個人ポートフォリオ

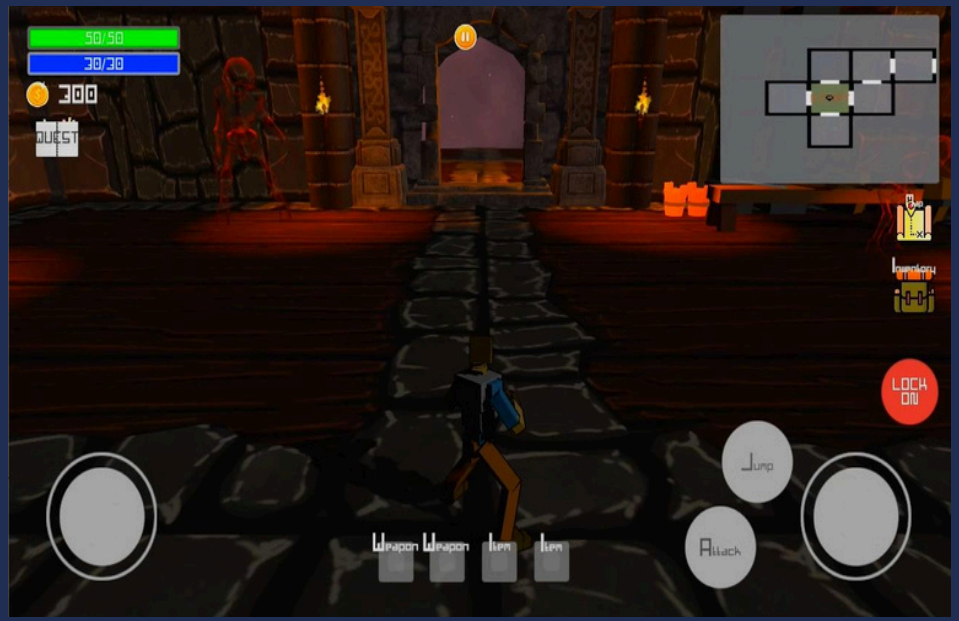
UNITY

eodud1992@gmail.com
キム・デヨン

目次

コンテンツ

ゲームの概要



制作期間　5週

バージョン　2019.3.15f1

その外用

phpMyAdmin
Visual Studio 2017

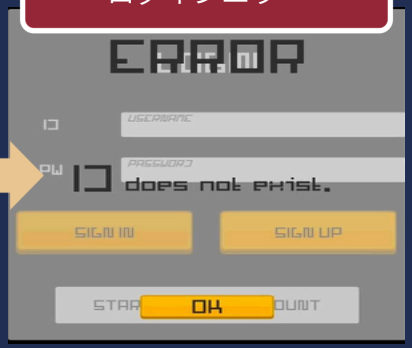
動画

https://youtu.be/mg2NKNVAPSg

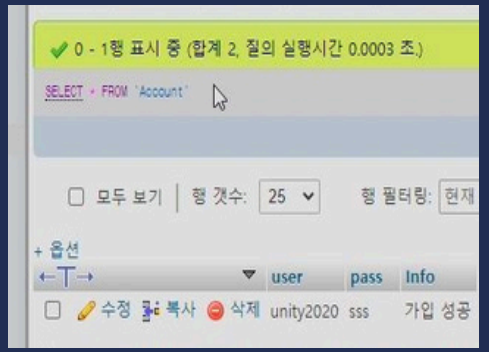
DB連動

アカウントがない場合

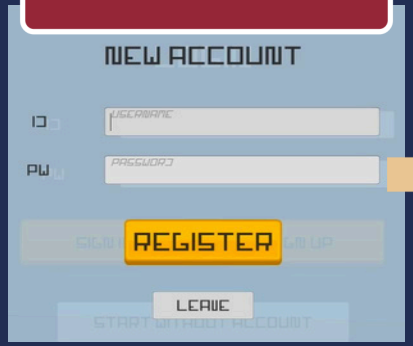
ログインエラー



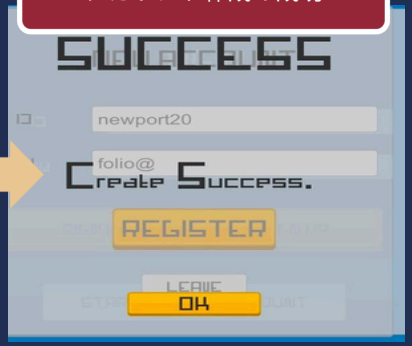
入力した
アカウントがDBに
存在しない場合は、
エラーメッセージ
を出力します。



新しいアカウントを作成

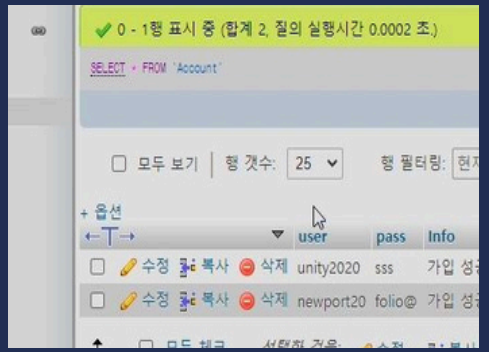


アカウント作成の成功



アカウント情報を
新しく
DBに追加できます

DBに保存するアカウント

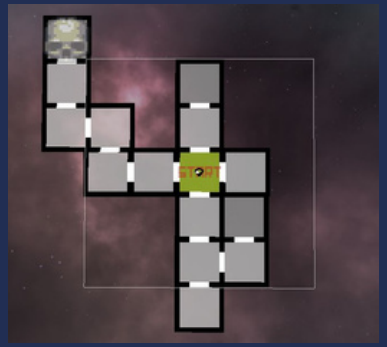


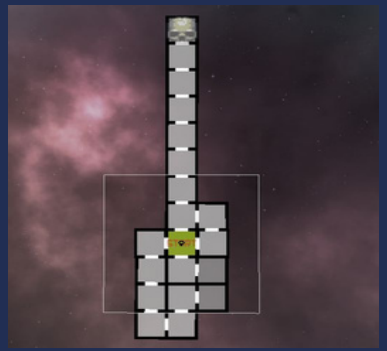
マップの作成

コード

```
private void Spawn()
{
    if(spawned == false)
    {
        if (openingDirection == 1)
        {
            rand = Random.Range(0,  templates.bottomRooms.Length);
            CheckRoom();

            Instantiate(templates.bottomRooms[rand],
                transform.position, templates.bottomRooms[rand].transform.rotation);

            spawned = true;
        }
        else if (openingDirection == 2)
        {
            rand = Random.Range(0, templates.topRooms.Length);
            CheckRoom();

            Instantiate(templates.topRooms[rand], transform.position,
                        templates.topRooms[rand].transform.rotation);

            spawned = true;
        }
        else if (openingDirection == 3)
        {
            rand = Random.Range(0, templates.leftRooms.Length);
            CheckRoom();

            Instantiate(templates.leftRooms[rand], transform.position,
                        templates.leftRooms[rand].transform.rotation);
            spawned = true;
        }
        else if (openingDirection == 4)
        {
            rand = Random.Range(0, templates.rightRooms.Length);
            CheckRoom();

            Instantiate(templates.rightRooms[rand], transform.position,
                        templates.rightRooms[rand].transform.rotation);

            spawned = true;
        }
    }
}
```

