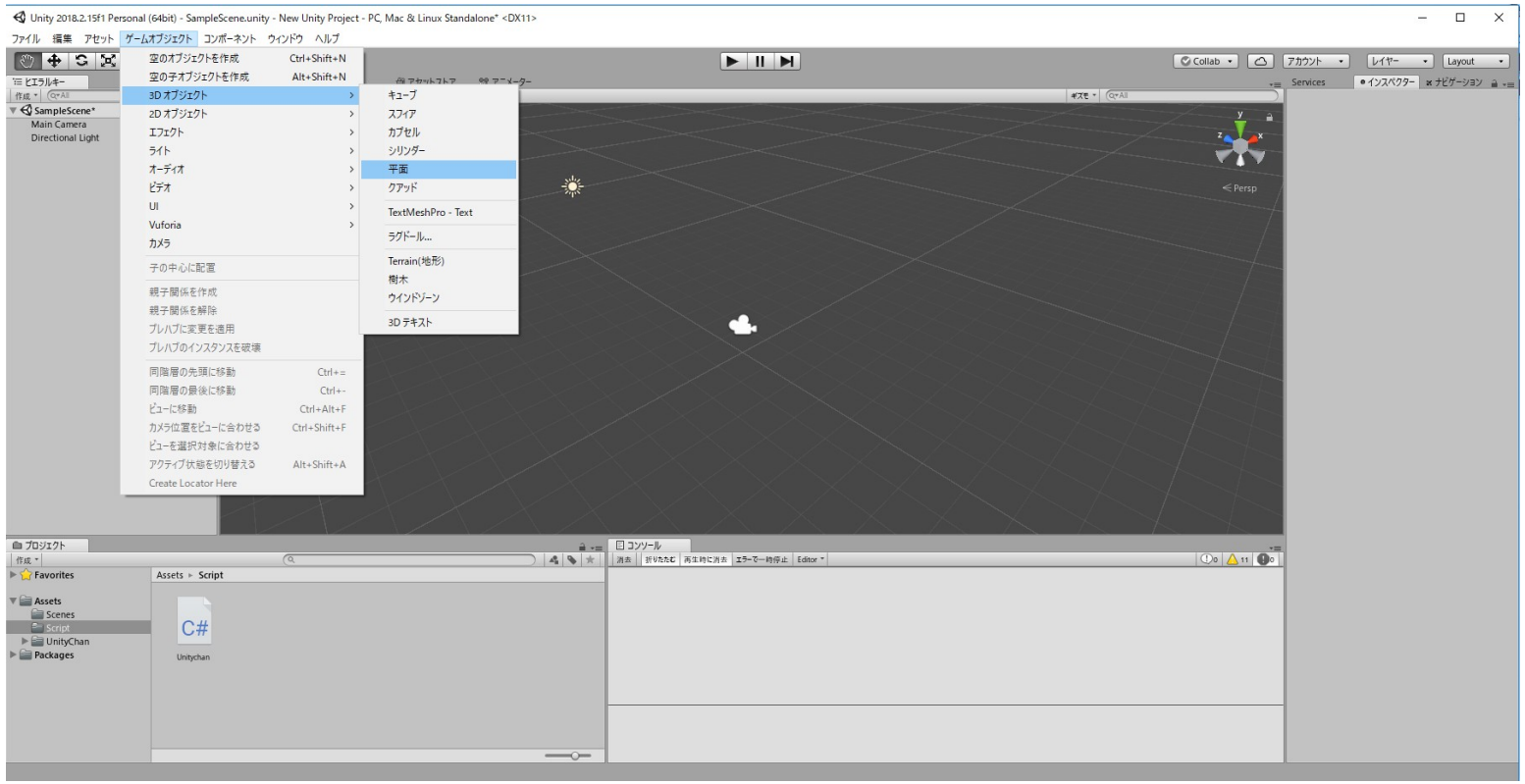
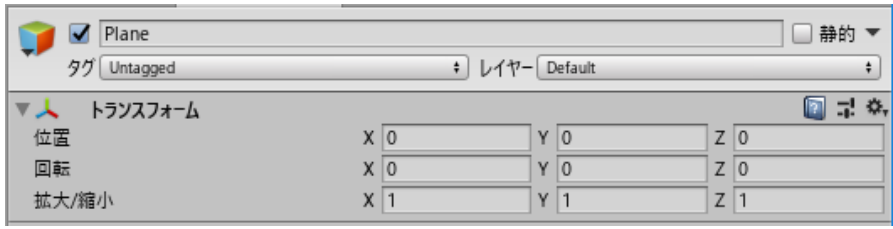


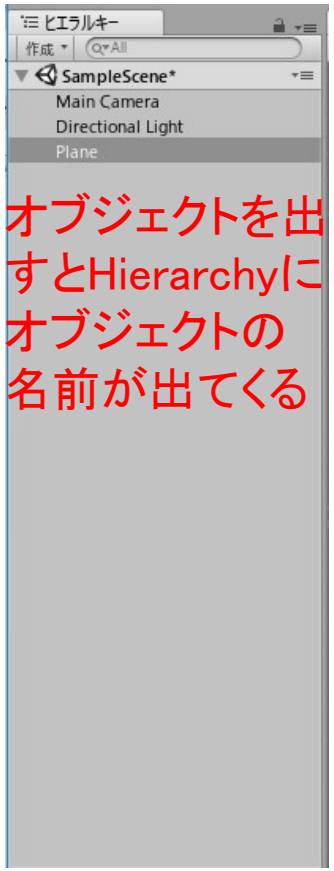
ゲームプロジェクトから
画面にオブジェクトを出していく

左から順に　QWERTYキーを押すと切り替わる

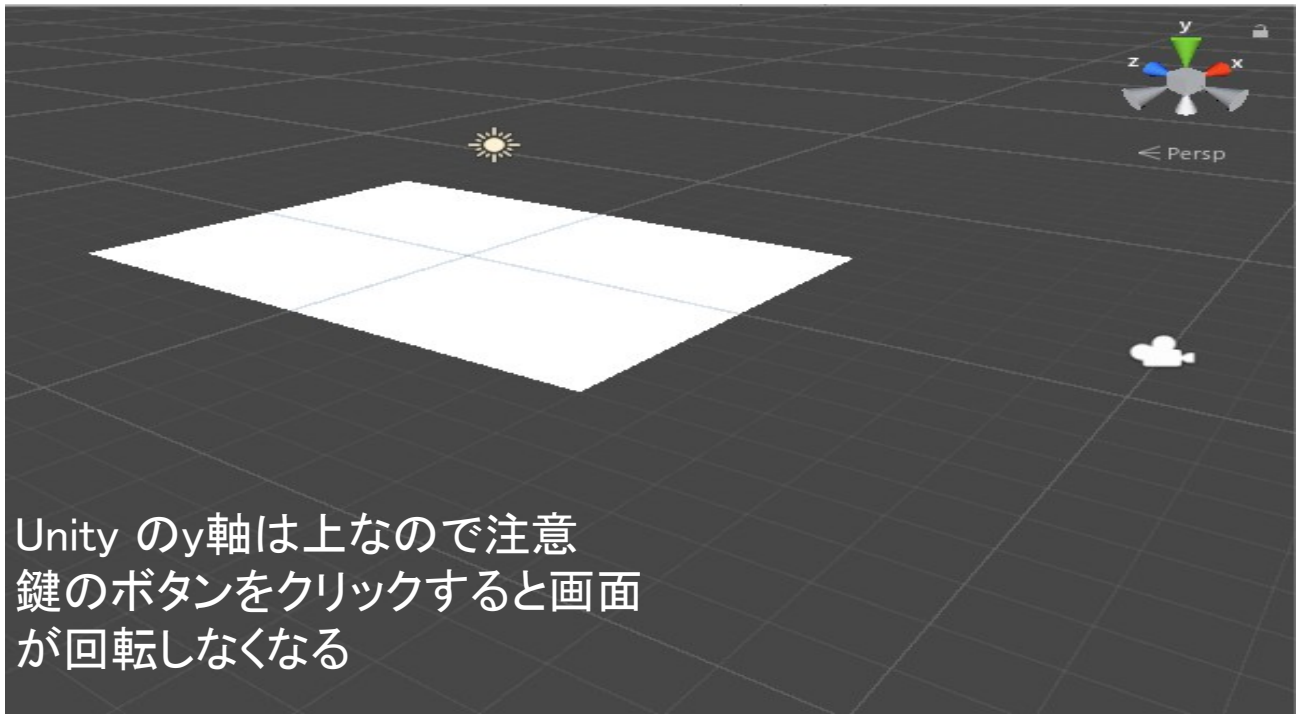
Q　右クリックでスクリーンをつまんで操作できる
W　オブジェクトを選択すると位置を変えることができる
E　オブジェクトを選択すると回転させることができる
R　オブジェクトを選択すると拡大縮小させることができる
T　オブジェクトの頂点を選んで拡大縮小させることができる
Y　WERの機能を一緒に使うことができる



