

2020年度 Unity講座(基礎編)



09回目

講師：幸田 将伍 (@MagurodonDev)

今回の講義の目的

2

- ▶ プログラムを自分で読めるようになる
- ▶ Unityを使って自分が実現したいことをできるようになる
- ▶ 自分一人でもゲームを作成できるレベルになる
- ▶ Unityの活用事例を学び、自分の進路に役立てる
- ▶ 実際のエンジニアがどういった仕事の進め方をしているかを知る
- ▶ ゲーム会社のクライアントエンジニアとして就職できるレベルになる

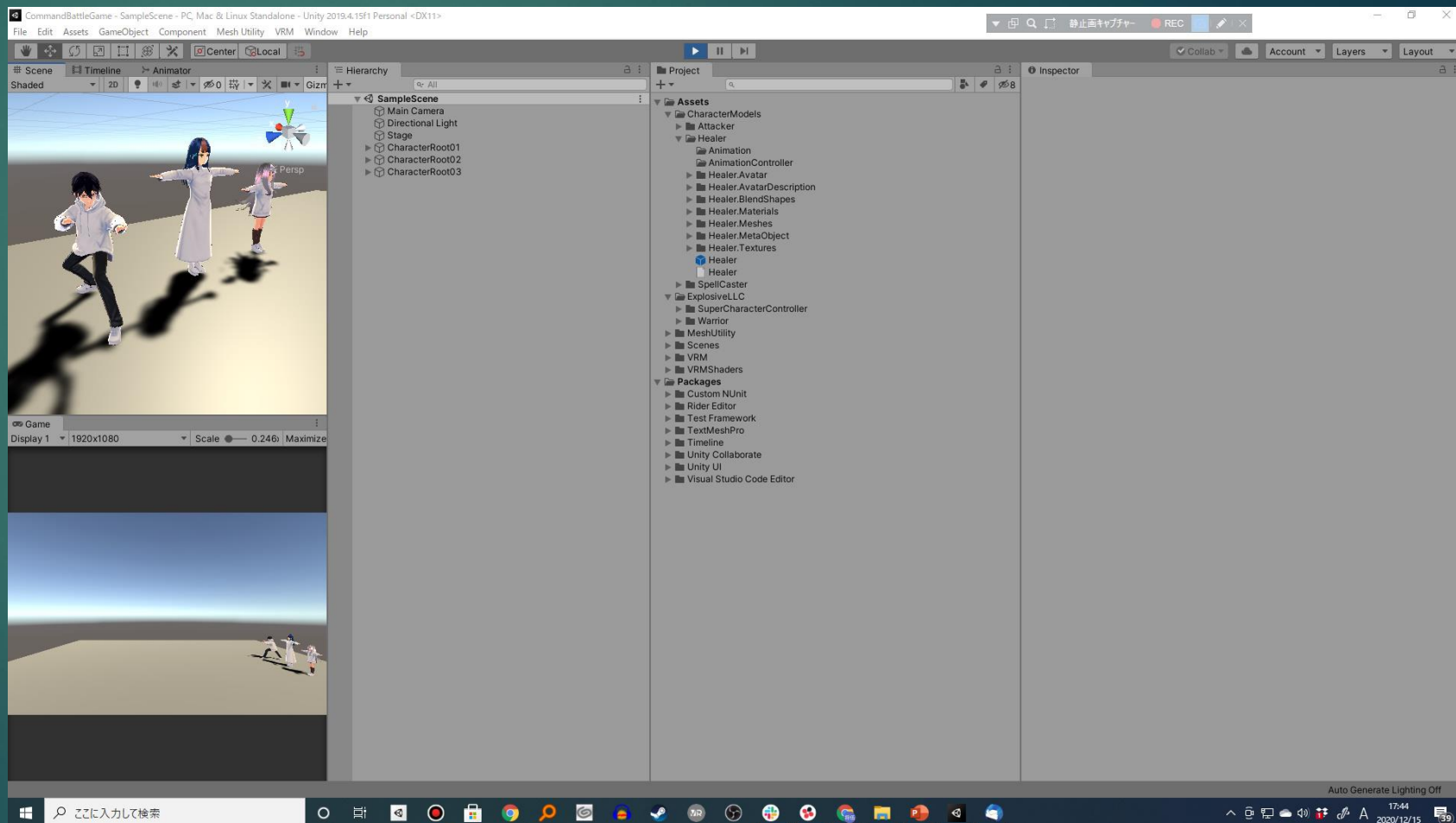
一緒にレベルアップして行きましょう！

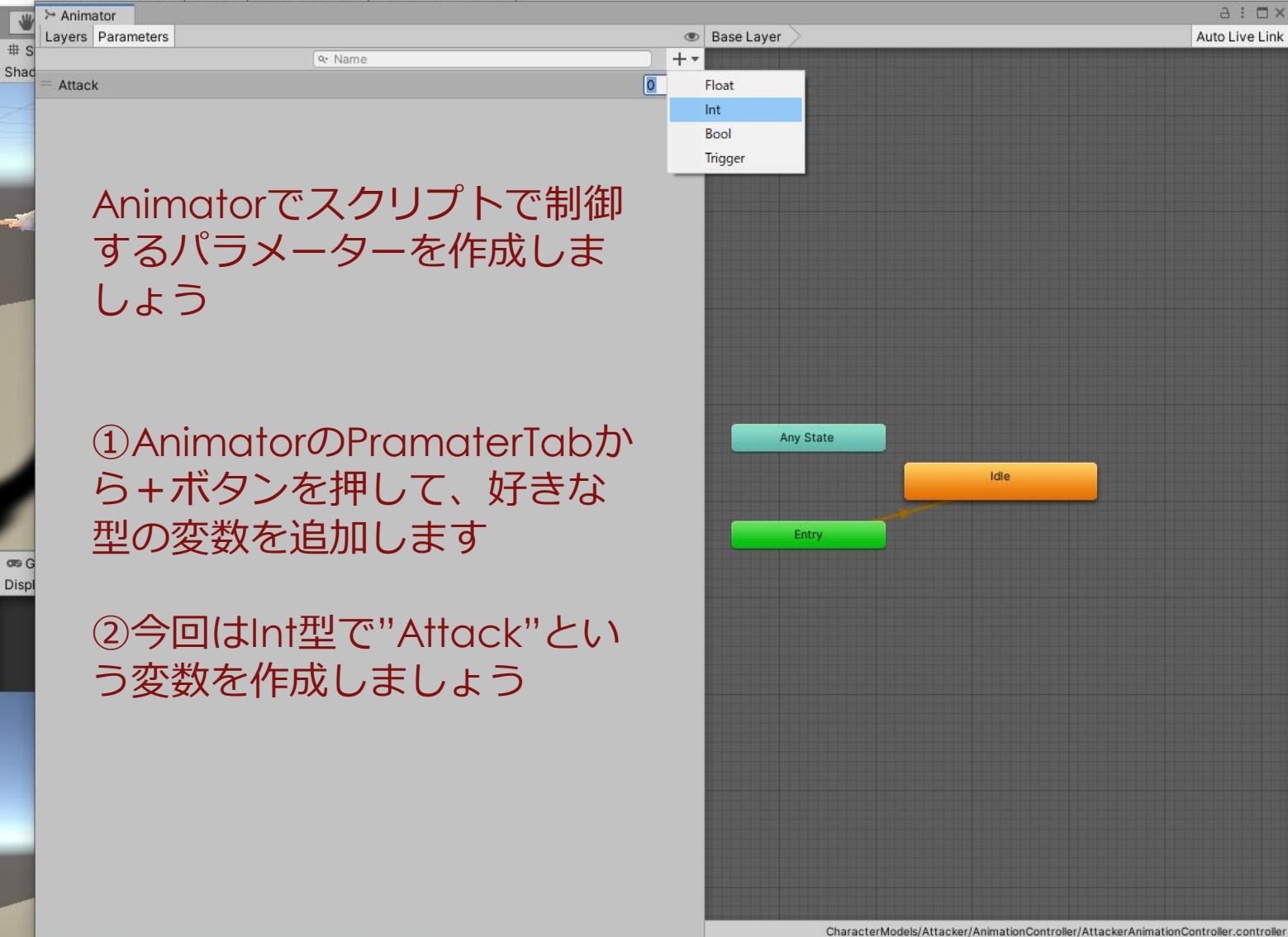
Unity

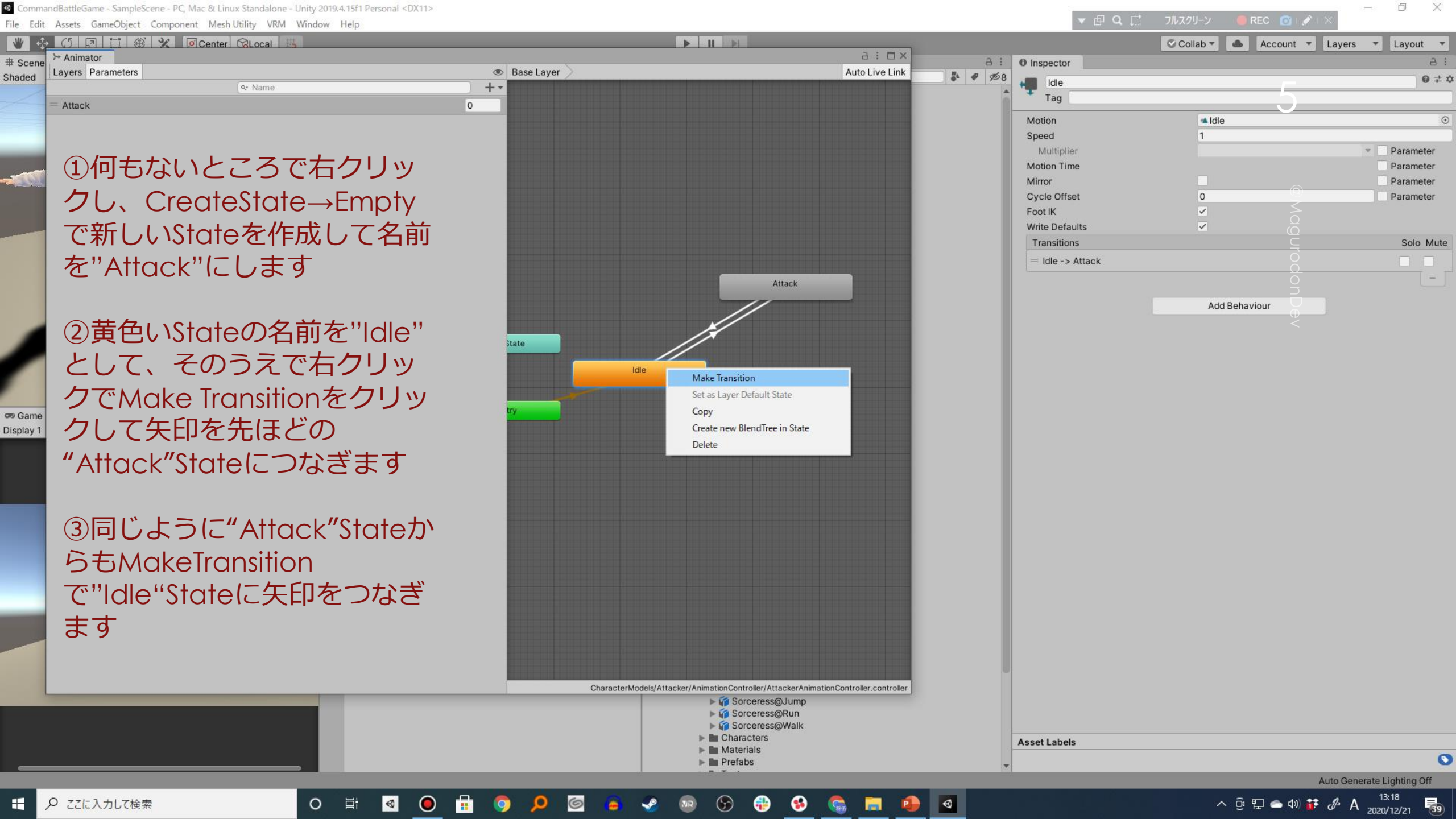
3

コマンドバトルゲーム

- 先週は3体のモデルを作成して、そのうちの一体に対して、アニメーションを設定しました
- 今回はアニメーションの設定、並びに、敵のもとへ行って Attackerに攻撃を行わせるところまで行こうと思います
- Attackerの設定をしますので、もし今日の授業が修了できれば他の二体は宿題でやってみてもらいます
- まずは、Attackerのアニメーションの設定をしていきましょう



