Unity と Wii リモコンで簡単アーケードゲーム制作

トミー (@Tomy_0331)　著

2019-01-26 版　　発行

目次

第 1 章

はじめに

1.1 この本について

「Unity を使えば簡単にゲームを作れますよ！」と聞いてダウンロードし，入門サイトを見ながら簡単なゲームはどうにかできたけど，その後どうすればいいの・・・と悩んでしまった方も多いのではないでしょうか．この本では「入門レベルよりちょこっとクオリティの高いゲームを作ってみたい」「弾転がしとかブロック崩しはもういい」「もっと Unity のいろんな機能を使いたい」というような Unity 初心者〜中級者手前の人をターゲットにしています．

また，タイトル通り文化祭などで"映える"アーケードゲームを作ることをメインとしています．アーケード，と言っても Wii リモコンと自作ペダルのみですがシンプルな操作性は子どもでも遊びやすくウケがいいです (体験談)．

なお，この本では Unity 画面の見方や操作など基本的な部分や機能の説明は省略することが多いです．プログラミングの C#における基本文法も説明を省かせていただくので，分からない場合適宜調べるかそのレベルに至ってない場合，一旦入門サイトでミニゲームを作ってみることをおすすめします．

1.2 開発環境

以下の環境・ソフトで開発しました．

Windows10

一般的な OS．Unity は Mac 版でも操作は基本的に同じですが，今回使う Wii リモコンやペダルの Mac での入力方法は対象外とさせてもらうのでご了承ください．Linux も同様です．

Unity2018.3.2f1

今回のメインで使うゲームエンジン．3D ゲームを作るのに最適で個人からゲーム会社まで広い場所で使われています．

blender

フリーの 3D モデリングソフト．造形からテクスチャのマッピング，アニメーショ

ン作成までできます．3D ビューの操作が独特で混乱しがち．

Visual Studio 2017

Unity でスクリプトを書くときに使います．Unity 環境の補完もしてくれます．

第 2 章

Unity でゲーム部分を作ろう

2.1　プロジェクトを作成する

Unity を起動し「New」から新規に作るプロジェクトの設定をします．Project name は「shoot」など任意の名前，Template は「3D」，他パスの設定などして Create project でプロジェクトを作成します．

2.2　地面を作る

ゲームの舞台となる地形を作ります．Unity には山や植物をペイントツールのように作る Terrain という機能があります．

Standard Assets のインポート

Terrain を使う前に，地面に適用するテクスチャや草木のデータを使うために公式の「Standard Assets」をインポートします．

Unity の Asset Store から「Standard Assets」で検索するとパッケージが出てくるのでダウンロードし，インポートします．インポート時に色々選択できますが，必要なテクスチャや草木データが入っている「Environment」のみでいいでしょう．

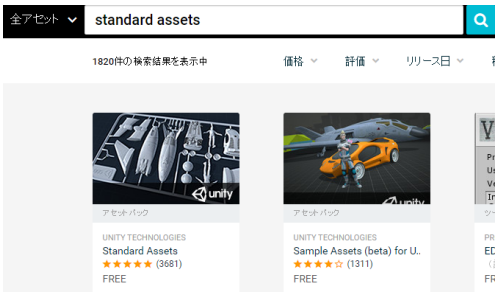

図 2.1: Standard Assets

Terrain の設置

「Hierarchy」の「Create」から「3D Object>Terrain」で作ります．「Inspector」の「Terrain」で設定ができ，「Paint Texture」を選択し「Edit Terrain Layers」から草のテクスチャ画像を指定しましょう．