Plane　オブジェクトの名前
トランスフォーム
位置　オブジェクトの置く位置を決める
回転　オブジェクトを回転させる
拡大縮小　オブジェクトを拡大縮小する

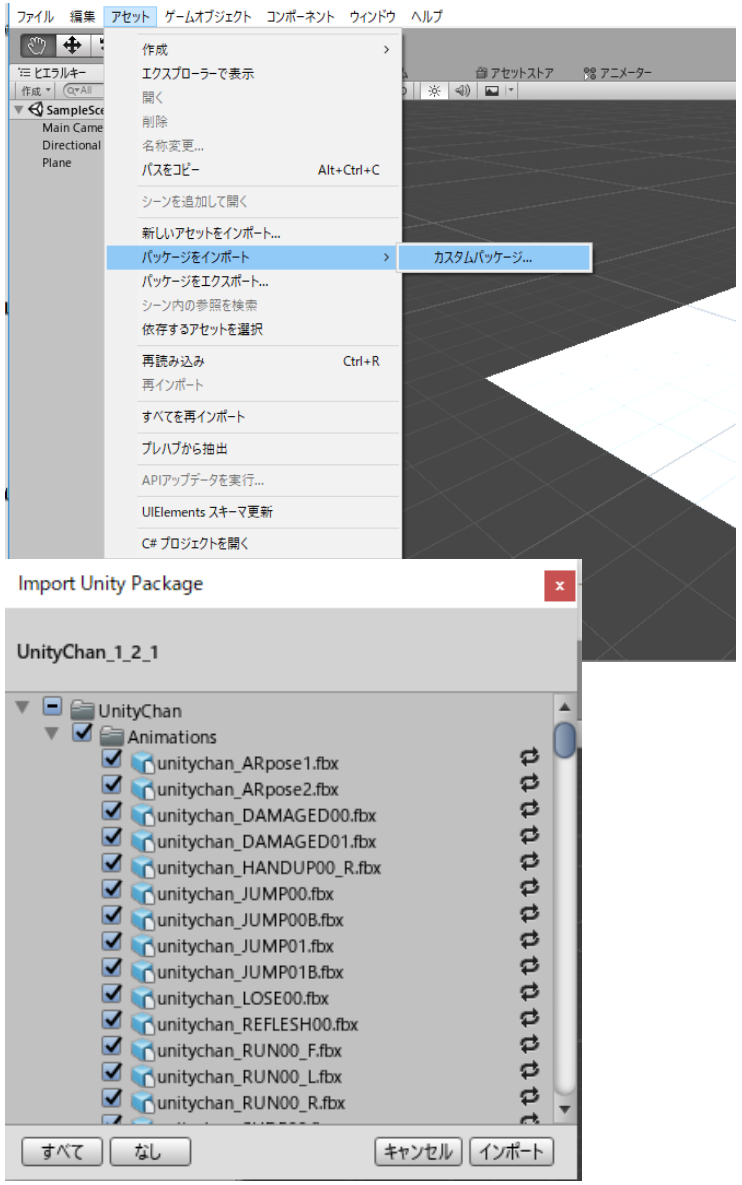


オブジェクトを出すとHierarchyにオブジェクトの名前が出てくる

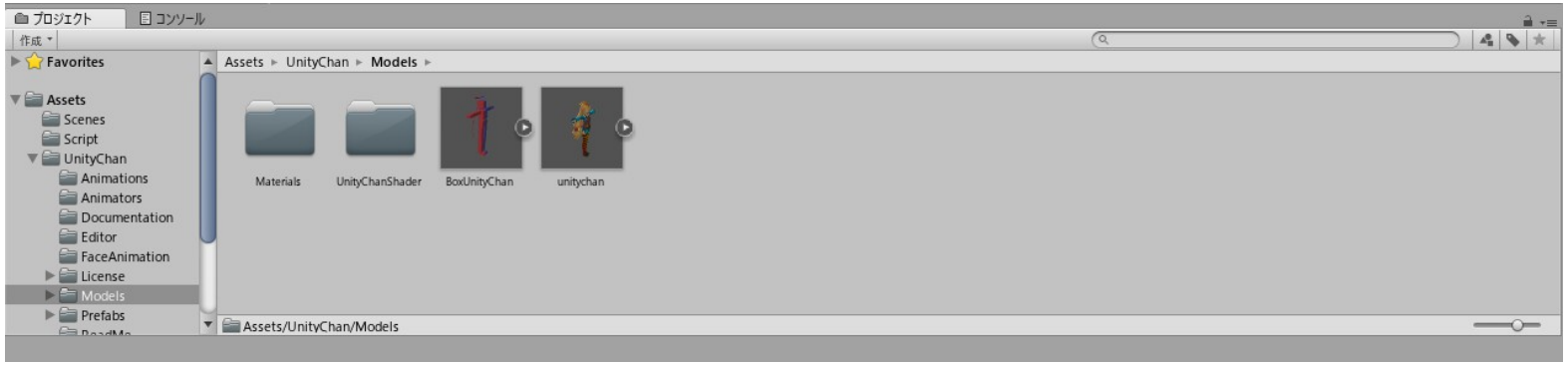


Unity のy軸は上なので注意
鍵のボタンをクリックすると画面が回転しなくなる

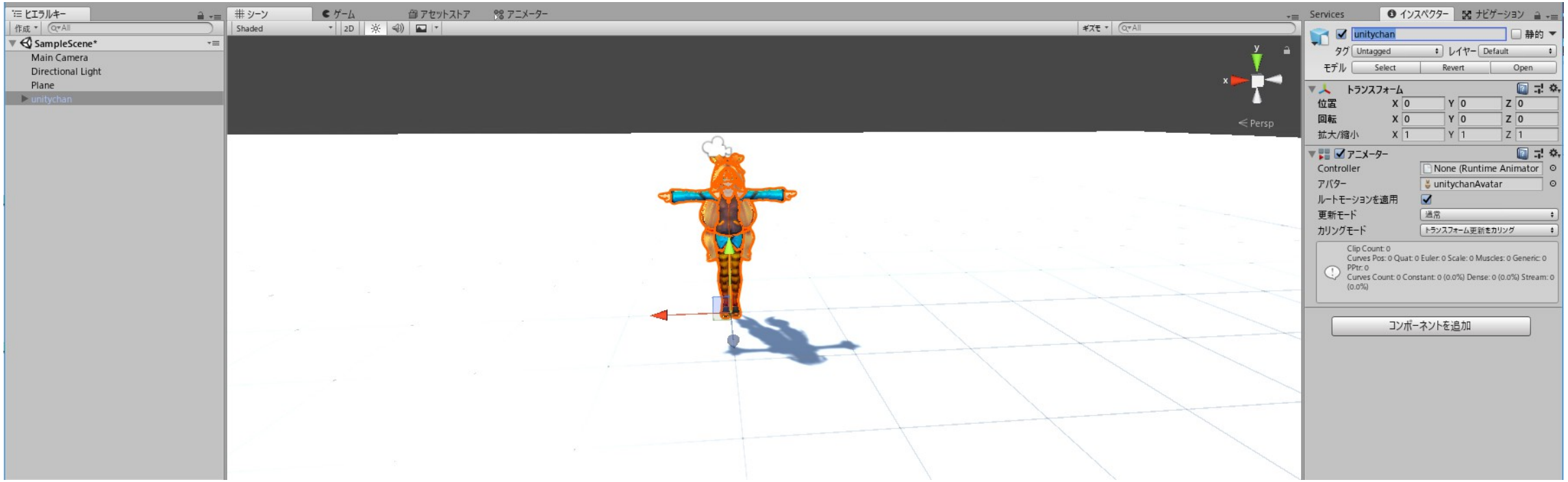
Unityちゃんをスクリーンに出す



・アセット→パッケージをインポート→カスタムパッケージ
ダウンロードしたUnityちゃんを選択する(http://unity-chan.com/)
インポート



下のプロジェクトから
UnityChan→Moders→Unitychanを探す
シーンもしくはヒエラルキーにドラッグ &ドロップ

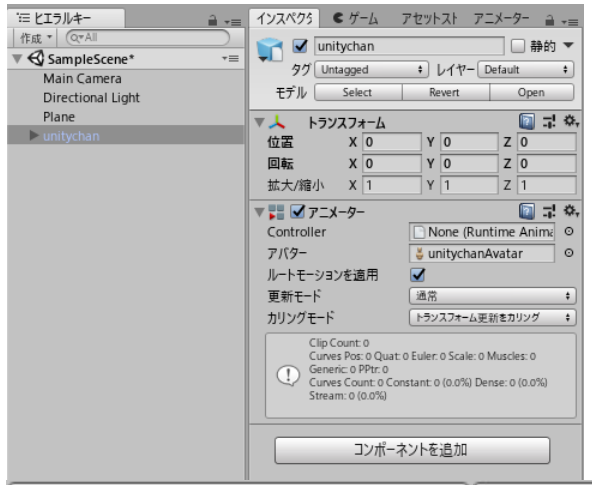


完成（ここまでで理解できること）

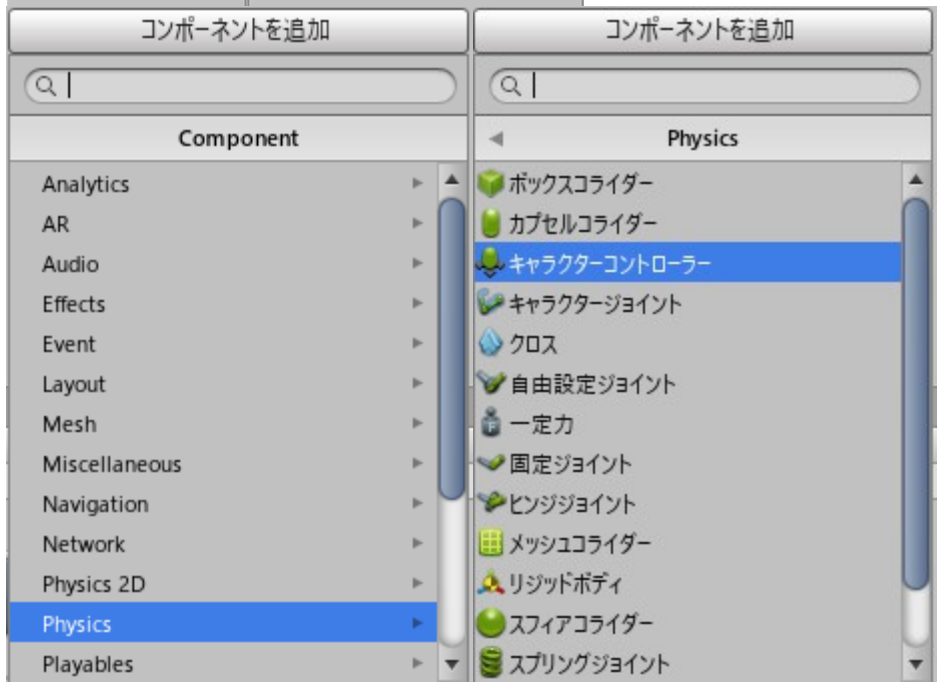
・初期画面の説明

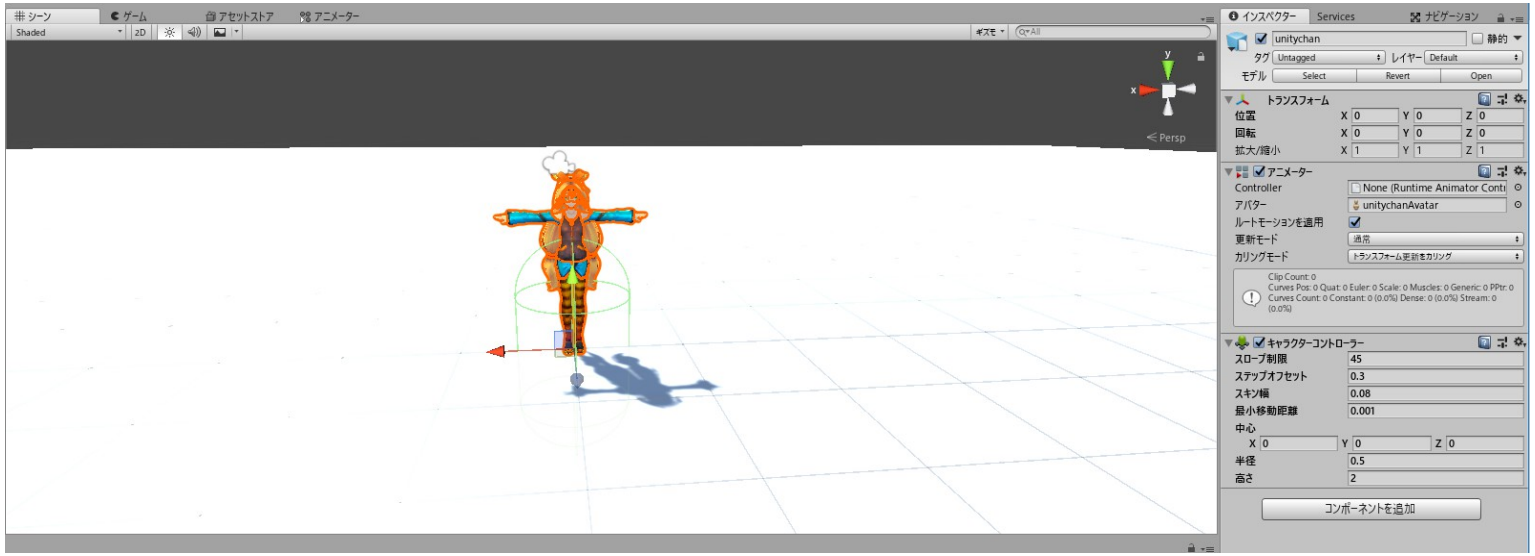
・インポートのやり方

Unityちゃんを動かしてみる



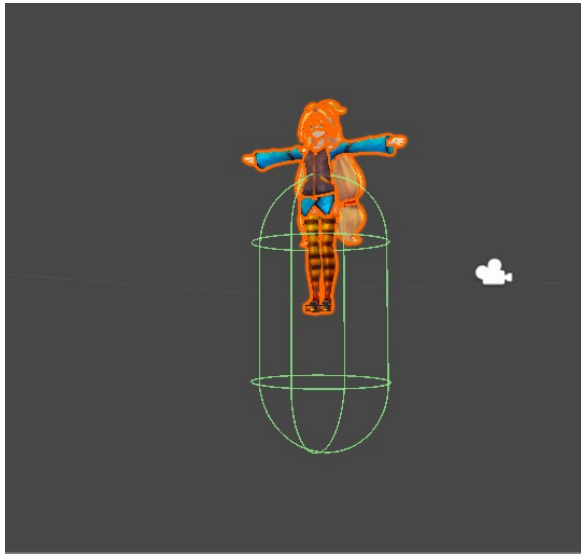
ヒエラルキーからunitychanを選択
インスペラクターの下にあるコンポーネントを追加からPhysics(物理) キャラクターコントローラーを追加する