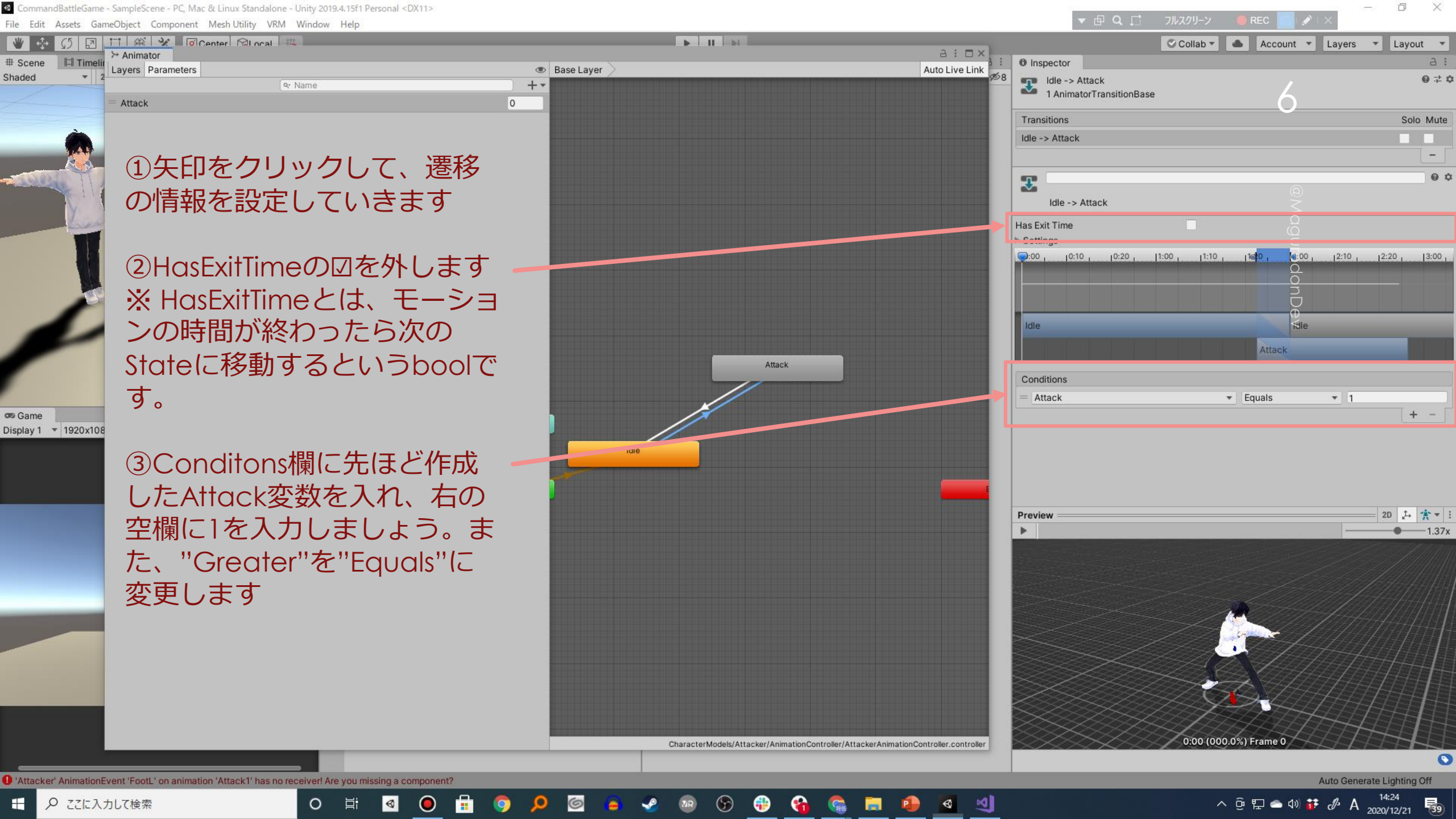




①何もないところで右クリックし、CreateState→Emptyで新しいStateを作成して名前を"Attack"にします

②黄色いStateの名前を"Idle"として、そのうえで右クリックでMake Transitionをクリックして矢印を先ほどの"Attack"Stateにつなぎます

③同じように"Attack"StateからもMakeTransitionで"Idle"Stateに矢印をつなぎます



①矢印をクリックして、遷移の情報を設定していきます

②HasExitTimeの☑を外します
※ HasExitTimeとは、モーションの時間が終わったら次のStateに移動するというboolです。

③Conditons欄に先ほど作成したAttack変数を入れ、右の空欄に1を入力しましょう。また、"Greater"を"Equals"に変更します

Inspector

Idle -> Attack
1 AnimatorTransitionBase

Transitions

Idle -> Attack

Has Exit Time ☐

Settings

0:00 | 0:10 | 0:20 | 1:00 | 1:10 | 1:20 | 1:30 | 1:40 | 1:50 | 2:00 | 2:10 | 2:20 | 3:00

