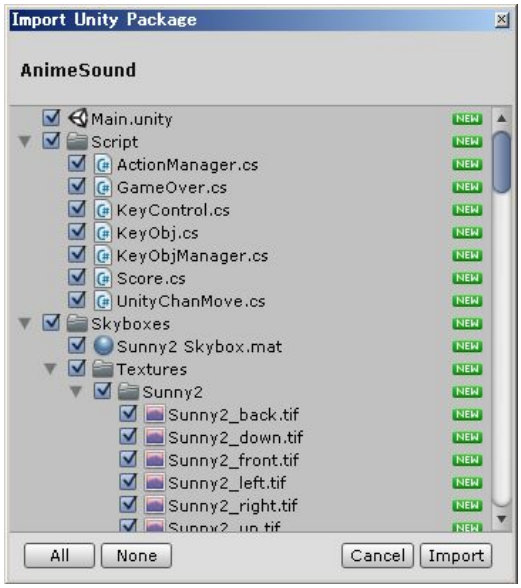


(1)準備

1-1)ゲームデータの準備。

Projectビュー > Assets フォルダ内で右クリックし、Import Package > Custom Package… を選択。

「AnimeSound.unitypackage」を選択します。



上写真の様にインポート内容が表示されますので、全てのファイルにチェックが入っている事を確認して、右下の Import ボタンをクリックして下さい。

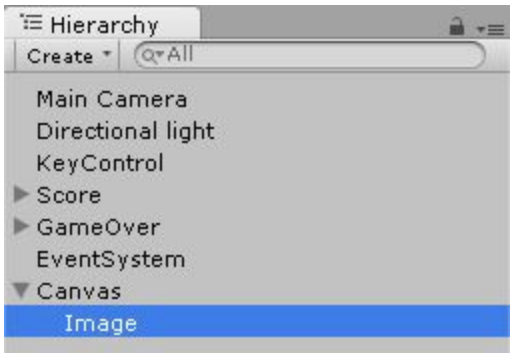
・Main.unity
・Scriptフォルダ
・Skyboxesフォルダ
・Textureフォルダ
・UnityChanフォルダ

が、Assets フォルダ内に表示されます。「Main.unity」をダブルクリックしてMainシーンに移行し、本格的にゲーム制作を開始します。

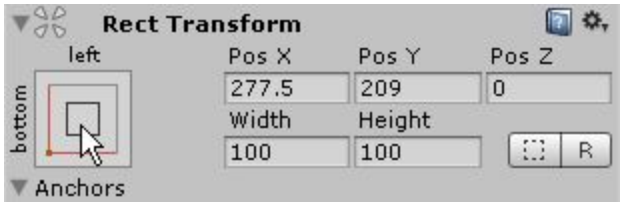
尚、Hierarchy ビューにはすでにいくつかのオブジェクトが作成されており、スコアやゲームオーバーの表示等が予め出来ています。よって、今回はゲームシステムの構築をメインに進めていきます。

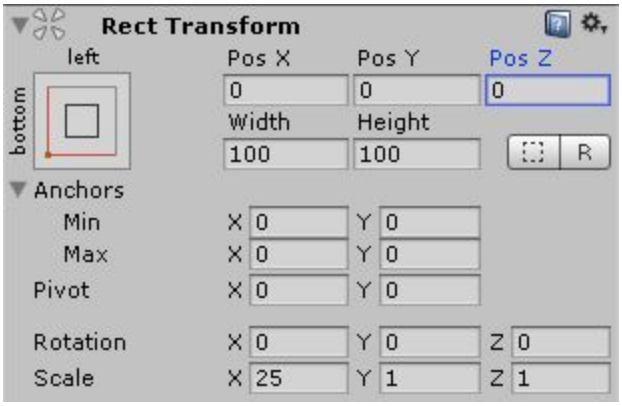
(2)アイコン及びゲームシステムの準備と設定

2-1）2Dのイメージの作成していきます。
Hierarchy ビュー > Create > UI > Image を選択します。すると自動的に、
▼Canvas（親）
　　・Image（Canvasの子）
これらが生成されます。また、Game ビューの中央に、白い四角が表示されます。この四角の中にスコアの表示を設定していきます。