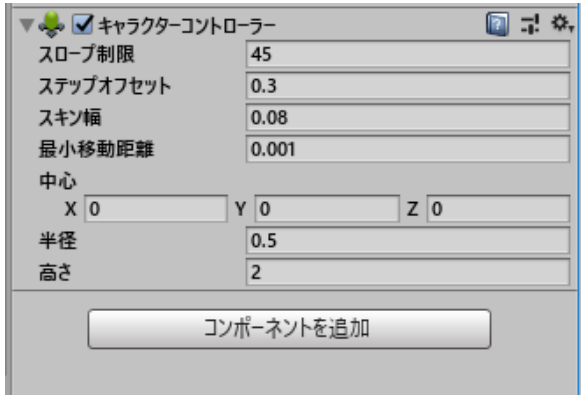




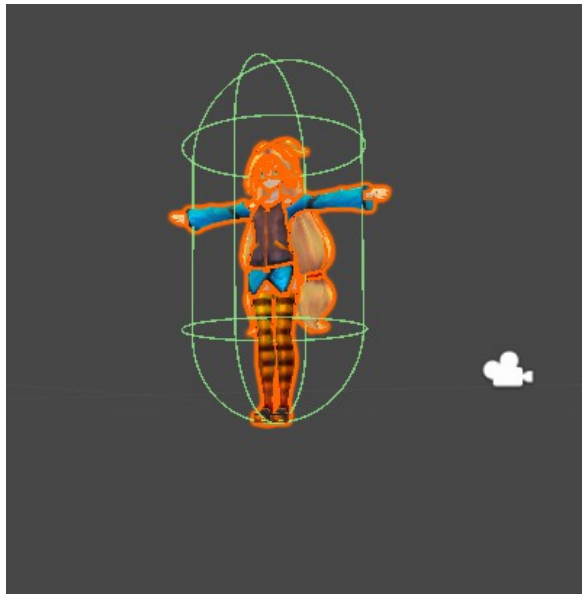
キャラクターコントローラーを追加した所まで

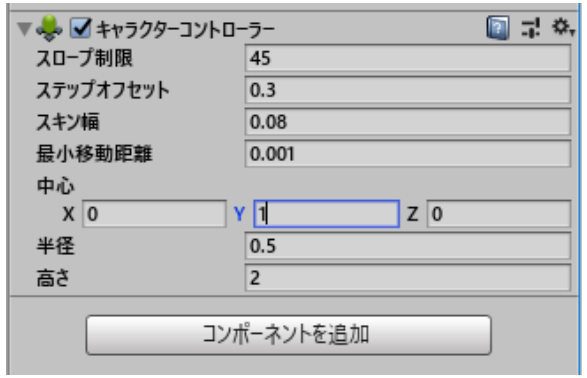
キャラクターコントローラーをUnityちゃんのたかさに合わせる



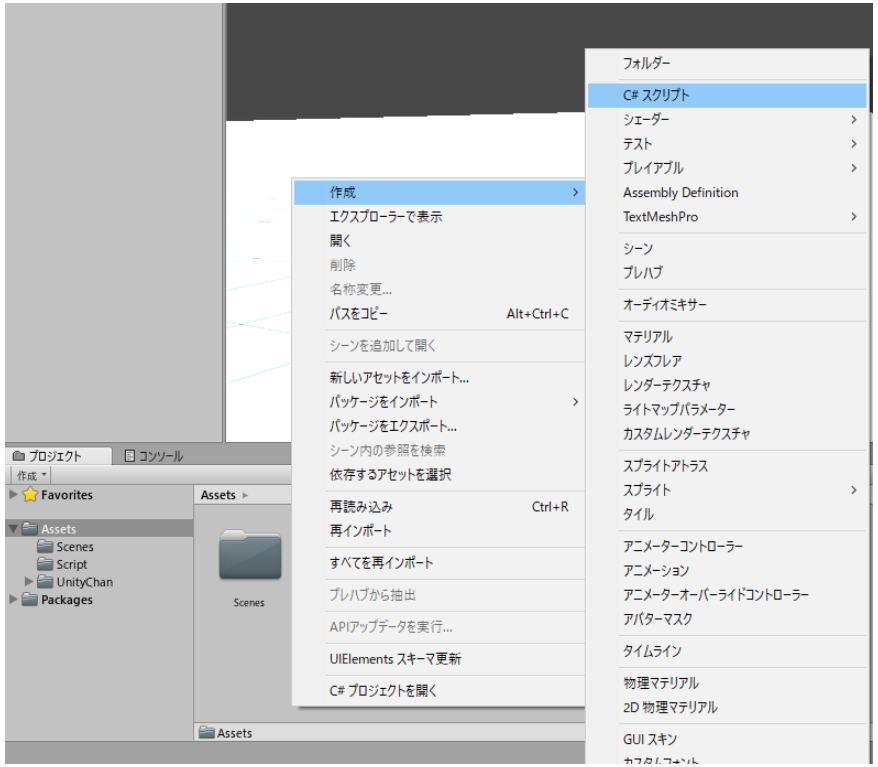


キャラクターコントロールがUnityちゃんのおなか辺りまでしか選択されていないので緑のカプセルがUnityちゃんを覆うようにキャラクターコントロールの中心をいじる

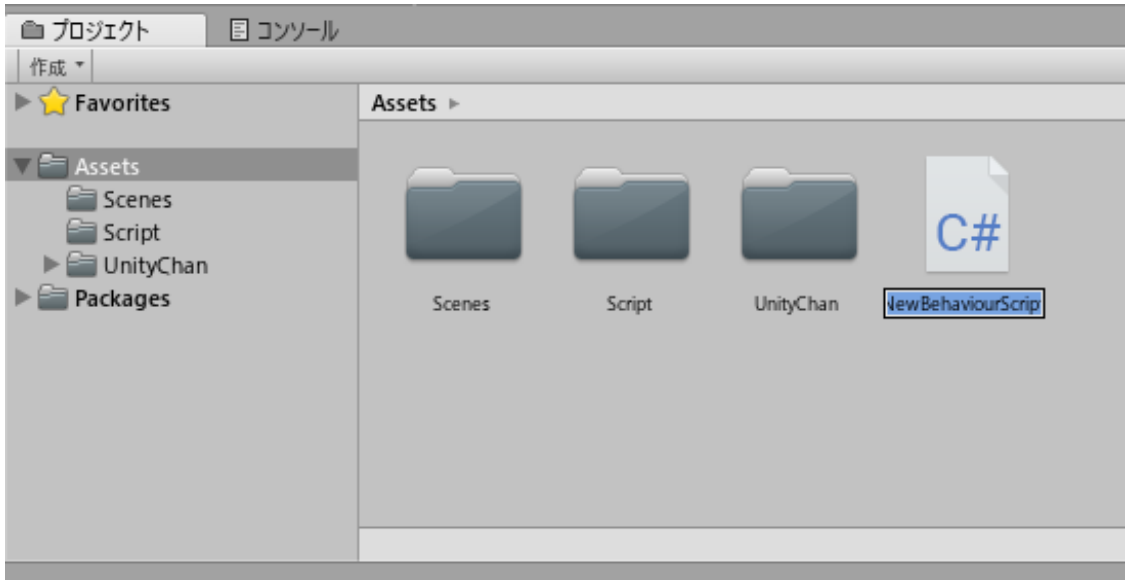




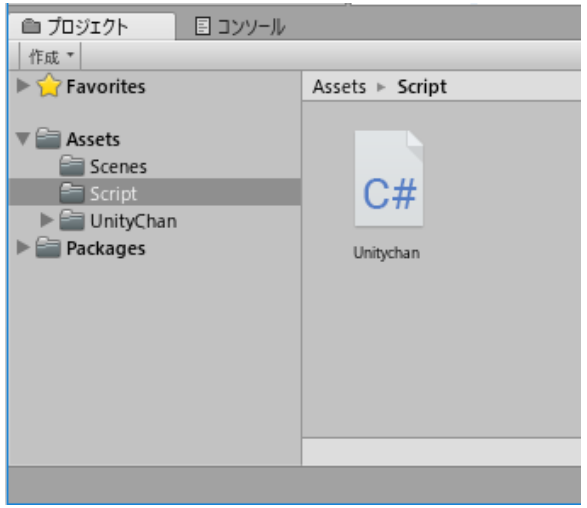
スクリプトを書いてUnityちゃんを動かすための準備



プロジェクト欄を右クリック
作成からC#スクリプトを選択



スクリプトを入れておくためのファイルをあらかじめ作っておくとよい
スクリプトの名前はなるべくわかりやすいものにすること



スクリプトを書く

スクリプトを開いたときに最初に出てくる画面
使用しているエディターによって見た目は異なる(Atomを使用)

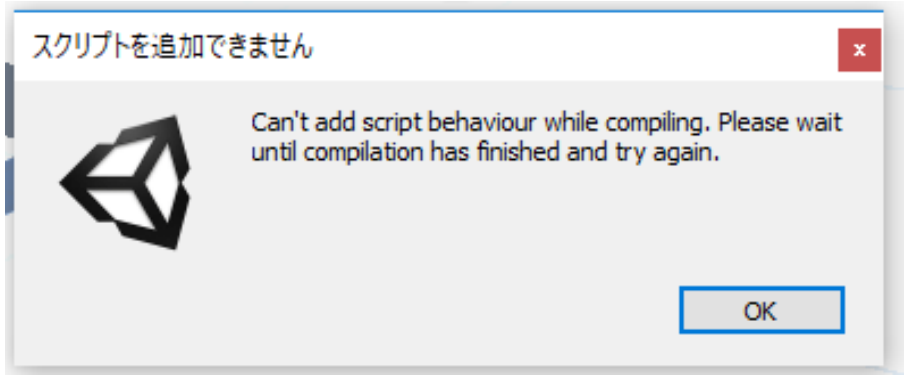
Unitychan.cs

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class Unitychan : MonoBehaviour {
6
7    // Use this for initialization
8    void Start () {
9
10   }
11
12   // Update is called once per frame
13   void Update () {
14
15   }
16 }
17
```

Unityのスクリプトを使用するために
using　UnityEngineが入力されている

```
public class Unitychan : MonoBehaviour
```

Unitychanの部分にスクリプトのシート名が入る
シート名を変更してしまうとエラーしてしまうので
むやみにシート名を変えないこと　変えた場合はすぐに書き直す。



スクリプトを書いた画面

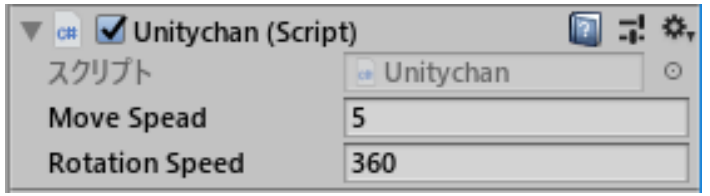
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Unitychan : MonoBehaviour {

  public float moveSpead = 5f;//歩くスピード
  public float rotationSpeed = 360f;//振り向くスピード

  CharacterController characterController;

  // Use this for initialization
  void Start () {
    characterController  = GetComponent<CharacterController>();
  }

  // Update is called once per frame
  void Update () {
    Vector3 direction = new Vector3(Input.GetAxis("Horizontal"),0,Input.GetAxis("Vertical"));
    if(direction.sqrMagnitude > 0.01f){
      Vector3 forward = Vector3.Slerp(
      transform.forward,
      direction,
      rotationSpeed*Time.deltaTime/Vector3.Angle(transform.forward,direction)
      );
      transform.LookAt(transform.position + forward);
    }
    characterController.Move(direction*moveSpead*Time.deltaTime);
  }
}
```

これと全く同じことを書けば
Unityちゃんが無事動いてくれます。
(public classは
自分のシート名にするように)

次のシートで自分が知る限りの説明をさせていただく

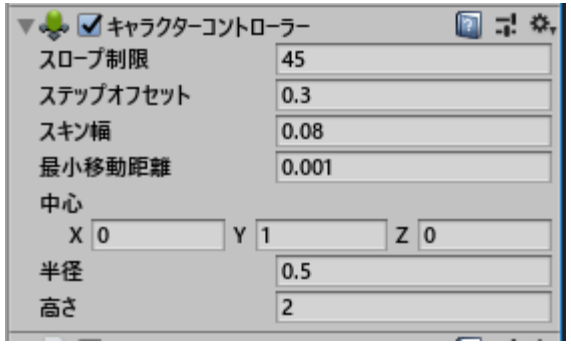
スクリプト説明

```
7     public float moveSpead = 5f;//歩くスピード
8     public float rotationSpeed = 360f;//振り向くスピード
```



コメントの通り動くスピードと振り向くスピードを変えることができる。
インスペクター欄でこの数値をいじることができるようになる。

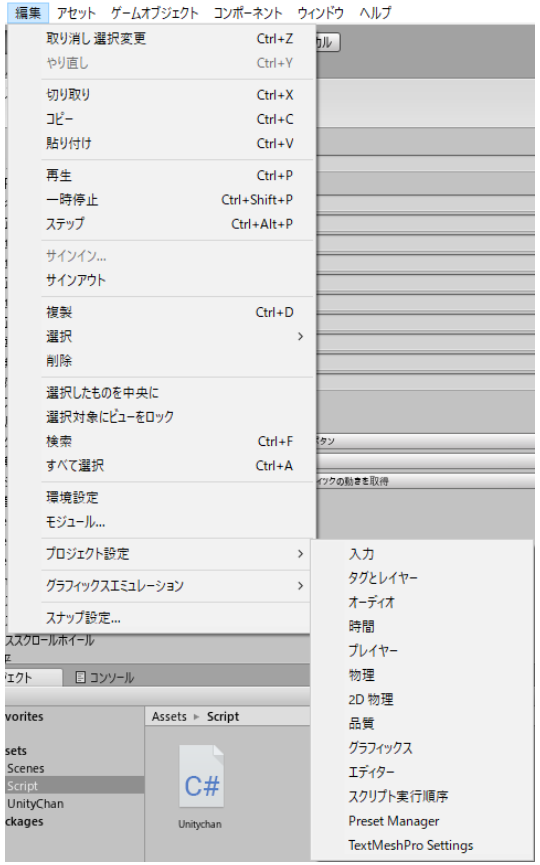
```
14        characterController  = GetComponent<CharacterController>();
15    }
```