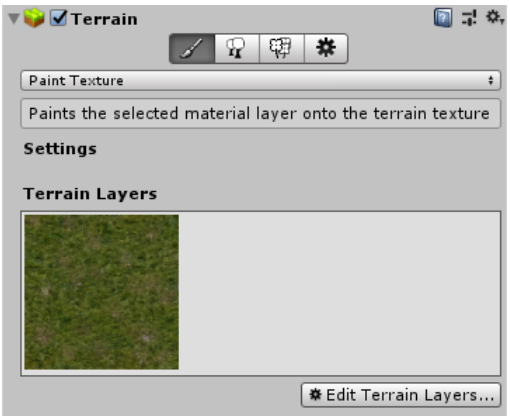

図 2.2: 草テクスチャの指定

2.3　敵 (ゾンビ) の用意

今回メインの作業である敵キャラの動作を作ります．

ゾンビが今回唯一登場するキャラクターです．人型の 3D モデルを扱う際，アニメーションやラグドール，移動 AI など初心者には難しいところがあるかもしれませんが，ここを作ることができると”ゲーム感”がとても出て感動も大きいはずです．

ゾンビのモデルをインポート

Asset Store で「zombie」と検索すると出てくる一番人気のアセットをインポートします．

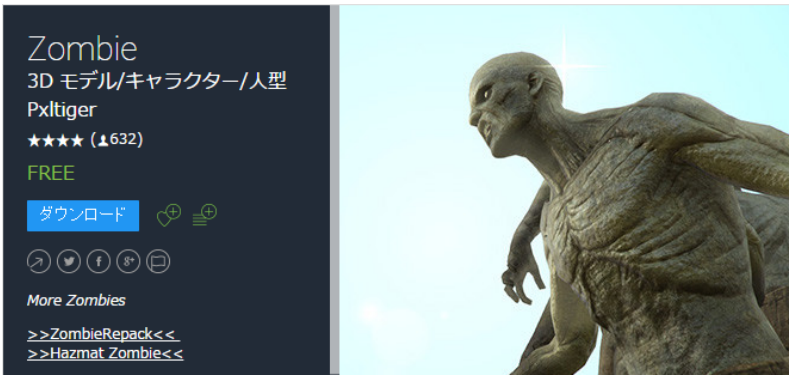

図 2.3: ゾンビの素材

筆者はこの時インポートしたファイル名の一つが大文字小文字の区別に関するエラーがなぜか出てきたので手動で修正しました.

ゾンビの死体を作る

恐ろしい見出し名ですが，ゾンビを撃った後に物理法則に従って倒れるようにラグドールを適用します．ラグドールとは，洋ゲーとかで敵が死ぬとぐったり人形のように崩れていくアレです．インポートしたアセットの Prefabs というフォルダにゾンビのプレハブがあるのでシーンに追加し，メニューの「GameObject>3D Object>Ragdoll...」から Create Ragdoll ウィンドウを開きます.

頭，腕，足など指定するためシーンのゾンビの階層を展開してドラッグアンドドロップで一つずつ体のパーツを当てはめていきます.