生成されたマップ

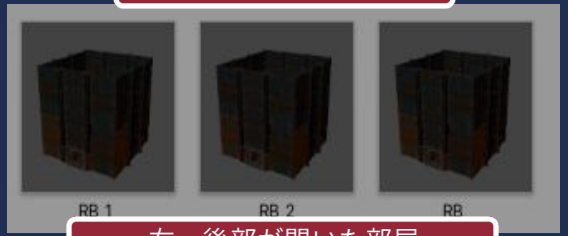




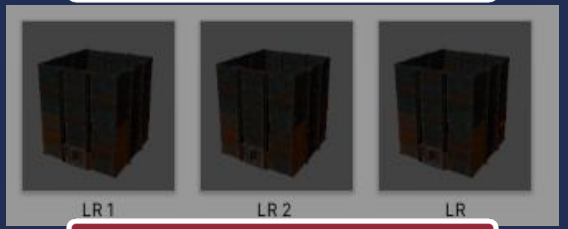
ゲームを開始するときにランダムに部屋を作成してプレイするたびに異なる形のマップを作成

マップの作成

左側が開いた部屋



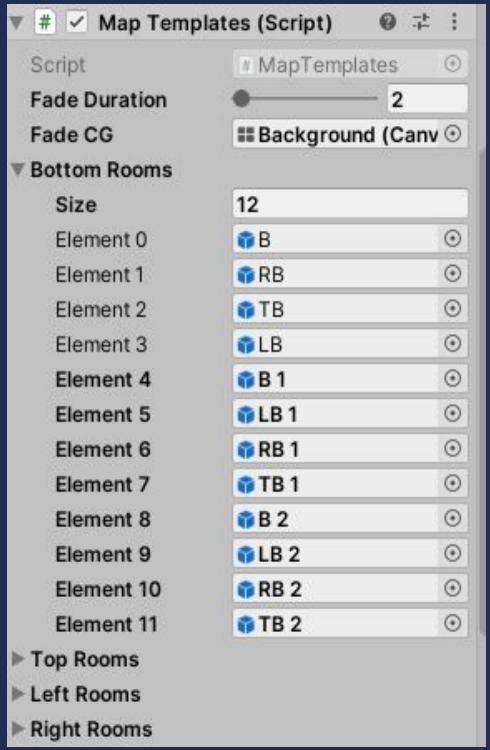
右、後部が開いた部屋



左、右が開いた部屋

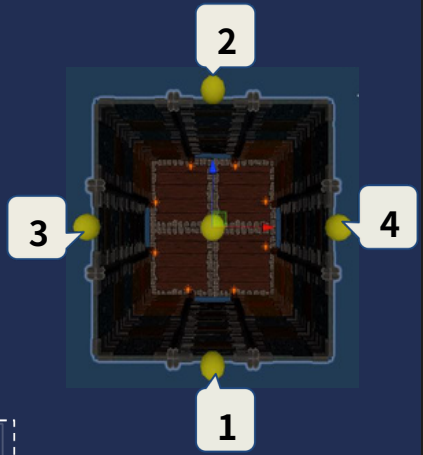
異なる構成の部屋を複数生成します。

事前に作られた部屋を出入り口の位置に応じて(前後左右)配列に分割して保存します。



マップの作成

出入口があるところに部屋の中心から一定距離離れたところにギズモを生成します。

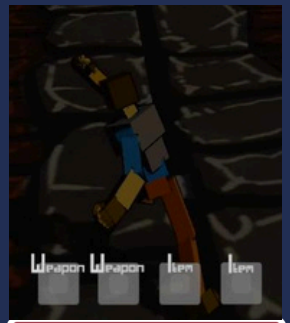


各ギズモの位置に応じて番号を付けます。

```
private void Spawn()
{
    if(spawned == false)
    {
        // 1번 (뒤쪽에 출입구가 있을 때)
        if (openingDirection == 1)
        {
            rand = Random.Range(0,  templates.bottomRooms.Length);
            CheckRoom();

            Instantiate(templates.bottomRooms[rand], transform.position,
              templates.bottomRooms[rand].transform.rotation);

            spawned = true;
        }
        // 2번 (앞쪽에 출입구가 있을 때)
        else if (openingDirection == 2)
        {
            rand = Random.Range(0, templates.topRooms.Length);
            CheckRoom();

            Instantiate(templates.topRooms[rand], transform.position,
                        templates.topRooms[rand].transform.rotation);

            spawned = true;
        }
        // 2번 (왼쪽에 출입구가 있을 때)
        else if (openingDirection == 3)
        {
            rand = Random.Range(0, templates.leftRooms.Length);
            CheckRoom();

