Unity勉強会 ver2.0

2Dアクションゲームの

攻撃アニメーションとコンボ攻撃の実装

2020/06/13 Ryo Katayama

自己紹介

レオ

関西10th Unity

ゲーム大好き

面白いゲーム
作りたい

プレゼンの内容

アクションゲームにおける

攻撃

の作り方を学ぶ！

具体的には

- 攻撃アニメーションの作り方
- コンボ攻撃の実装

サンプルプロジェクトの紹介

GitHub：https://github.com/chocsar/SmashUnityChan.git

目次

目次

Unityちゃんの基本構成

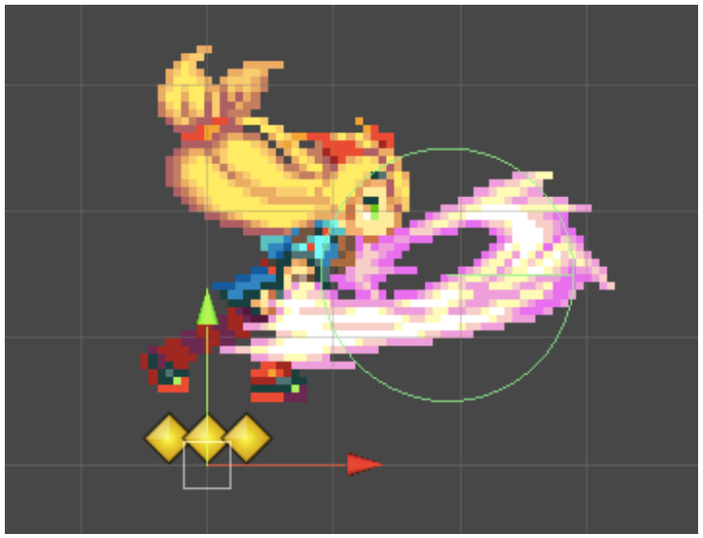
攻撃アクションを作る前に、

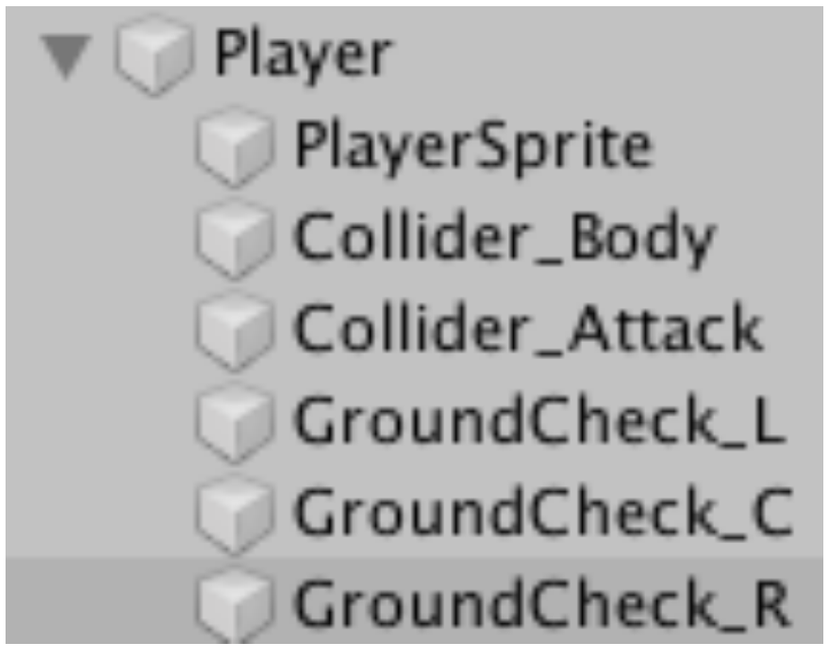
まずは**Unityちゃんの基本構成**を見ていきます！

- **オブジェクト**の構成
- **スクリプト**の構成
- **Animator**の構成

Unityちゃんの基本構成

オブジェクトの構成





- ・・・ **親**（Rigidbody2D, Animator, スクリプトをアタッチする）
- ・・・ **Sprite**（Unityちゃんの見た目）
- ・・・ **Collider**（体）
- ・・・ **Collider**（剣）

