

# PORTFOLIO

ポートフォリオ

ASOポップカルチャー専門学校  
ゲーム・CG・アニメ専攻科

Tanaka

田中

Isami

矯

# <目次>

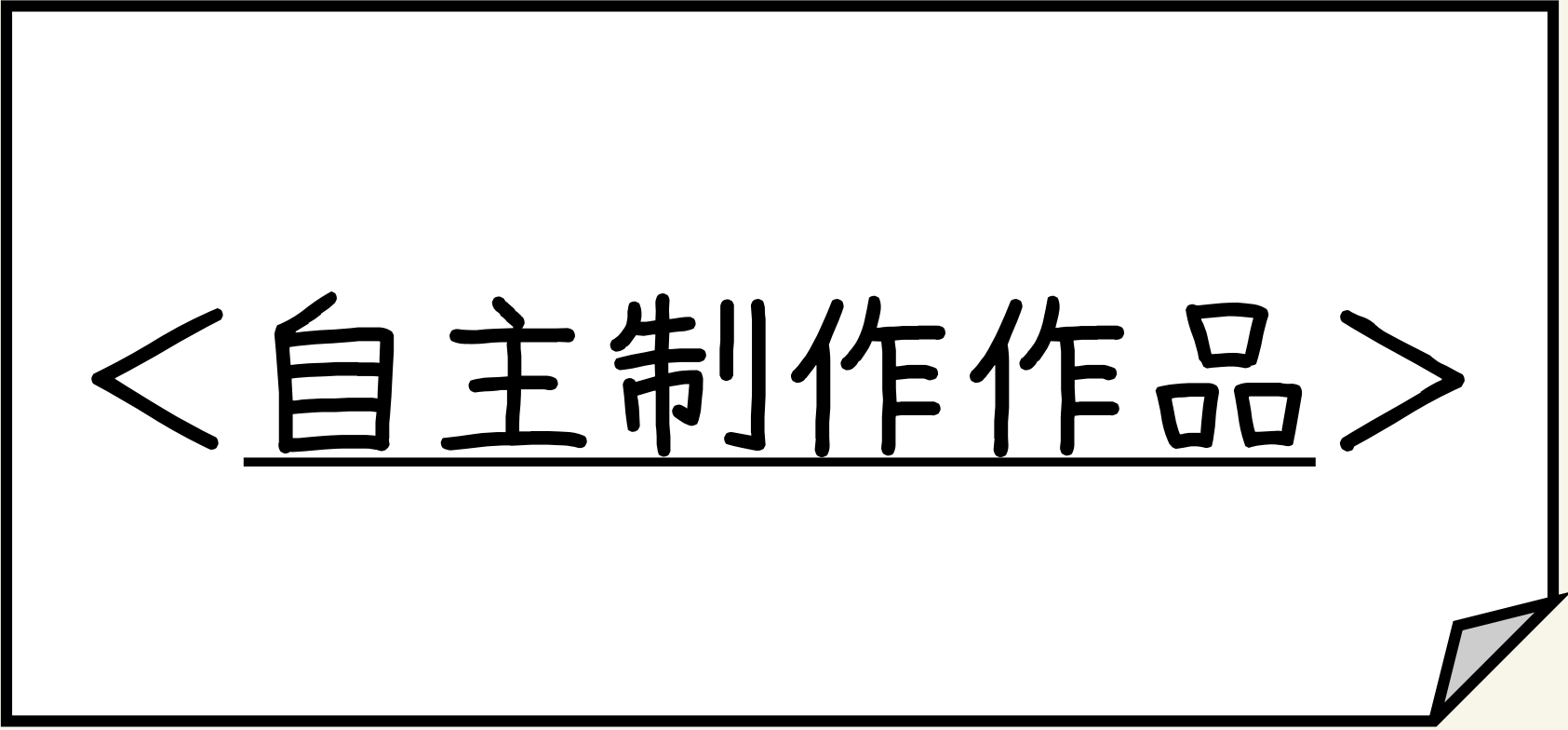
## 1. 自主制作作品.....2

- ◆ Dive Chase Down...3
- ◆ Horrific House...10
- ◆ Unity Dash!!.....15
- ◆ Lock' n' Role.....16