            Instantiate(templates.leftRooms[rand], transform.position,
                        templates.leftRooms[rand].transform.rotation);
            spawned = true;
        }
        // 2번 (오른쪽에 출입구가 있을 때)
        else if (openingDirection == 4)
        {
            rand = Random.Range(0, templates.rightRooms.Length);
            CheckRoom();

            Instantiate(templates.rightRooms[rand], transform.position,
                        templates.rightRooms[rand].transform.rotation);

            spawned = true;
        }
    }
}
```

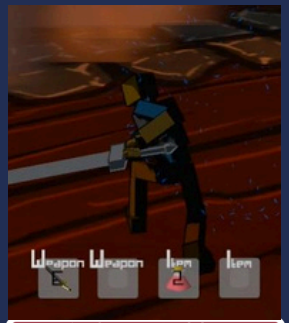
ギズモの固有番号にしたがって、配列に保存されている部屋をランダムに生成します。

プレイヤー



攻撃(非武装)

武器を装備して
攻撃する場合、
ダメージと攻撃範囲が
上昇します。



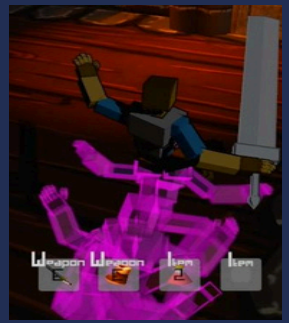
攻撃(武装)

敵ロックオン

カメラが
敵に注目します。

視界から逃さず
集中的に

攻撃することができます。



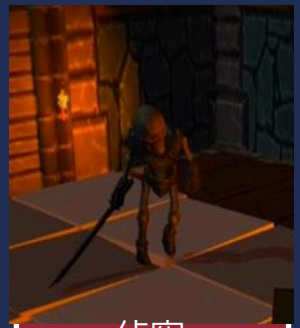
ダッシュ

ダッシュ(ロックオン状態)

敵の攻撃を回避できます。

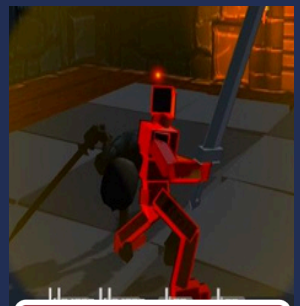
プレイヤーのメッシュを
ベークしてキャプチャされた
モーションを取得し、
残像効果を出しました。

敵



偵察

プレイヤーが遠い場合は、
周囲を偵察します。



攻撃

プレイヤーが
近づくと、
追跡しながら
攻撃します。

```
public List<Transform> wayPointList;
```

偵察位置を保存

```
iNextIdx = ++iNextIdx % wayPointList.Count;
```

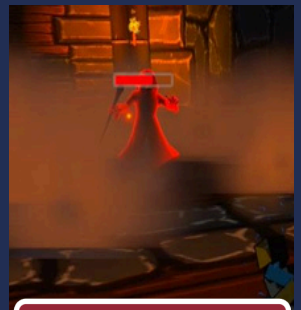
次の偵察の場所を探します。

```
void MoveWayPoint()
{
    if (agent.isPathStale)
        return;

    agent.destination = wayPointList[iNextIdx].position;
    agent.isStopped = false;
}
```