- ・・・ 地面チェック用オブジェクト

機能ごとに階層化することで管理しやすくなる！

Unityちゃんの基本構成

スクリプトの構成

動かす

PlayerController

移動・ジャンプ・攻撃の
動作を実装

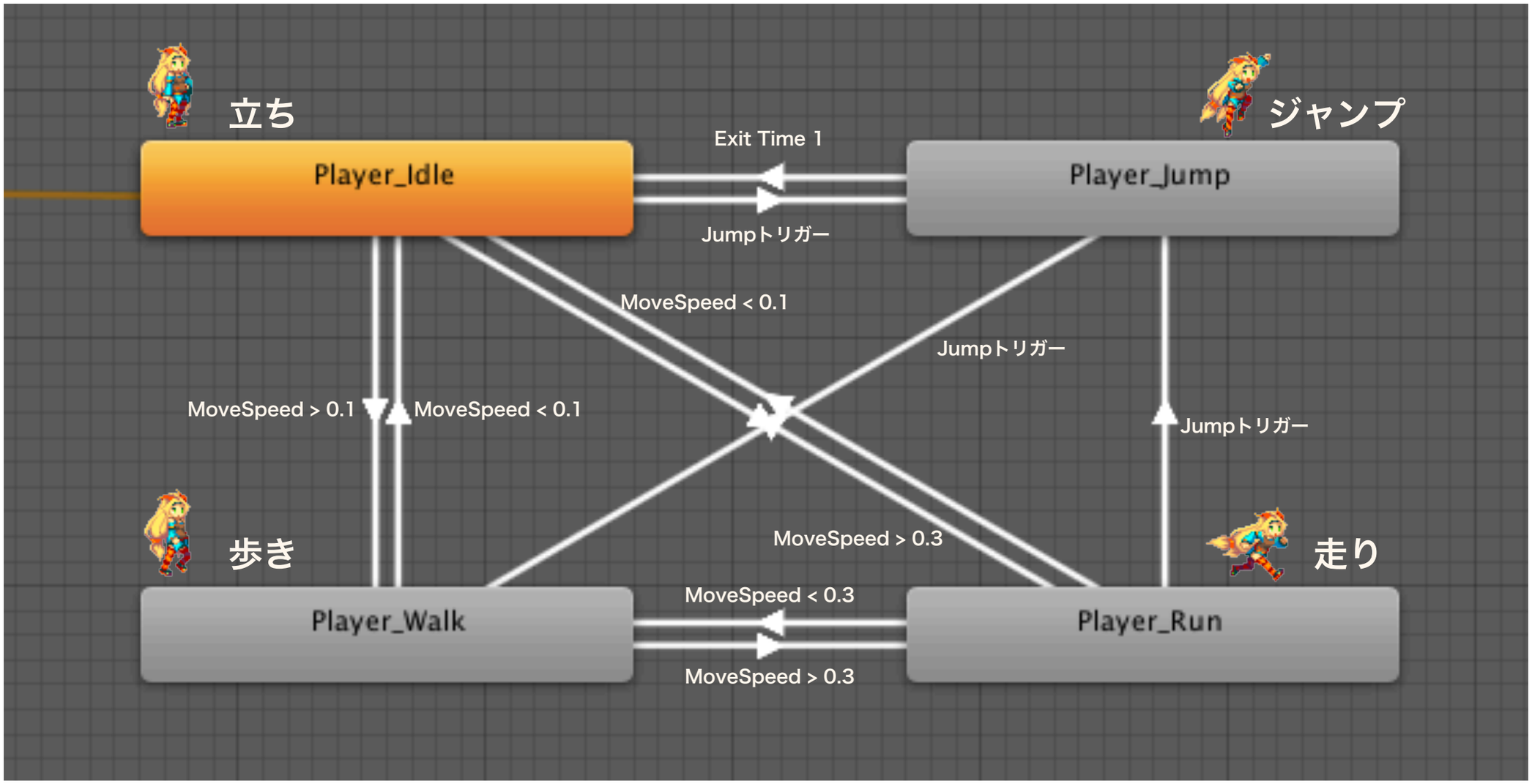
攻撃の部分を詳しく解説します！

PlayerMain

キー入力を取得し、
PlayerControllerを動かす

Unityちゃんの基本構成

Animatorの構成



あとで攻撃のアニメーションを追加していきます！

Unityちゃんの基本構成

基本アクションのみを実装したスクリプト
→PlayerController_Basic

これをベースに攻撃アクションを加えていきます！

目次

攻撃アニメーションの作り方

←完成形

攻撃A,B,Cの3種類を作る予定

ポイント

- Spriteの並べ方
- Collider(剣)の動かし方

順番に説明していきます！

攻撃アニメーションの基礎知識

ニュートラル　テークバック　フォワード スイング　インパクト　フォースルー　硬直　戻りモーション

予備動作　攻撃判定　硬直

最低限、4枚で成立（ニュートラル, 予備動作, インパクト, 硬直）

スプライトの並べ方：ポイントは「ツメタメ」

- 力の入りどころ、抜けどころを意識するとリアルに
- 攻撃前後のスキの長さ→ゲームバランス

今回はUnityちゃんの素材が
充実してるので
