FPSゲームを作ろう

Let's create FPS!

Unity3D で FPS ゲームを作ろう。
僕のワークショップは、割と難しいかもしれないけど、3D のゲームを作る上で必要なことなので、覚悟してね。

ひとまず StandardAsset(最初からインポートできる部品)を使ってゲームを作っていきましょう。
Import→Characters→Characters のチェックをいったん外す。Characters の中の FirstPersonCharacter にのみチェックを入れて、Import

Create->light->DirecitonalLight

Create->3D->Terrain

シーンが 2D ビューになっていることがあるので、その場合はトグルを外してください。これから 3D ゲームを作るので。

あと、MainCamera の AudioListener は外しておく。

ローポリピストルと、ゾンビを用意します。

後述する「アセット」を使えば、より楽により本格的に Unity ゲームを作っていけますが、初歩のうちは、使いこなせないと思いますので、ある程度

最初からインポートできるアセットでもそれなりのものは作っていけますが、
AssetStore からいろいろなパーツをダウンロードすると、よりゲームがそれっぽくなっていきます。

AssetStore には、様々な無料アセットがあり、お金をかけなくても結構いいものが使えます。PM 4O などの PBR 系のモデルはマシンパワーをアホほど使いますので、お勧めしません。
マシンパワーに不安がある人は「ローポリ」系のモデルを使用するといいでしょう。
リアル感は落ちますが、動かない、重いよりもましです。

とりあえず FPS を作りたいので、銃がないと始まりません。一番シンプルで軽い
CustomizableFirearmSet」を使いましょう。デフォルトはピストルを使用します。

単体で見るとチャチに見えるかもしれないけど、ゲーム画面に乗っけるとそれっぽく見えます。

ひとまずシーンの中に入れたら、FirstPersonController の中にドラッグ&ドロップして、
ピストルの座標を(0,0,0)にします。

ゾンビインポート

ゾンビ歩きは1周すると動きがとまるため、Project->Z@Walk->animation にて、Loop をオンにする。

そのままだと、動かない的(マト)でつまらないため、AI(知能)をもたせましょう。
お待ちかね「プログラミング」のお時間です。

FPS 等の敵の動きのアルゴリズムを単純に説明すると

- 徘徊行動(ウロウロ)
- 索敵行動(どこかなー)
- 追跡行動(見つけたーっ)
- 攻撃行動(ぶっとばす)

の4つに分けられます。
プレイヤーとの位置関係(主に距離)や、その時の状況でそれぞれの行動を切り替えていきます。これにより「意志を持った」かのような行動を実装することができます。

コンピュータ用語で、状態が変化することを「状態遷移」と言いますが、今回はこれを実装します。

それぞれ説明すると「徘徊行動」とはランダムウォーク。つまりあてもなく彷徨うことです。これをプログラミングで実装するならば、目隠しして、ぐるぐる回って止まった時の前方方向に向かって歩く。
もしくは酔っ払いの千鳥足のようにある程度の方向は持っているようだが、やっぱり定まらない。そんな実装でいいでしょう。

索敵行動では、ある程度「匂い」などによって、こちらに近づいてこようとするが、確信的でないため「まっすぐこちらへ向かってくる」ほどではない。
立ち止まってキョロキョロするのもこの行動。

次に追跡行動ですが、これは完全に「目視され」てしまって、まっすぐ近づいてきてしまうような状態ですね。

この状態になったら、とにかくゾンビを殺すしかない。いつか食べられてしまいます。

最後に「攻撃行動」ですが、もうこれは攻撃範囲内に入ったら、一足飛びに攻撃する。です。
シンプルですね。

ひとまずは、ランダムからやってみましょう。

ゾンビ選択して、AddComponent→New Script にて、ZombieAI と入力してください。
ZombieAI をダブルクリックすると、以下のようなスクリプトがでてくると思います。
ここにゴリゴリプログラムを書いていくんです。

今回使用するプログラミング言語はC#と言って、割とゲーム業界でも使用されている言語です。現状のゲーム業界で使われている順では

1. C++
2. C言語
3. Java
4. ObjectiveC++
5. C#
6. Javascript
7. BluePrint

などですが、Unityで開発する会社も増えてきており、習得の容易さと、C/C++に近い点から今回はC#を使います。

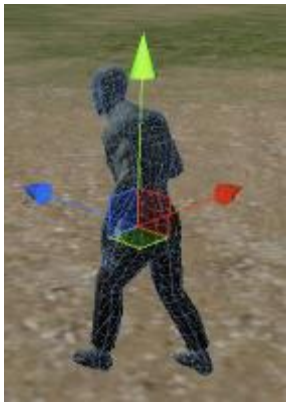
```
using UnityEngine;
using System.Collections;

public class ZombieAI : MonoBehaviour {

    // Use this for initialization
    void Start () {

    }

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

    }
}
```