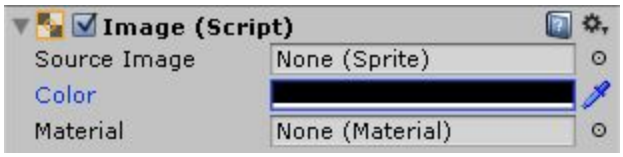


まず Image を選択し、名前を「Back」と変更します。
そして、Inspector ビュー > Rect Transform 内の四角い枠をクリックして下さい。Anchor Presets と言う項目が出てきますので、そこで bottom left をクリックして閉じます。
続いて、Rect Transform 内の Anchors と Scale 項目、及び Pos XYZ と Width・Height を下記の値に変更します。
Anchors
Pivot　X: 0　Y: 0
Scale　X: 25　Y: 1　Z: 1
Pos X: 0　Pos Y: 0　Pos Z: 0
Width: 100　Height: 100
（注）必ず Anchors を先に設定して下さい。Anchors を基準に Pos が変動するので、設定順序が逆になると位置も変化してしまいます。





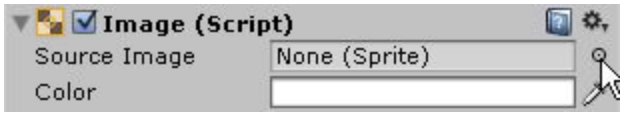
さらに、Inspector ビュー > Image (Script) > Color 項目左の白いエリアをクリックして下さい。Color ビューが表示されますので、色を下記の様に変更して下さい。
Color　: R: 0　G: 0　B:0　A:255

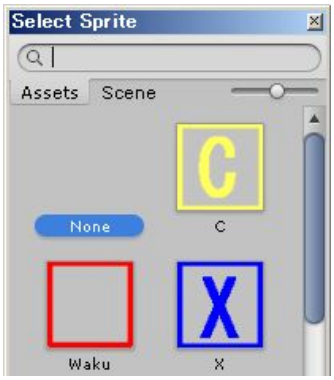


これで、白い四角だった Image が、黒の細長い四角に変更され、画面下に移動しました。

2-2)以後作成予定であるキーアイコンを認識する為の枠を作成します。
Hierarchy ビュー > Create > UI > Image を選択し、自動的に「Canvas」内に生成された Image の名前を「Waku」に変更します。
Waku を選択し、Inspector ビュー > Rect Transform 内の設定を、それぞれ下記に変更して下さい。
四角い枠 : bottom left
Anchors
Pivot　X: 0　Y: 0
Pos X: 0　Pos Y: 0　Pos Z: 0
Width: 100　Height: 100

さらに、Inspector ビュー > Image (Script) > SourceImage の項目の右にある◎をクリックすると、Select Sprite ビューが出てきますので、Assets タブ内の「Waku」をダブルクリックして下さい。Game ビューにて、白い四角が選んだ画像に変わります。



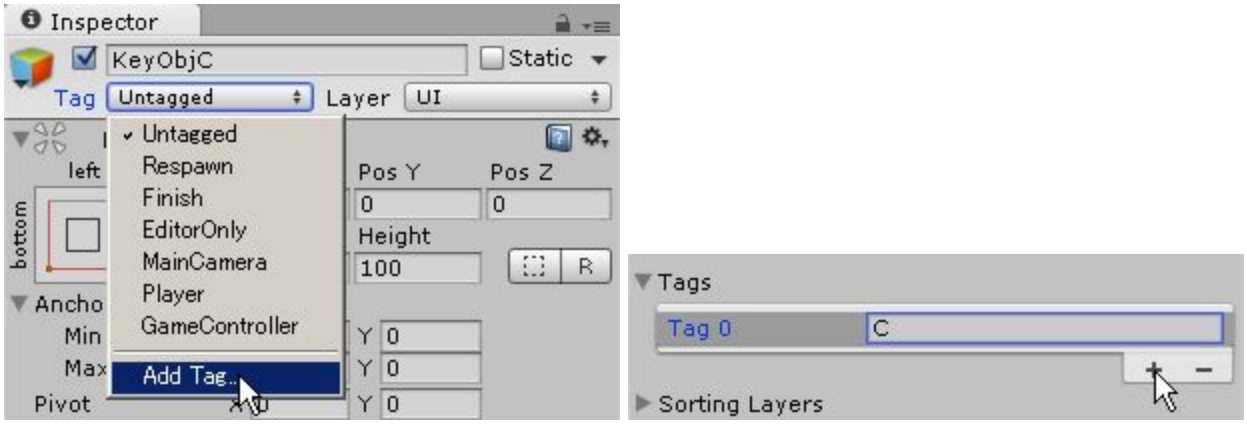


2-3)キーアイコンを作成します。
Waku 作成時と同様、Hierarchy ビュー > Create > UI > Image を選択し、自動的に Canvas 内に生成された Image の名前を「KeyObjC」に変更します。
次に、Inspector ビュー > Rect Transform 内の設定を、それぞれ下記に変更して下さい。
四角い枠 : bottom left
Anchors
Pivot　X: 0　Y: 0
Pos X: 0　Pos Y: 0　Pos Z: 0
Width: 100　Height: 100