パスが有効であることを
確認して移動します。

攻撃

プレイヤーが
一定距離以内に
近づくと
遠距離攻撃を
します。



瞬間移動

プレイヤーが
近づくと、
別の位置に
瞬間移動します。

ボス

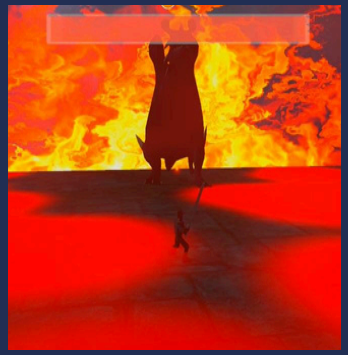
コード

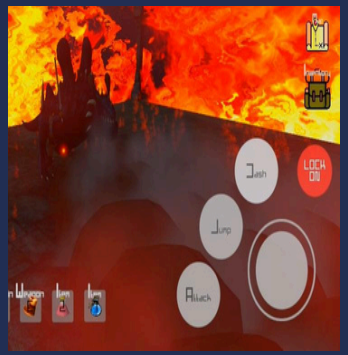
```
switch (state)
{
    case State.IDLE:
        anim.SetBool(hashIdle, true);
        anim.SetBool(hashRetreat, false);
        moveAgent.Stop();
        break;
    case State.METEOR:
        anim.SetBool(hashIdle, false);
        anim.SetBool(hashRetreat, false);
        moveAgent.Stop();
        anim.SetTrigger(hashScream);
        break;
    case State.EARTHQUAKE:
        anim.SetBool(hashIdle, false);
        anim.SetBool(hashRetreat, false);
        moveAgent.Stop();
        anim.SetTrigger(hashEarthquake);
        break;
    case State.ATTACK:
        anim.SetBool(hashIdle, false);
        anim.SetBool(hashRetreat, false);
        moveAgent.Stop();
        anim.SetTrigger(hashAttack);
        break;
    case State.JUMP:
        anim.SetBool(hashIdle, false);
        anim.SetBool(hashRetreat, true);
        moveAgent.Stop();
        anim.SetTrigger(hashJump);
        break;
    case State.LOOKAT:
        anim.SetBool(hashRetreat, false);
        anim.SetBool(hashIdle, false);
        moveAgent.Stop();
        isretreat = false;
        LookAtTarget();
        break;
    case State.RETREAT:
        anim.SetBool(hashIdle, false);
        anim.SetBool(hashRetreat, true);
        Retreat();
        break;
    case State.DEATH:
        anim.SetBool(hashRetreat, false);
        anim.SetBool(hashIdle, false);
        moveAgent.Stop();
        isretreat = false;
        anim.SetTrigger(hashDeath);
        EventManager.instance.Die(anim, hashDie, null, info);
        break;
}
```