Idle

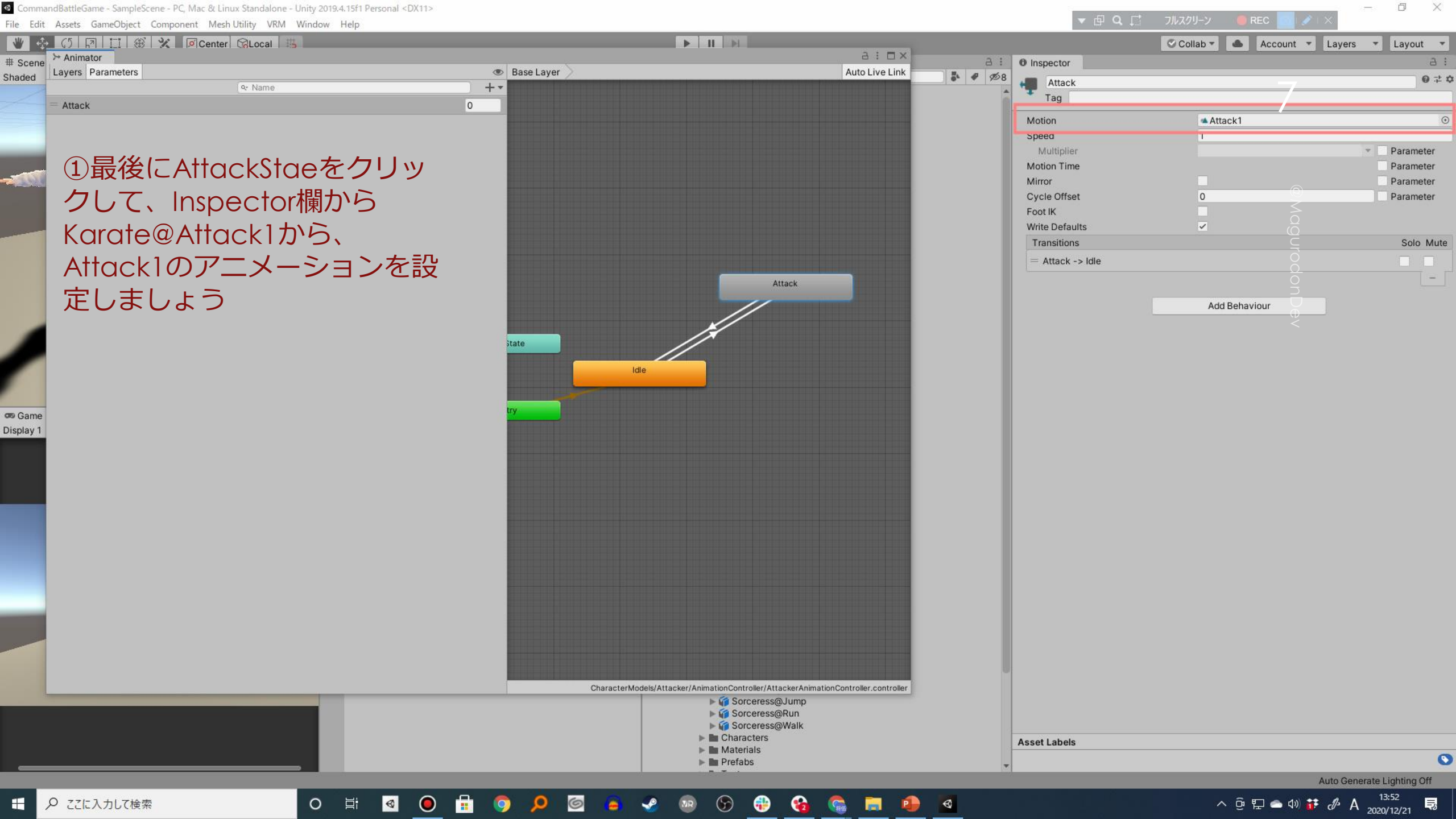
Attack

Conditions

= Attack Equals 1

Preview

0:00 (000.0%) Frame 0



①最後にAttackStateをクリックして、Inspector欄からKarate@Attack1から、Attack1のアニメーションを設定しましょう

Inspector

Attack

Tag

Motion Attack1

Speed 1

Multiplier

Motion Time

Mirror

Cycle Offset 0

Foot IK

Write Defaults

Transitions

Attack -> Idle

Add Behaviour

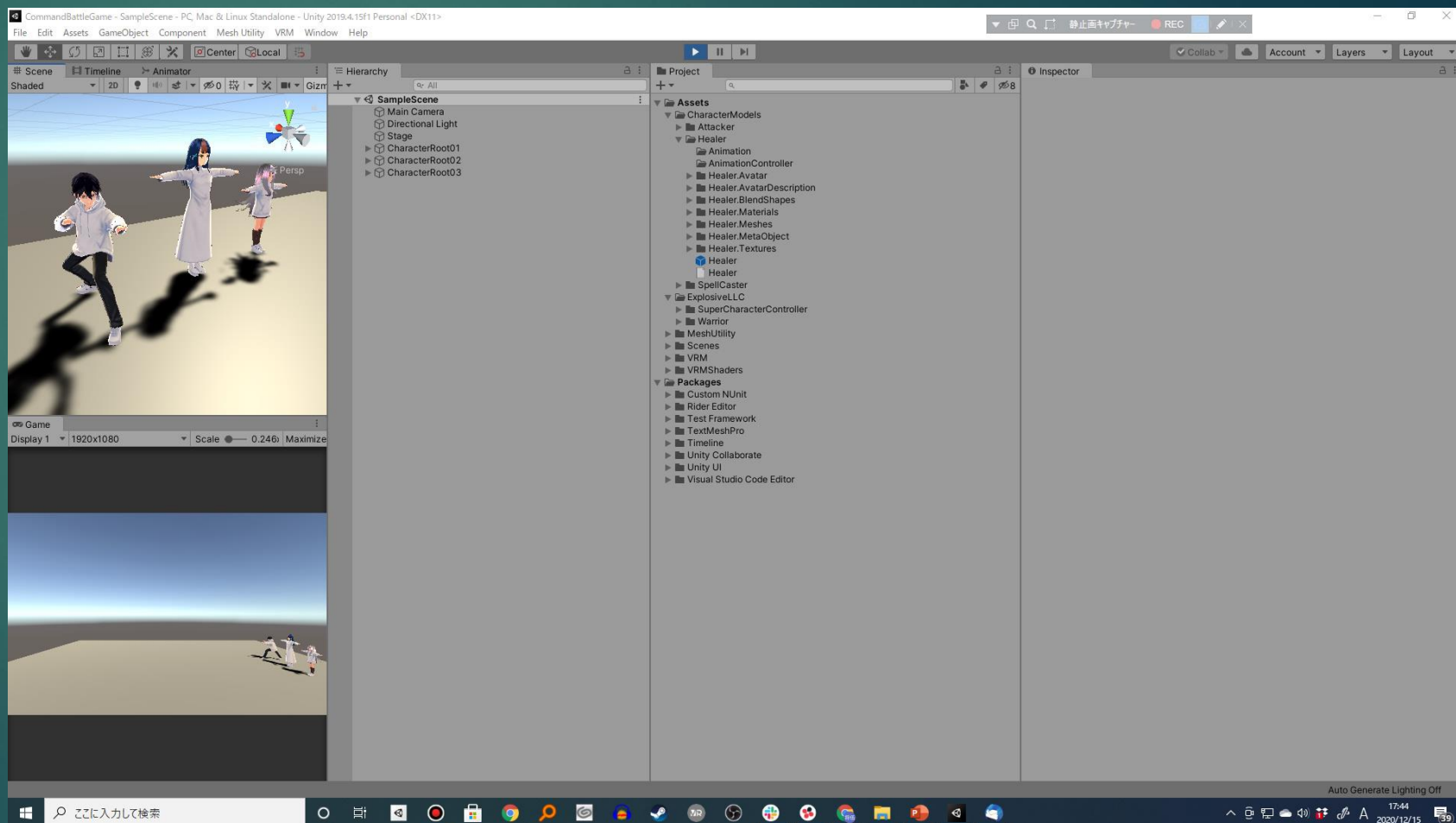
Asset Labels

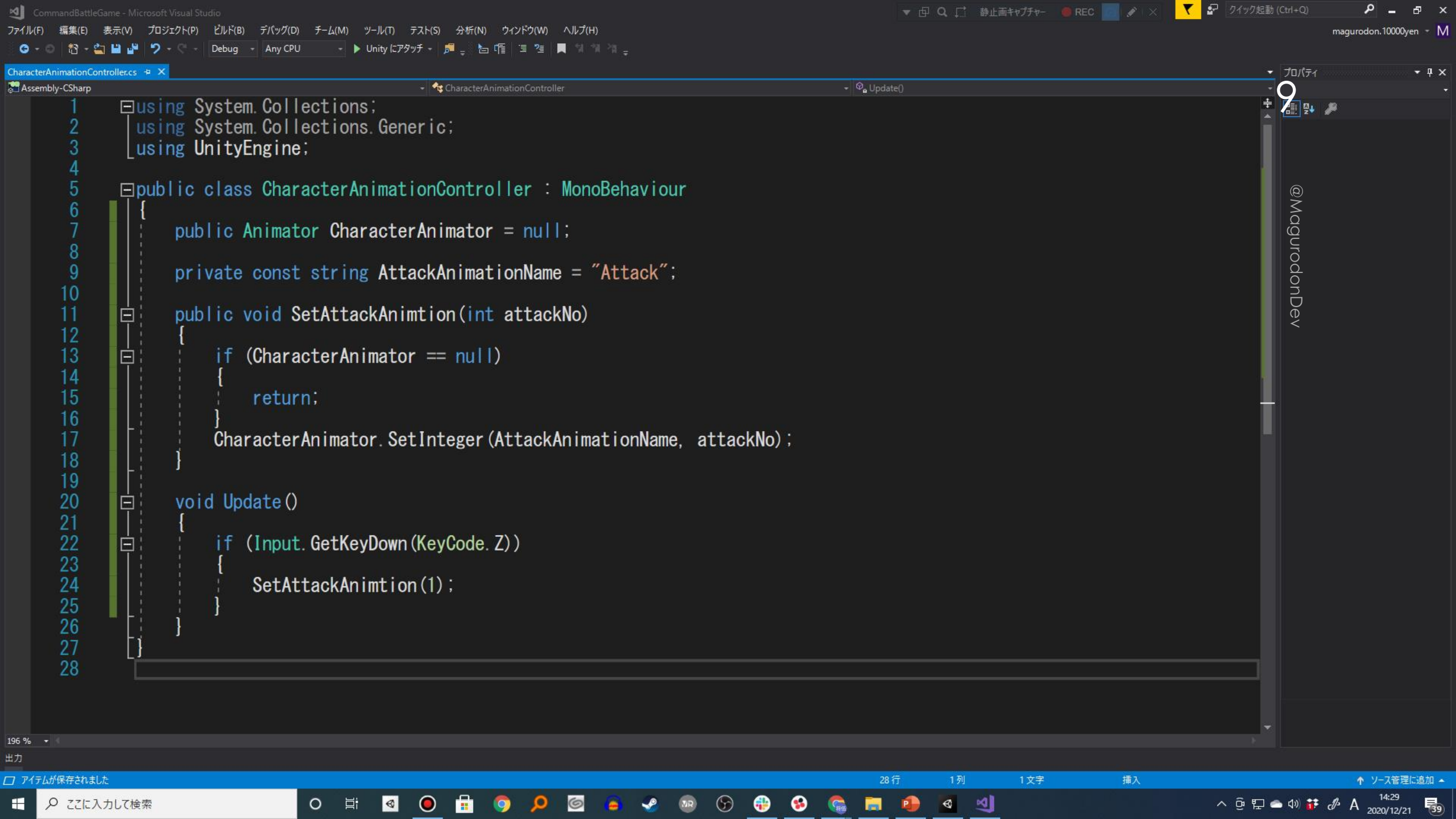
Unity

コマンドバトルゲーム

8

- ここまでできましたら、キャラクターのAnimationを制御するスクリプトを書いていきます
- Assetsフォルダ以下にScriptsフォルダを作成し、新しくCharacterAnimatonController.csを作成しましょう





Unity

10

コマンドバトルゲーム

- ※Animator型
- Animatorで設定したフラグや値を操作するクラスです
- 例) SetInteger("Name",1)
- "Name"のInt型の変数に1を設定します
- 例) Setfloat("Name",0.1f)
- "Name"のfloat型の変数に0.1fを設定します
- 例) Setbool("Name",false)
- "Name"のbool型の変数にfalseを設定します
- 例) SetTrigger("Name")
- "Name"のTriggerをオンにします→すぐにオフに変わります

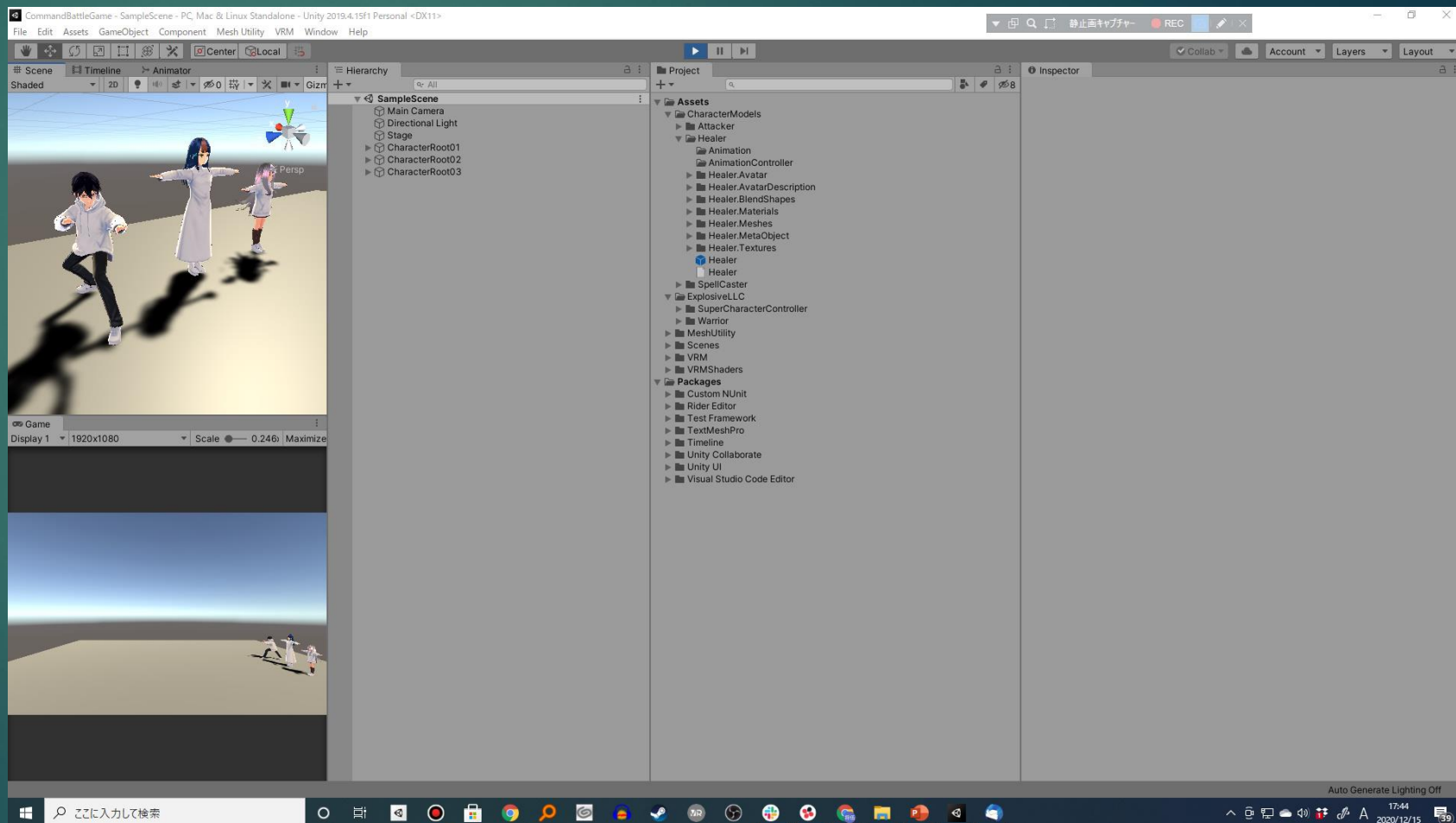
```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class CharacterAnimationController : MonoBehaviour
6 {
7     public Animator CharacterAnimator = null;
8
9     private const string AttackAnimationName = "Attack";
10
11     public void SetAttackAnimtion(int attackNo)
12     {
13         if (CharacterAnimator == null)
14         {
15             return;
16         }
17         CharacterAnimator.SetInteger(AttackAnimationName, attackNo);
18     }
19
20     void Update()
21     {
22         if (Input.GetKeyDown(KeyCode.Z))
23         {
24             SetAttackAnimtion(1);
25         }
26     }
27 }
28
```

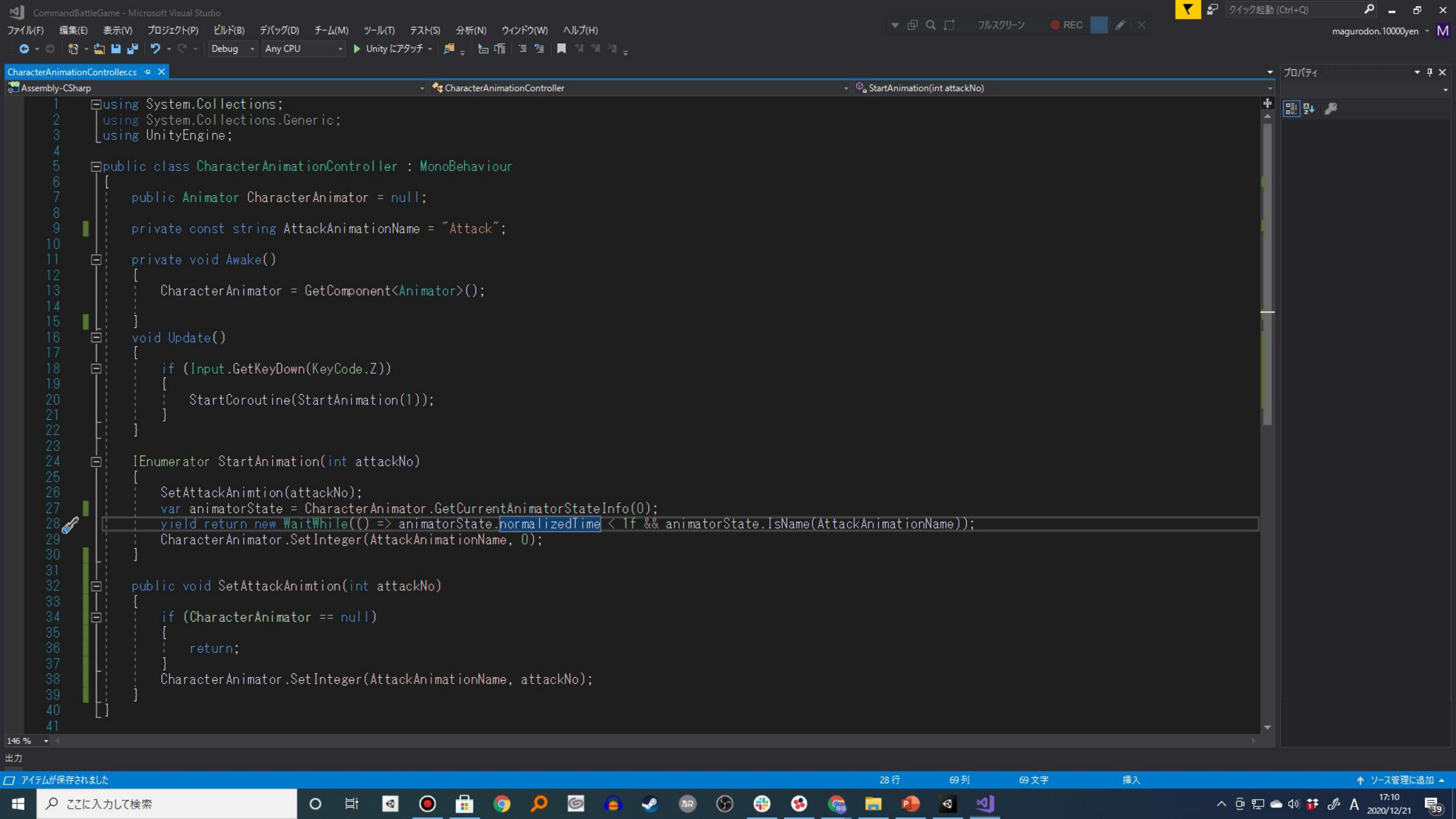
Unity

11

コマンドバトルゲーム

- Attacker.prefabに
CharacterAnimationController
をAddComponentします
- CharacterAnimatorを自身の
Animatorで設定します
- 再生してZキーを押すと、アニメーションが再生されます
- (今の時点ではエラーは無視してください)
- ただ今のままですと、キャラクターがアニメーションをずっと流し続けるので、戻さないとはいけません。
- アニメーションの待ちを実装しましょう。