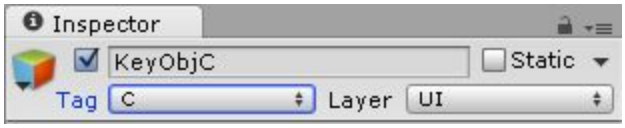
そして、Inspector ビュー > Image (Script) > SourceImage 項目より、(2-2)の作業と同様にSelect Sprite 画面を出し、「C」を選択します。

続いて、キーアイコンに動きを追加する為、スクリプト（プログラム）を導入します。
Inspector ビュー > Add Component > Scripts > Kye Obj を選択して下さい。

これで黄色の C の文字枠の作成は完了ですが、最後に C オブジェクトに「C」というタグ付けを行ないます。
Inspector ビュー > Tag 項目を開き、Add Tag… を選択します。すると、Inspector ビューの内容が変わりますので、Tags の項目を開いて List を追加します。 + ボタンを押して、Tag 0 を追加し、名前を「C」に変えて下さい。



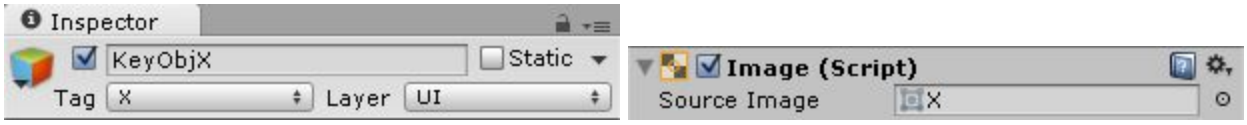
変えたら、Hierarchy ビュー > KeyObjC オブジェクトを選択し、Inspector ビュー > Tag 項目を開きます。C が追加されていると思いますので、これを選択して下さい。こうする事によって、「C」の Image に “C” というタグが付けられ、スクリプト（プログラム）内で、“C” のタグが付いたオブジェクトに何らかの処理をする際に必要となります。



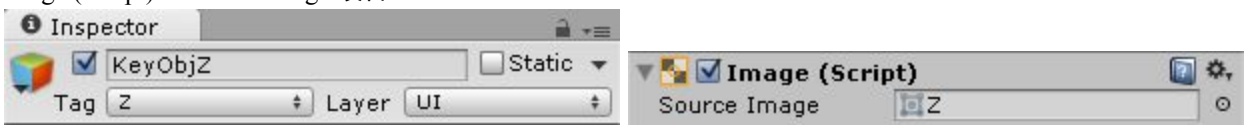
ここで一度、ゲームを再生してみて下さい。画面下の黒い四角の右端から「C」のアイコンが左に流れてくれば大丈夫です。

2-4)キーアイコンが1種類だけではゲームとして面白くない為、さらにキーアイコンを2つ作成します。
作り方ですが、(2-3)と同じ手順で作成するのは面倒です。そこで、「KeyObjC」オブジェクトを複製していきます。

Hierarchy ビュー > KeyObjC を右クリック > Duplicate（複製） を選択。
これを2回行ない、新たに同じオブジェクトが2個生成されますので、それぞれの名前と、Inspector ビュー内の設定を、下記に変更して下さい。
※Tag 名 “X” “Z” は(2-3)の作業を参考に、新規で作成して下さい。
名前: KeyObjX
Tag: X
Image (Script) > Source Image 項目: X