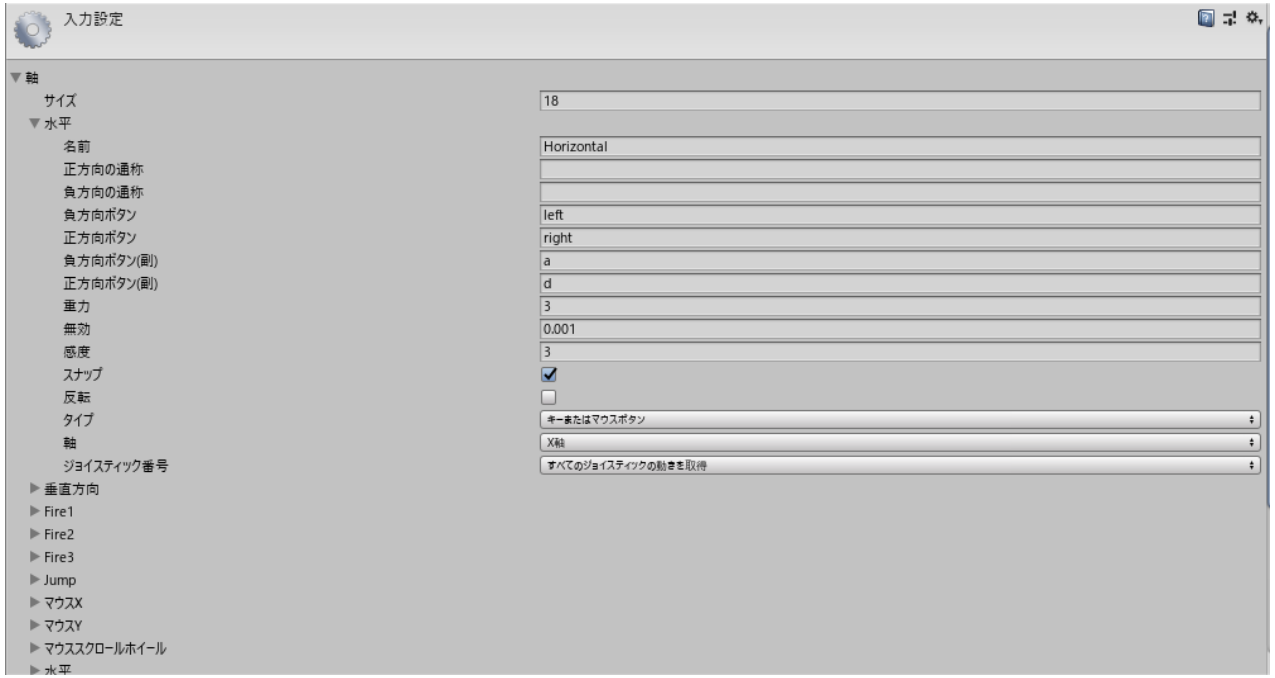
先ほどUnityちゃんと同じ高さに合わせた緑のカプセルのこと
スクリプト内で命令させるために書く
他のコンポーネントを追加する際にも
GetComponent<コンポーネント名>を書く

```
19        Vector3 direction = new Vector3(Input.GetAxis("Horizontal"),0,Input.GetAxis("Vertical"));
```

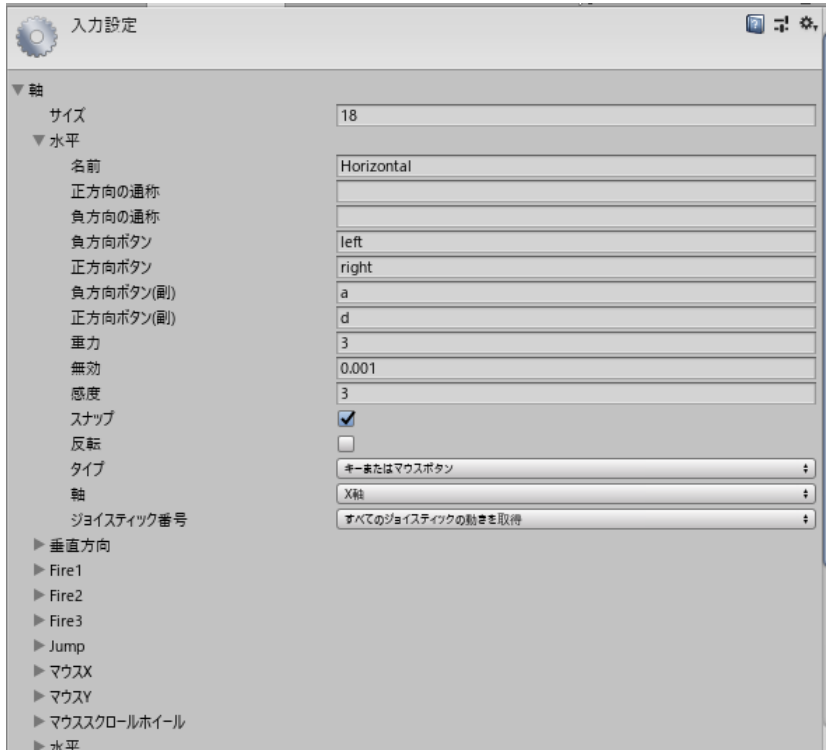
Vector(xyz)の移動の仕方を決める
Input.GetAxis（"Horizontal"）
Horizontal　horizontalキーを押したときにX方向に動くように設定されている
0　Y軸には何も設定されていないので動かすことができない。
Vertical　Verticalキーを押したときにZ方向に動くように設定されている



Horizontal　Verticalキー
キーボードに上記二つの
キーなどないので何を押
せばいいのか。
編集からプロジェクト設定
入力を選択



インスペクターに入力
設定という画面が出て
くる



水平の場所を見てみると
名前　Horizontal
負の方向正の方向を見ると　left　right　a　dと書いてある
つまり、デフォルトの設定では、←、aを押せば左に→,dを押せば右に移動することができる。この部分を変えれば自分で移動したいキーを設定することも可能。
Horizontalの名前を変えてしまうと、Input.GetAxisに書いたものを変えなくてはいけないので注意

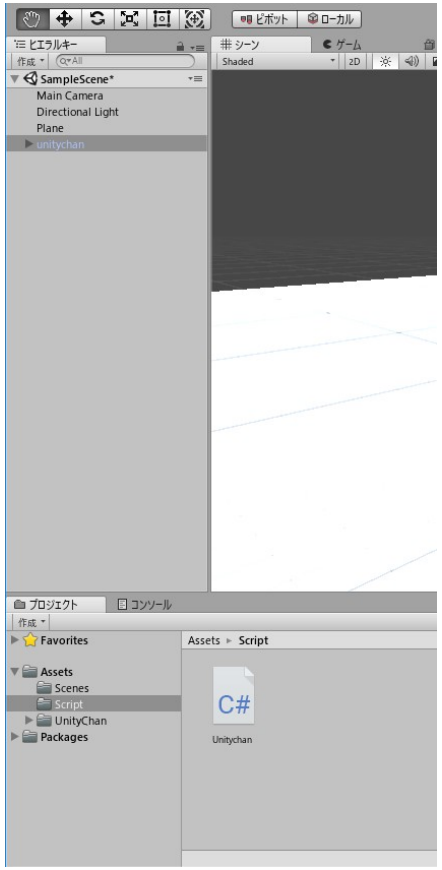
垂直方向を見ればVerticalの設定を見ることができる。

19行目以降は説明できるようになったら説明
書いてあることは振り向きについて

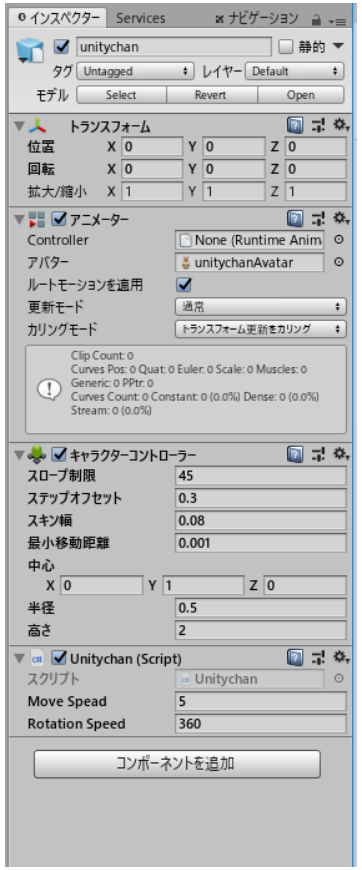
```
22      if(direction.sqrMagnitude > 0.01f){
23        Vector3 forward = Vector3.Slerp(
24        transform.forward,
25        direction,
26        rotationSpeed*Time.deltaTime/Vector3.Angle(transform.forward,direction)
27        );
28        transform.LookAt(transform.position + forward);
29      }
30      characterController.Move(direction*moveSpead*Time.deltaTime);
31      animator.SetFloat("Speed",characterController.velocity.magnitude);
32    }
33  }
34
```

direction.sqrMagnitude　0.01f　方向を変えるための命令
Vector3.Slerpで円を描くように方向を変えることができる。
Slerpの部分を変えると直覚に曲がることもできるので詳しくは調べたし
If下4行の計算式によって方向を計算して
Transform.LookAt()によってUnityちゃんが向きを変える。
いろいろとっかえひっかえすると進行方向とは逆のほうに向きながら歩くこともできる。

