研究班番号【１１９】

Unity と C#言語を用いてのゲーム開発

情報班：瓜生 泰基　梶川 雅央　大塚 夏以

要約

近年 e-sport としてゲームへの注目が高まり、単なるエンターテイメントではなく競技となってきている。そこで簡単な fps(first person) gun game を作ろうと思い制作した。素人 3 人による 0 からのスタートであり、緻密なプログラムが組まれたものではないので基本的なものだけ制作した。今回の制作を通じて、世の中にあふれるプログラムに対する興味、関心、理解が少しでも深まるのではないかと考えた。

Abstract

Today, games are attracting attention by ordinary people or people who work as professional gamer. Then we decided to create fps (first person) gun game. We created hard because we had no big deal knowledge and assistance of university professor. Through this experience, we thought that it will improve our interest and understanding for programs which exist around the world.

１．序論

ゲーム制作をするとなるとゲーム制作エンジンを必要とする。私たちは、3D ゲームの製作において様々な手助けとなる情報がインターネット上に豊富で容易にゲームを作ることのできる「unity」を用いてゲーム制作をしようと試みた。

２．作成方法

今回はゲームエンジンと C#言語を用いた。

Asset を用いて１人称のプレイヤー(3D オブジェクト)を作った。そのオブジェクトに Camera の動作をつかさどるプログラムを付帯させ、同様に銃の形をした 3D オブジェクトを付属させた。処理を軽くするために銃弾は実際に発射されるわけでなく、Ray とういう不可視のレーザーが着弾点にあたれば物体に応じてアクションを起こすようにした。壁に当たれば弾痕を表示し、敵に当たればエフェクトがでるようにした。敵キャラクターはランダム生成、自動追尾、自動攻撃、消滅するプログラムを組み込んだ。フィールドについては、平面の物理マテリアルを生成し凹凸をつけ岩のテクスチャを張り付け、草と木の Asset を設置した。

３．成果

・1 人称視点で AWDS キーとマウスで操作できる。

・銃を撃てる。
・敵が自動で追尾して攻撃してくる。
・フィールド上に敵キャラクターを自動出現させる。
・敵の体力が 0 になったら消滅するようにする。

４．今後の展望

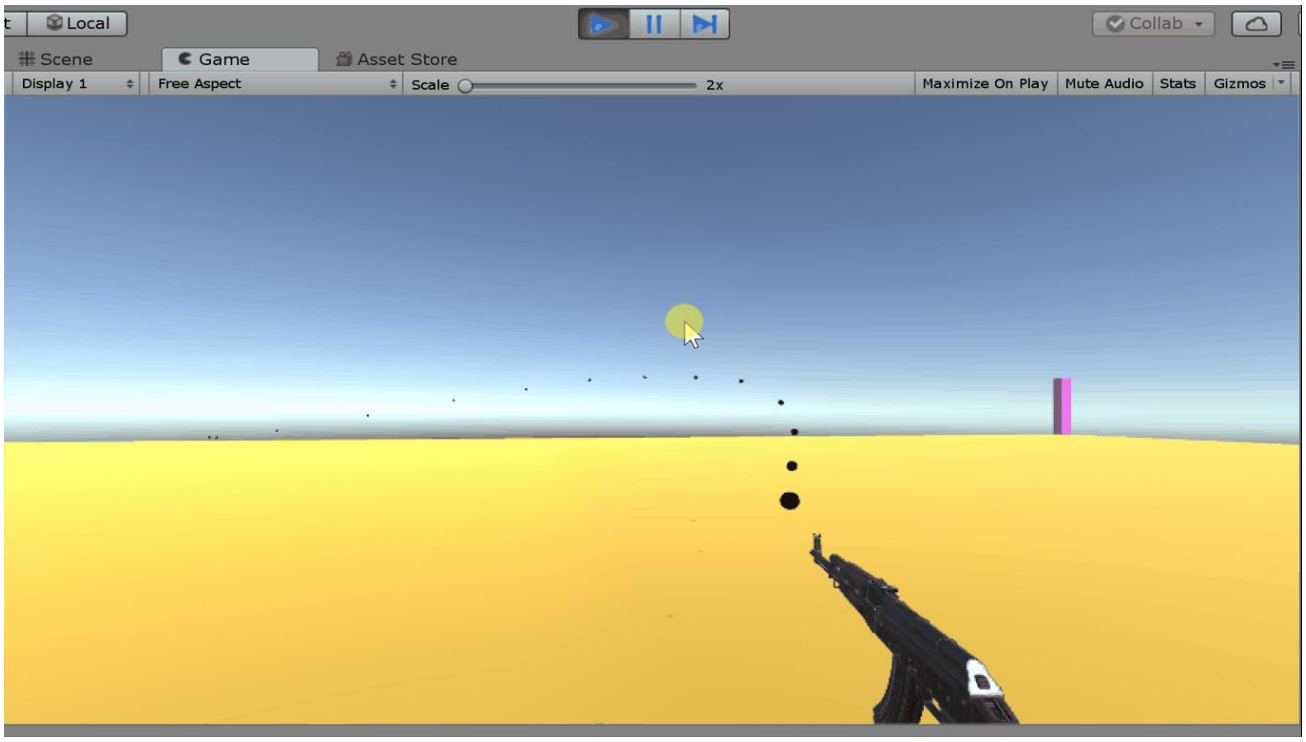
ゲーム性を充実さるために、ゲームオーバー、スタート画面追加、操作キャラクターHP 実装、武器のレパートリーを増やす。
ゲームとして個性のないものに仕上がってしまっているので、独自の要素を追加する必要がある。