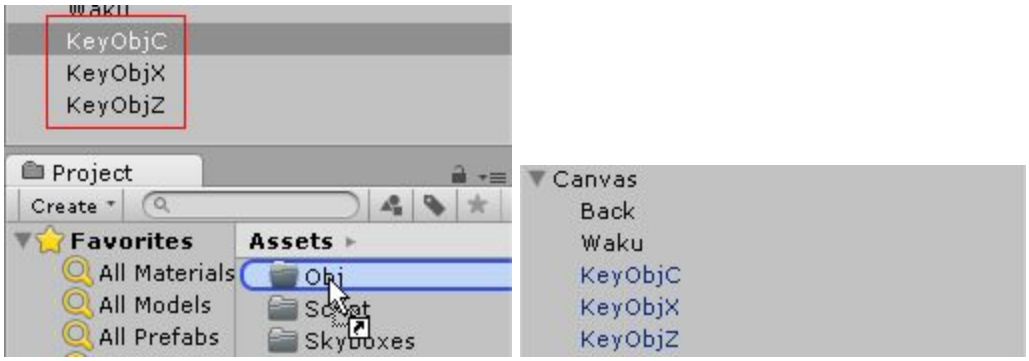
名前: KeyObjZ
Tag: Z
Image (Script) > Source Image 項目: Z



これで、計3種類のキーアイコンが作成出来ました。
※余談ですが、Game ビューを見ると、（奥）C → X → Z（手前） の順で表示されていますが、これは Hierarchy ビューでのオブジェクトの配列順と連動しています。試しに、Canvas オブジェクトを開くと、
▼Canvas
- Back（以下 Canvas の子）
- Waku
- KeyObjC
- KeyObjX
- KeyObjZ

の順で子オブジェクトが並んでいると思いますが、「Back」を「Canvas」内の一番下段に移動させてみて下さい。Game ビューにて「Back」が最前面に来てしまい、他のオブジェクトが見えなくなると思います。この様に、3D の場合だと Image や Text 等の2Dオブジェクトでは、順序が下にいく程手前に位置していく仕様となっています。

2-5)キーアイコンの作成が完了したので、これらをプレハブ化します。
まず、Project ビュー > Assets 内で右クリックし、Create > Folder を選択。名前を「Obj」として下さい。
そして、Hierarchy ビューの「KeyObjC」「KeyObjX」「KeyObjZ」の3つをそれぞれ、
Project ビュー > Assets > Obj フォルダの中にドラッグ&ドロップしていきます。



Hierarchy ビューの各 Image の名前が青色に変わったら、プレハブ化は成功です。

※プレハブ化する事で、プログラム処理にて該当するオブジェクトを何度も呼び出す事が可能となります。また、シーンをまたいで同じオブジェクトを使用したい場合にも、プレハブ化しておくと容易に呼出せます。
プレハブ化後は、Hierarchy ビューにある各アイコンは不要となりますので、KeyObjC・KeyObjX・KeyObjZ は削除して下さい。

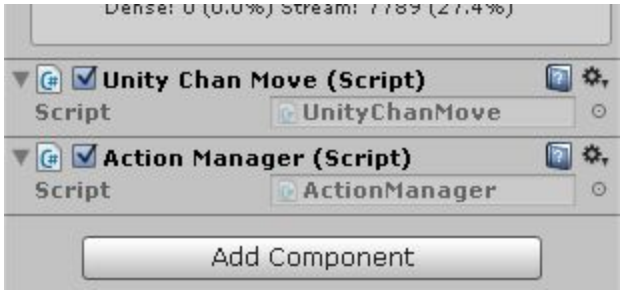
2−6)アイコンキーを自動生成するシステムを作成します。
Hierarchy ビュー > Canvas を選択し、Inspector ビュー > Add Component > Scripts > Kye Obj Manager を選択して下さい。
追加された Key Obj Manage (Script) 内の Waku Obj 項目より Select GameObject ビューを開き、Scene タブ内の Waku を選択します。次に、Key Obj Original 項目を開くとさらに Size 項目が出てくるので、値を 3 に変更します。すると、新たに Element の項目が3つ出現しますが、Select KeyObj ビューからは選択出来ないので、それぞれに各アイコンのプレハブを直接ドラッグ&ドロップして下さい。
Element 0 : KeyObjC（プレハブ）
Element 1 : KeyObjX（プレハブ）
Element 2 : KeyObjZ（プレハブ）
尚、設定の順番は問いません。