まぁ、英語が出てきて、びびったかもしれない。マジで。そうはいってもプログラミングなんてそんなもんだ。そこは割り切ってアキラメロン。

さて、ひとまずはここにプログラムを追加して、ランダムウォークしてもらいます。最初に必要な処理は「向いている方向に移動する」だと思いますので、それを実装しましょう。

向いている方向とはどこの方向なんだろう？

Unity でゾンビを選択すると矢印がでてくると思うが、これがゾンビが持っている「座標系」となる。



青い矢印が「Z 方向」と言う。ゾンビが向いている方向が Z 方向だ。これを覚えておいてくれ。とにかくこの Z 方向に移動しないと、後ろ歩き、カニ歩きみたいな動き動きをして不自然だ。
さて、動かしてみよう。ちなみに、今回の話では 3D のためなんどもベクトル(Vector3)が出てくるが、これはつまり xyz 方向をまとめて表現しているものだと思ってくれ。

Update(){
　　の中に、プログラムを書くと、ゾンビにいろいろな行動をとらせることができる。

とりあえずは前に動かそう。移動は transform.Transform 関数を呼ぶことで可能だ。

試しに、
transform.Transform(new Vector3(0,0,1));

と書いてみてくれ。実行してみればわかると思うけど、ゾンビが超スピードで駆け抜けてしまう。
newVector3(0,0,1)ってのは Vector3(X 方向,Y 方向,Z 方向)の移動量を決めているのだが、Z 方向の 1 ってのが大きすぎるようだ。
さすがにこれではいけない(笑)

こういう時は数値をかけて(乗算して)速さを調整する。

```
public class ZombieAI : MonoBehaviour {
　　//ここに以下のコードを書いてくれ
```

```
public float speed=0.5f;
```

と。
このspeedは速さ調整のための変数で、先ほどのベクトルにかけることによってスピードを調整できる。

```
transform.Transform(new Vector3(0,0,1)*speed);
```

で、このspeedの値はエディタからも調整できる。

で、0.5でもやっぱり相当に速い。さすがにこれでは扱いづらい。なぜこのようになっているのかというと、Update関数はおよそ1/60秒という超短時間でコールされるためだ。これを秒単位調整したい場合は、Time.deltaTimeをさらにかけることにより、秒単位での調整ができる。

```
transform.Transform(new Vector3(0,0,1)*speed*Time.deltaTime);
```

これでだいぶいい感じの移動量になったのではないかと思う。ちょっと踏み込み時に滑っている感じがするかもしれないが、これに対応するのは1日のワークショップじゃ無理なので、今後の課題としてほしい。

これで前方向への移動はできた。次は向きの決定だ。
ランダムにしたい。ところで、毎フレームランダムにすると、それはそれでせわしない。数秒おきに方向を変更したい。どうすればいいだろうか？

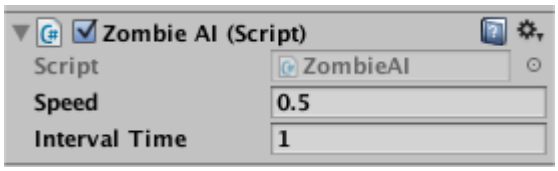
インターバル(間隔)を最初に決めておき、その時間ごとに処理をするには？
とりあえず、最初に間隔は定義すべきなので、

```
public class ZombieAI : MonoBehaviour {
	public float speed=0.5f;
	//ここに以下のコードを書いてくれ
	 public float intervalTime=3.0f;//時間間隔
```

を定義しておく。さらに今度はprivateで現在の経過時間を定義する。

```
public class ZombieAI : MonoBehaviour {
        public float speed=0.5f;
         public float intervalTime=3.0f;//時間間隔
         //ここに以下のコードを書いてくれ
         private float passedTime=0.0f;//経過時間
```

public と private の違いについて話すと長くなるので、とりあえず今はこう打ち込んでおいてくれ。簡単に言うとこういうことだ。



ご覧のように、public で宣言したものは外から値の変更が可能であり、private で宣言したものは、値の変更ができないというわけだ。インターバルは調整したいからね。逆に経過時間は外から扱われると、正確な時間が測定できず、困るというわけ。

Update 関数の最初で

```
if (Time.time > passedTime) {
            passedTime = Time.time + intervalTime;
}
```

こういう風に書いてくれ給え。
これがどういうことを意味しているかわかるだろうか？

Time.time とは現在時刻を表す。とするとどうなるかな？しばらく考えてください。これが 1/60 秒の間隔で、何度もなんども実行されたとしたら、どういう風に動くかな？