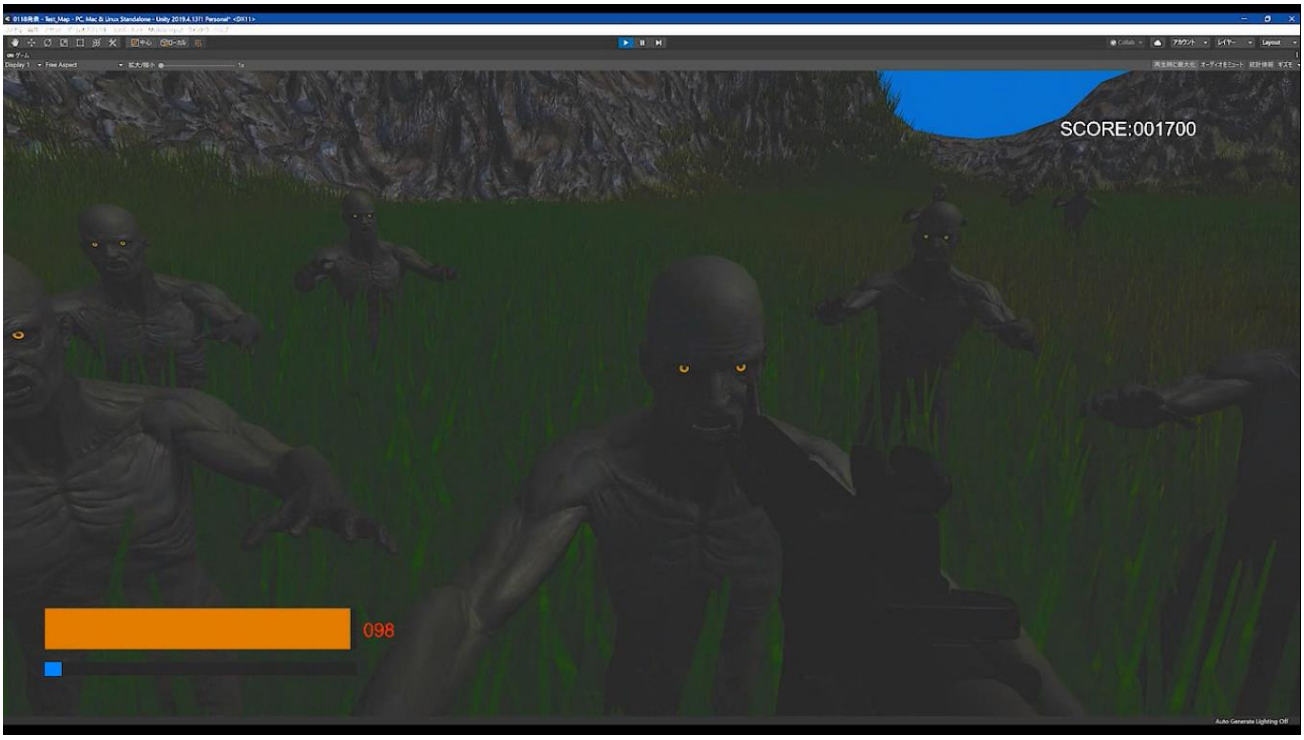
５．参考文献ならびに参考 Web ページ

くろくまそふと. https://moon-bear.com/2020/01/23/unity-fps/
Unity Asset store https://assetstore.unity.com/
BYTE http://bytejp.com/
Qiita https://qiita.com/yando/items/ef76c200bb50005170d5