ゲームを再生してみて下さい。「C」「X」「Z」のアイコンが右からランダムに出現するのが確認出来ると思います。

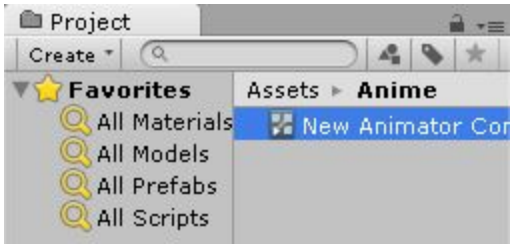
(3)キャラクターの配置と、アニメーション作成の準備

3-1)無料の3D モデルである Unity ちゃん(女の子キャラクター)を画面に配置します。
Project ビュー > UnityChan > Prefabs > unitychan を、Hierarchy ビュー内にドラッグ&ドロップして下さい。
Game ビューの中央に Unity ちゃんが表示されます。もし Unity ちゃんの位置や向きがおかしかったら、Inspectorビュー > Transform 内の各項目を下記の値に修正して下さい。
Position X: 0　Y: 0　Z: 0
Rotation X: 0　Y: -180　Z: 0
Scale　X: 1　Y: 1　Z: 1

3-2) Unity ちゃんがボタンと連動してポーズを取るよう、アニメーションを作成します。
Hierarchy ビュー > unitychan を選択し、Inspector ビュー > Add Component > Scripts から、Unity Chan Move と Action Manager の2つのスクリプト を追加します。



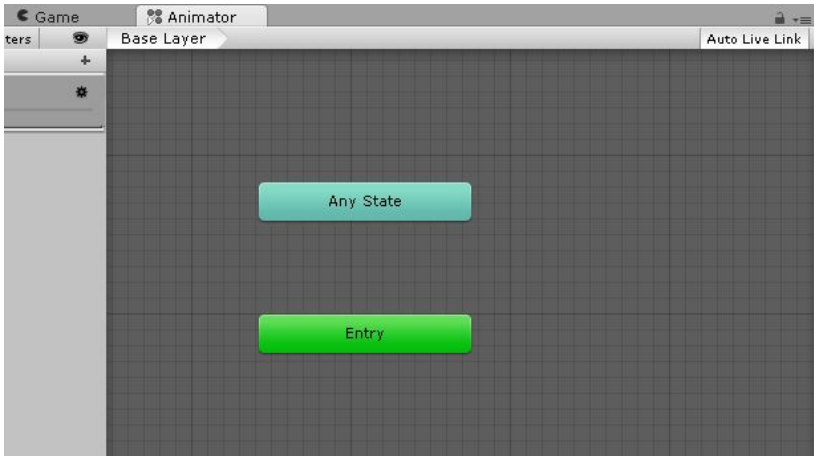
次に、アニメーションシステムを作成します。Project ビュー > Assets 内にて「Anime」フォルダを作成して下さい。
作成したら、Anime フォルダ内で右クリックし、Create > Animator Controller を選択して、新規の Animator Controller を生成します。



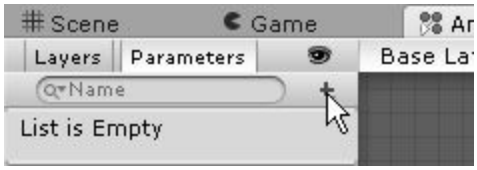
New Animator Controller をダブルクリックし、ここからは実際に各アニメーションの指定と発生条件（アニメーションの遷移）を設定していきます。

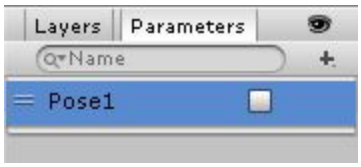
(4)アニメーションの遷移の作成

4-1)ダブルクリックすると、Game ビューの隣りに、新たに Animator ビューが出現します。

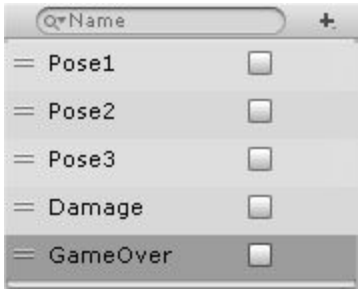


まず、アニメーションに使用するパラメータを準備します。
Animator ビュー > Paramaters タブをクリックすると、「List is Empty」という表示が出てきます。新たにパラメータを追加していくので、Paramaters タブ内の ＋ をクリックし、Bool を選択して下さい。New Bool が新規に生成されますので、名前を「Pose1」に変更します。