あまり関係ないです

攻撃アニメーションの基礎知識

ニュートラル　テークバック　フォワード スイング　インパクト　フォースルー　硬直　戻りモーション

予備動作　攻撃判定　硬直

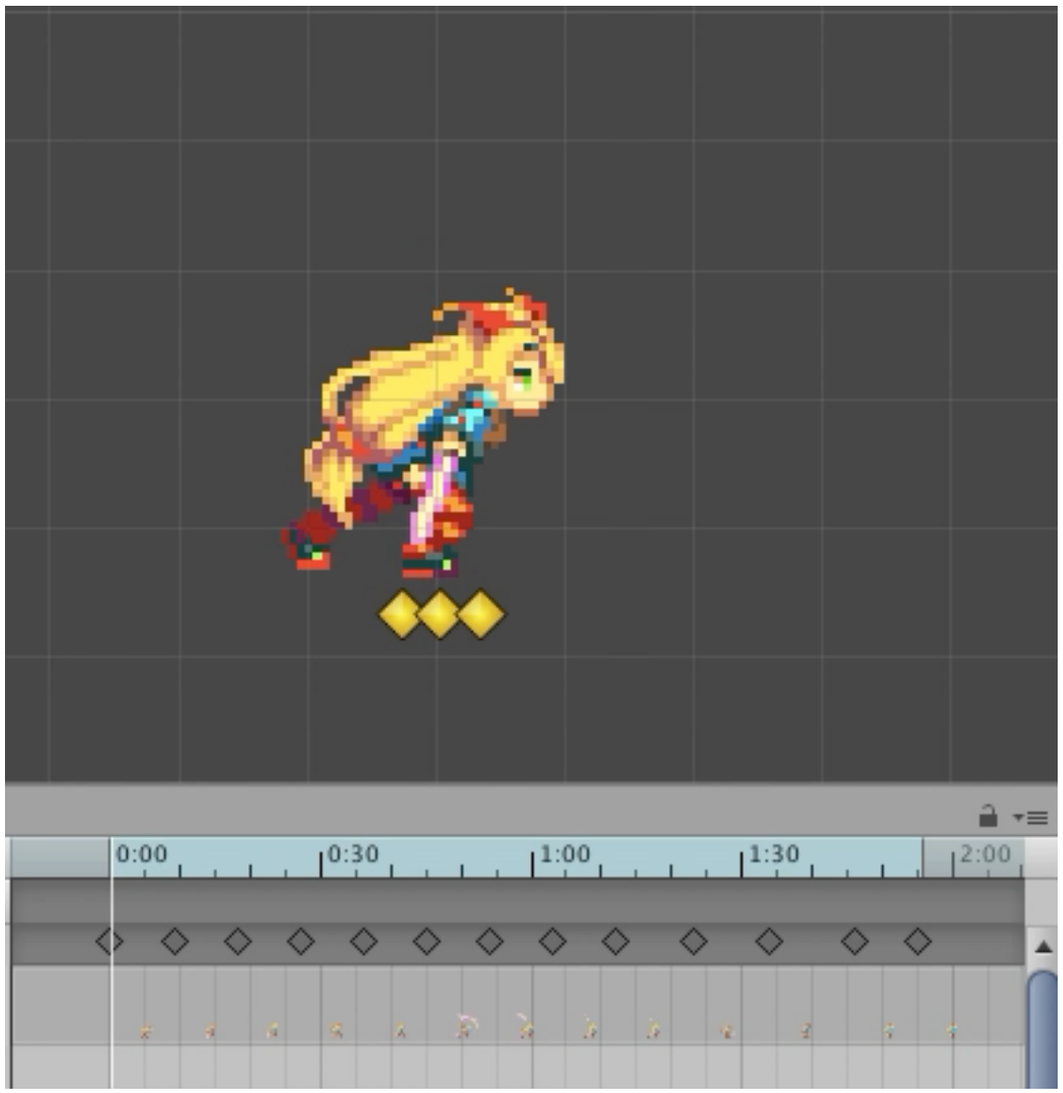
今回意識したこと

・連続攻撃の気持ち良さを高めたい・・・予備動作を**ツメ**る

・大振りの攻撃・・・攻撃後の硬直を**タメ**る

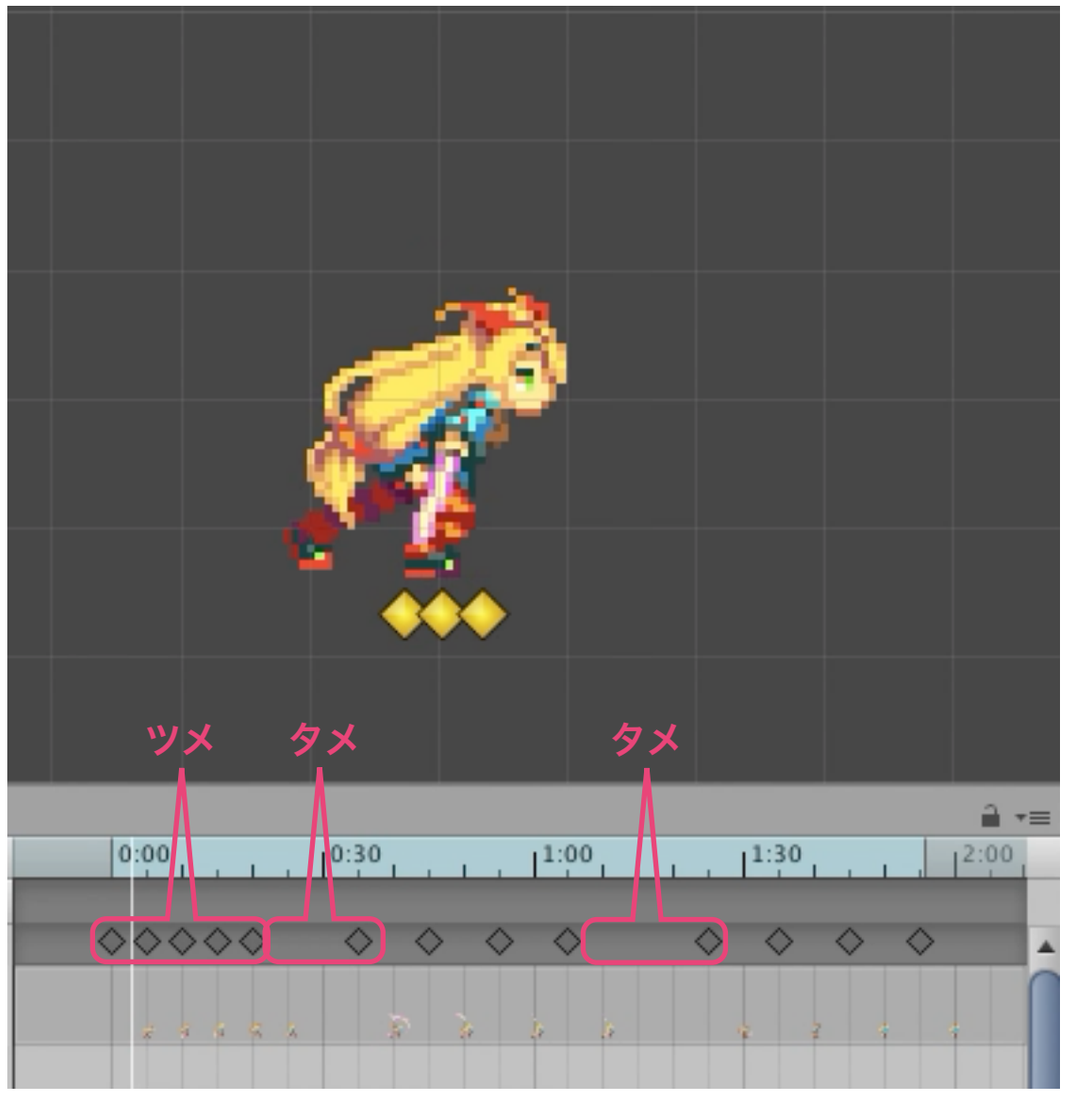
アニメーションの作成

Spriteを等間隔に並べたver



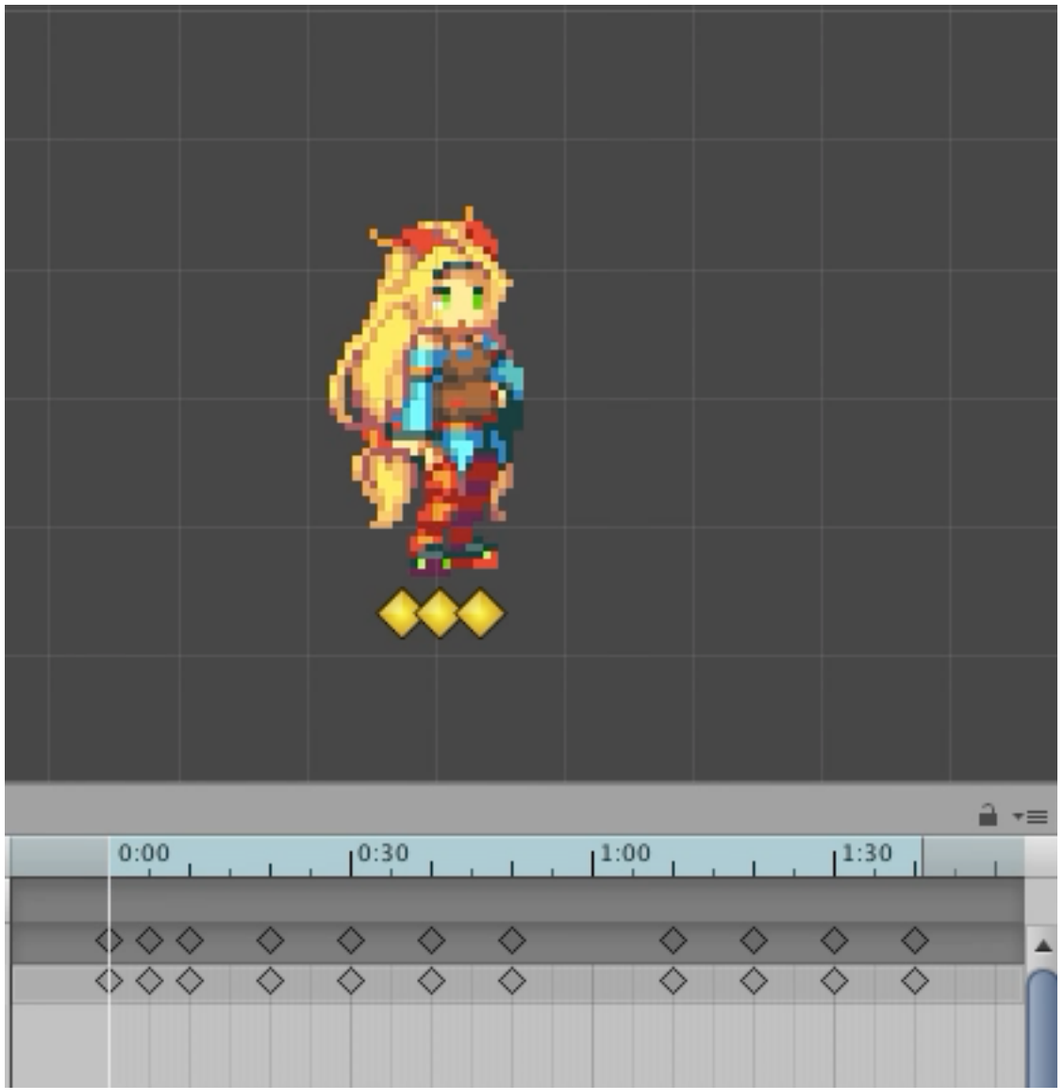
予備動作　攻撃判定　硬直

ツメ・タメを意識したver



予備動作　攻撃判定　硬直

アニメーションの作成



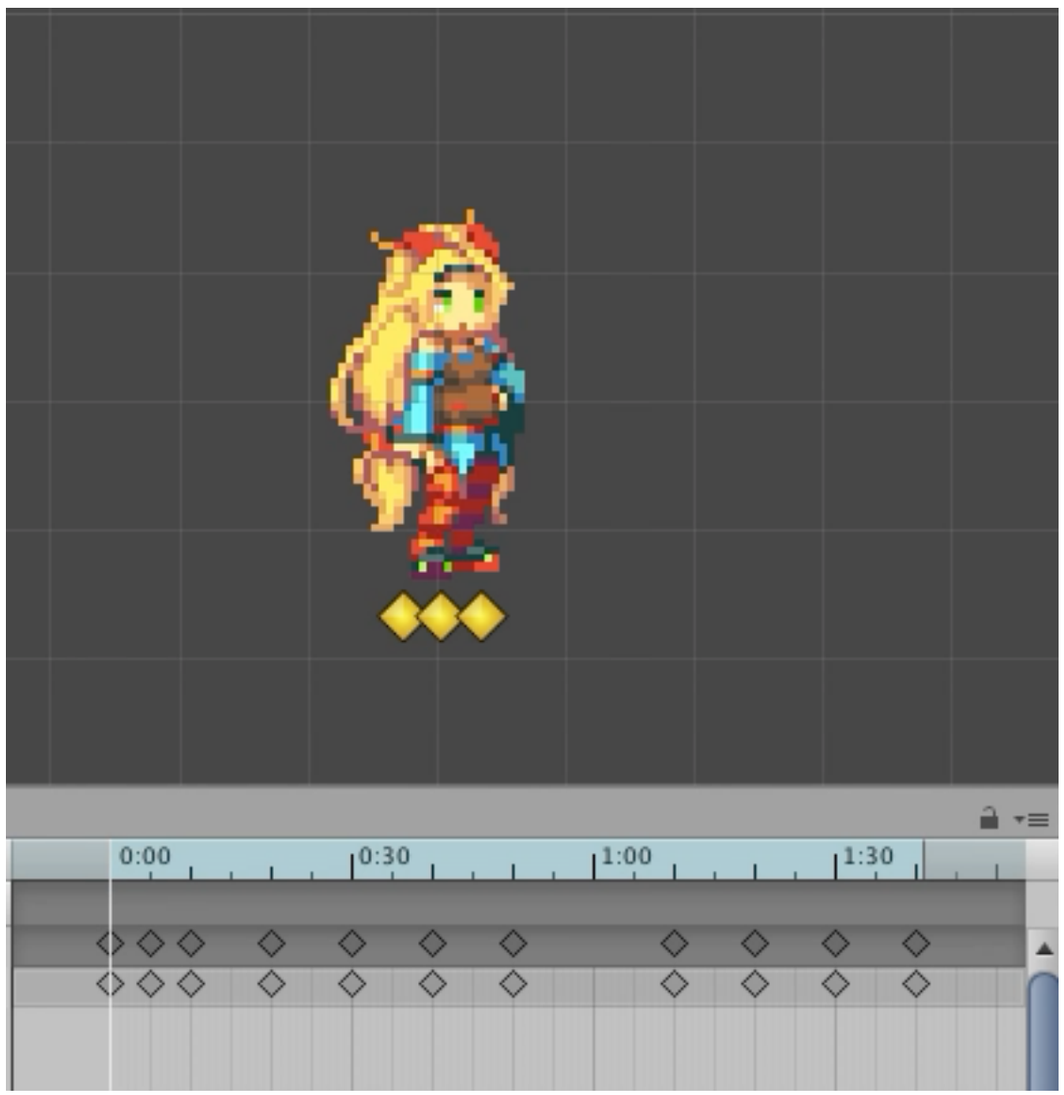
ツメタメを意識して並べた！

実はある違和感が…

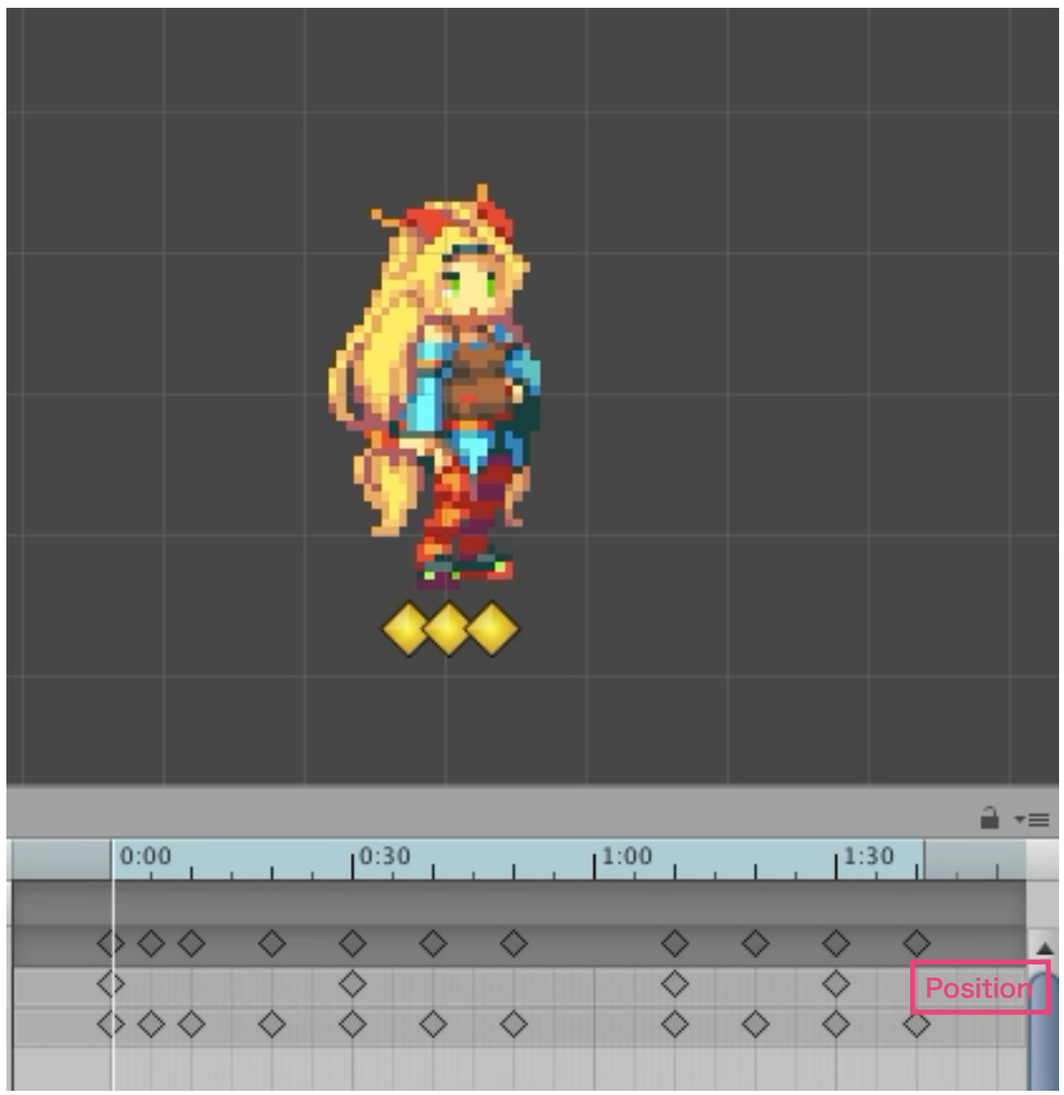
みなさん、どう思いますか？

アニメーションの作成

さっきのやつ



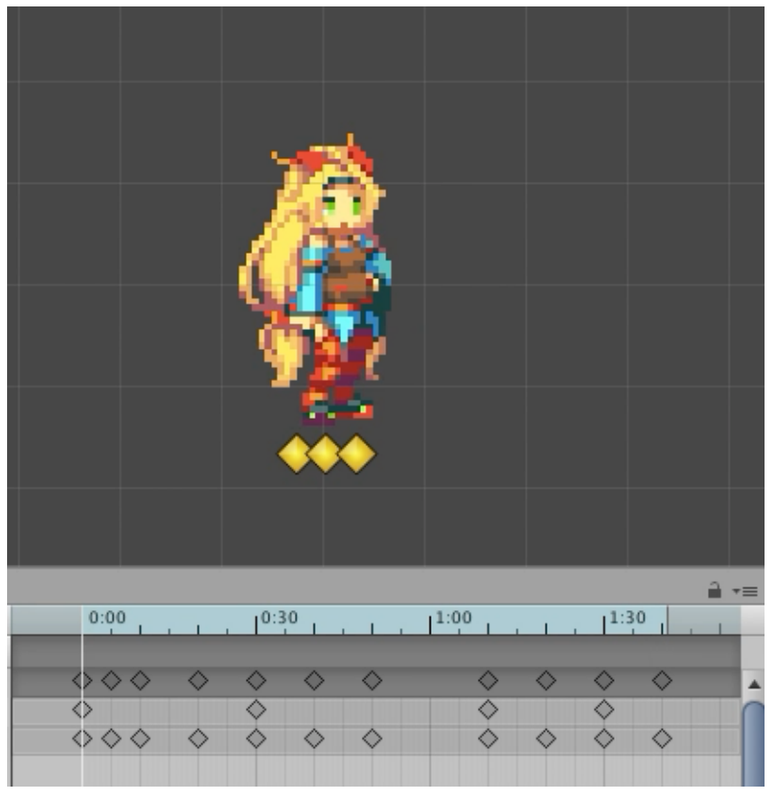
Positionを調整したver

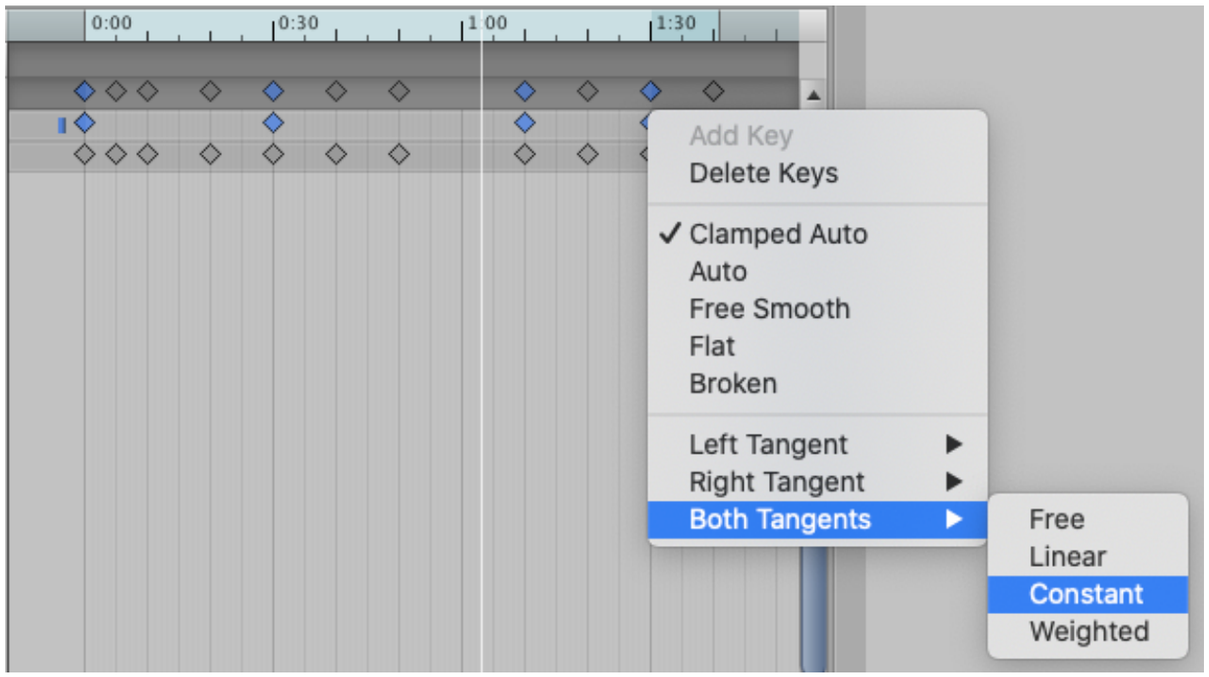


足の位置を合わせるように調整すると良い感じ！

アニメーションの作成