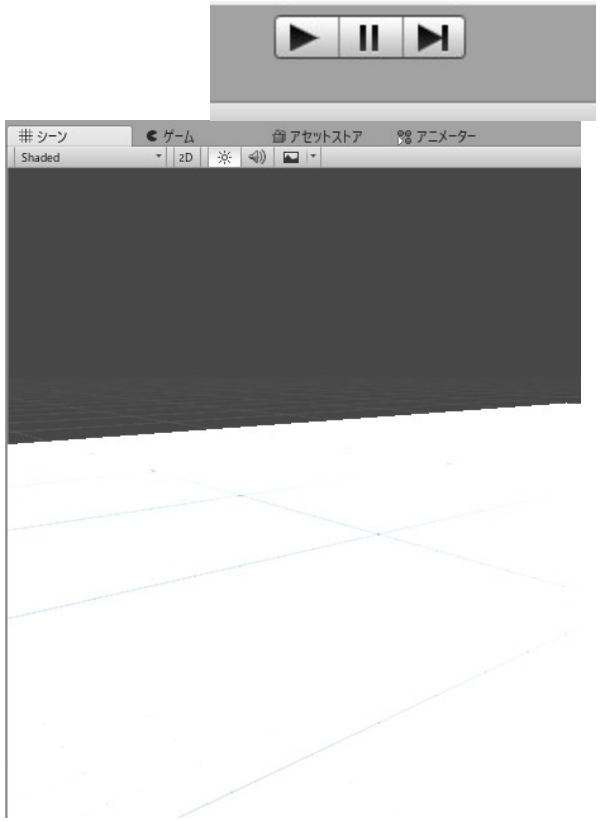
(不完全)修正必



書いたスクリプトを
ヒエラルキーのUnitychanに
ドラック＆ドロップ

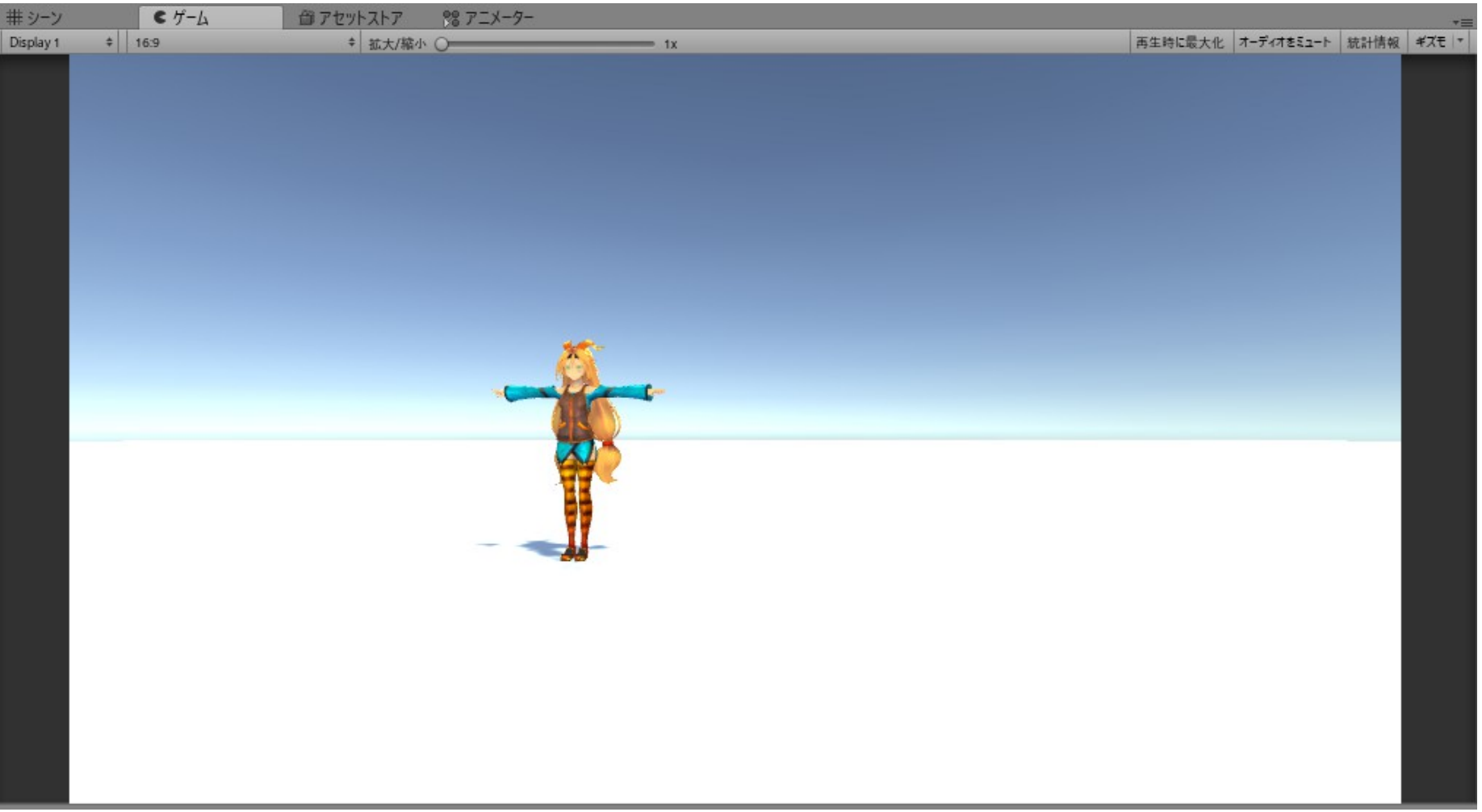


インスペクター欄に先ほど
書いたスクリプトが追加され
ている



シーン画面からゲームを選
び、再生ボタンを押す。

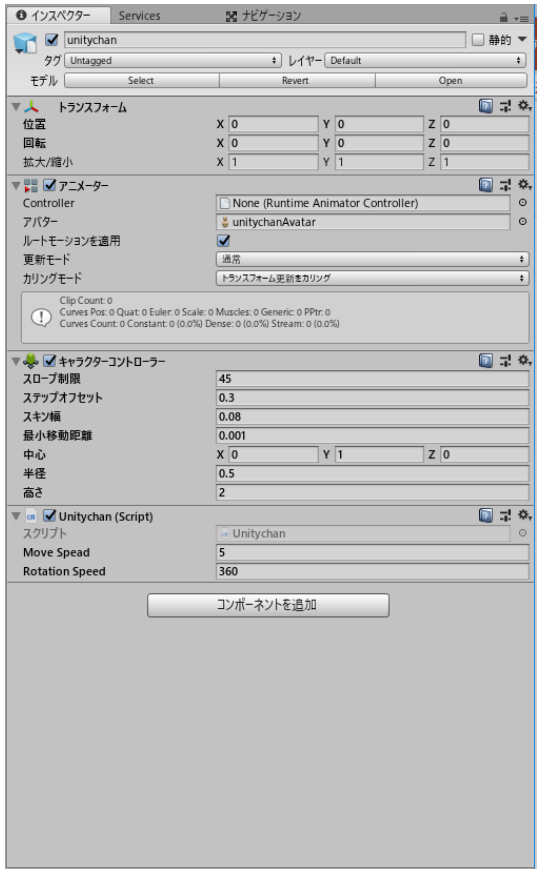
キーボードの上下左右ボタ
ンもしくはWASDボタンで
Unitychanを動かすことがで
きる。



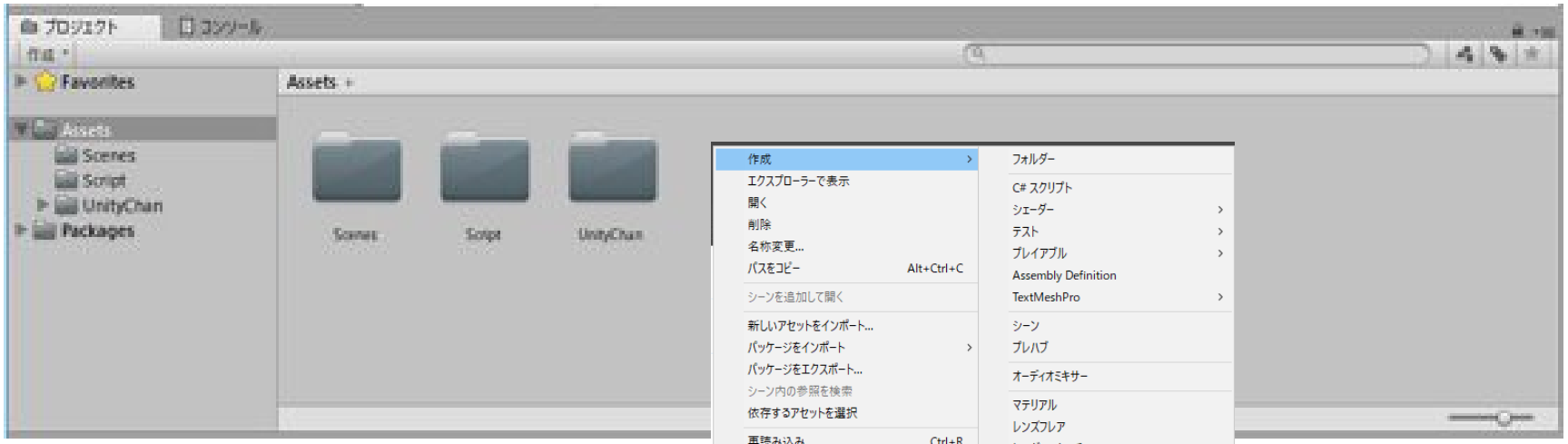
完成
(今回理解したこと)
・スクリプトの書き方
・スクリプトをオブジェクトに読ませる方法
・Input.GetAxisの使い方
・Unityちゃんが動くけどUnityちゃんの体は動かないこと

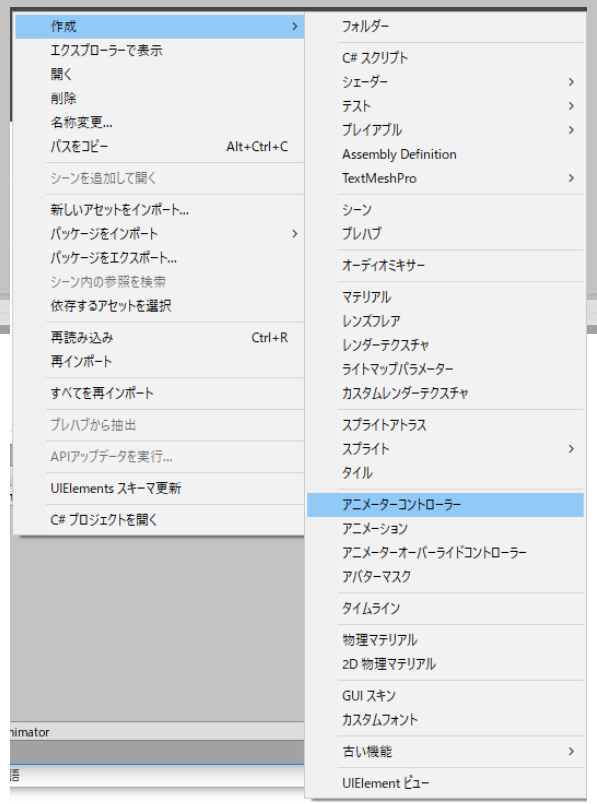
キャラクターを動かす方法はこのほかにもいろいろとあるので調べておくこと

前回のことでオブジェクトにスクリプトを読ませる方法が分かった。
今回はオブジェクトにモーションを付けてみる。



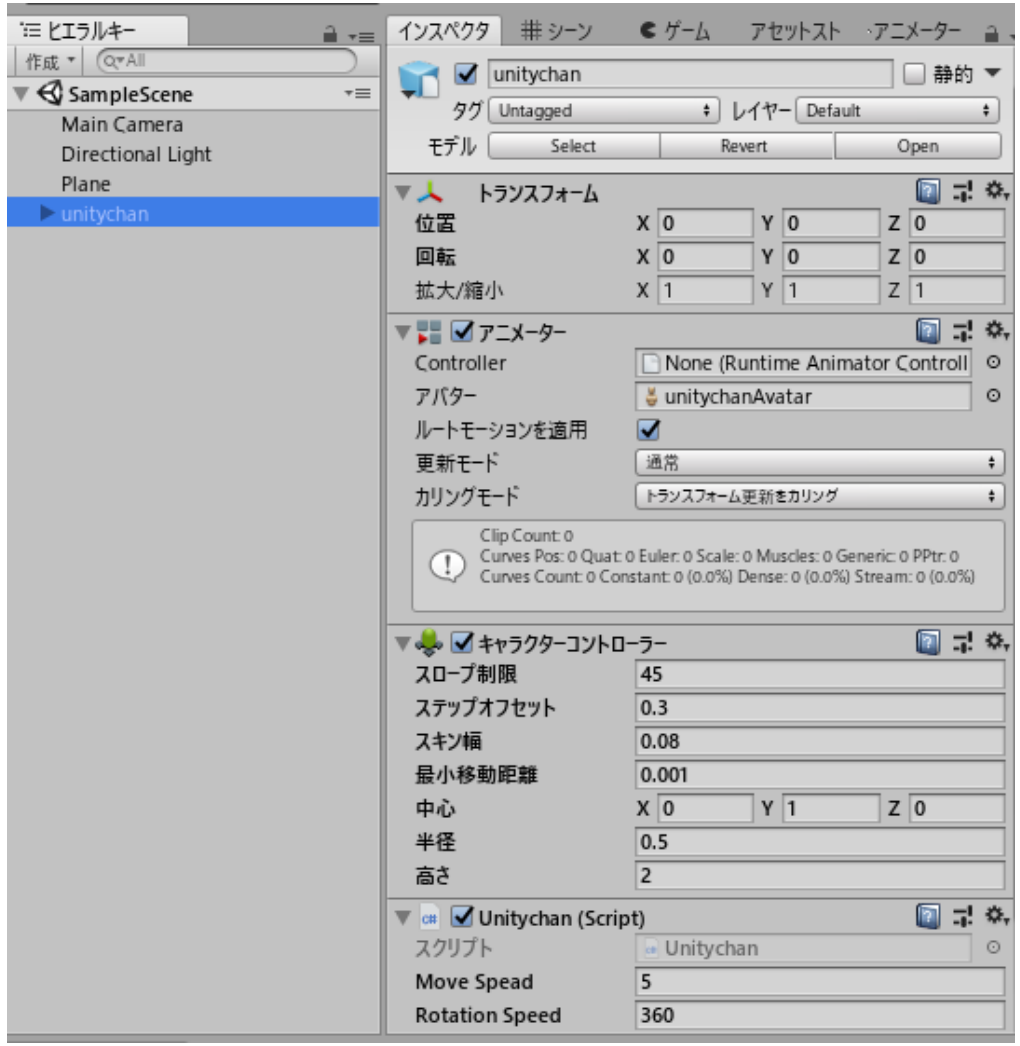
Unityちゃんの前回までのコンポーネントがこちら
今回はアニメーターを使用する。