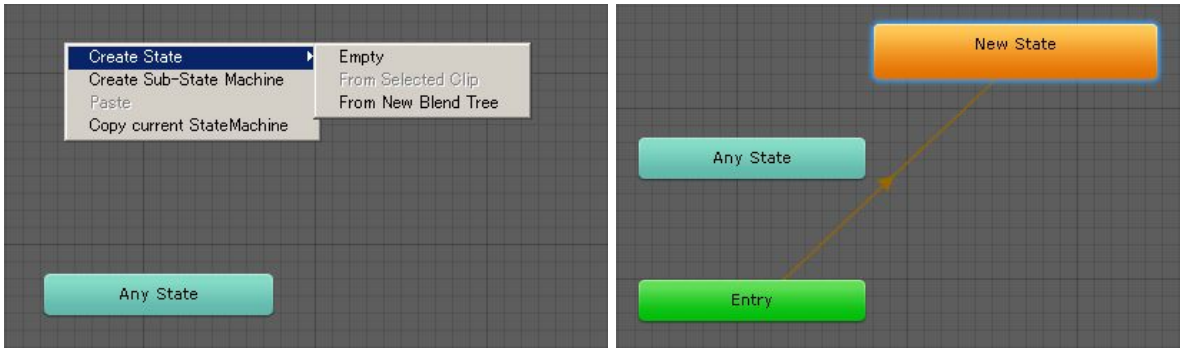




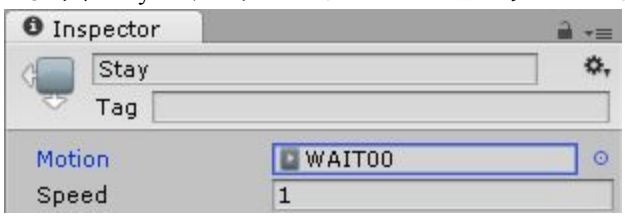
同様の手順で、Pose1 の他に4つの Bool パラメータを生成します。名前はそれぞれ下記の様に変更して下さい。
・Pose1（生成済み）
・Pose2
・Pose3
・Damage
・GameOver
（注）ここは特に1字1句間違えず入力して下さい。



4-2)次に、待機時のモーションを作成します。
Animator ビュー内で右クリックし、Create Status > Empty と選択して下さい。新たに「New State」というステータスが表示されます。



まず、「New State」をクリックし、Inspector ビュー内の一番上にある名前を「Stay」に変更します。続いて、Inspectorビュー > Motion 項目にて Select Motion ビューを開き、「WAIT00」を選択して下さい。これにより、Stay ステータスでは WAIT00 というモーションが適用されます。



4-3)続いて、各キー入力成功時、及びミスやゲームオーバー時のモーションを作成します。
(4-2)ですでに待機時のモーションは作成しましたので、これと同じ要領で、他のモーションも作成していきます。Animator ビュー内で右クリックし、Create Status > Empty と新たにステータスを5つ作成し、State 名、Inspectorビュー > Motion 項目を下記にそれぞれ変更して下さい。