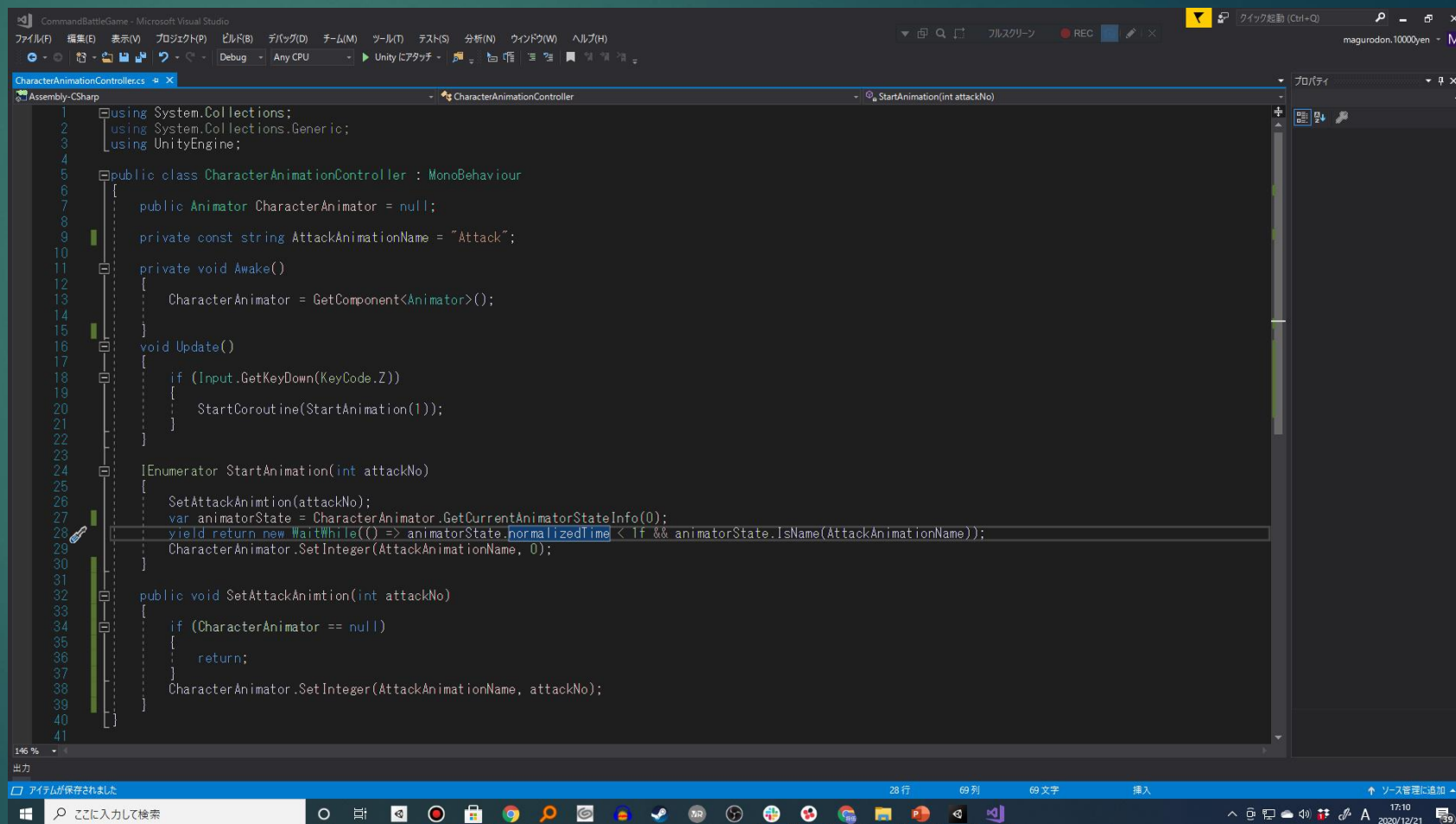


Unity

13

コマンドバトルゲーム

- ※IEnumerator型
- 何かを条件などで待つ際にはCoroutineを使うと良いです
- Coroutineで操作できる型をIEnumeratorと言います
- IEnumerator HogeHogeと続く場合、何かを待ちたいんだな、と読むことができます
- ※yield return new HogeHoge
- このHogeHogeの部分にいろいろと待つ条件が入りますので、次ページで紹介していきます



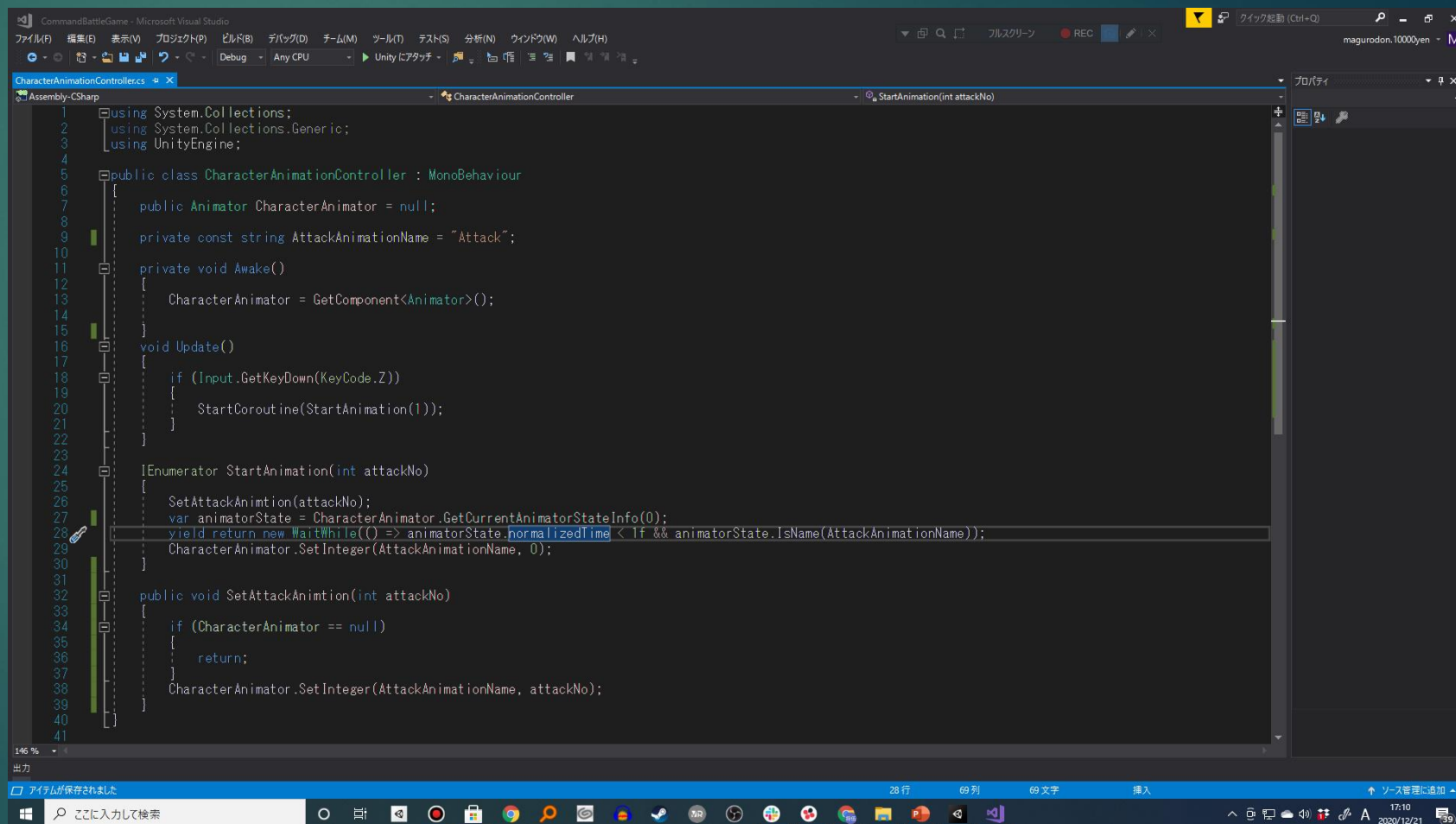
```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class CharacterAnimationController : MonoBehaviour
6 {
7     public Animator CharacterAnimator = null;
8     private const string AttackAnimationName = "Attack";
9     private void Awake()
10     {
11         CharacterAnimator = GetComponent<Animator>();
12     }
13     void Update()
14     {
15         if (Input.GetKeyDown(KeyCode.Z))
16         {
17             StartCoroutine(StartAnimation(1));
18         }
19     }
20     IEnumerator StartAnimation(int attackNo)
21     {
22         SetAttackAnimtion(attackNo);
23         var animatorState = CharacterAnimator.GetCurrentAnimatorStateInfo(0);
24         yield return new WaitForSeconds(1);
25         if (animatorState.IsName(AttackAnimationName))
26         {
27             CharacterAnimator.SetInteger(AttackAnimationName, 0);
28         }
29     }
30     public void SetAttackAnimtion(int attackNo)
31     {
32         if (CharacterAnimator == null)
33         {
34             return;
35         }
36         CharacterAnimator.SetInteger(AttackAnimationName, attackNo);
37     }
38 }
39
40
41
```

Unity

14

コマンドバトルゲーム

- ※ WaitForSeconds(float)
秒数で待ちます
- ※ While(()=>条件)
条件の間待ちます
- ※ Until(()=>条件)
条件になるまで待ちます
- ※ WaitForEndOfFrame()
1フレームの終わりまで待ちます