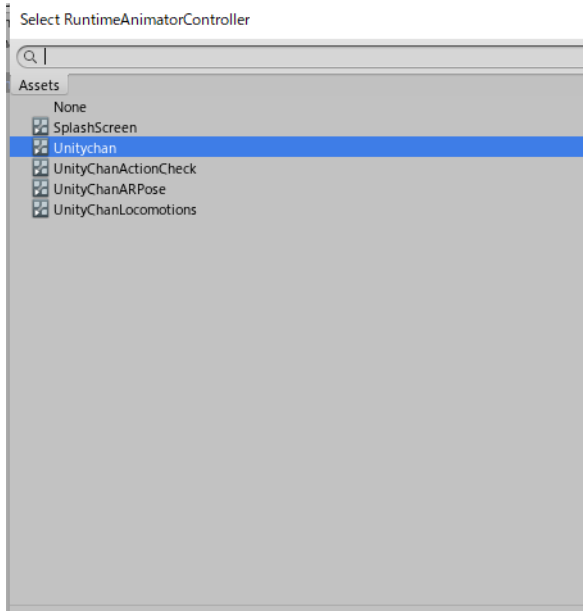


下のプロジェクト欄を右クリック
作成からアニメーターコントローラーを選択

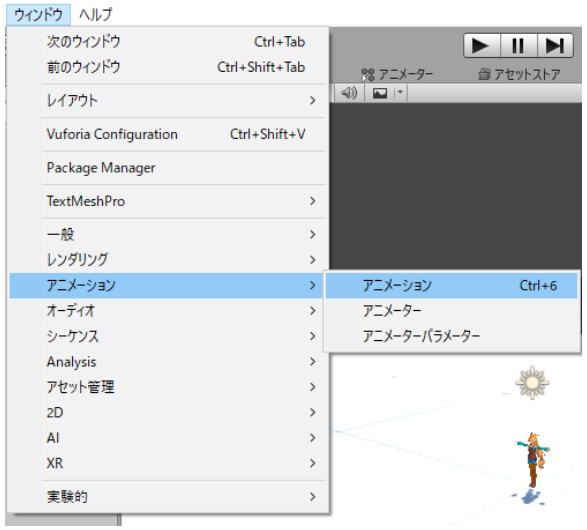
名前はわかりやすいものにすること



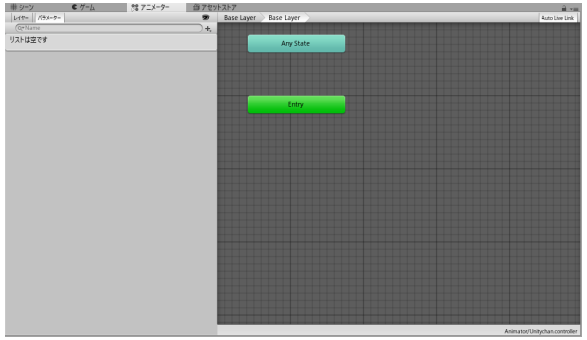
ヒエラルキーからunityちゃんを選択
インスペクタのアニメーター
→Controllerの右にある◎をクリック
すると下の画面が出てくるので先ほ
ど作ったアニメコントローラーを選択
ルートモーションを適用のチェックを
外しておく



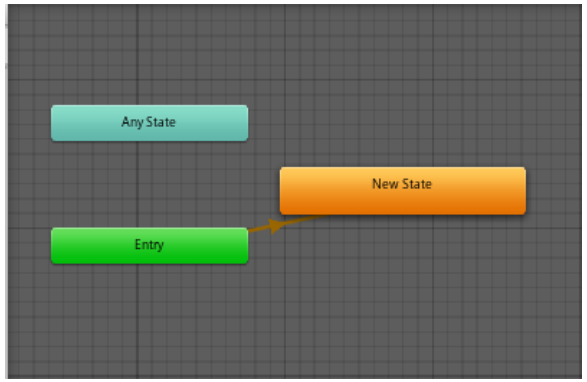
animatorを設定する



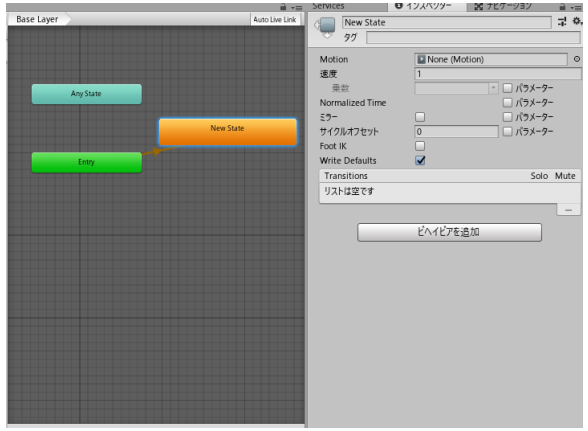
ウインドウアニメーションからアニメーターを選択する



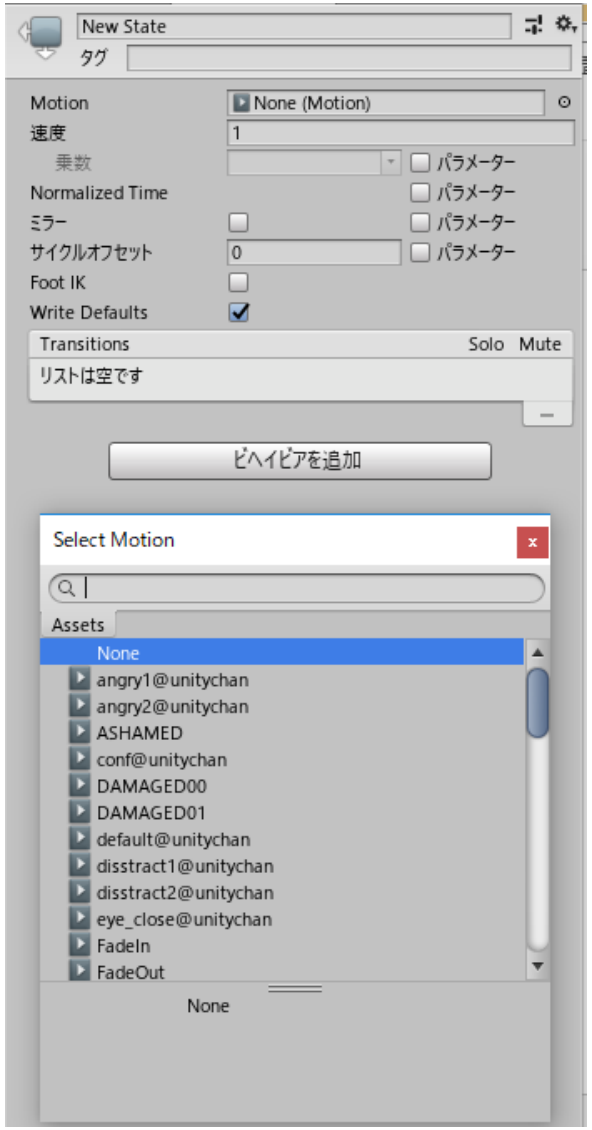
アニメーションビューが作成される
Altを押しながらマウス操作で画面を操作できる