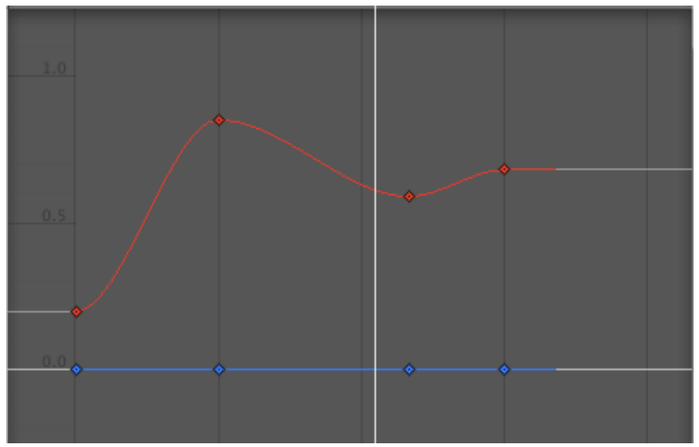
デフォルトのまま



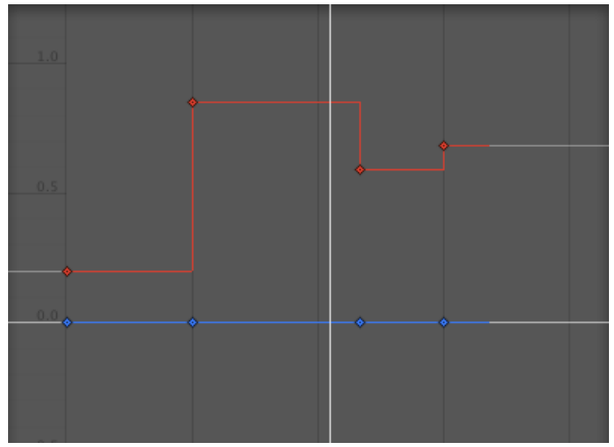


キーを選択して

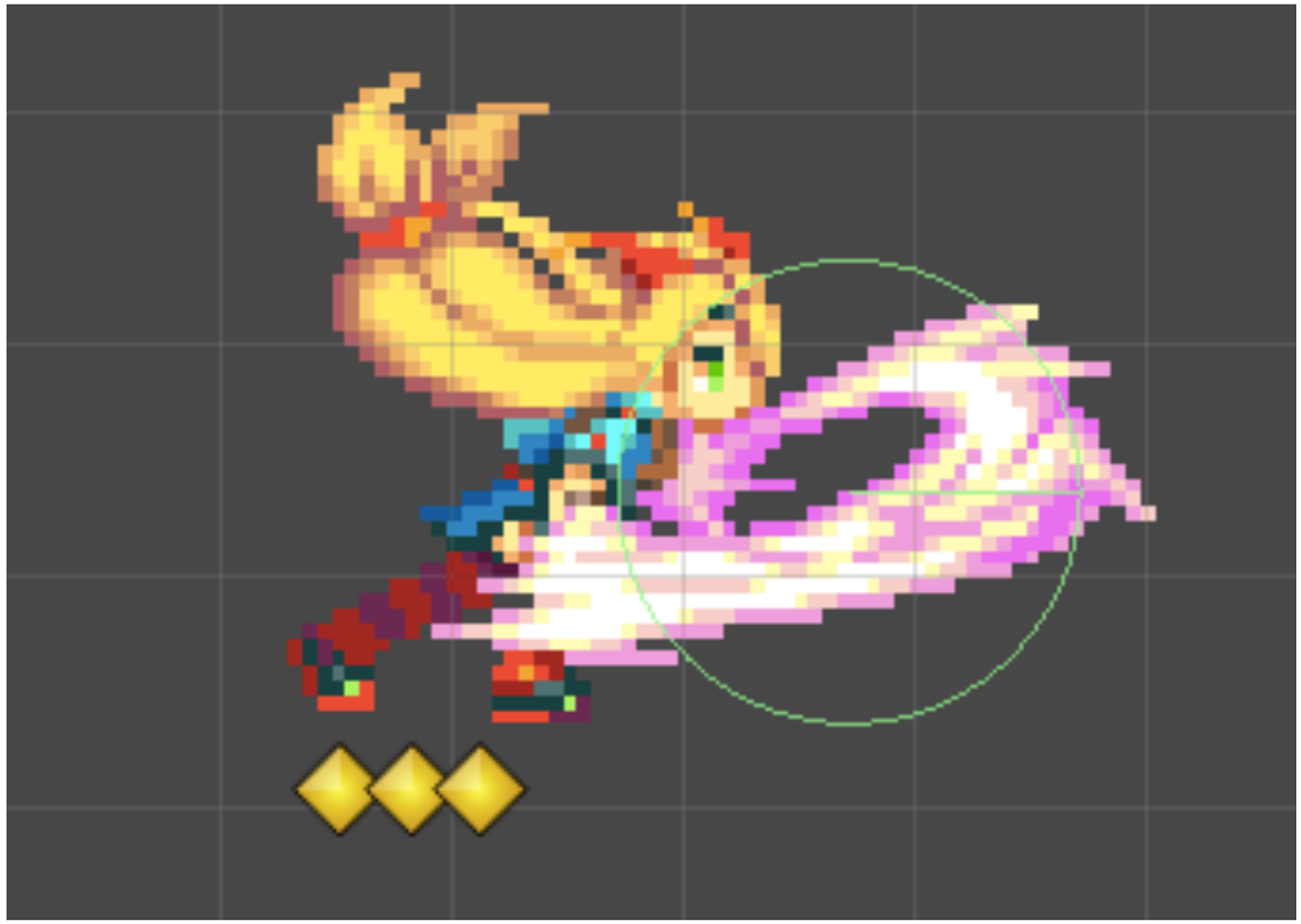
Both Tangents → Constant



間が補完されてしまう…



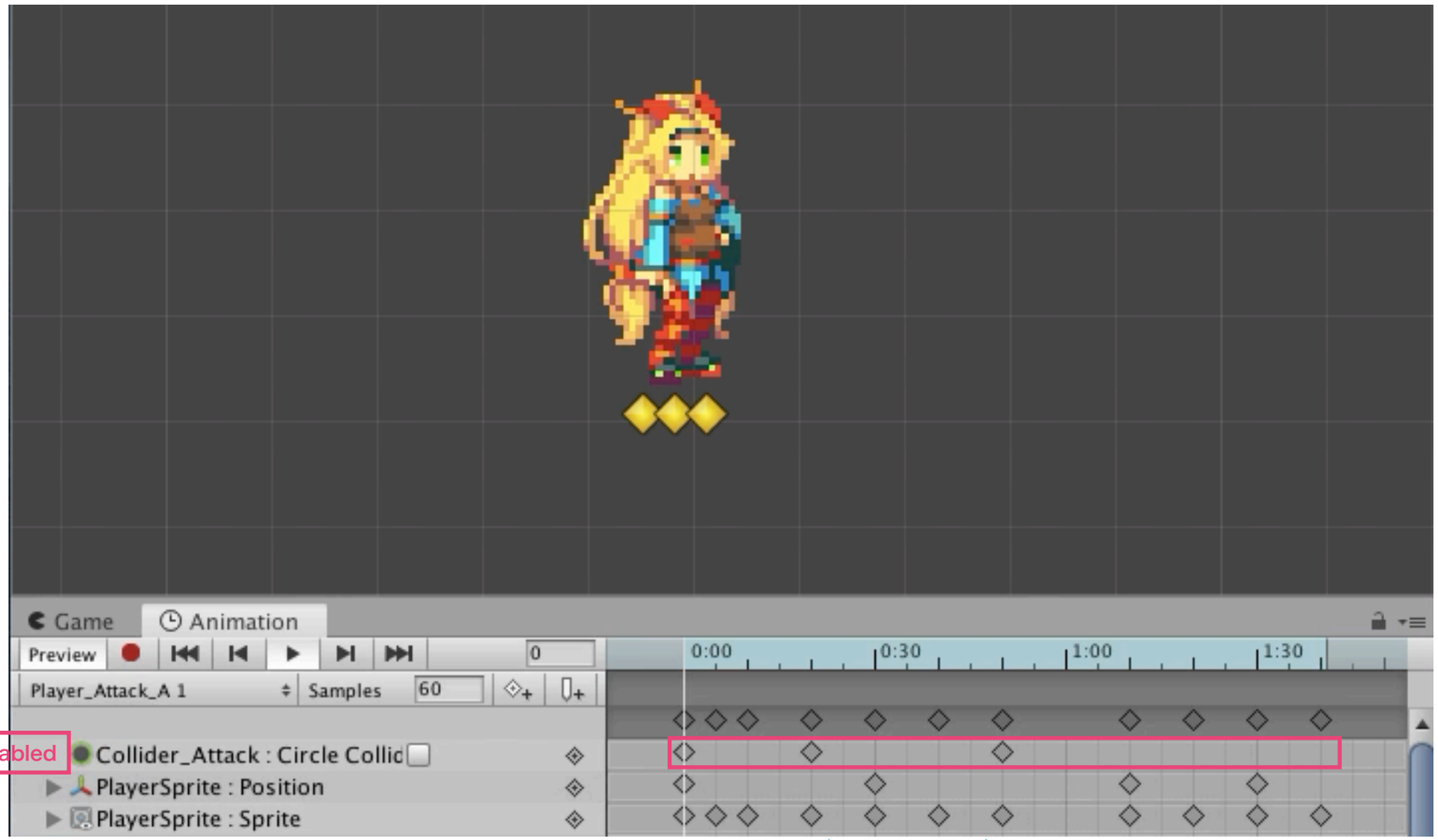
アニメーションの作成



Spriteを並べ終わったので、
これからColliderの動きを追加していきます！

アニメーションの作成

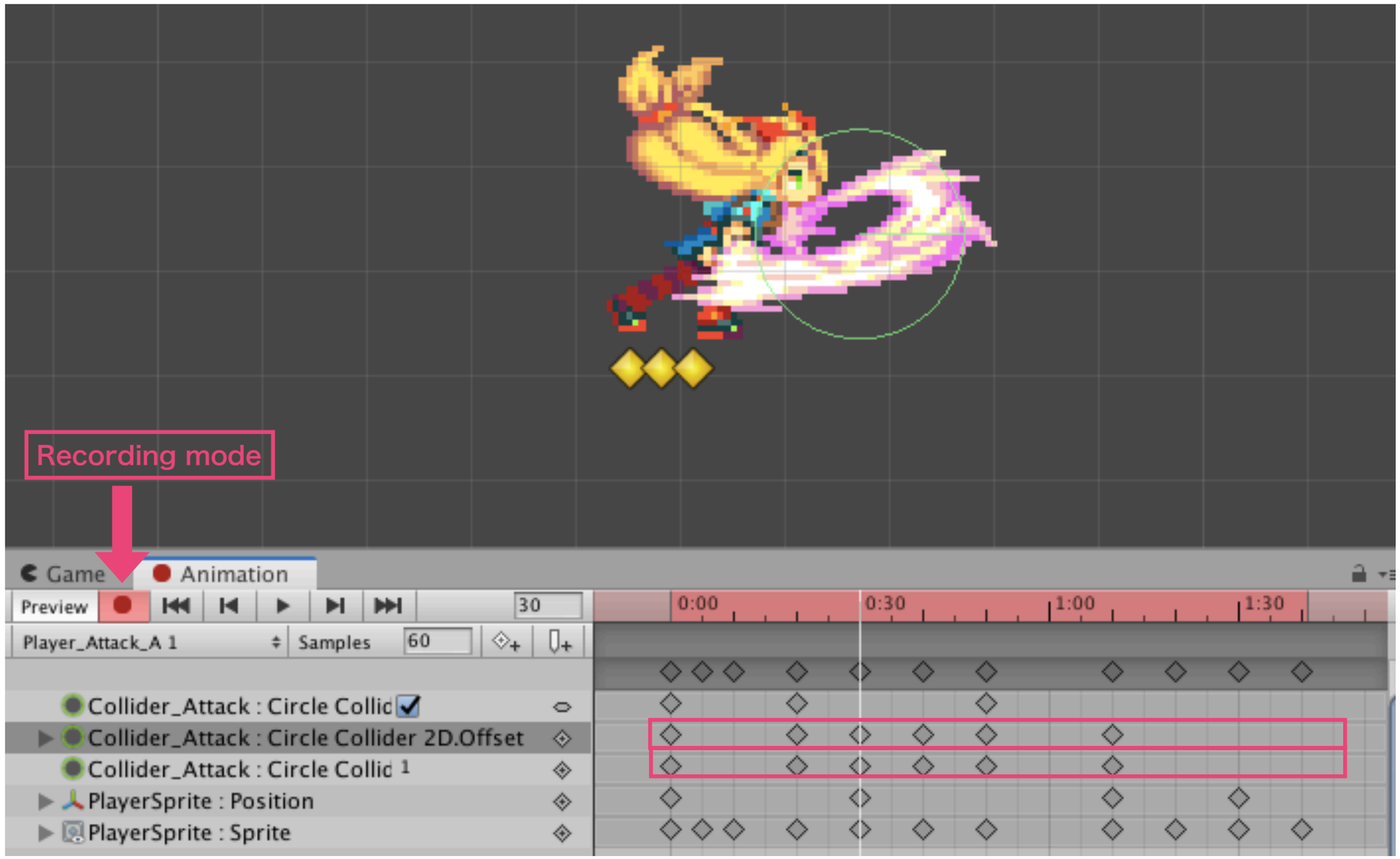
Colliderのアニメーション追加：Enabled



攻撃判定

アニメーションの作成

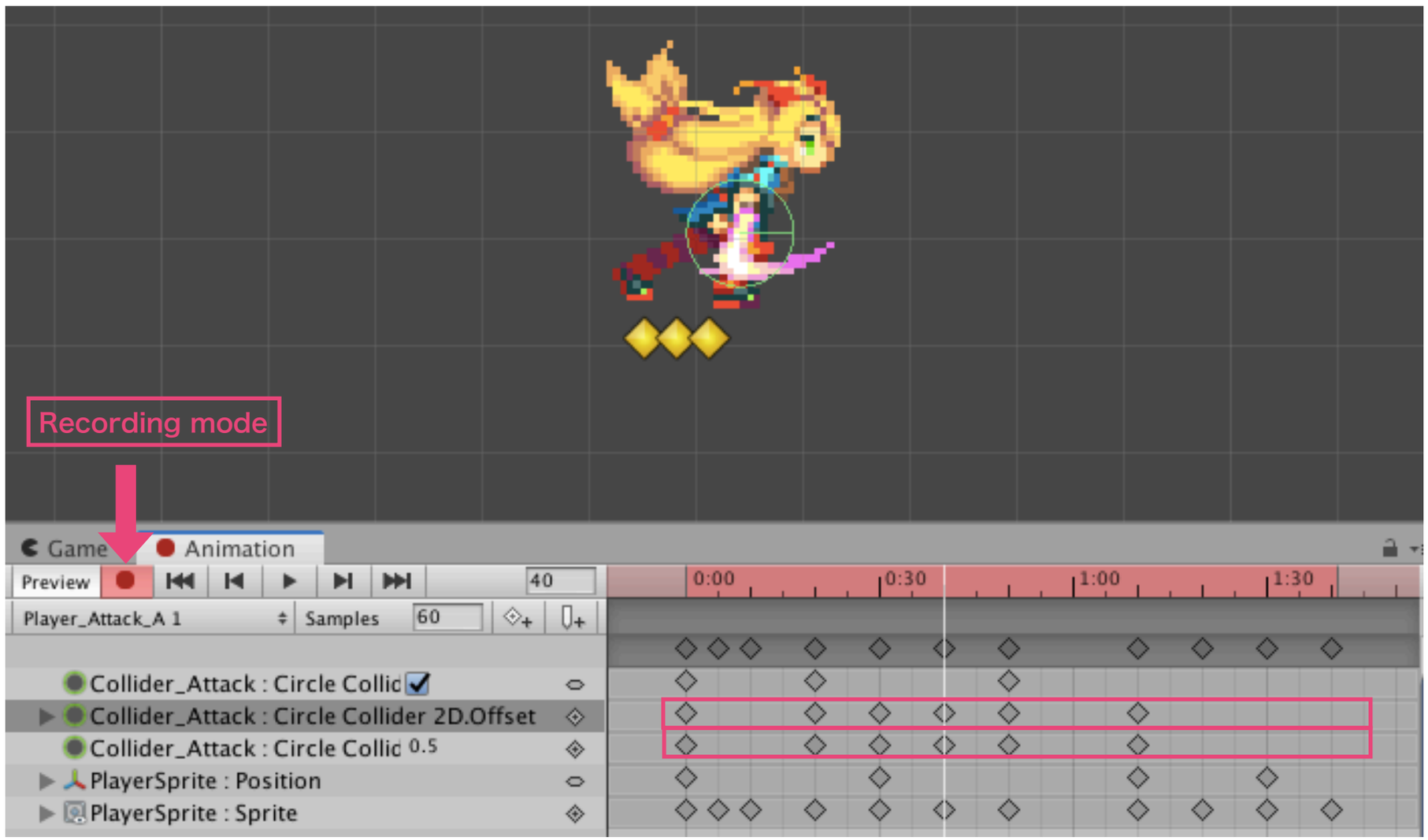
Colliderのアニメーションを追加：Radius, Offset



各Spriteのキーに合わせて、レコーディングモードで編集

アニメーションの作成

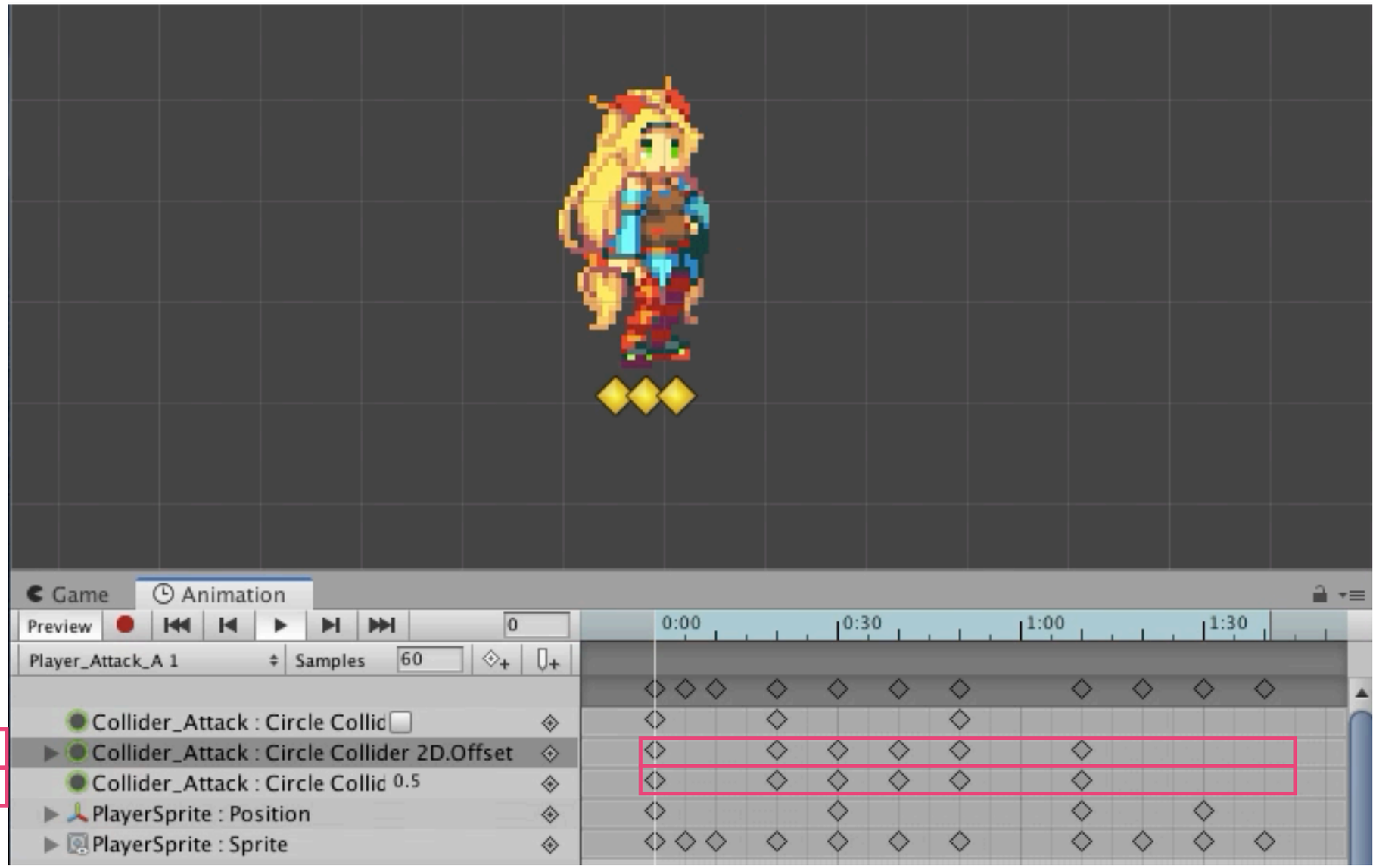
Colliderのアニメーションを追加：Radius, Offset



各Spriteのキーに合わせて、レコーディングモードで編集

アニメーションの作成

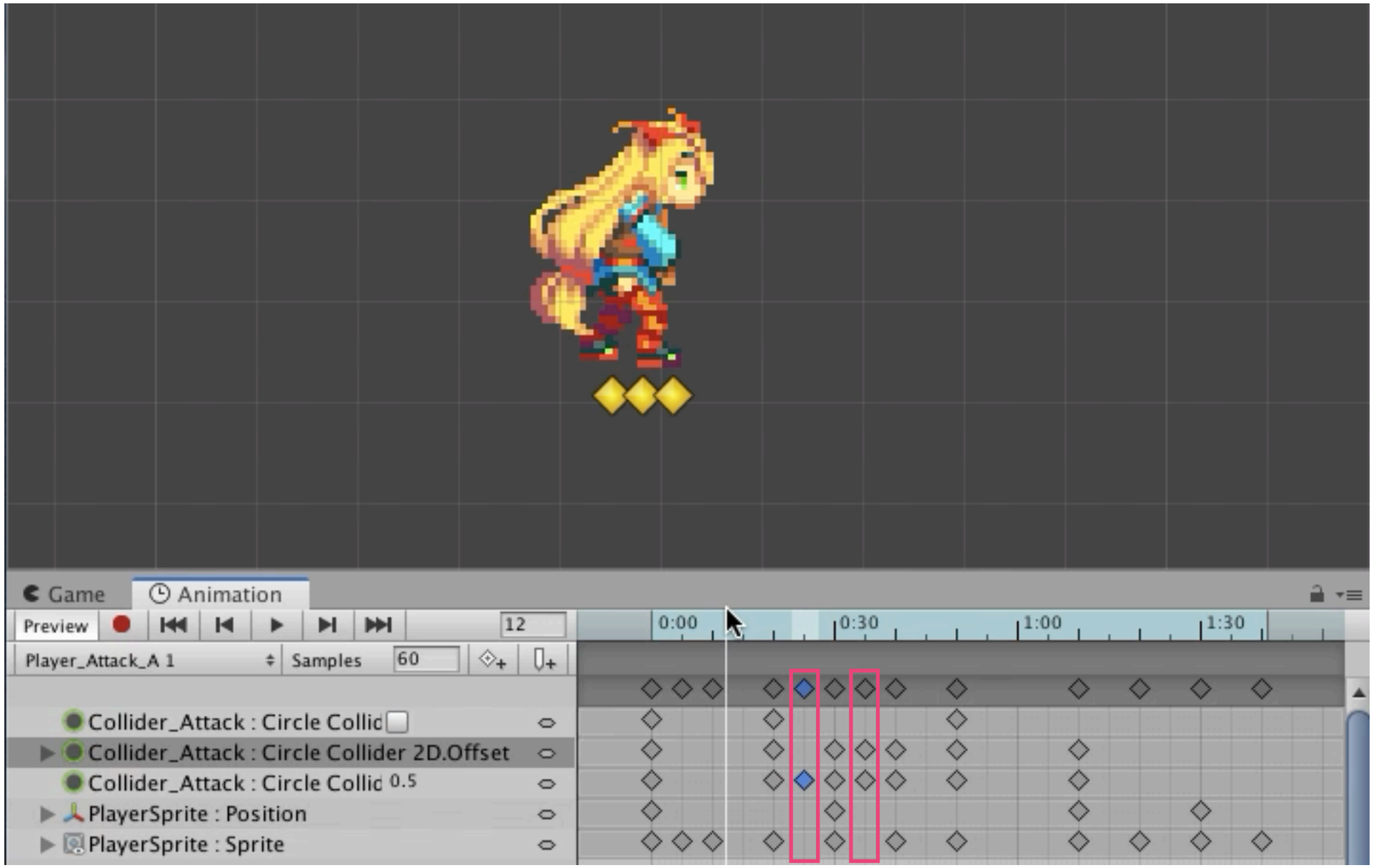