## 2. 授業作品.....17

- ◆ 数学作品.....26





<自主制作作品>

# DIVE CHASE DOWN

・ダイブ・チェイス・ダウン・

Dive Chase Down: 1/7

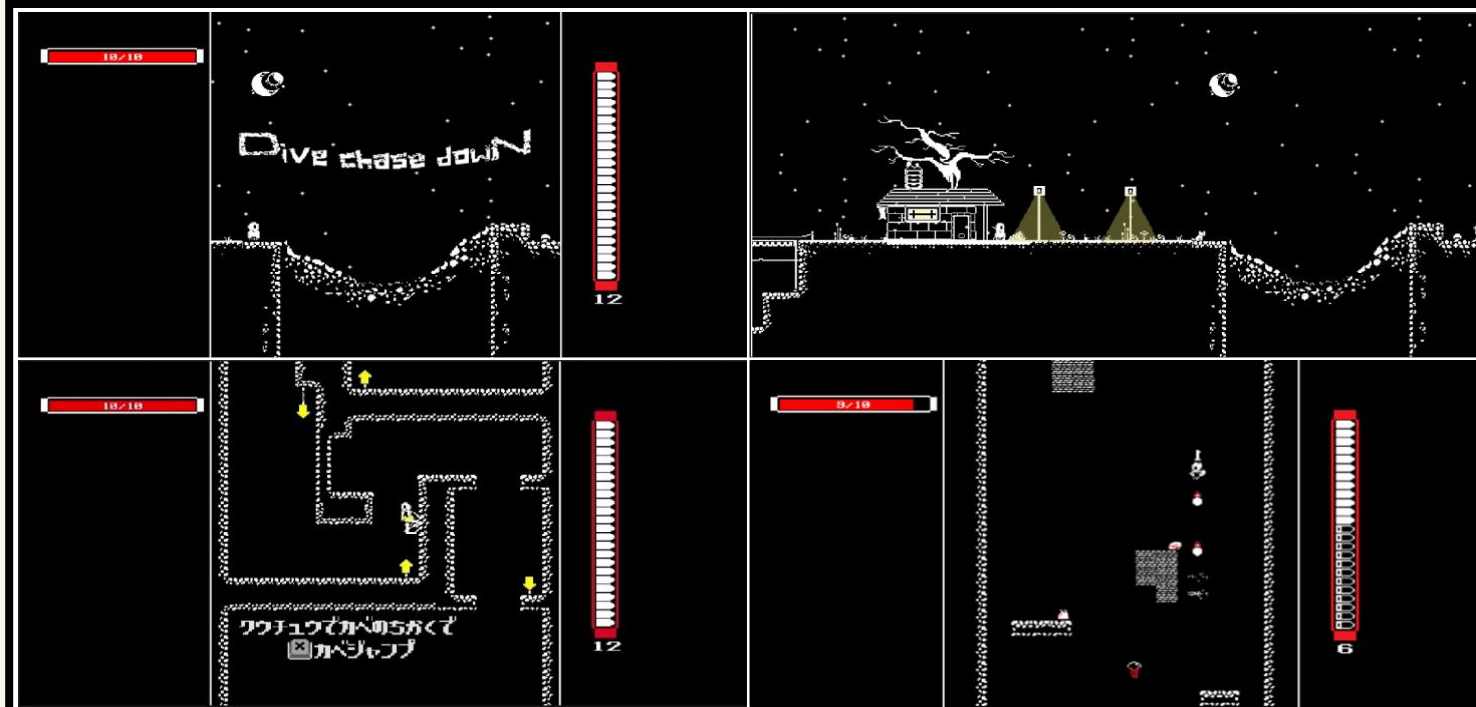
3年生の前期に学内コンテストとGC甲子園に向けて制作した作品です。

学内コンテストでは約80作品の中から第1位に入賞し、GC甲子園では、総合月間ランキング第2位になったこともあり、商品化希望もありました。

落下を主軸にしたアクションゲームです。下方向にしか攻撃ができず、体力が0にならないように、最下層にいる猫を探しに追いかける事が目標です。

雰囲気や演出・表現に力を入れ、ゲームの世界観を魅せるよう意識して制作しました。また、プレイヤー自身に目的を理解できるようカメラワークによる視線誘導にも力を入れています。

色を極端にして、敵とプレイヤーの視認性を高めると共に、レトロゲームのような雰囲気が出るように意識して制作しました。



プラットフォーム : PC

ゲームエンジン : Unity

使用言語 : Unityスクリプト (C#)

制作人数 : 1人

制作期間 : 4ヶ月

企画(1ヶ月)/プログラム(3ヶ月)

ジャンル : 下スクロールアクション

[プレイ動画] [ P V ]