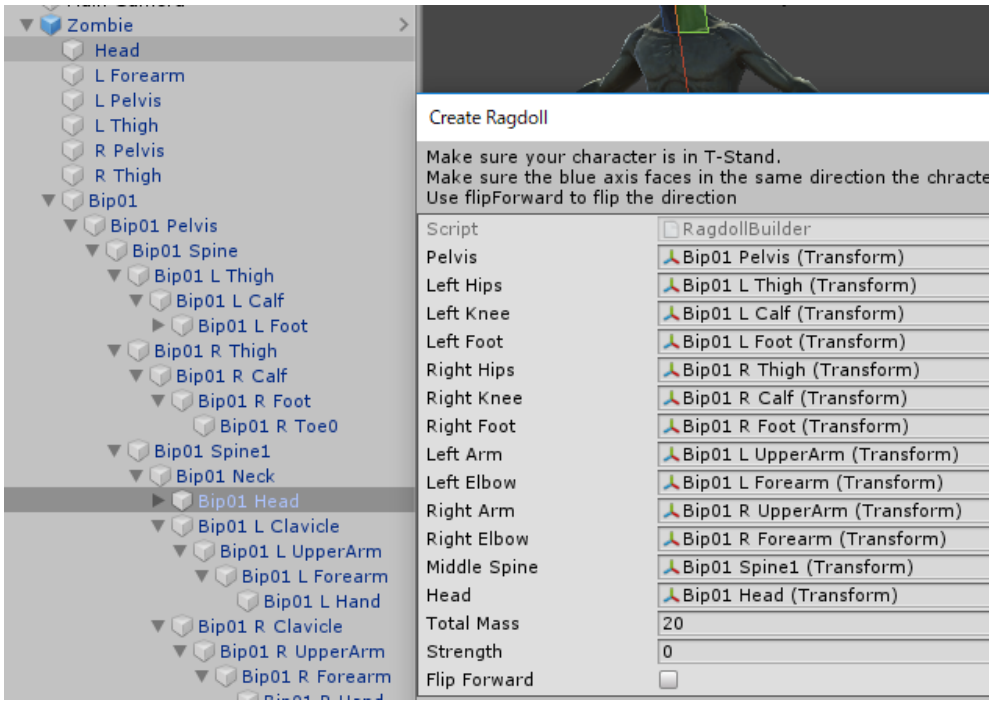

図 2.4: ラグドールの設定

タグを付ける

この後ゾンビの撃つスクリプトを書きますが，プログラムで識別するためのタグをつけます．インスペクタの「Tag ＞ add tag」から「enemy」と名付けたタグを追加し，ゾンビのコライダがついているオブジェクト全てに適用しましょう．階層を shift キーで推しながら複数選択してから Tag を設定すると一度に付けることができます．

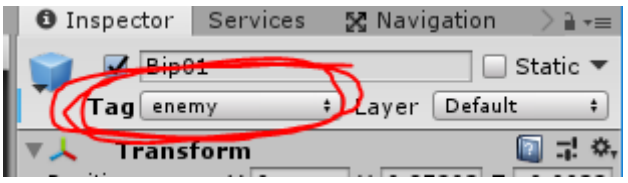

図 2.5: tag の設定

2.4　ゾンビを撃つスクリプトを書く

ここでゲームの動作部分になるプログラムを作ります.「ShotCam」という名前でスクリプトを新規作成し[*1]，以下のコードを書きます.

スクリプトは Main Camera にアタッチします.

ShotCam.cs

```
public class ShotCam : MonoBehaviour {
  void Start() {
  }
  void Update() {
    GameObject clickObject=getClickObject();
    //クリックしたのが敵なら
    if (clickObject!=null && clickObject.gameObject.tag == "enemy") {
      Vector3 vec = clickObject.transform.position - this.transform.position;
      //射撃した部位に力を加える
      clickObject.GetComponent<Rigidbody>().velocity = vec.normalized*30;
      //ゾンビ側のスクリプトの death() 呼び出し
      clickObject.transform.root.GetComponent<Zombie>().death();
    }
  }
  // 左クリックしたオブジェクトを取得する関数
  public GameObject getClickObject() {
    GameObject clickObject = null;
    if (Input.GetMouseButtonDown(0)) {  //左クリック
      Ray ray = Camera.main.ScreenPointToRay(Input.mousePosition);
      RaycastHit hit = new RaycastHit();
      if (Physics.SphereCast(ray, 0.1f, out hit)) {
        clickObject = hit.collider.gameObject;
      }
      Debug.Log(clickObject);
    }
    return clickObject;
  }
}
```

続いてゾンビのオブジェクト (一番上の階層) に Zombie というスクリプトを作成し以下のクラスを追加しアタッチします.

Zombie.cs

[*1] Unity では C#の他 JavaScript,Boo といったプログラミング言語を使えましたが，廃止となっています.

```
public void death() {
  GetComponent<Animator>().enabled = false; //アニメーション無効
  Invoke("destroyObject", 5f);  //5 秒後に消滅させる
}
void destroyObject() {
  Destroy(gameObject);  //オブジェクトを消す
}
```

プログラム解説

まずは ShotCam の説明をします.

getClickObject 関数は左クリックされると画面上に Ray と呼ばれるレーザーのような線を出し，衝突したオブジェクトを取得し return で返しています．毎フレーム呼び出される Update 関数では getClickObject 関数でクリックしたオブジェクトを検知し続け，if 文にて enemy タグが付いているかを判別します.