６．資料

〈一人称画面〉





〈プログラム〉

銃の動作

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
public class FirstPersonGunController : MonoBehaviour
{
    public enum ShootMode { AUTO, SEMIAUTO }
    public bool shootEnabled = true;
    [SerializeField]
    ShootMode shootMode = ShootMode.AUTO;
    [SerializeField]
    int maxAmmo = 100;
    [SerializeField]
    int maxSupplyValue = 100;
    [SerializeField]
    int damage = 1;
    [SerializeField]
    float shootInterval = 0.1f;
```

```
[SerializeField]
float shootRange = 50;
[SerializeField]
float supplyInterval = 0.1f;
[SerializeField]
Vector3 muzzleFlashScale;
[SerializeField]
GameObject muzzleFlashPrefab;
[SerializeField]
GameObject hitEffectPrefab;
[SerializeField]
Image ammoGauge;
[SerializeField]
Text ammoText;
[SerializeField]
Image supplyGauge;
bool shooting = false;
bool supplying = false;
int ammo = 0;
int supplyValue = 0;
GameObject muzzleFlash;
GameObject hitEffect;
public int Ammo
{
    set
    {
        ammo = Mathf.Clamp(value, 0, maxAmmo);
        //UIの表示を操作
        //テキスト
        ammoText.text = ammo.ToString("D3");
        //ゲージ
        float scaleX = (float)ammo / maxAmmo;
        ammoGauge.rectTransform.localScale = new Vector3(scaleX, 1, 1);
    }
    get
    {
        return ammo;
    }
```

```
}
public int SupplyValue
{
    set
    {
        supplyValue = Mathf.Clamp(value, 0, maxSupplyValue);
        if (SupplyValue >= maxSupplyValue)
        {
            Ammo = maxAmmo;
            supplyValue = 0;
        }
        float scaleX = (float)supplyValue / maxSupplyValue;
        supplyGauge.rectTransform.localScale = new Vector3(scaleX, 1, 1);
    }
    get
    {
        return supplyValue;
    }
}
void Start()
{
    InitGun();
}
void Update()
{
    if (shootEnabled & ammo > 0 & GetInput())
    {
        StartCoroutine(ShootTimer());
    }
    if (shootEnabled)
    {
        StartCoroutine(SupplyTimer());
    }
}
public void InitGun()
{
    Ammo = maxAmmo;
    SupplyValue = 0;
```

```
    }
    bool GetInput()
    {
        switch (shootMode)
        {
            case ShootMode.AUTO:
                return Input.GetMouseButton(0);
            case ShootMode.SEMIAUTO:
                return Input.GetMouseButtonDown(0);
        }
        return false;
    }
    IEnumerator ShootTimer()
    {
        if (!shooting)
        {
            shooting = true;
            //マズルフラッシュON
            if (muzzleFlashPrefab != null)
            {
                if (muzzleFlash != null)
                {
                    muzzleFlash.SetActive(true);
                }
                else
                {
                    muzzleFlash = Instantiate(muzzleFlashPrefab, transform.position,
transform.rotation);
                    muzzleFlash.transform.SetParent(gameObject.transform);
                    muzzleFlash.transform.localScale = muzzleFlashScale;
                }
            }
            Shoot();
            yield return new WaitForSeconds(shootInterval);
            //マズルフラッシュOFF
            if (muzzleFlash != null)
            {
                muzzleFlash.SetActive(false);
```