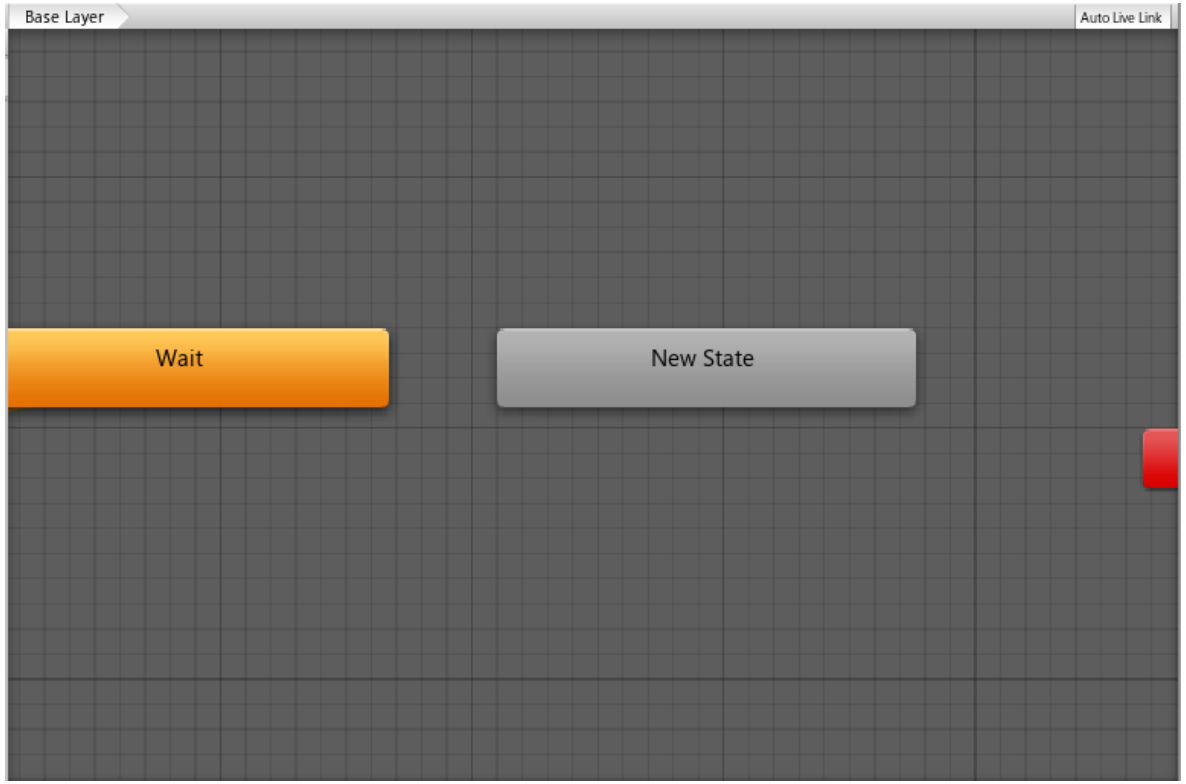
右クリック→ステートの作成→空　でステートを作る



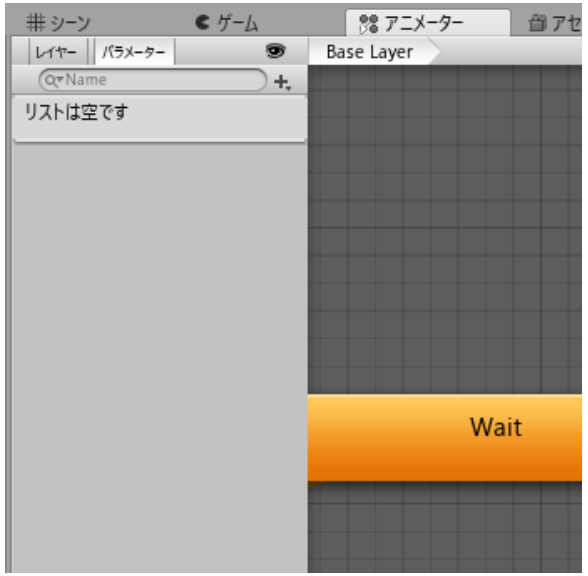
作ったステートをクリックするとインスペクターでいじれるようになる。



Motionの右にある◎をクリックすると
モーションが選べるようになるので、
Wait02を選択する。
ステートの名前はわかりやすいように
変えておく



同じようにステーツを作って次は
Run00_Fを選択してみる

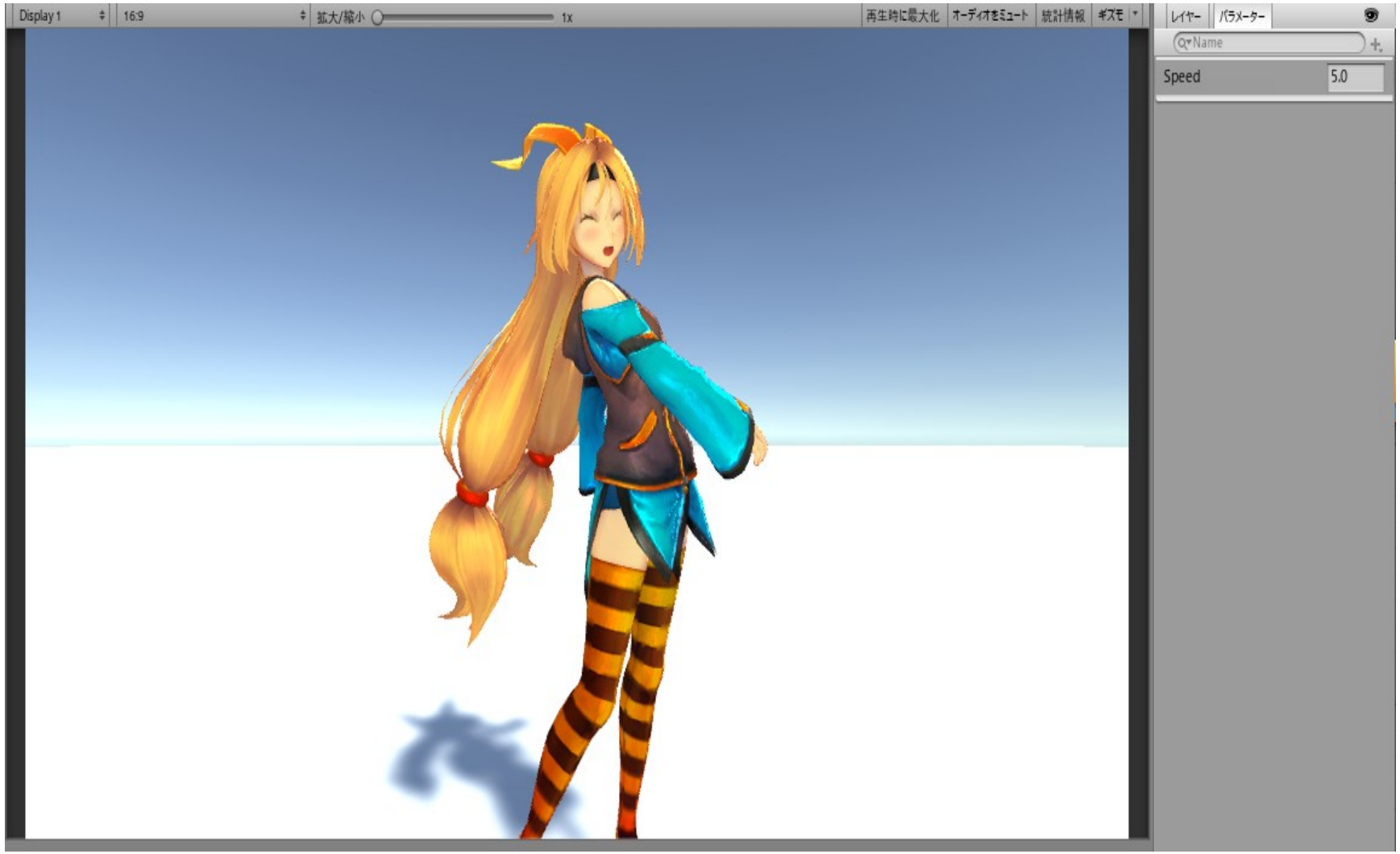


パラメーターを選択して右側にある
+ボタンからFloatを選択してSpeedという名前を付ける

スクリプトを編集する

```
7     public float moveSpead = 5f;//歩くスピード
8     public float rotationSpeed = 360f;//振り向くスピード
9
10    CharacterController characterController;
11    Animator animator;
12
13    // Use this for initialization
14    void Start () {
15      characterController  = GetComponent<CharacterController>();
16      animator =GetComponentInChildren<Animator>();
17    }
18
19    // Update is called once per frame
20    void Update () {
21      Vector3 direction = new Vector3(Input.GetAxis("Horizontal"),Input.GetA
22      if(direction.sqrMagnitude > 0.01f){
23        Vector3 forward = Vector3.Slerp(
24        transform.forward,
25        direction,
26        rotationSpeed*Time.deltaTime/Vector3.Angle(transform.forward,directi
27        );
28        transform.LookAt(transform.position + forward);
29      }
30      characterController.Move(direction*moveSpead*Time.deltaTime);
31      animator.SetFloat("Speed",characterController.velocity.magnitude);
32    }
33  }
34
```

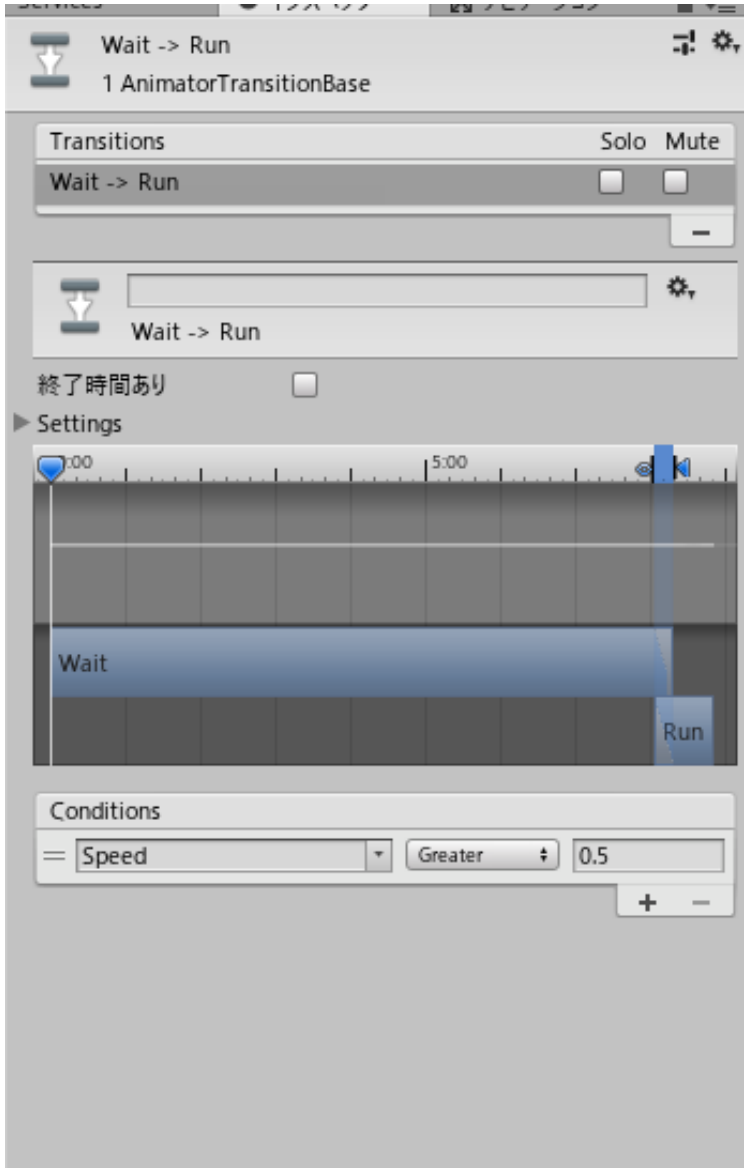
11,16,31の3行を追加する
前回の説明と同じようにコンポーネントを読み込むために16行目を書き加える。



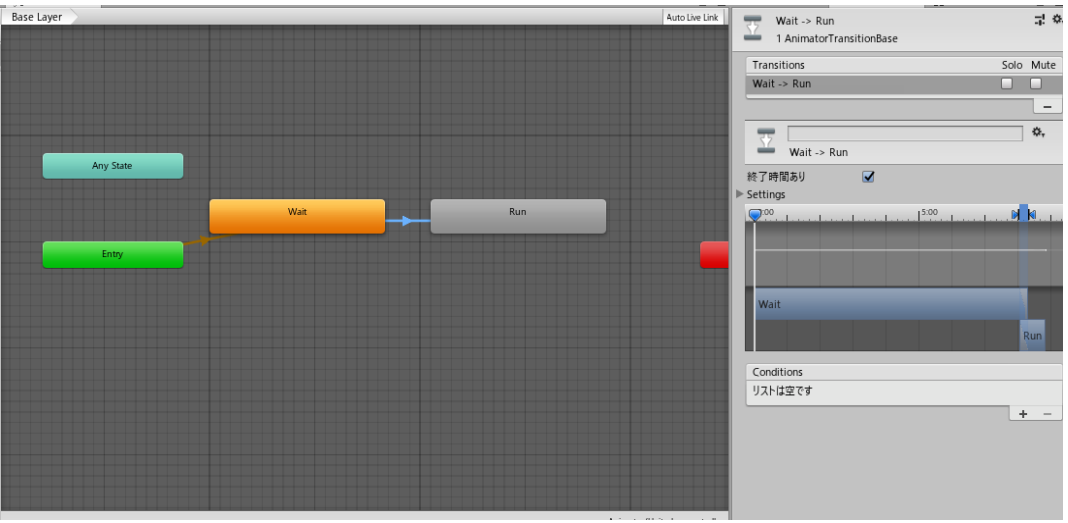
ゲームを再生してUnityちゃんが動いたこととアニメーションビューのSpeedの数字が変わることを確認する。

Unityちゃんがうごくようになったが、次は移動の時に走るように設定いたい。

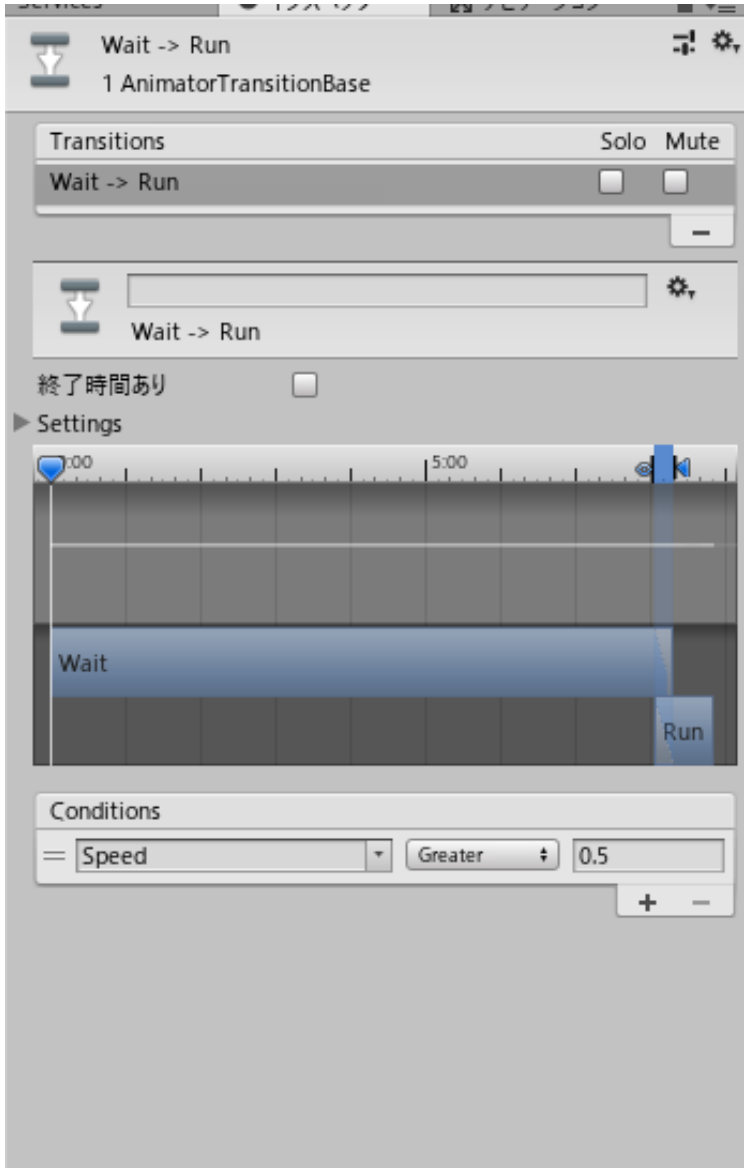
Waitを右クリックして転移を選ぶ
矢印が引っ張られるので
先ほどRunのモーションを入れたス
テートを選択する

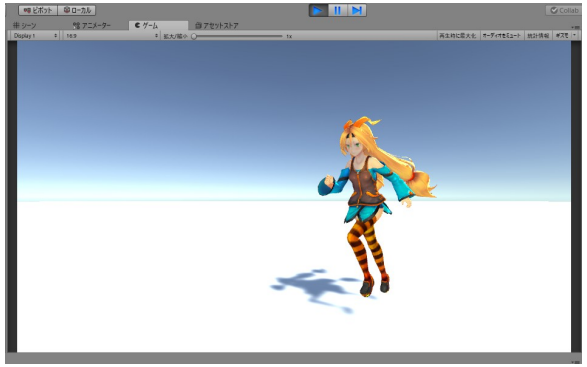


矢印を選択すると
状態が変わる条件を入れることができ
る。(選択した矢印が青くなる)