変数 vec は撃った部位に力を加えて吹き飛ぶ演出をするためのベクトルを保持しています．撃った部位の座標からゲーム画面を映すカメラの座標を減算することでベクトルが取得できます．そして normalized で単位ベクトル化し Rigidbody コンポーネントで力を加えています．「*15」は力の大きさです．試しに大きくして遊んだり自分好みの値にしてみるのもいいでしょう.

enemy タグはゾンビの胴体，手足，頭につけているので clickObject.transform.root でいずれからも一番上の階層のオブジェクトから Zombie.cs の関数 death を呼び出せます.

関数 death が呼び出されるとアニメーションを無効にして 5 秒後に消す処理を行います．アニメーションが無効になると魂が抜けたようにふっと体が崩れ (=ラグドールにより倒れ) ます.

シーンのカメラに見える位置にゾンビを設置し，Unity の三角ボタンで実行してみましょう．クリックするとゾンビが吹き飛ぶと OK です.

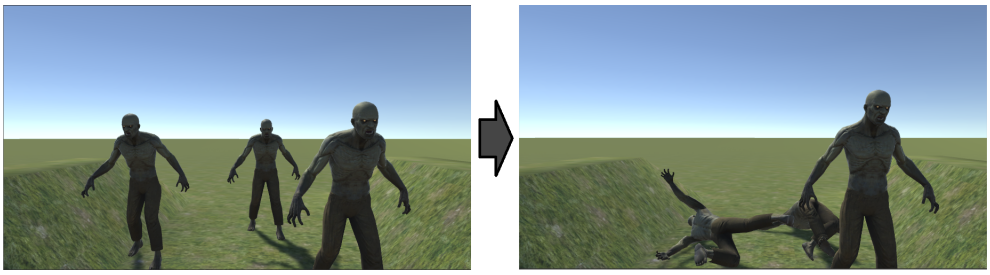

図 2.6: 実行結果 (クリックすると倒れる)

2.5　ゾンビが歩くようにする

ゾンビをプレイヤーまで歩かせるために，経路に沿って移動していく Agent とアニメーションとして体が動く Animator の二つの機能を使います．=== 経路探索 AI の実装 Unity には嬉しいことに経路探索 AI が自動で入っています．まず，ゾンビに「Nav Mesh Agent」のコンポーネントを追加し図のように設定します．

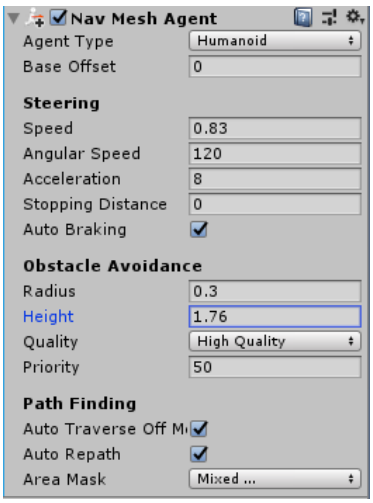

図 2.7: Nav Mesh Agent の設定

次に Terrain に経路探索のために必要な作業をします．Terrain を選択するとインスペクタの隣に「Navigation」というタブが出てきます．選択し「Bake」からパラメータを調整して Bake することでゾンビが歩ける場所が青く表示されます．

なお，試しに障害物として画像では Cube を置いていますがインスペクタから「static」

にチェックを入れるのを忘れないようにしてください.