そう、指定した時間間隔ごとに if のカッコ内を実行するのだ。ここに方向転換コードを書けば、ゾンビが一定時間ごとに方向を変えるわけ。

C#では、ランダム値は Random.range(最小値,最大値)で取得できます。

角度は 0　360 で指定する。

回転は transform.Rotate(new Vector3(X 回転,Y 回転,Z 回転));で指定できる。
今回は Y 回転だけ指定すれば、ゾンビは自分の方向を変更する。

んだけど、それだと、あまりにもピコッピコッって動いてゾンビっぽくないので、動きに制限をつける。
だいたい-15　15 でいいんじゃないかな。

それでもピコピコ動く場合の対処法として、Quaternion.Slerp 関数を使って補間する方法があるが、これはちょいと難しいので、今回時間が余ったらにしよう。

ともかく、これでランダムに徘徊するコードができた。

次の動きに移りたいが、ひとまずは FPS らしく「うちころす」挙動を実装しよう。

まず、敵にコライダを設定する。AddComponent->CapsuleCollider で OK。

次に、プレイヤー側でマウス位置を取得し、マウスの座標と、自分の座標を結ぶ直線が敵に当たるかどうかの当たり判定を作りましょう。

FPSController に AddComponent->NewScript->Player を作ります。

今回もいじるのは Update
とりあえず、こんなコードを書いてください。

```
//マウスレイを取得
 Ray ray=Camera.main.ScreenPointToRay(Input.mousePosition);
 RaycastHit hitInfo;
 //なんかに対するあたり判定とその情報(hitInfo)の取得
if (Physics.Raycast (ray, out hitInfo, 10)) {
        Debug.Log( "なんかに当たった" );
 }
```

このコードを見て、なにやってるかだいたいわかった人はえらい。大雑把に言うと、マウス座標を元に光線を飛ばして、それが何かしらのオブジェクトに当たるかどうかを判定している。
ちなみに上の 10 は対象物との最遠距離限界

ちょっと実行してみよう。
ゾンビにある程度近づいて、ゾンビを狙えば「当たった」と出力されるはずだ。
だが、このままだと、ゾンビ以外にも反応してしまうため区別できるようにしておく。
ゾンビを選んだ状態で一番上の「Tag」を見てくれ。Untagged 担っているので、AddTag する。そうすると、タグ追加画面になるので、Enemy というタグを追加。
あくまでもタグ追加したのみなので、きちんとタグの選択まで完了しておくことを忘れずに。
そうすると、もし、Enemy のみに反応したければ上のコードをこう書き換える。

```
if (Physics.Raycast (ray, out hitInfo, 10)) {
 if (hitInfo.collider.gameObject.tag == "Enemy") {
                Debug.Log ("敵に当たったやで");
        }
 }
```

ここまで書いて、ゾンビを狙った時にのみ、メッセージが出れば正解や。割と敵の真ん中を狙わんと当たってくれへんので、あとあと調整が必要や

ただ、これは FPS ゲーム。メッセを出すのが目的ではない。敵を殺すのが目的である。

というわけで、左クリックしたら、先ほどの処理を行い、敵に当たったら、敵を消してみよう。
左クリックは
`if(Input.GetMouseButtonDown(0)){ //左クリック`
で、取得でき、

敵を消すには、さっきのあたり判定のところで、Debug.Log の代わりに

Destroy(敵オブジェクト);

ってやってやれば消える。

さて、これで、基本的な部分ができてきたわけだが、このままでは爽快感もへったくれもない。

こっからはいろいろとエフェクトを追加していこう。
まずはぶっ放した時に音を出そう。WeaponSoldierSoundsPackってのがミリタリー系の音を持っている。リアルなのかもしれないけど、あまりいい音じゃないね。

さて、これで音はいいんだが、敵が死んだ時の演出がちょっと薄い。
死ぬ時にティウンティウンさせてみよう。

と、まぁ、色々と書いてきたわけですが、ぶっちゃけこの授業「学生さんの制作の負担にならないようにしてください」と言われております。
つまり、自主制作有線くらいで考えておいてほしいという事です。ただ、かといって何もなしじゃ評価の仕様がないので、授業を受けた証拠として何かしらは提出物は欲しい所です。

というわけで、「特に何も作る気はないけど、単位だけは取得してやり過ごしたい」という不届きな野郎どものために、「アセットフリップゲーム」というのを紹介します。

何かというと、Unity には無数のアセットがありますが、中には完成ゲームがそのままあるものも少なくありません。

なので、何かしら購入し、ちょこっと変更してあたかも自分が1から作りました、な顔をして提出しましょう。

ただし、これをゲームコンテストに使うようなことはしないでください。あくまでも単位の間に合わせのためだけに使いましょう。以上です。