ボスのパターンは
FSMでそれぞれの
行動を区分しました。
プレイヤーとの
距離に応じて
近距離/遠距離攻撃、
位置移動をします。

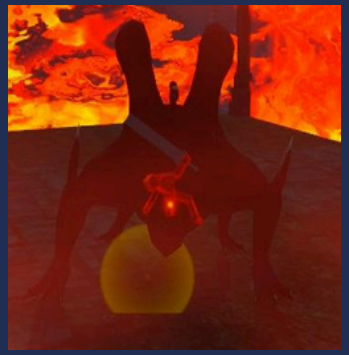
ボスのすべての攻撃は
プレイヤーを
ダウンさせます。

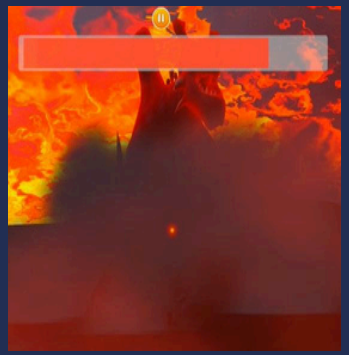
遠距離攻撃



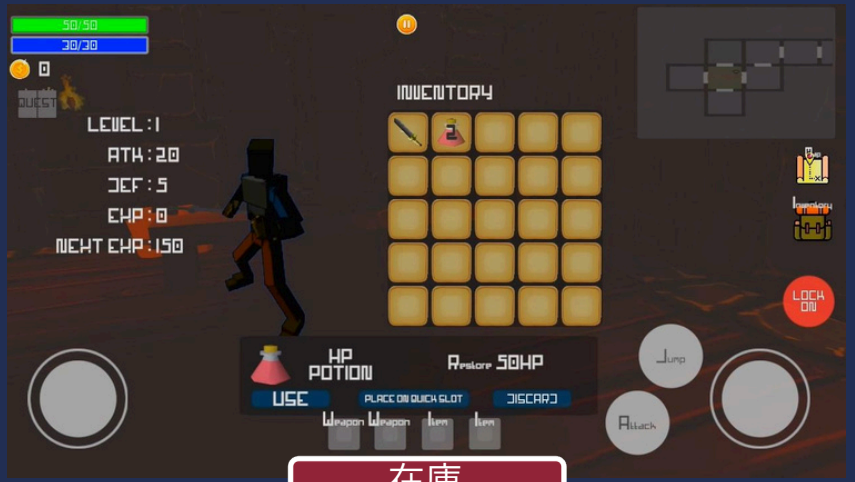


近距離攻撃





在庫



在庫

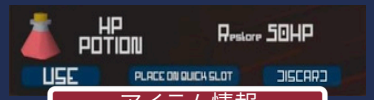
入手した武器/アイテムの情報を保存して
すぐに使用したり、クイックスロットに移動したり
できます。