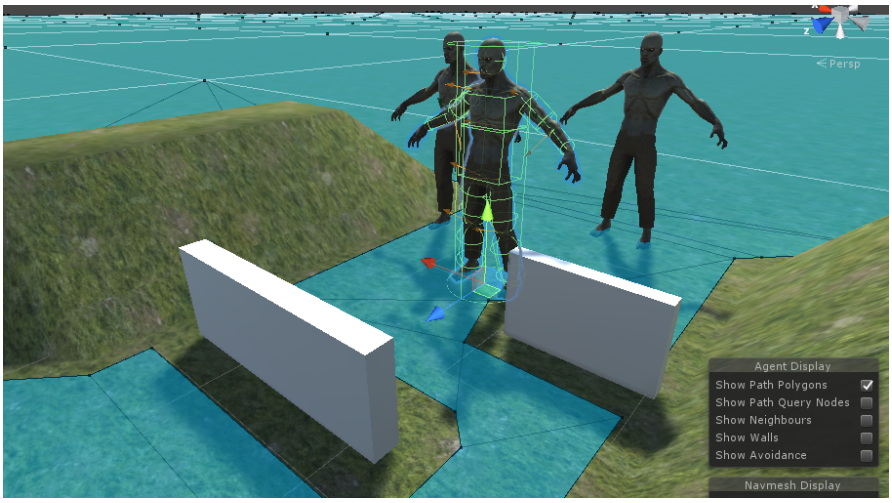

図 2.8: 経路となる領域を Bake した様子

アニメーション遷移の実装

移動中には歩くアニメーション，止まるときには待機のアニメーションを適用するため Animator の設定をします．アセットに walk,idle,atack のアニメーションが含まれているのでノードと遷移を設定します．矢印はノードを右クリックし，「Make Transition」を選択します．矢印を選択したときにインスペクタに表示される Conditions で「state」と名付けた遷移条件変数を追加し walk への矢印には Equals0,idle へは Equals1,attack へは Equals2 を割り当てます.

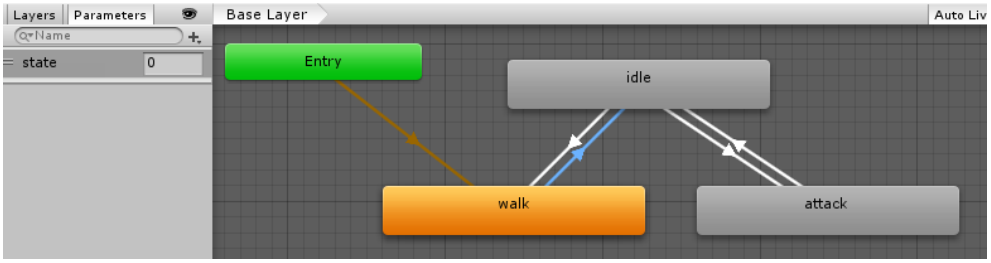

図 2.9: Animator の設定

スクリプトを書き換える

経路探索のために AI 機能をインポートします

Zombie.cs

```
using UnityEngine.AI;
```

そして以下のように書き換えていきます

Zombie.cs

```
public class Zombie : MonoBehaviour {
private new GameObject camera;
private NavMeshAgent agent;
private bool stop;
private enum state { walk,idle,atack }  //アニメーションの状態
private Animator animator;
void Start() {
  camera = GameObject.Find("Main Camera").gameObject;
  agent = GetComponent<NavMeshAgent>();
  agent.SetDestination(camera.transform.position);  //目標座標を設定
  stop = false;
  animator = GetComponent<Animator>();
  SetKinematic(true);  //物理演算を無効にする
}
void Update() {
  //ゾンビが目標点まで 2m 近づいたら立ち止まる
  if (!stop &&
      Vector3.Distance(camera.transform.position, this.transform.position) < 2f)
  {
    animator.SetInteger("state", (int)state.idle);
    Vector3 p = camera.transform.position;
    p.y = this.transform.position.y;
    transform.LookAt(p);
    agent.isStopped = stop = true;
  }
}
//死ぬ処理
public void death() {
  GetComponent<Animator>().enabled = false; //アニメーション無効
  Invoke("destroyObject", 5f);  //5 秒後に消滅させる
  SetKinematic(false);  //物理演算を付ける
  agent.enabled = false;
}
void destroyObject() {
  Destroy(gameObject);  //オブジェクトを消す
```

```
  }

  public void SetKinematic(bool newValue) {
    Component[] components = GetComponentsInChildren(typeof(Rigidbody));
    foreach (Component c in components) {
      (c as Rigidbody).isKinematic = newValue;
    }
  }
}
```

