Unity を用いたゲーム制作

情報班：北村良太　堤優太

１．はじめに

ゲームを制作することにはたくさんの苦労があるということを聞いた。そこでそのたくさんの苦労の中のほんの少しでも軽減させたいと考えた。ゲームを制作することには、具体的にどのような苦労があるのだろうか、またどのような技術が詰め込まれているのだろうか。それを知ることで今まで苦労していた点が楽になる方法が見つかるのではないかと考えた。そこで、Unity という環境でゲームを制作しようと決心した。

２．準備

まず、今回はモデリング（キャラを作ること）ではなくプログラミング（データを処理するための命令）を目標としていたので、Unity ちゃん(歩く等の基本動作といくつかのモーションがプログラムされている仮想キャラ)をインストールし、実際に使うモーション等を調節して動ける状態にカスタマイズした。

ゲーム制作の技術を身に付け、オリジナルゲーム制作につなげるために、「ブロック崩しゲーム」と「玉転がしゲーム」という二つの簡単なゲームを「はじめての Unity」というサイトを見て制作した。この経験を生かし、オリジナルゲームに使う床を作り、表面にテクスチャ（床の模様）を貼り付け、この後行うプログラミングの足がかりとした。

３．制作

(1) ボールを発射する装置を作る

・指定した座標からボールが飛んでいくようにプログラミングをした（図１）。さらに、重力を働かせるように設定をした。また、このままでは何もない場所からボールが出てくることになるので、発射装置を作り、見た目が不自然にならないようにした。（図２）

(2) 当たり判定をつける

・Unity ちゃんが攻撃モーションを行っているときにボールに触れると、ボールが飛んでいくようにしようとしたが、技術がなかったため何度もエラーが発生した。そこで、Unity ちゃんにボールが当たった時にボールが破壊されるようにプログラミングした。（図３）

(3) 得点制にしてゲームとして完成させる

・攻撃によってボールを飛ばす事ができたら得点を加算する。

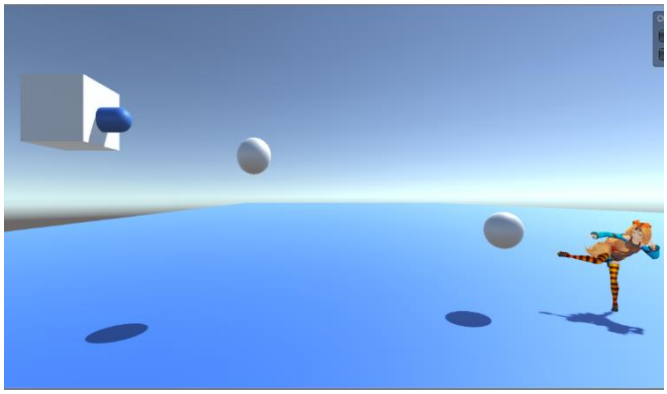
・飛ばされたボールが的に当たるとさらに得点を加算する。

・「制限時間内にこれら二つの得点の合計が一定に達したらゲームクリアとする」というプログラミングをする。

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

public class NewBehaviourScript : MonoBehaviour {
    //public GameObject score_object = null;
    //public int score_num = 0;
    void Start () {
    }
    void OnCollisionEnter(Collision collision) {
        if (collision.gameObject.tag == "mars") {
            Destroy (this.gameObject);
        }
        else if (collision.gameObject.tag == "Player") {
            //Instantiate (particle, transform.position, transform.rotation);
            //Text score_text = score_object.GetComponent<Text> ();
            //score_num += 1;
            //score_text.text = "Score:" + score_num;
            Destroy (this.gameObject);
        }
    }
}
```

（図１）



（図２）

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

public class Chooting : MonoBehaviour {

    // bullet prefab
    public GameObject bullet;


    // 弾丸発射点
    public Transform muzzle;

    // 弾丸の速度
    public float speed = 1000;
    float countTime = 0;
    float launch = Random.Range(0.5f,0.5f);

    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {
        countTime += Time.deltaTime;

        // z キーが押された時
//      if (Input.GetKeyDown(KeyCode.Z)){
        if(countTime > launch){
            countTime = 0;
            launch = Random.Range (5.0f, 5.0f);

            // 弾丸の複製
            GameObject bullets = Instantiate(bullet) as GameObject;

            Vector3 force;

//          force = this.gameObject.transform.forward * speed;
            force = this.gameObject.transform.forward * (int)(Random.Range(5.0f,10.0f)) * 300;

            // Rigidbodyに力を加えて発射
            bullets.GetComponent<Rigidbody>().AddForce(force);

            // 弾丸の位置を調整
            bullets.transform.position = muzzle.position;
        }

    }
}
```

（図３）

４．結果と考察

３の(2)で破壊されたときにエフェクトをつけようとしたが、ボールが破壊された座標を検知し、その座標にエフェクトを発生させることが困難だったため、制作を断念した。

３の(3)は(2)の制作が上手くいかなかったため制作に至らなかった。

どんなゲームでも使われているはずの当たり判定というものですら、完成させることができなかったことから、色々なゲームに使われている技術はわれわれが考えつく部分から考えつかない部分まで、莫大な技術が詰め込まれていることを知った。

一番難しかったのはエラーが一度に何度も出てきてしまったことで、それが重なると何が悪いのかがわからず一回のエラーで一時間程度悩まされたことが何度もあった。文章の最後にセミコロンをつけなければ、それだけでエラーが出たりするので、スクリプト（データを処理するための命令が書かれた文章）を隅から隅まで見直さなければならないことがわかった。

現実世界では当たり前なこともゲームを作るときには一つ一つ考えなくてはならないと感じた。

５．参考文献ならびに参考 Web ページ

・Unity　https://unity3d.com/jp

・[超初心者向け]Unity チュートリアル「はじめての Unity」のブロック崩しと同等を C#で::
https://qiita.com/JunShimura/items/cbb0db8087a5cc75735e

・Unity の Rigidbody と Collider で衝突判定

https://qiita.com/yando/items/0cd2daaf1314c0674bbe

※　研究を進めるにあたり、大阪工業大学　田熊隆史先生にご指導いただいた。