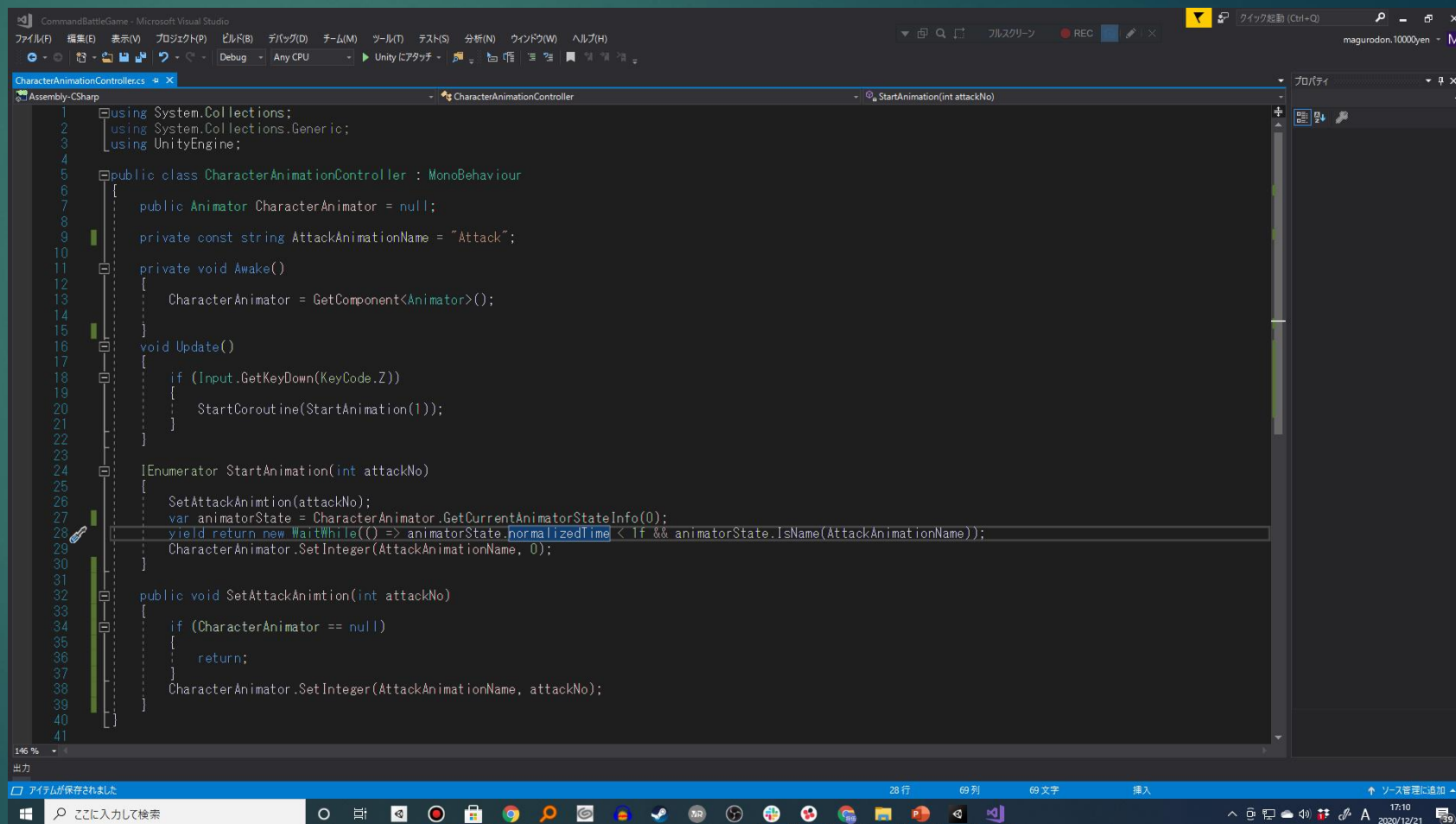
```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class CharacterAnimationController : MonoBehaviour
6 {
7     public Animator CharacterAnimator = null;
8     private const string AttackAnimationName = "Attack";
9
10    private void Awake()
11    {
12        CharacterAnimator = GetComponent<Animator>();
13    }
14
15    void Update()
16    {
17        if (Input.GetKeyDown(KeyCode.Z))
18        {
19            StartCoroutine(StartAnimation(1));
20        }
21    }
22
23    [Enumerator] StartAnimation(int attackNo)
24    {
25        SetAttackAnimtion(attackNo);
26        var animatorState = CharacterAnimator.GetCurrentAnimatorStateInfo(0);
27        yield return new While(() => animatorState.normalizedTime < 1f && animatorState.IsName(AttackAnimationName));
28        CharacterAnimator.SetInteger(AttackAnimationName, 0);
29    }
30
31    public void SetAttackAnimtion(int attackNo)
32    {
33        if (CharacterAnimator == null)
34        {
35            return;
36        }
37        CharacterAnimator.SetInteger(AttackAnimationName, attackNo);
38    }
39
40 }
41
```

Unity

15

コマンドバトルゲーム

- 今回の場合は、アニメーションのStateの正規化された時間が1以下の場合で、かつ、Animationの名前が"Attack"の間、待つという処理を行いました
- 正規化とはどういうことか
- モーションの時間はバラバラなので、それを正規化すると0から1fまでのfloat型で取ることができます。
- 0が始まりで、1fを超えるとアニメーションは終了しているという感じで判定することができます



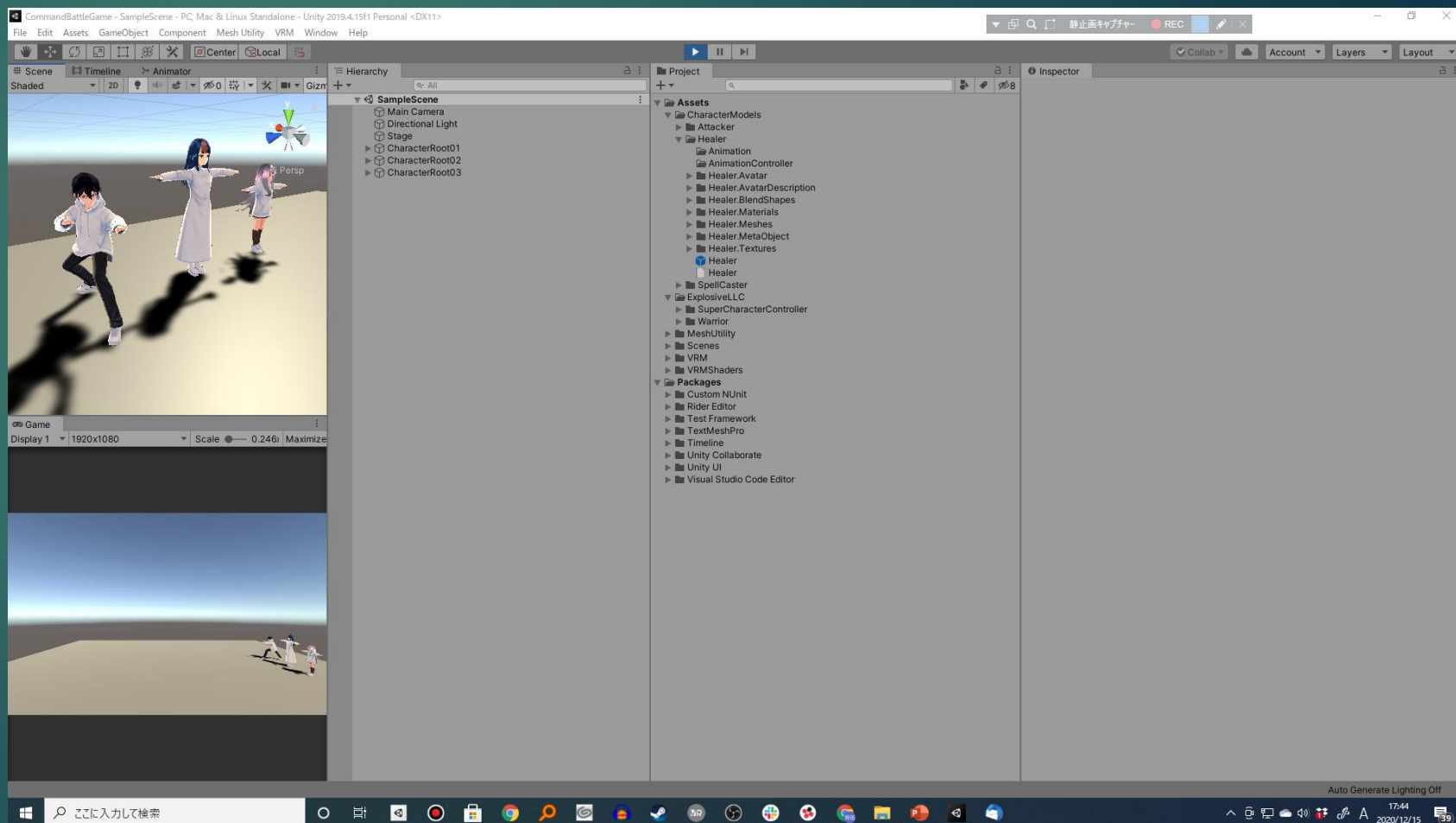
```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class CharacterAnimationController : MonoBehaviour
6 {
7     public Animator CharacterAnimator = null;
8
9     private const string AttackAnimationName = "Attack";
10
11     private void Awake()
12     {
13         CharacterAnimator = GetComponent<Animator>();
14     }
15
16     void Update()
17     {
18         if (Input.GetKeyDown(KeyCode.Z))
19         {
20             StartCoroutine(StartAnimation(1));
21         }
22     }
23
24     [Enumerator] StartAnimation(int attackNo)
25     {
26         SetAttackAnimtion(attackNo);
27         var animatorState = CharacterAnimator.GetCurrentAnimatorStateInfo(0);
28         yield return new WaitForSeconds(1f - animatorState.normalizedTime < 1f && animatorState.IsName(AttackAnimationName));
29         CharacterAnimator.SetInteger(AttackAnimationName, 0);
30     }
31
32     public void SetAttackAnimtion(int attackNo)
33     {
34         if (CharacterAnimator == null)
35         {
36             return;
37         }
38         CharacterAnimator.SetInteger(AttackAnimationName, attackNo);
39     }
40
41 }
```

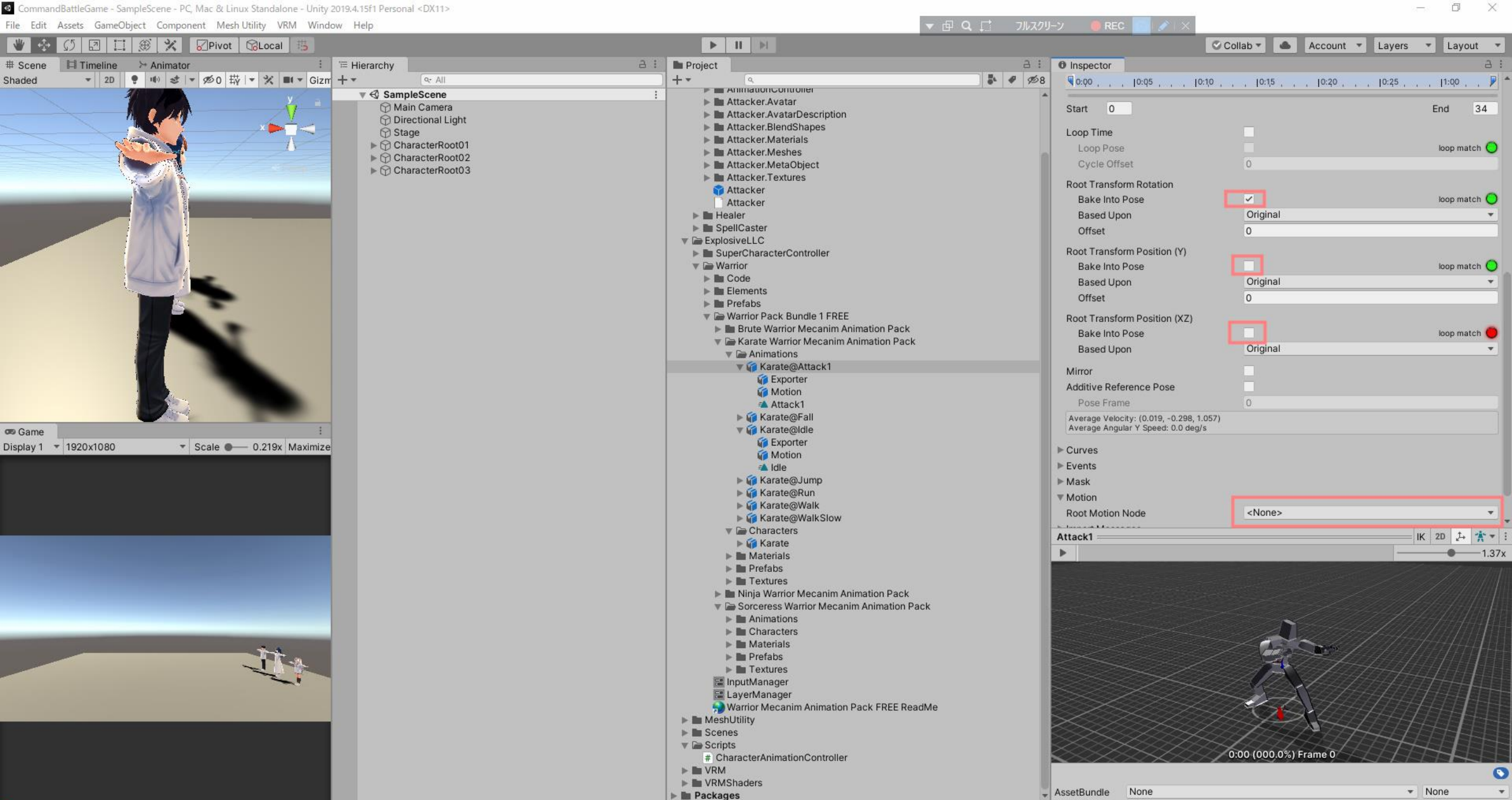
Unity

16

コマンドバトルゲーム

- では、Zキーを押すと一度だけ、アニメーションが再生されることになると思います
- この時、アニメーションが前に進んでしまうことがありますので、モーシヨンの調整をしています
- まあ基本的にはデザイナーさんの仕事なのですが、こちらで触る場合もありますのでお伝えしておきます