```
            }
            //ヒットエフェクトOFF
            if (hitEffect != null)
            {
                if (hitEffect.activeSelf)
                {
                    hitEffect.SetActive(false);
                }
            }
            shooting = false;
        }
        else
        {
            yield return null;
        }
    }
    void Shoot()
    {
        Ray ray = new Ray(transform.position, transform.forward);
        RaycastHit hit;
        //レイを飛ばして、ヒットしたオブジェクトの情報を得る
        if (Physics.Raycast(ray, out hit, shootRange))
        {
            //ヒットエフェクトON
            if (hitEffectPrefab != null)
            {
                if (hitEffect != null)
                {
                    hitEffect.transform.position = hit.point;
                    hitEffect.transform.rotation = Quaternion.FromToRotation(Vector3.forward,
hit.normal);
                    hitEffect.SetActive(true);
                }
                else
                {
                    hitEffect = Instantiate(hitEffectPrefab, hit.point, Quaternion.identity);
                }
            }
```

```
            //★ここに敵へのダメージ処理などを追加
            string tagName = hit.collider.gameObject.tag;
            if (tagName == "Enemy")
            {
                EnemyController enemy = hit.collider.gameObject.GetComponent<EnemyController>();
                enemy.Hp -= damage;
            }
        }
        Ammo--;
    }
    //--中略--
    IEnumerator SupplyTimer()
    {
        if (!supplying)
        {
            supplying = true;
            SupplyValue++;
            yield return new WaitForSeconds(supplyInterval);
            supplying = false;
        }
    }
}
```

敵キャラクターの動作

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

[RequireComponent(typeof(Animator))]
[RequireComponent(typeof(Rigidbody))]
[RequireComponent(typeof(NavMeshAgent))]
public class EnemyController : MonoBehaviour
{

    public bool moveEnabled = true;
```

```
[SerializeField]
int maxHp = 3;
[SerializeField]
int ammoDamage = 5;
[SerializeField]
int attackInterval = 1;
[SerializeField]
int score = 100;
[SerializeField]
string targetTag = "Player";
[SerializeField]
float deadTime = 3;

bool attacking = false;
int hp;
float moveSpeed;
Animator animator;
BoxCollider boxCollider;
Rigidbody rigidBody;
NavMeshAgent agent;
Transform target;
GameManager gameManager;
FirstPersonGunController player;

public int Hp
{
    set
    {
        hp = Mathf.Clamp(value, 0, maxHp);

        if (hp <= 0)
        {
            StartCoroutine(Dead());
        }
    }
    get
    {
        return hp;
```

```
        }
    }

    void Start()
    {
        animator = GetComponent<Animator>();
        boxCollider = GetComponent<BoxCollider>();
        rigidBody = GetComponent<Rigidbody>();
        agent = GetComponent<NavMeshAgent>();

        target = GameObject.FindGameObjectWithTag(targetTag).transform;
        gameManager = GameObject.FindGameObjectWithTag("GameController").GetComponent<GameManager>();
        player =
GameObject.FindGameObjectWithTag("Player").GetComponentInChildren<FirstPersonGunController>();

        InitCharacter();
    }

    void Update()
    {
        if (moveEnabled)
        {
            Move();
        }
        else
        {
            Stop();
        }
    }

    void InitCharacter()
    {
        Hp = maxHp;
        moveSpeed = agent.speed;
    }

    void Move()
    {
```

```
        agent.speed = moveSpeed;
        animator.SetFloat("Speed", agent.speed, 0.1f, Time.deltaTime);

        agent.SetDestination(target.position);
        rigidBody.velocity = agent.desiredVelocity;
    }

    void Stop()
    {
        agent.speed = 0;
        animator.SetFloat("Speed", agent.speed, 0.1f, Time.deltaTime);
    }

    IEnumerator Dead()
    {
        moveEnabled = false;
        Stop();
        gameManager.Score += score;
        animator.SetTrigger("Dead");
        boxCollider.enabled = false;
        rigidBody.isKinematic = true;
        yield return new WaitForSeconds(deadTime);
        Destroy(gameObject);
    }

    private void OnCollisionEnter(Collision collision)
    {
        if (collision.gameObject.tag == "Player")
        {
            StartCoroutine(AttackTimer());
        }
    }

    IEnumerator AttackTimer()
    {
        if (!attacking)
        {
            attacking = true;
```

```
            moveEnabled = false;

            animator.SetTrigger("Attack");
            player.Ammo -= ammoDamage;
            yield return new WaitForSeconds(attackInterval);

            attacking = false;
            moveEnabled = true;
        }

        yield return null;
    }
}
```

プレイ画面