Colliderのアニメーションを追加：Radius, Offset



剣より、コライダーの方が速く見える…？

アニメーションの作成

Colliderのアニメーションを追加：間の調整



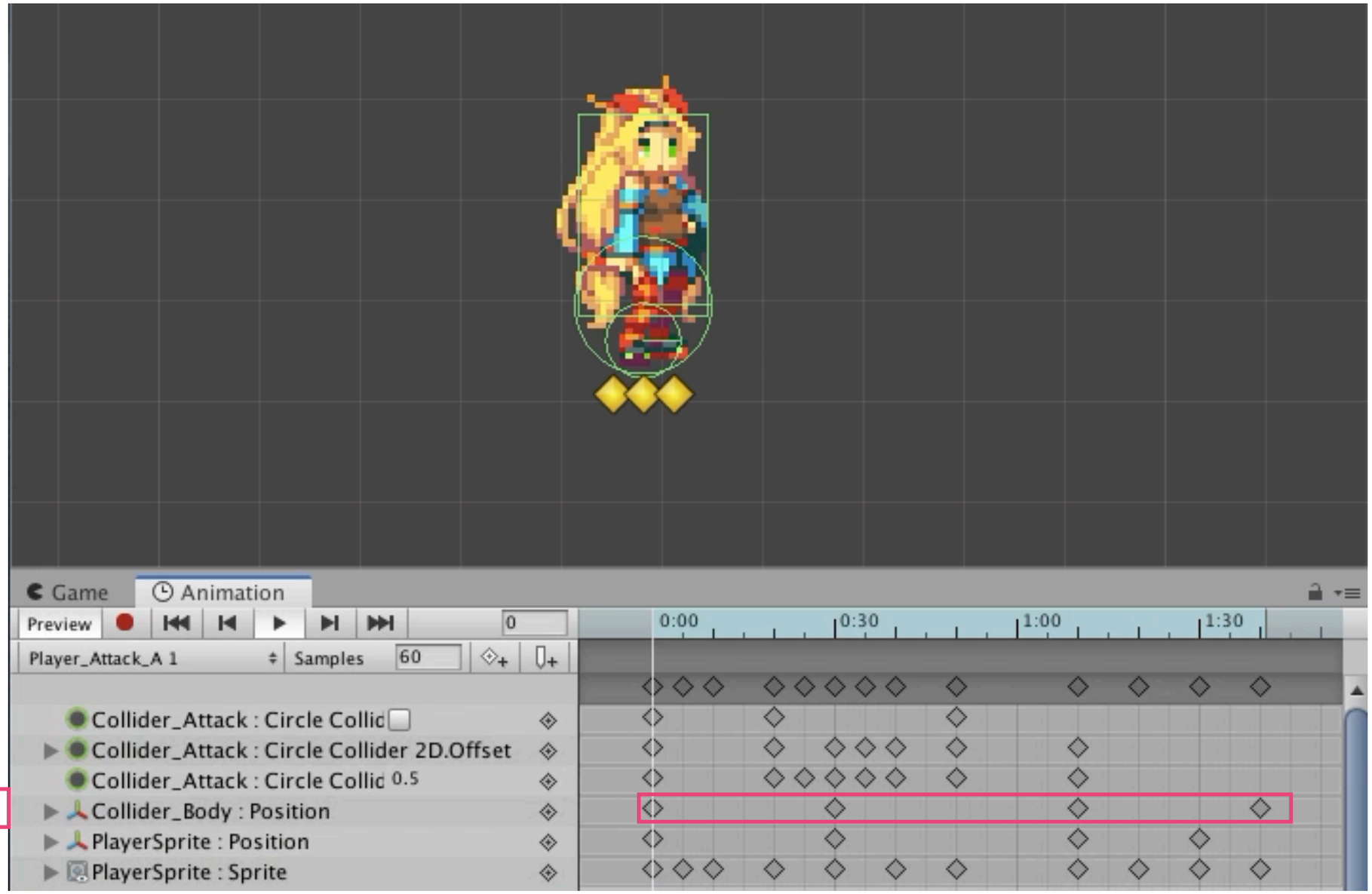
Radius　0　0　1　1　.5

アニメーションの作成

Colliderのアニメーションを追加：間の調整

アニメーションの作成

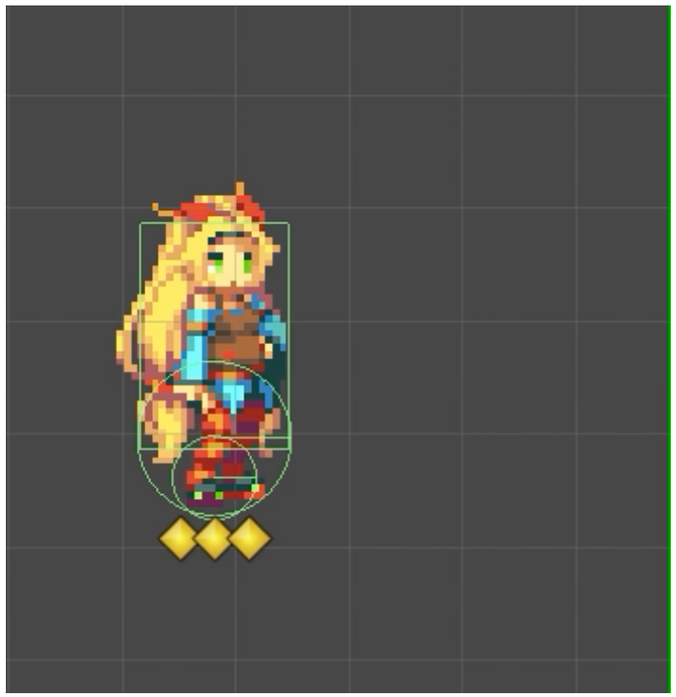
Colliderのアニメーションを追加：Collider(体)のPosition



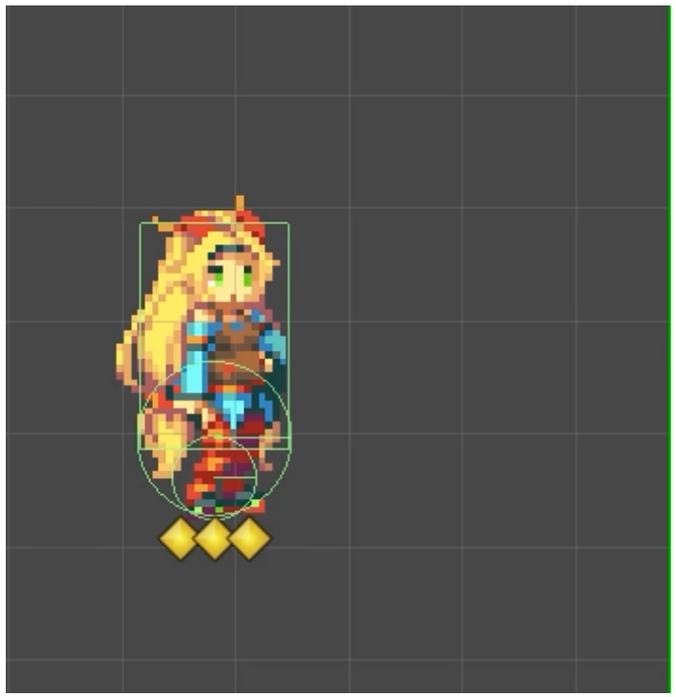
Sprite Positionと同様にConstantに設定

アニメーションの作成

同じ手順で3種類作成！



攻撃A



攻撃B

攻撃C

小振り ⟷ 大振り

攻撃範囲 小
硬直 小

攻撃範囲 大
硬直 大

目次

コンボ攻撃の実装

アニメーションのステートを把握する

```
public static int ANISTS_Attack_A = Animator.StringToHash("Base Layer.Player_Attack_A");
```

```
public void ActionAttack()
{
    AnimatorStateInfo stateInfo = animator.GetCurrentAnimatorStateInfo(0);

    if(stateInfo.fullPathHash != ANISTS_Attack_A)
    {
        animator.SetTrigger("Attack_A");
    }
    else
    {
        animator.SetTrigger("Attack_B");
    }


}
```

ステイト名の文字列から生成したハッシュ値で比較する

(文字列をそのまま比較するより速い)

コンボ攻撃の実装

具体的な実装

```
public static int ANISTS_Attack_A = Animator.StringToHash("Base Layer.Player_Attack_A");
public void ActionAttack()
{
    AnimatorStateInfo stateInfo = animator.GetCurrentAnimatorStateInfo(0);

    if(stateInfo.fullPathHash != ANISTS_Attack_A)
    {
        animator.SetTrigger("Attack_A");
    }
    else
    {
        animator.SetTrigger("Attack_B");
    }

}
```