WaitからRunに変える条件
終了時間ありのチェックを外す
Conditionsの＋をクリックすると
アニメーションのパラメーターで書いた
Speedが出てくるので選択
真ん中のボタンからGreaterを選択
入力ボックスに0.5と入力する。

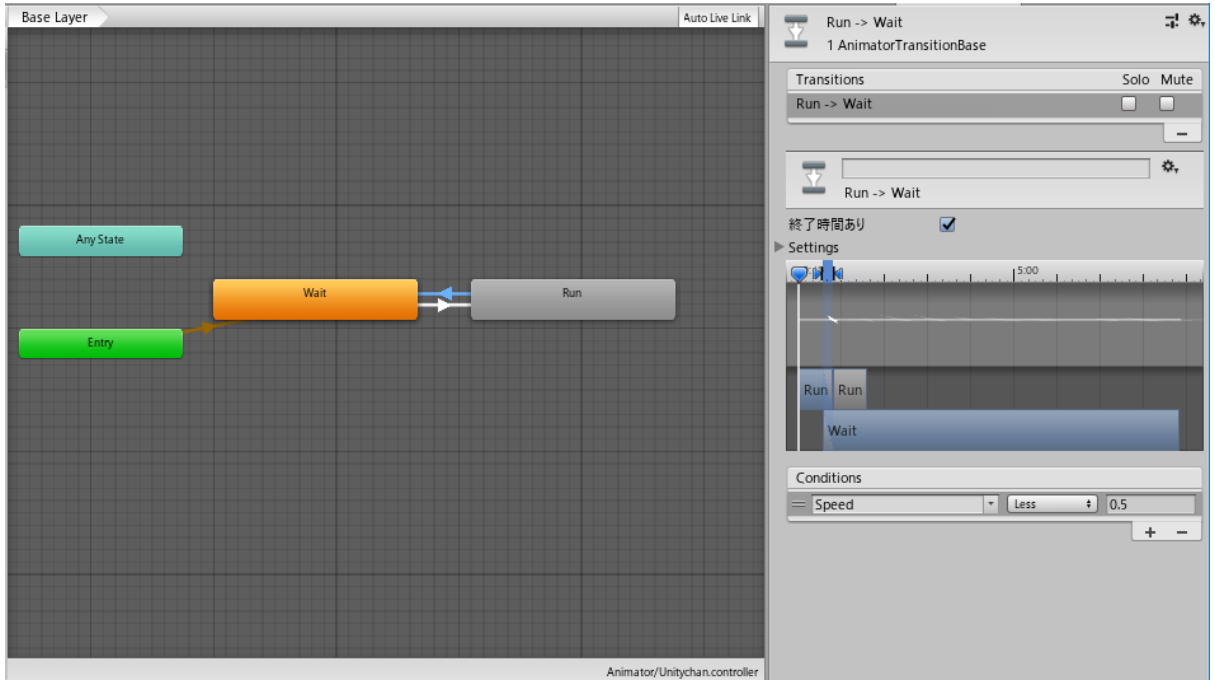


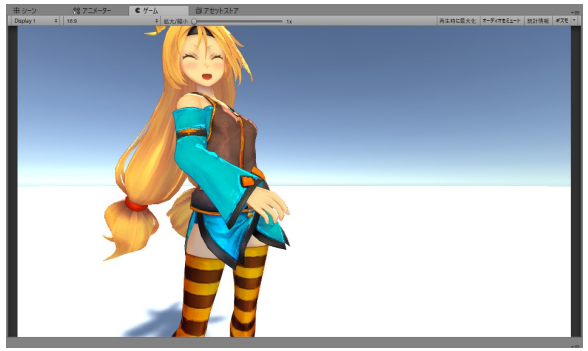


歩き出すとUnityちゃんが走るようになった。
しかし、
このままだと一度0.5以上の数字になると走るモーションがずっと続いてしまうので止まった時に待機モーションになるようにしたい

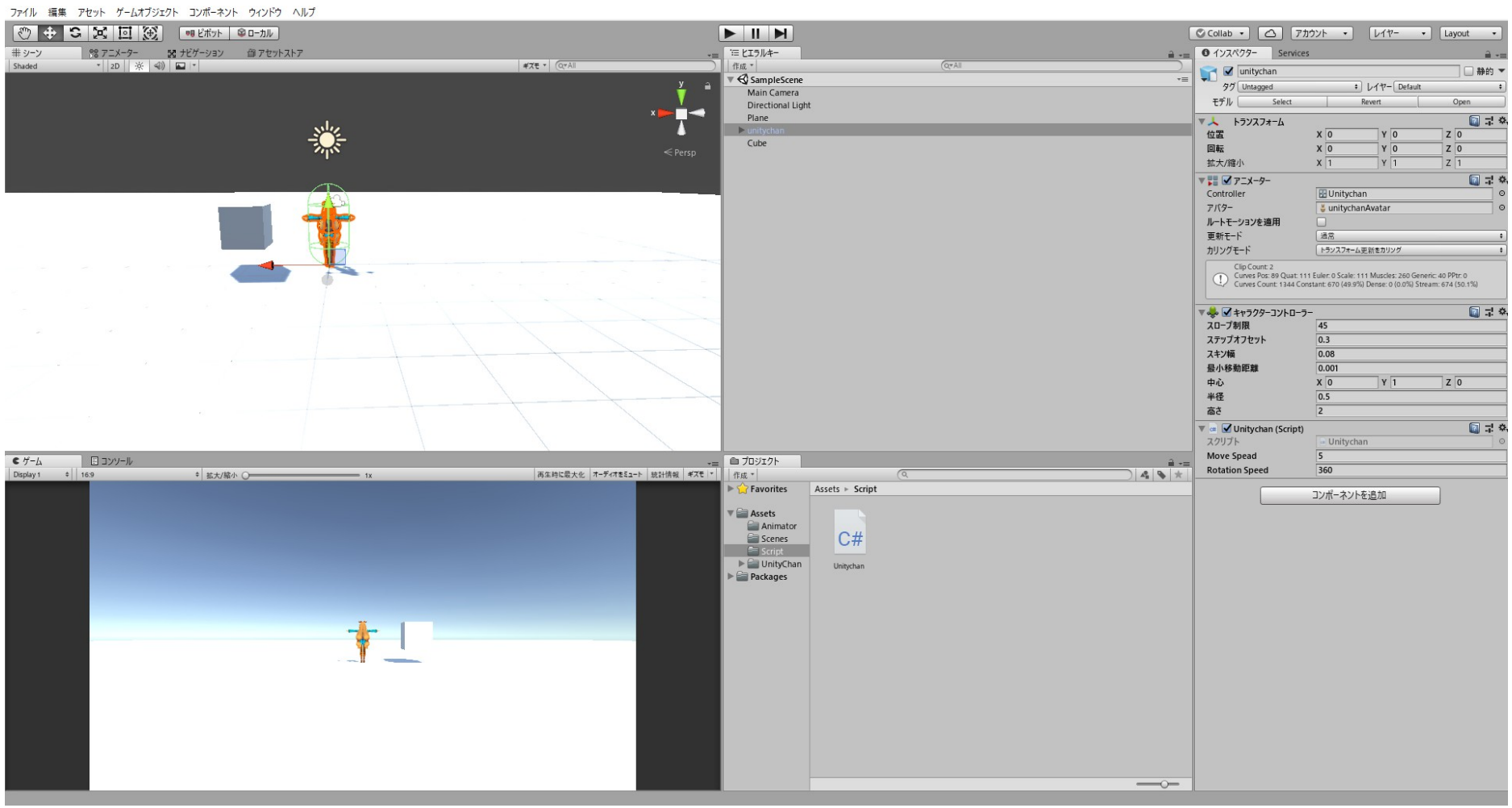
パラメーターが0.5以上の数値になるとモーションRunを読み込むようになる。
そのために
パラメーター、スクリプトの名前が一致している必要がある
パラメーターの名前がWalkなら
ConditionsにはWalkがでてくる。
しかし、スクリプトのほうがSpeedのままだとパラメーターの数字が動かないのでRunモーション働かない

今度はRunから矢印を作りwaitまで
引っ張る
先ほどの設定はGreaterだったが
今度はLessに設定する。





完成(今回理解したこと)
・アニメーションの付け方
・アニメーションの切り替え方



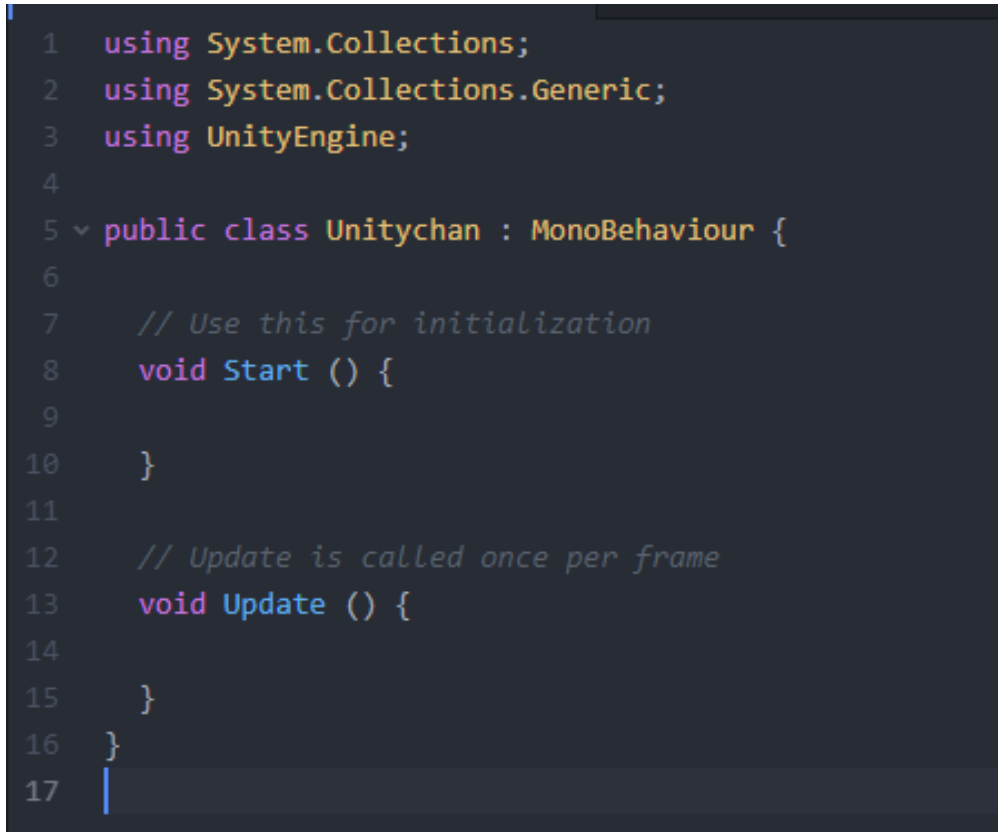
UIは好みの形に変えられるので使いやすい形に変えること

フロートとはコンポーネントのスクリプト欄に追加
されていく要素のこと
スクリプトでいうと上のほうにある
pubic　float　という部分
public　froat　で追加するとコンポーネントのほう
に要素が増えていく


```
7     public float moveSpead = 5f;//歩くスピード
8     public float rotationSpeed = 360f;//振り向くスピード
9     public float jumpPower;
10
```

わかりやすい名前を付けておくこと

```
public float moveSpead = 5f;//歩くスピード
public float rotationSpeed = 360f;//振り向くスピード
public float jumpPower;
public float test;
```



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

public class Unitychan : MonoBehaviour {

  // Use this for initialization
  void Start () {

  }

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

  }
}
```

public class Unitychan : MonoBehaviour
Unitychanの部分にシート名を入れる(拡張子は外す)
シート名以外のものを入れてしまうとエラーになる
同じファイル名を使うことはできない
(同ファイル内に同じ名前のものを作れないのと同じ原理)
void Start
コンポーネントに追加したものをスクリプト上で読み込むために使う
変数=GetComponent〈コンポーネント名〉();
void Update
スクリプトから直接命令を送るために使う
Startに入れたコンポーネントの変数を利用して命令を追加する

(ただしUpdateに直接GetComponentを入れているものもあるので
断定できていない)調査中
Startは変数化するため？