プレイヤーのステータス

現在プレイヤーの
レベル、攻撃力、
防御力、経験値、
次のレベルアップまでに
必要な経験値を表示します。

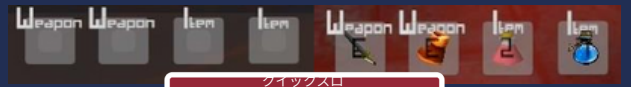
アイテムリスト

プレイヤーが入手した
武器とアイテムが
保存する
リストです。



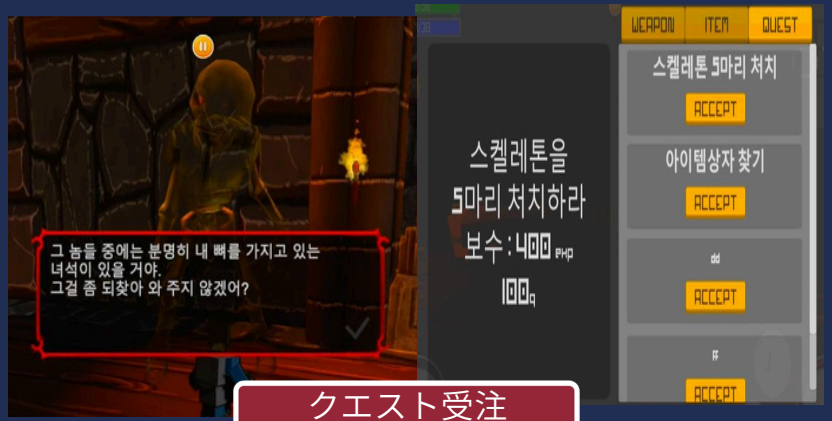
アイテム情報

選択したアイテムの情報を
示します。
使用/装備、クイックスロット
配置、廃棄機能があります。

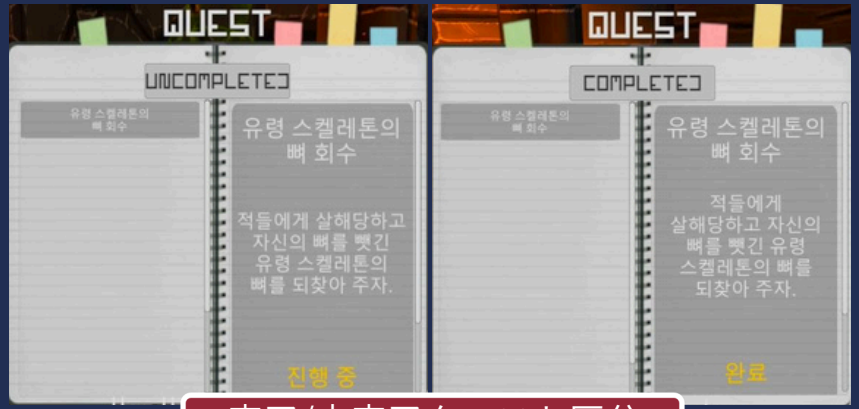


クイックスロット

クエスト



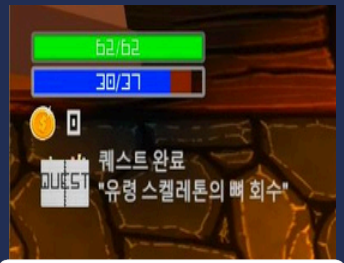
クエスト受注



完了/未完了クエスト区分

NPCとの会話や店舗のQUEST欄で
クエストを受けることができます。

クエストに応じて経験値、お金、
特定アイテムを入手します。



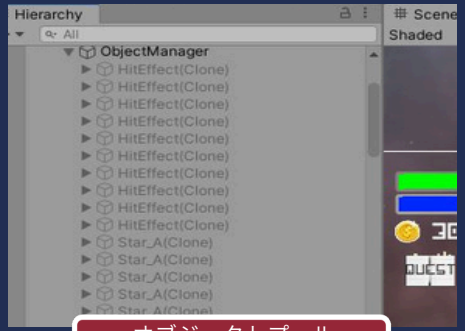
クエスト完了通知

クエストを状況を

表示します。

達成時、通知します。

その他

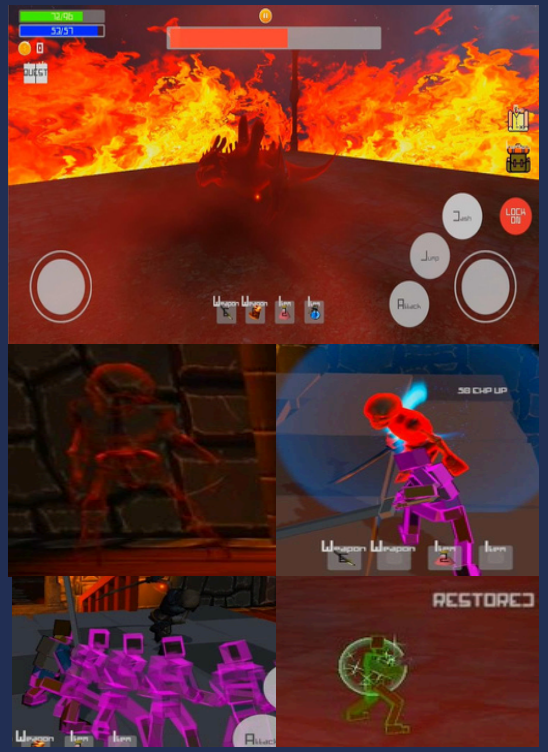


オブジェクトプール

オブジェクト

事前に作成しておき、
必要に応じて
動作します。

シェーダー



リムライト効果を利用して
ダメージ、アイテム使用、
レベルアップ
などのイベント発生時
それぞれ異なる色値を
与えて特定の効果を
付与したり
ゴーストNPCへ
ホログラム効果を
与えました。
また、UV値を調整して
背景効果を表現しました。

その他

オクルージョンカリング

オクルージョン領域を
設定してカメラの
領域に入った
オブジェクトのみ
出力しました。

ライトプローブ

事前にライトマップ
をベークして、
光源効果は
ライトプローブで
表現しました。

レイキャスト

レイキャストを使って
カメラが壁に
衝突しているかどうかを
調べます。

衝突点を検出して
それ以上のカメラの
位置を調整しました。

壁のアルファ値を
調整して半透明の
状態にして、

プレイヤーが見える
ように しました。

THANK YOU

eodud1992@gmail.com
キム・
デヨン