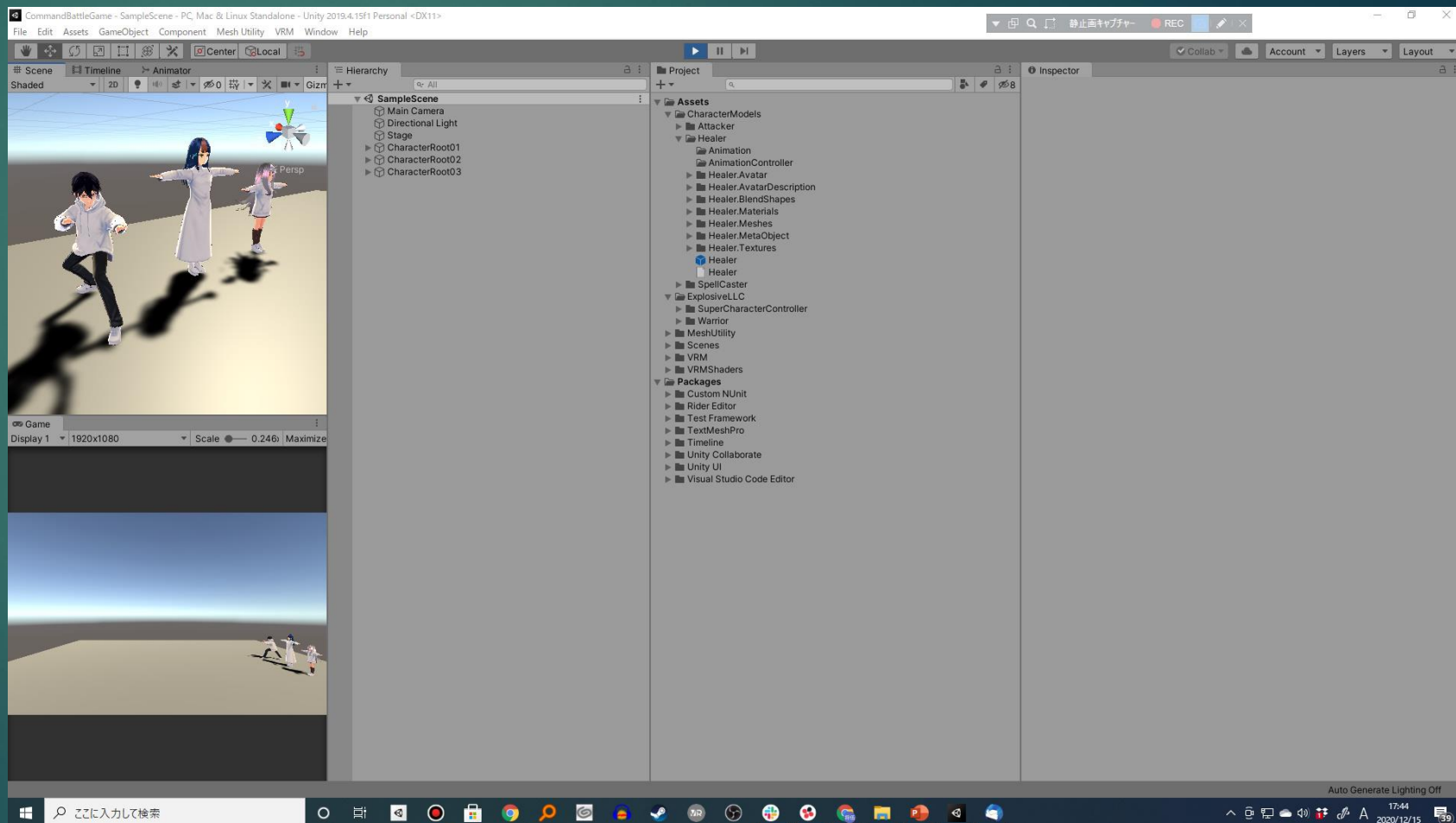


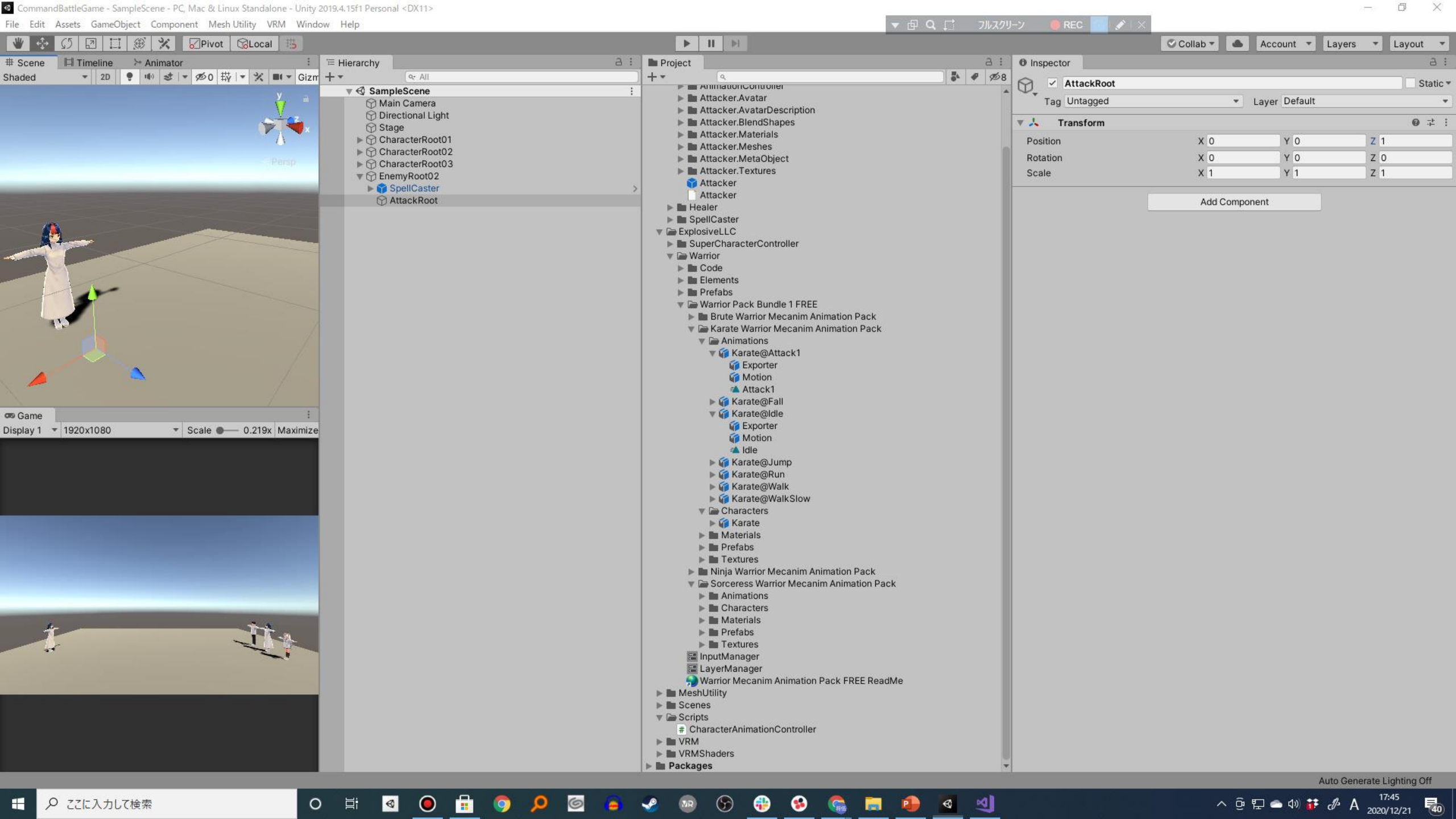
Unity

18

コマンドバトルゲーム

- 次に、敵のルートを作成して、
そこまでAttackerがいけるようにします
- CharacterRoot02をCtrl+Dで複製し、EnemyRoot02として、配置します
- Position
- X:-7 y:-2 z:0
- そして、EnemyRoot2の直下にAttackRootを設定します



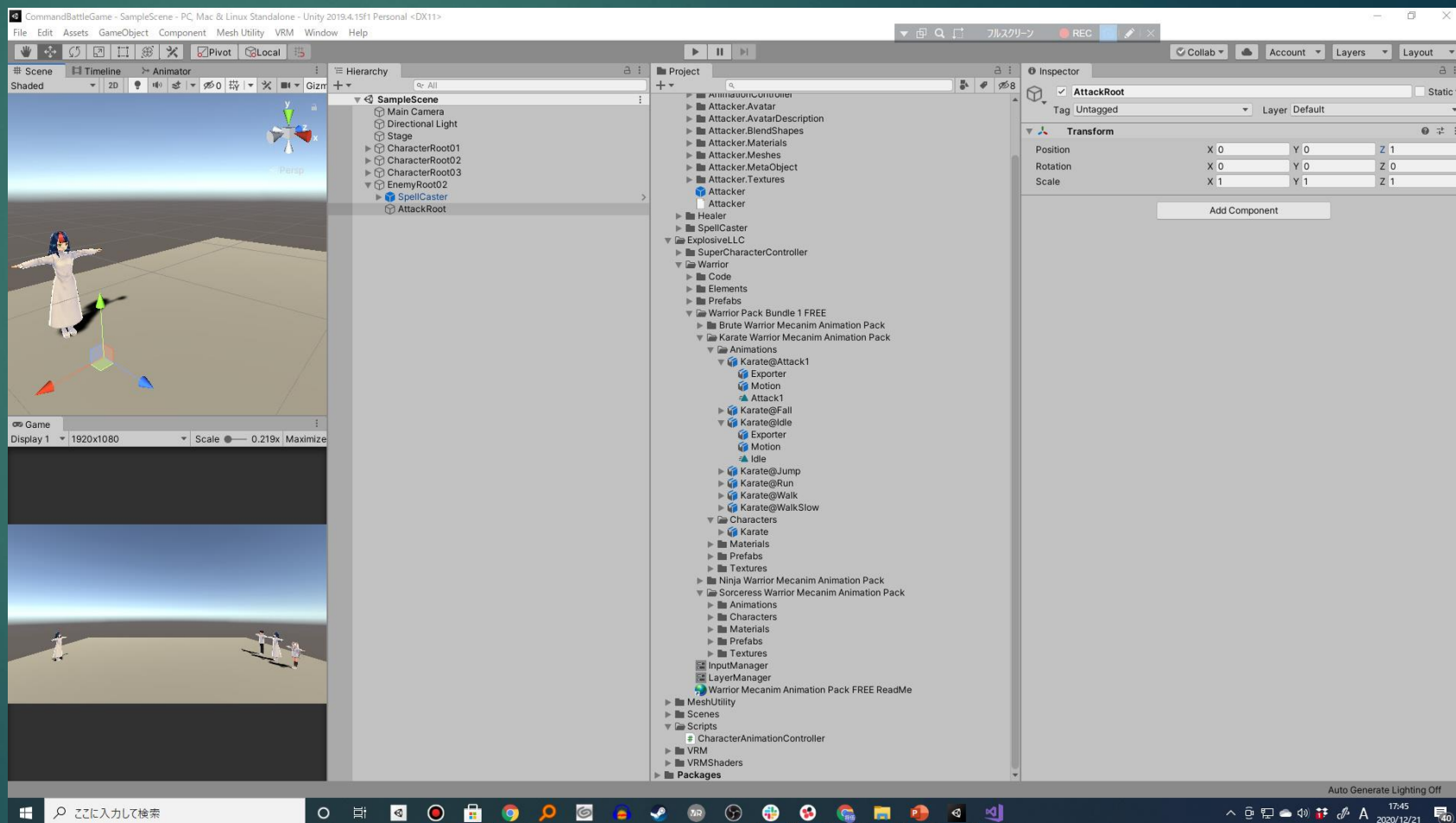


Unity

20

コマンドバトルゲーム

- AttackRootの役割はAttackerがAttackerRootの場所まで移動でして、その後攻撃を行うという役割をします
- 次はスクリプトで制御していきます
- CharacterAnimationControllerを修正していきます