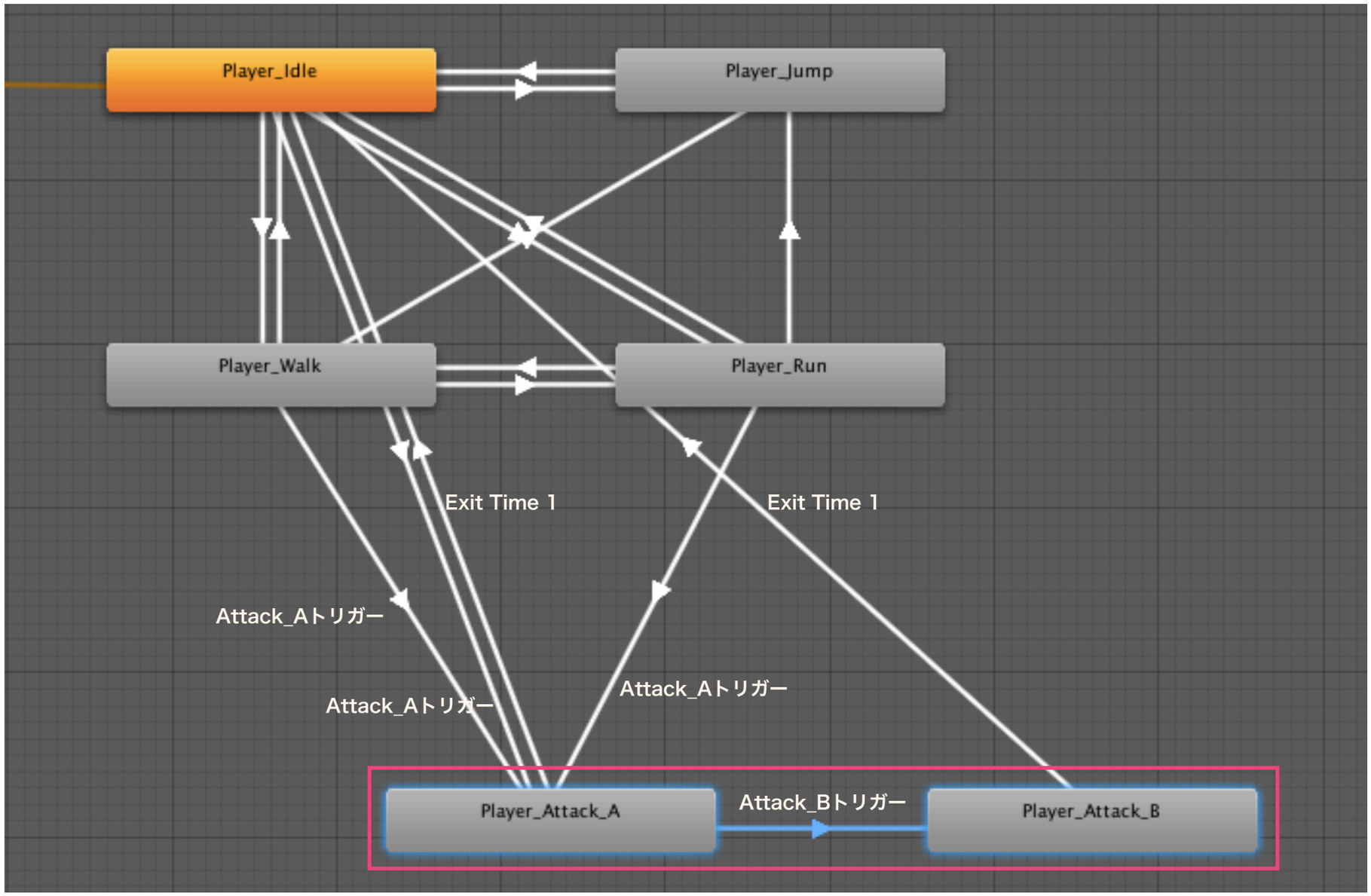
Action Attack：攻撃ボタンが押されたら実行される関数

今のステートが、

- 攻撃A以外なら、攻撃Aトリガー**ON**！
- 攻撃Aなら、攻撃Bトリガー**ON**！

コンボ攻撃の実装

Animatorの設定



この実装で本当に大丈夫…？

コンボ攻撃の実装

実はダメな実装例！

理由1：アニメーション処理が別スレッドで行われている

攻撃ボタンを素早く連打したら…？

攻撃Aトリガーを**ON**にした瞬間、
攻撃Aのアニメーションが再生されるわけじゃない

ズレが生じる

理由2：遷移タイミングが遅すぎる

攻撃Aが戻りモーションまで終了してから攻撃Bへ遷移…

理想は攻撃Aの途中で硬直キャンセルして攻撃Bへ

コンボ攻撃の実装

実装の方針：コンボ入力受付フレームの設定

攻撃A

コンボ入力受付

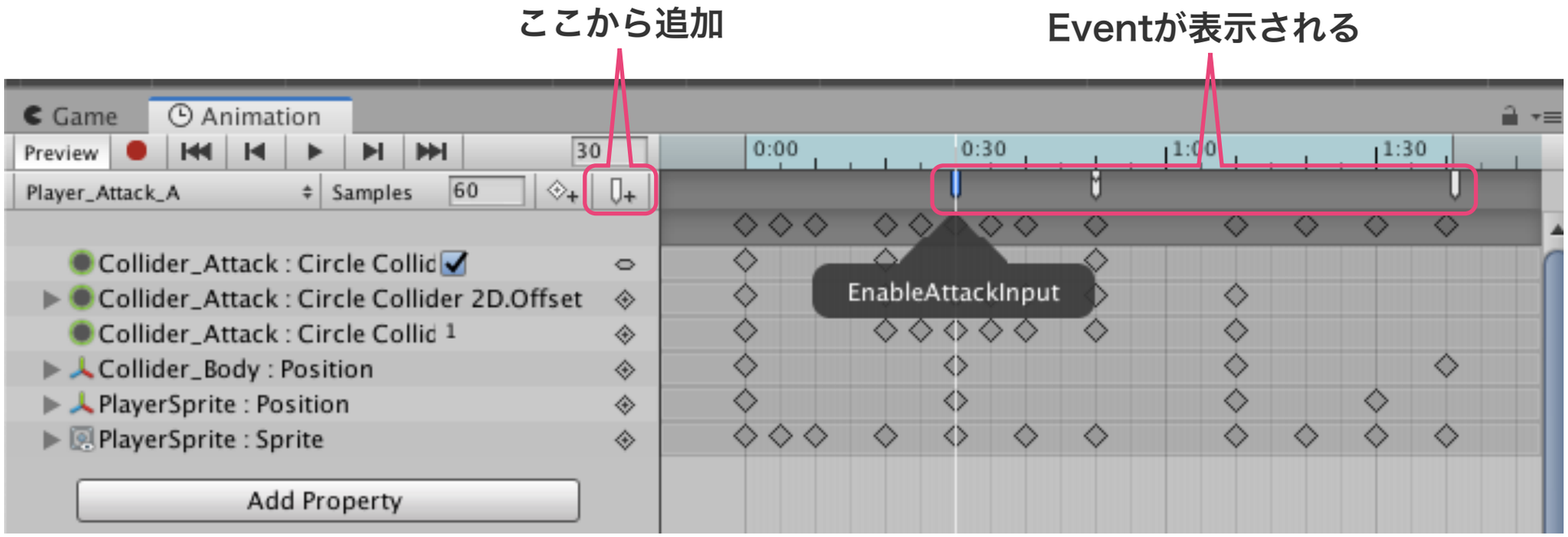
- 入力があったら攻撃Bへ
- アニメーション側で遷移処理を呼び出す
→ズレが起きない
- 硬直キャンセル→OK

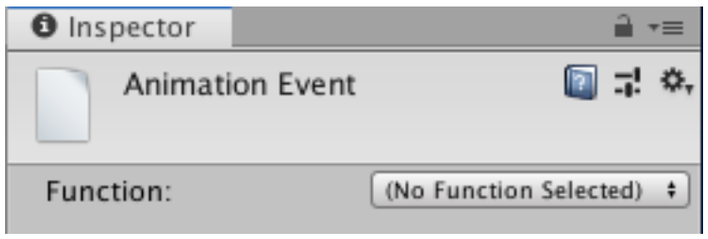
攻撃B

コンボ攻撃の実装

実装の方針：Animation Eventについて

特定のタイミングでスクリプトの関数を呼び出せる機能





呼び出す関数はInspectorで選択する

コンボ攻撃の実装

具体的な実装：Animation Event用の関数

```
private bool atkInputEnabled = false; //trueの間、コンボ入力受付
3 references
private bool atkInputNow = false; //コンボ入力が行われたかどうか
```

```
public void EnableAttackInput()
{
    atkInputEnabled = true;
}

0 references
public void DisableAttackInput()
{
    atkInputEnabled = false;
}
```

```
0 references
public void SetNextAttack(string stateName)
{
    if(atkInputNow == true)
    {
        atkInputNow = false;
        animator.Play(stateName);
        ResetPosition();
    }
}
```

2種類のフラグを用意！

EnableAttackInput

↕ コンボ入力受付フレーム

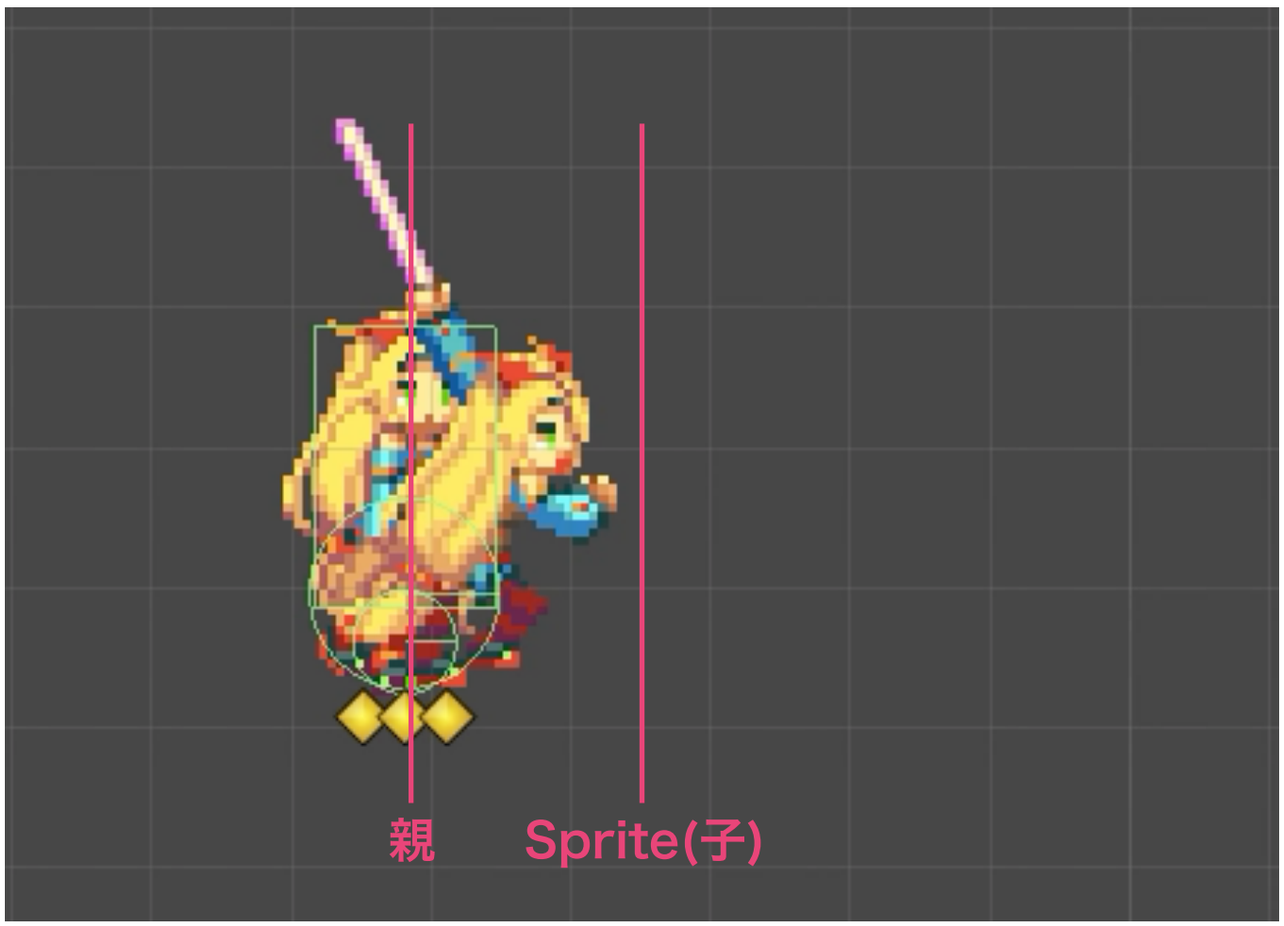
DisableAttackInput

Animator.Play関数
→強制的にステイトを遷移する

ResetPositionについて→次のスライド

コンボ攻撃の実装

具体的な実装：Animation Event用の関数



アニメーション前後で子の位置が変化する

ResetPosition：親と子の位置関係のズレをリセットする処理

コンボ攻撃の実装

具体的な実装：ActionAttack（攻撃ボタンを押されたら実行される関数）

```
public void ActionAttack()
{
    AnimatorStateInfo stateInfo = animator.GetCurrentAnimatorStateInfo(0);

    if(stateInfo.fullPathHash == ANISTS_Idle ||
        stateInfo.fullPathHash == ANISTS_Walk ||
        stateInfo.fullPathHash == ANISTS_Run)
    {
        animator.SetTrigger("Attack_A");
    }
    else
    {
        if(atkInputEnabled)
        {
            atkInputEnabled = false;
            atkInputNow = true;
        }
    }
}
```