State 名: POSE1
Motion: POSE04

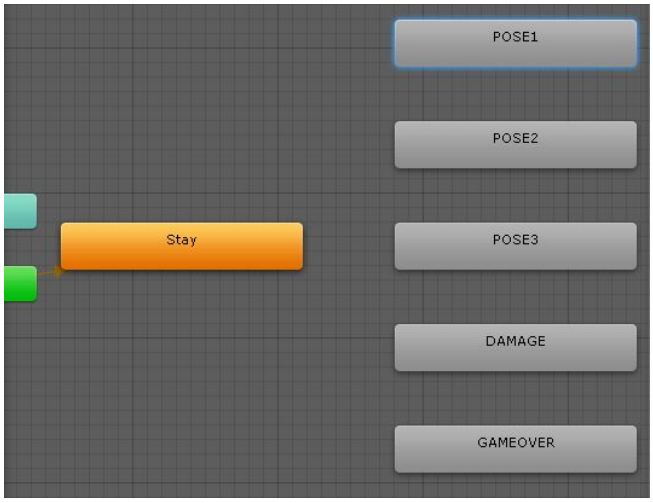
State 名: POSE2
Motion: POSE05

State 名: POSE3
Motion: POSE30

State 名: DAMAGE
Motion: DAMAGE00

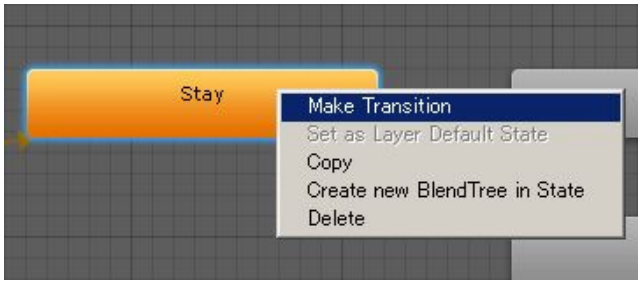
State 名: GAMEOVER
Motion: LOSE00

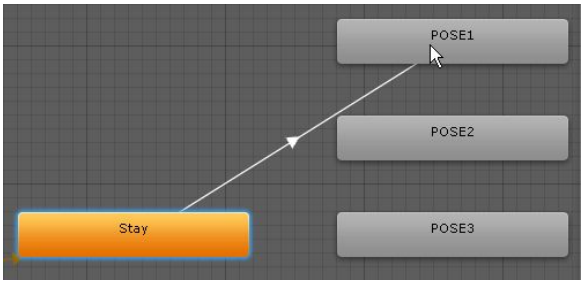
全て作成したら、位置を写真の様に調整しておきます。



4-4)Stay を始めとして、計6つのモーションを作成しましたが、どのモーションをどのタイミングで出すかを設定しないとUnity ちゃんは動きません。よって、最後の作業として、各種モーションの関連付けを行ないます。

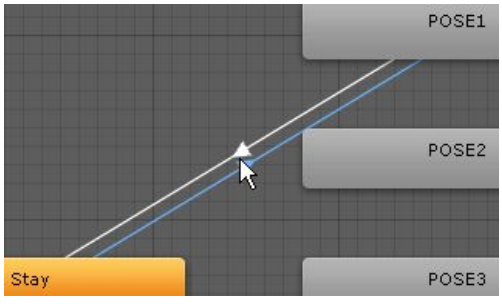
まず、Animator ビュー > Stay ステータスを右クリックし、Make Transition を選択します。その状態でカーソルを動かすと、ステータスから矢印が引っ張り出されてくるので、POSE1 ステータスまで持って行き、クリックして結合させます。





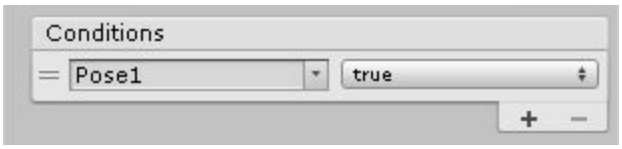
続いて POSE1 ステータスを右クリックし、同じくMake Transition を選択して、State ステータスに結合させます。

ここからは、モーションを移行する条件を設定します。Stay から POSE1 に向かう矢印の線をクリックして下さい。矢印が水色に変われば、選択している状態です。

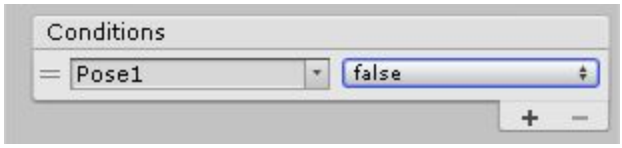


選択したら、Inspector ビュー > Has Exit Time 項目のチェックを外して下さい。また、Inspector ビュー > Conditions 項目が List is Empty となっていますので、右下の + ボタンをクリックして List を作成し、Bool 欄（初期では New Bool ） より Pose1 を選ぶと共に、右側を true にして下さい。
※この設定をする事で、Stay の状態からスクリプト処理等で 「Pose1 が true になった時に POSE1 のモー

ションに変わる」という指示を出す事が出来ます。



今度は逆のモーション遷移を設定します。Animator ビューより、POSE1 から Stay に向かう矢印の線をクリックして下さい。そして同じく、Inspector ビュー > Has Exit Time 項目のチェックを外し、Conditions 項目の + ボタンを押して List を作成して Pose1 を選び、右の欄は false を選択します。これで Stay と POSE1 の2つのモーションが繋がりました。



※これにより、スクリプト処理でフラグを管理し、フラグの ON（true） or OFF（false） でモーションを遷移出来る様になります。
4-5)残りのステータスについても、Stay ステータスと結び付けていきます。(4-4)と同様の作業で、各ステータスとの結合及び設定を下記の様に行なって下さい。

【POSE2】----------
Stay → POSE2
Has Exit Time : OFF（チェックを外す）
Conditions : Pose2 true