CommandBattleGame - Microsoft Visual Studio

ファイル(F) 編集(E) 表示(V) プロジェクト(P) ビルド(B) デバッグ(D) チーム(M) ツール(T) テスト(S) 分析(N) ウィンドウ(W) ヘルプ(H)

Debug Any CPU Unity にアタッチ

CharacterAnimationController.cs

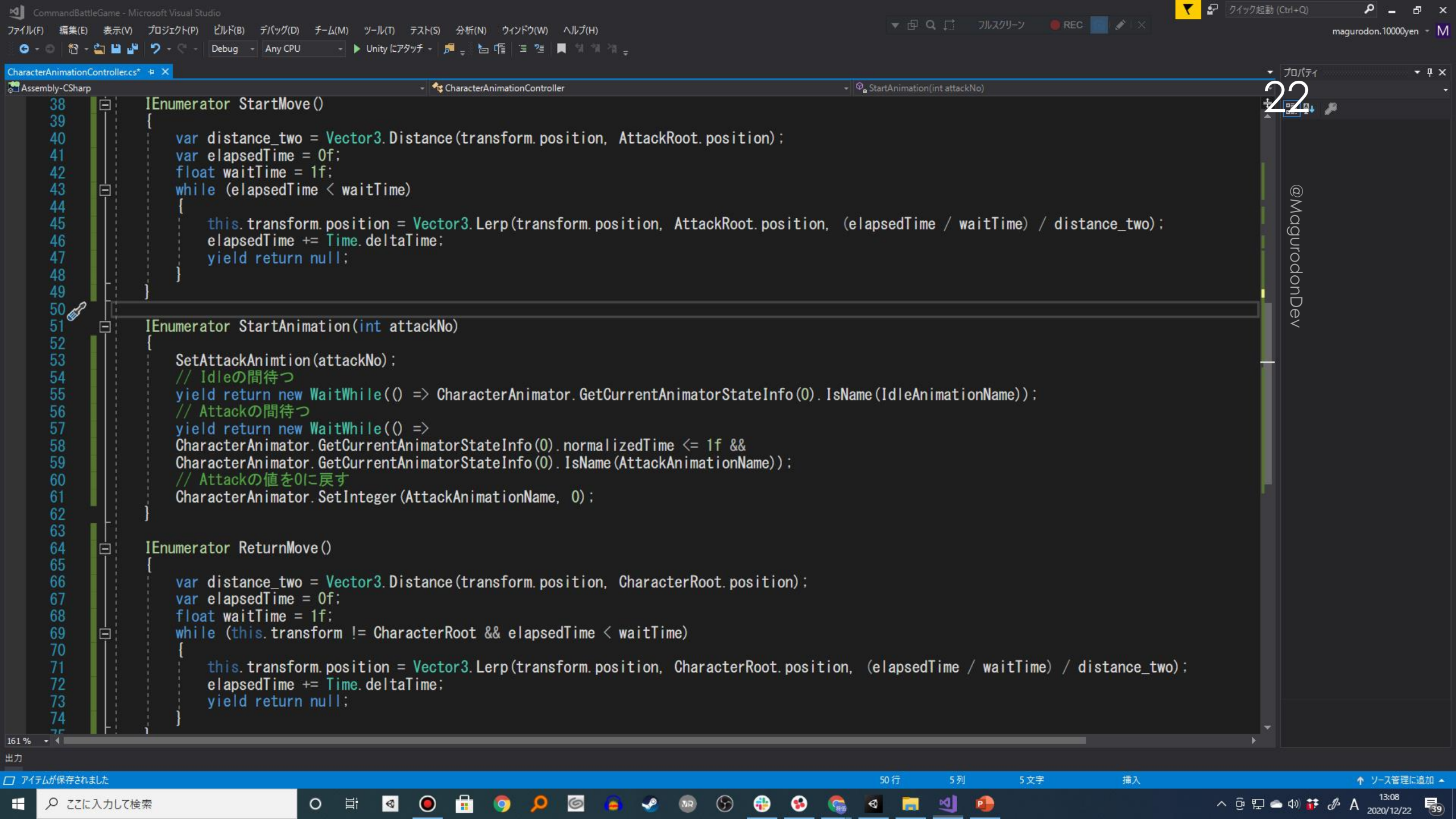
```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class CharacterAnimationController : MonoBehaviour
6 {
7     public Animator CharacterAnimator = null;
8
9     private const string AttackAnimationName = "Attack";
10
11     private const string IdleAnimationName = "Idle";
12
13     public Transform CharacterRoot = null;
14
15     public Transform AttackRoot = null;
16
17     private void Awake()
18     {
19         CharacterAnimator = GetComponent<Animator>();
20     }
21
22     void Update()
23     {
24         if (Input.GetKeyDown(KeyCode.Z))
25         {
26             StartCoroutine(StartAttackAnimation(1));
27         }
28     }
29
30     IEnumerator StartAttackAnimation(int attackId)
31     {
32         yield return StartCoroutine(StartMove());
33         yield return StartCoroutine(StartAnimation(attackId));
34         yield return StartCoroutine(ReturnMove());
35     }
36 }
```

①新しく変数を作成してください
必要なのは今いるTransformと敵への攻撃を行うTransformです

②IEnumerator型の
StartAttackAnimationメソッドを作成します

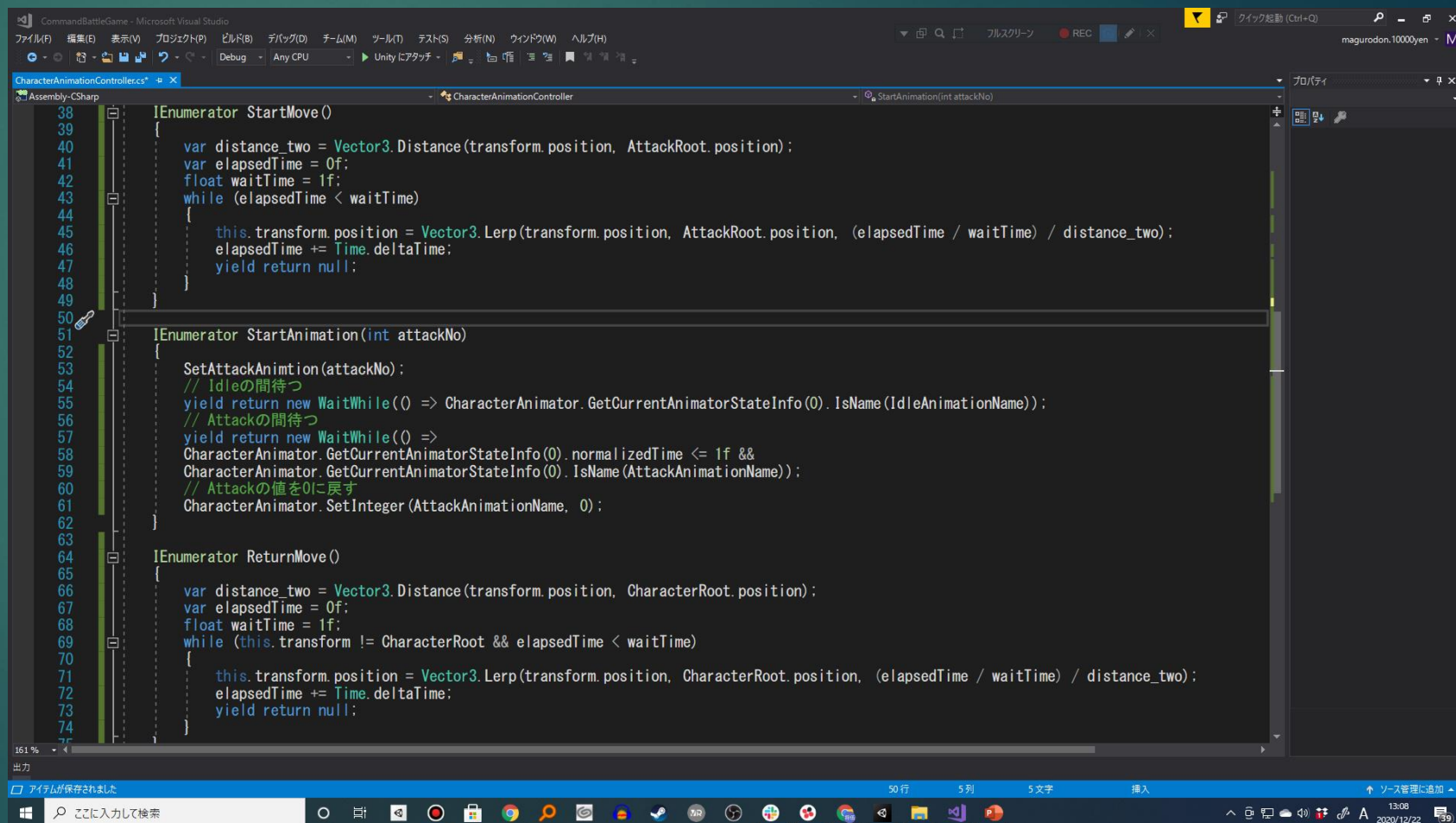
※引数の概念
・ (int hoge)等
引数を渡すことによって、メソッド内でその値を使用する事ができます
今回は呼び出し元の値をメソッド内に渡しているという感じですね

プロパティ



コマンドバトルゲーム

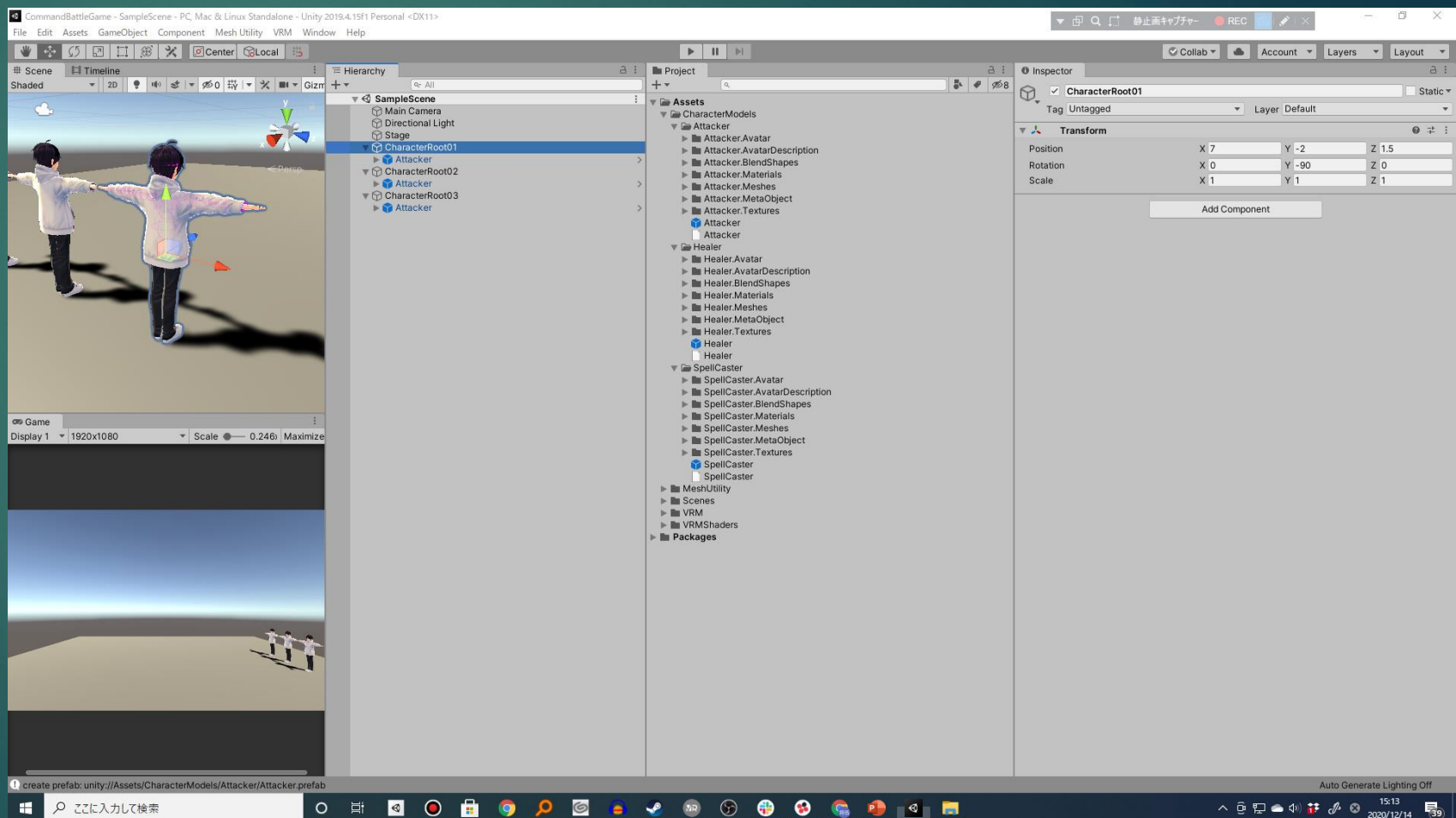
- ※Vector3.Distance(P1,P2)
- P1からP2までの距離をfloatで返してくれます
- ※while(条件){return null;}
- while文という、条件の間待つという構文です
- ※Vector3.Lerp(P1,P2,Time)
- P1からP2をなめらかに移動させます
- ※Animator.GetCurrentAnimatorStateInfo(int)
- 現在のAnimatorの状態を取得します