立ち, 歩き, 走りの時
↓
攻撃A

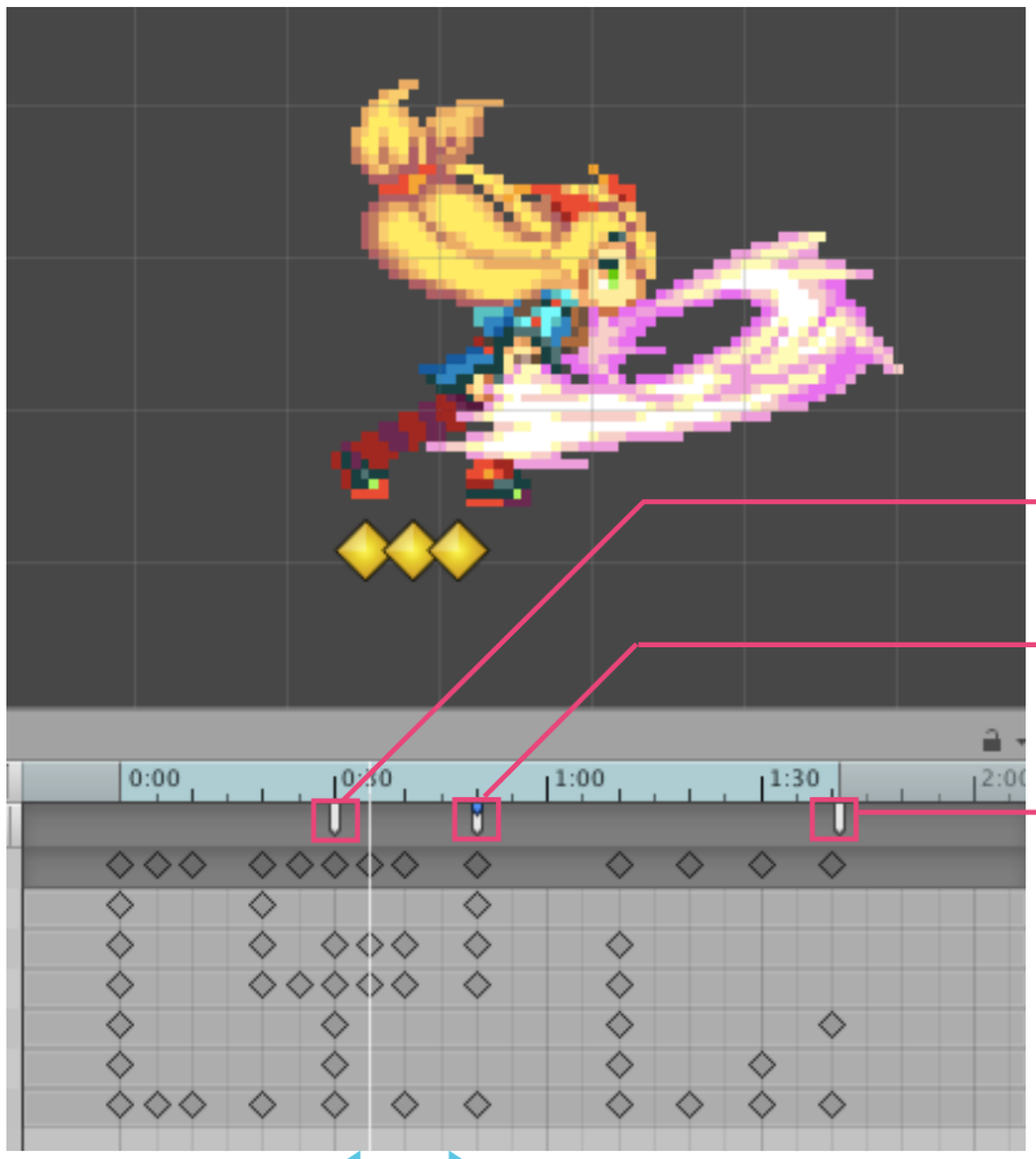
コンボ入力受付フレームのとき
↓
フラグ変更

ActionAttackでは**フラグを変えるだけ**でOK

コンボ攻撃の実行はAnimation Eventで行われる

コンボ攻撃の実装

具体的な実装：Animation Eventの設定（攻撃A）



EnableAttackInput

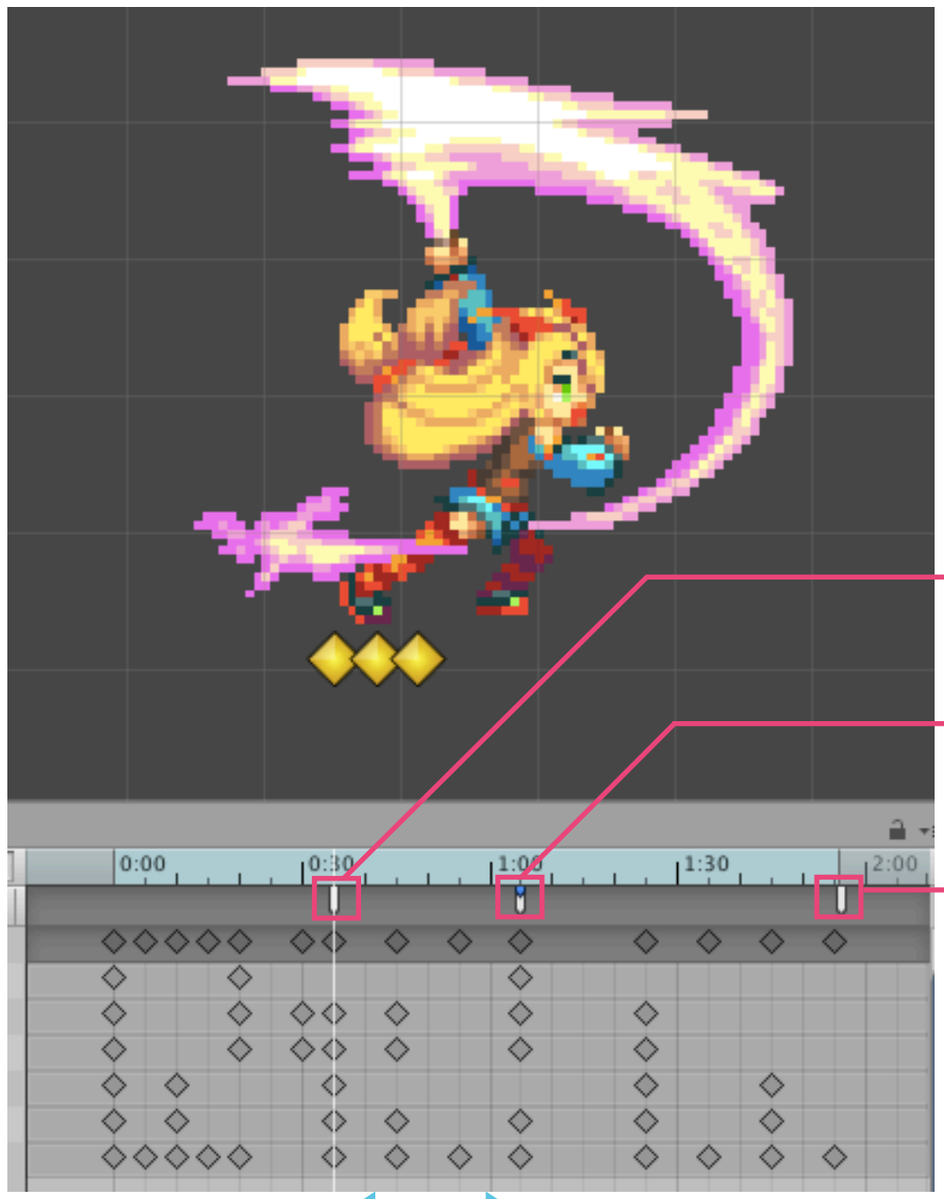
DisableAttackInput
SetNextAttack(攻撃B)

ResetPosition

コンボ入力受付

コンボ攻撃の実装

具体的な実装：Animation Eventの設定（攻撃B）



EnableAttackInput

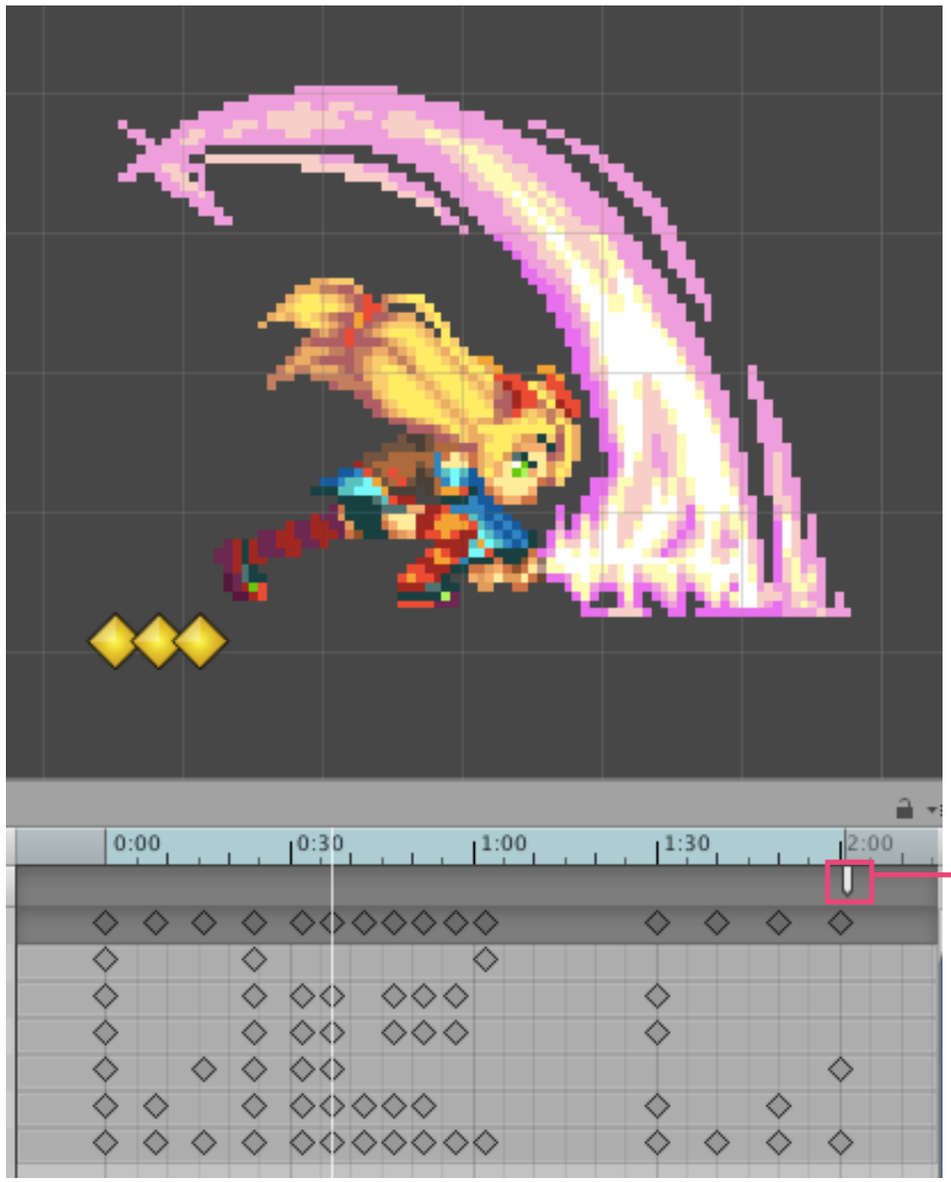
DisableAttackInput
SetNextAttack(攻撃C)

ResetPosition

コンボ入力受付

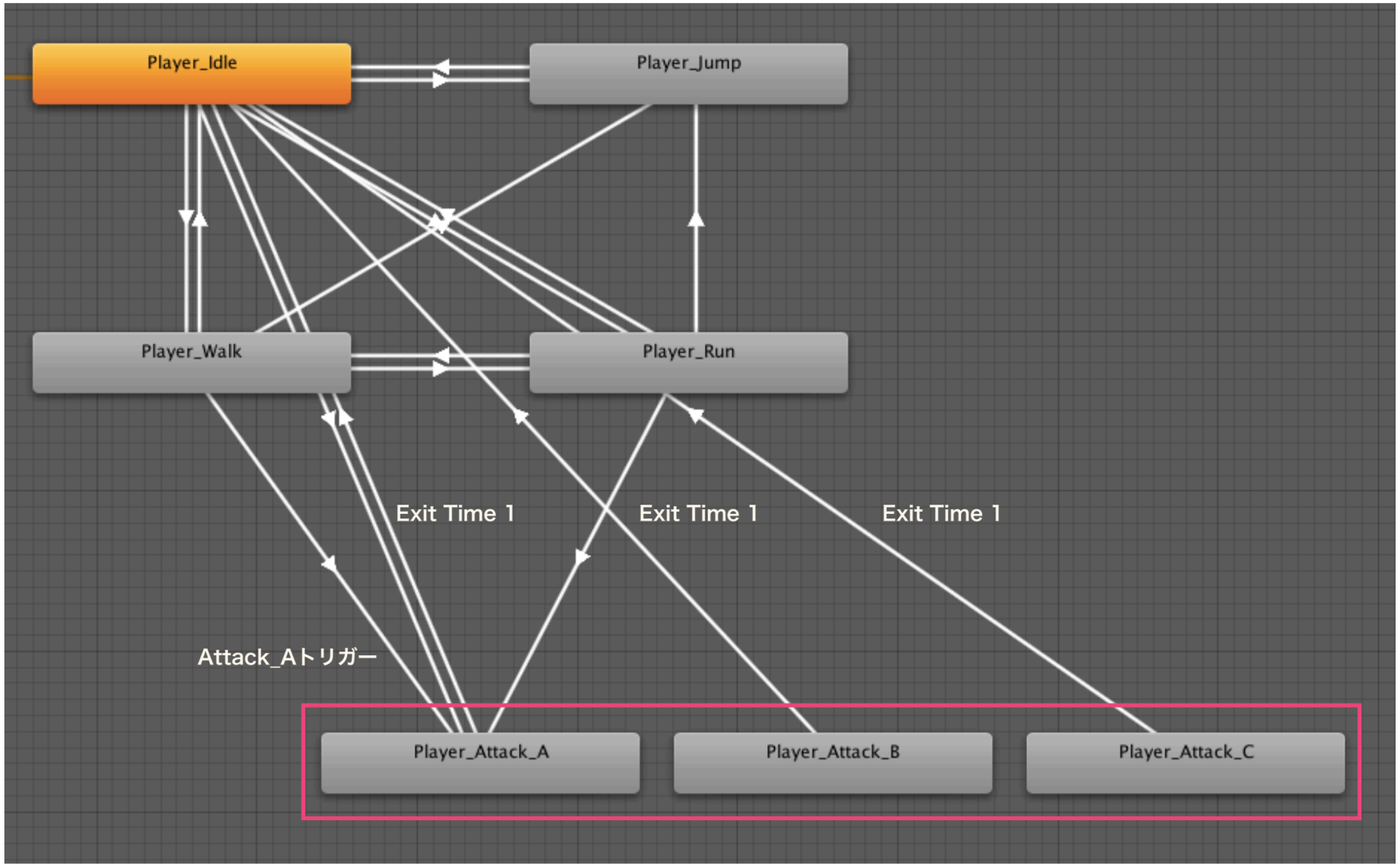
コンボ攻撃の実装

具体的な実装：Animation Eventの設定（攻撃C）



コンボ攻撃の実装

Animatorの全体図



コンボ攻撃はAimator.Playで実行する

→**攻撃B, Cへの遷移は作らない**（戻る経路だけでOK）

コンボ攻撃の実装

これにてコンボ攻撃の実装が完了です！

終わりに

その他のトピック
- 2段ジャンプ
- ジャンプ攻撃
- 地面チェック
- 攻撃中の移動停止
- コンボエフェクト
- 敵のスクリプト
- 敵のアニメーション
- 敵のノックバック
- ヒットエフェクト

興味ある人↓

GitHub：https://github.com/chocsar/SmashUnityChan.git

後日、記事にもまとめようかなと思ってます🤔

FIN.