POSE2 → Stay
Has Exit Time : OFF
Conditions : Pose2 false

【POSE3】----------
Stay → POSE3
Has Exit Time : OFF
Conditions : Pose3 true

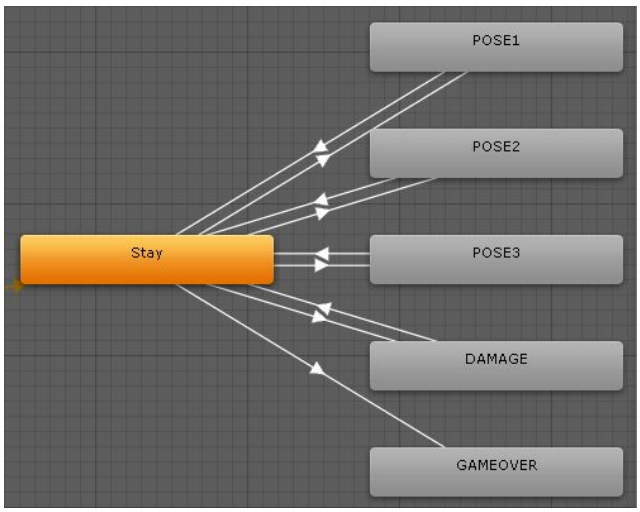
POSE3 → Stay
Has Exit Time : OFF
Conditions : Pose3 false

【DAMAGE】----------
Stay → DAMAGE
Has Exit Time : OFF
Conditions : Damage true

DAMAGE → Stay
Has Exit Time : OFF
Conditions : Damage false

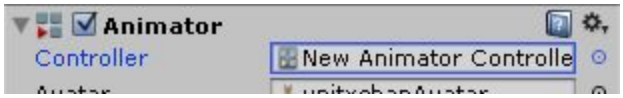
【GAMEOVER】----------　（注）GAMEOVER は一方通行です。

Stay → GAMEOVER
Has Exit Time : OFF
Conditions : GameOver true



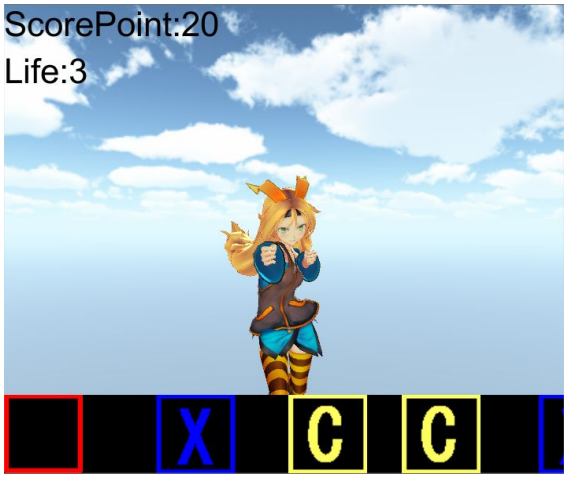
これで全てのアニメーションの設定が完了しました。

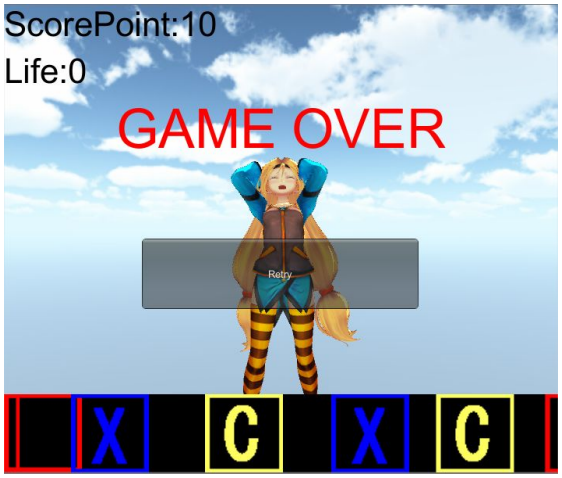
4−6)最後です。Unityちゃんに、作成したアニメーションをアタッチします。
Hierarchyビュー > unitychan を選択し、Inspector ビュー > Animator > Controller 項目より、New Animator Controller を選択します。



ゲームを再生して下さい。Unity ちゃんの普段のポーズが変わります。
C・X・Z アイコンがランダムで出現右下端から出現しますので、左下端の赤枠のエリアに入ったタイミングで、各アイコンに一致したキーボードのキー（C アイコン なら C のキー）を押して下さい。OK と判断され、Unity ちゃんがポーズを決めてくれると同時に、スコアが加算されます。

逆に、アイコンが赤枠内に来る前にキーを押したりアイコンをスルーしてしまうと NG となり、Life が減ると同時にダメージのモーションを取る事が確認出来ると思います。
最終的に Life が 0 になると GAME OVER となり、頭を抱えるモーションを取ります。





上記アニメーションが正しく作動したら完了です。ありがとうございました。