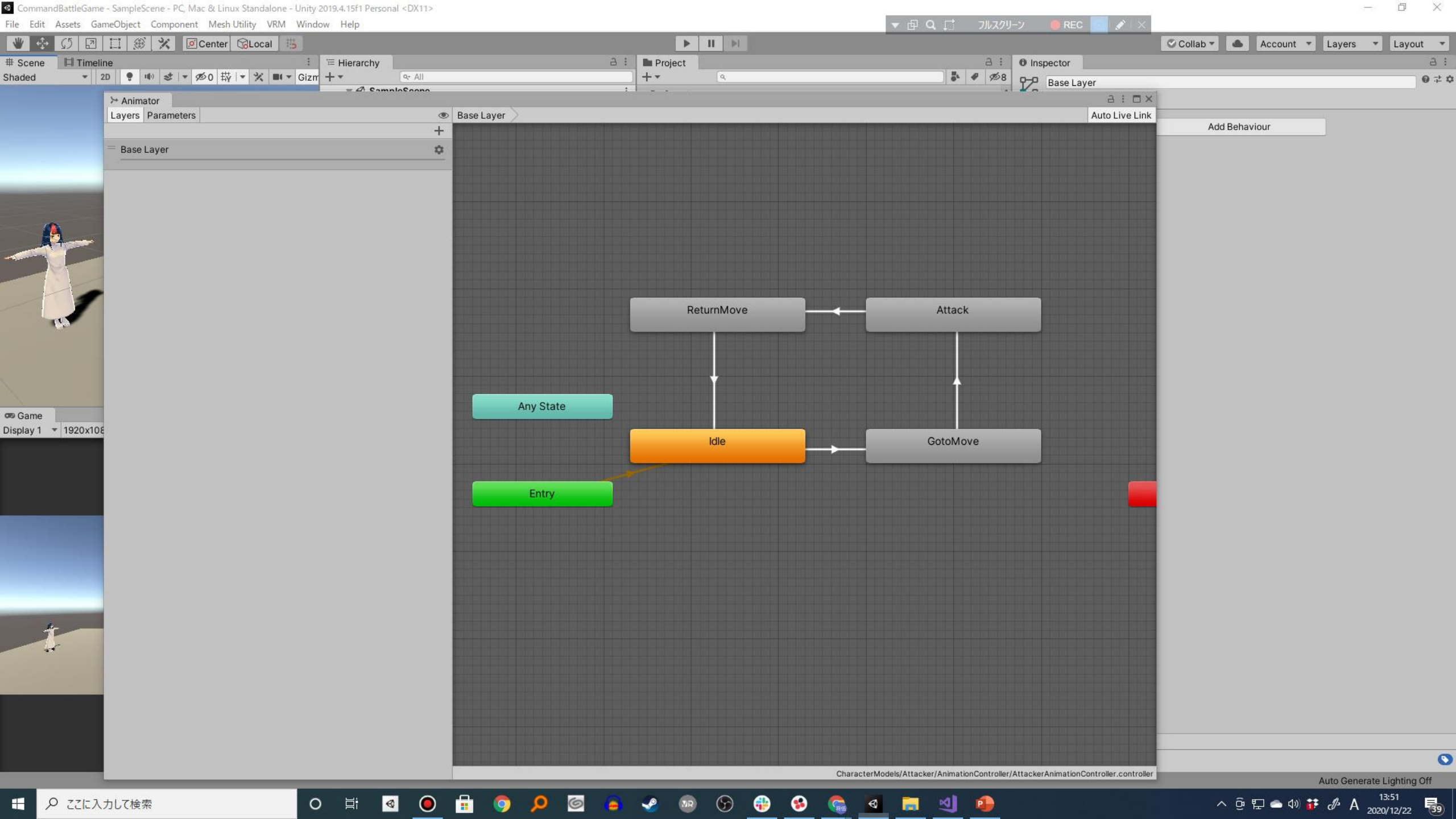


```
38 | IEnumerator StartMove()
39 | {
40 |     var distance_two = Vector3.Distance(transform.position, AttackRoot.position);
41 |     var elapsedTime = 0f;
42 |     float waitTime = 1f;
43 |     while (elapsedTime < waitTime)
44 |     {
45 |         this.transform.position = Vector3.Lerp(transform.position, AttackRoot.position, (elapsedTime / waitTime) / distance_two);
46 |         elapsedTime += Time.deltaTime;
47 |         yield return null;
48 |     }
49 | }
50 |
51 | IEnumerator StartAnimation(int attackNo)
52 | {
53 |     SetAttackAnimtion(attackNo);
54 |     // Idleの間待つ
55 |     yield return new WaitWhile(() => CharacterAnimator.GetCurrentAnimatorStateInfo(0).IsName(IdleAnimationName));
56 |     // Attackの間待つ
57 |     yield return new WaitWhile(() =>
58 |         CharacterAnimator.GetCurrentAnimatorStateInfo(0).normalizedTime <= 1f &&
59 |         CharacterAnimator.GetCurrentAnimatorStateInfo(0).IsName(AttackAnimationName));
60 |     // Attackの値を0に戻す
61 |     CharacterAnimator.SetInteger(AttackAnimationName, 0);
62 | }
63 |
64 | IEnumerator ReturnMove()
65 | {
66 |     var distance_two = Vector3.Distance(transform.position, CharacterRoot.position);
67 |     var elapsedTime = 0f;
68 |     float waitTime = 1f;
69 |     while (this.transform != CharacterRoot && elapsedTime < waitTime)
70 |     {
71 |         this.transform.position = Vector3.Lerp(transform.position, CharacterRoot.position, (elapsedTime / waitTime) / distance_two);
72 |         elapsedTime += Time.deltaTime;
73 |         yield return null;
74 |     }
75 | }
```

コマンドバトルゲーム

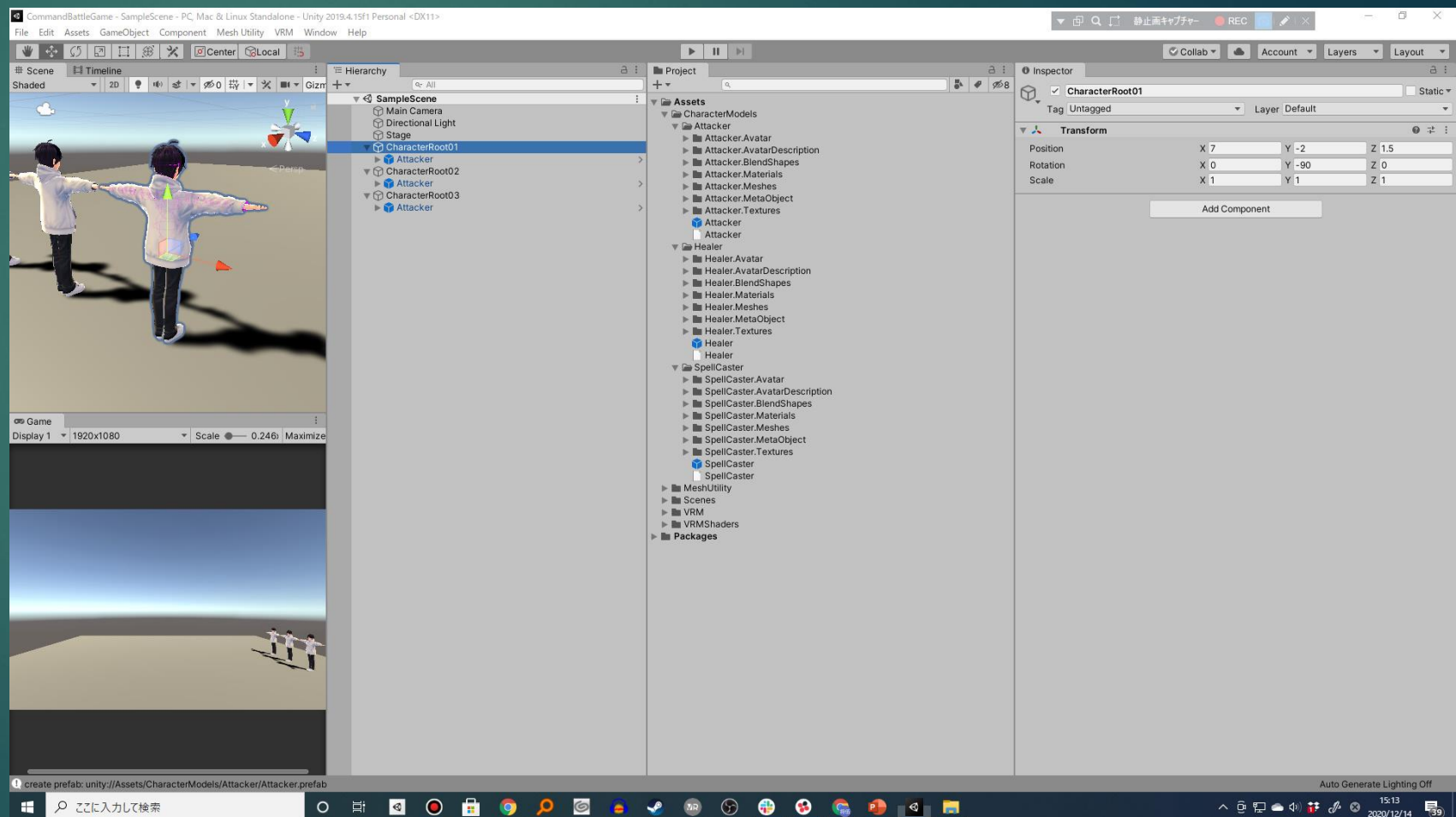
- Unity上でAttackerのCharacterRootとAttackRootを設定して、再生してください
- Attackerが1秒でAttackRootまで行って攻撃モーションを行い、戻ってくるまでを実装できたと思います
- 次は行きのモーションと帰りのモーションを設定しましょう





コマンドバトルゲーム

- 先ほどと違い、Animationを入れるとなると新しくStateを作成する必要があります
- ここからは先ほどまでお伝えしたことを使って、正しく動作するようにしてみてください
- 主な手順としては以下
 - ①Stateに移動する際のAttackの値を変える
 - ②スクリプトで調整
- ここまでできましたら本日は終了です



Unity

27

コマンドバトルゲーム

- もし、先ほどの修正が終わった場合、他の三人のアニメーションの調整も行ってください。
- CharacterAnimationControllerの中で自分がAddされているのはAttackerかHealerかSpellCasterを判定し、または指定してアニメーションのコントロールを行ってみましょう