プログラム解説

Start 関数ではゾンビが目的地とするカメラの取得と座標設定，Animator の取得をしています．

Update 関数ではゾンビがカメラまで近づいたら動作する処理があり，state を待機 (idle) にアニメーションを切り替えカメラの方向を向き，移動を停止しています．

そして death 関数の変更と SetKinematic という関数の追加も行っています．これは，NavMesh Agent と Animator，ラグドールを併用したために必要となったプログラムです．公式ドキュメントでは，NavMesh Agent は Physics と合わせて使用するのを非推奨としています．キャラの動きを NavMesh Agent で行っているのにそれに物理トリガを加えたりアニメーション動作を加えると競合してしまうからです．そこで，NavMesh Agent で動かしている，すなわちゾンビが生きている (?) 状態では Rigidbody の Is Kinematic をオンにして物理演算を無効にし，撃たれて死ぬ状態では NavMesh Agent をオフにしつつ Is Kinematic をオンにすることでラグドールを動作させるやり方にしています．SetKinematic 関数はそのために子オブジェクトが持つ Rigidbody コンポーネント全ての物理演算を切り替えるものです．

これでプレビューを実行してみると，ゾンビがカメラに向かって歩きだし，近づくと立ち止まります．

2.6　エフェクトを付ける

ゲームの見栄えを良くするためにエフェクトを使うことは効果的です．プレイヤーに”撃った感”を感じてもらうために着弾点に爆発のようなエフェクトを付けます．エフェクトは Unity のパーティクルシステムという機能で作成することも可能ですが，今回は Asset Store で無料の素材をお借りしましょう．

Asset Store で「War Fx」と検索すると爆発や炎，銃撃などが入ったアセットが見つかります．

図 2.10: War Fx

インポートしたら次に，ShotCam.cs を改造します．クラスのメンバ変数として

ShotCam.cs

```
  public GameObject hitEffect;
```

と宣言し，getClickObject 関数内の if 文に以下のように一行追加します

ShotCam.cs

```
if (Physics.SphereCast(ray, 0.1f, out hit)) {
        clickObject = hit.collider.gameObject;
        //[追加] 着弾点エフェクト
        Instantiate(hitEffect, hit.point, Quaternion.identity);
      }
```

そして hitEffect は public で宣言しているため Main Camera のインスペクタから手動でアタッチする必要があります．先ほどインポートしたアセットの「JMO Assets>WarFX>_Effects>Explosions>WFX_Explosion Small」をドラッグ&ドロップでアタッチします．実行してみるとわかるのですが，エフェクトが大きすぎて不自然です．そこでエフェクトの Scale を x,y,z 全部 0.3 にしたら丁度よくなりました．エフェクトを変えてもプログラム自体は変える必要ないので他のアセットやサイズ，オリジナルエフェクトなど試してみるのもいいでしょう．

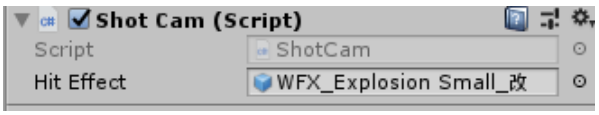

図 2.11: War Fx

■コラム: タイトル

ここにコラムをかける

Unity と Wii リモコンで簡単アーケードゲーム制作

2019 年 1 月 26 日　初版第 1 刷　発行

著　者　　トミー (@Tomy_0331)