<https://youtu.be/Cr1jhQIKZ34>



<https://youtu.be/n3NNqBSZwQ0>

# ゲームの流れ

Dive Chase Down:2/7

## Scene説明

○LogoScene  
ロゴ画像の描画

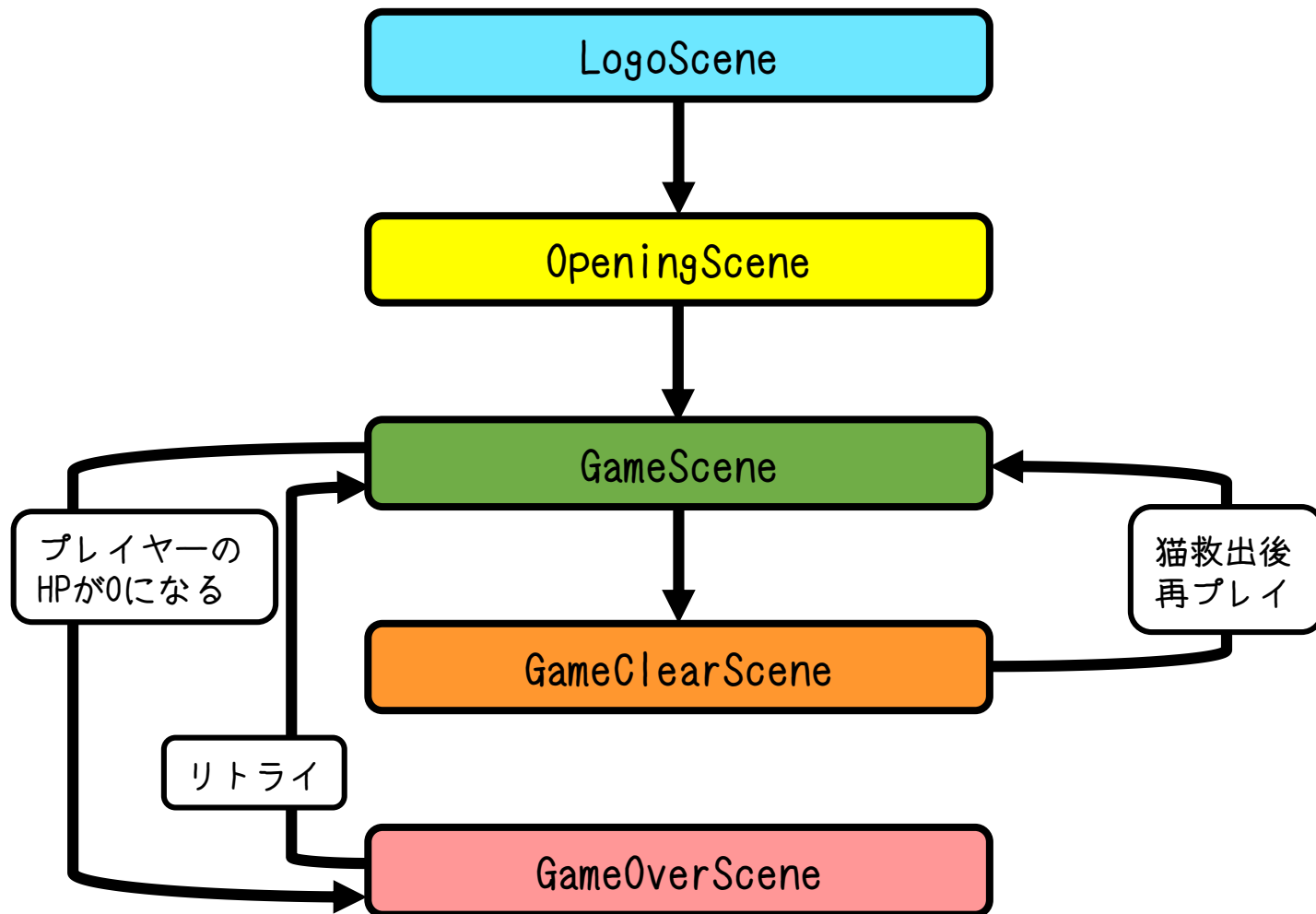
○OpeningScene  
オープニングの演出・描画

○GameScene  
タイトル・ゲームステージの描画

○GameClearScene  
クリア演出・画像の描画

○GameOverScene  
ゲームオーバー演出・画像の描画

## Scene遷移



# 企画

Dive Chase Down:3/7

## ○コンセプト

- ・2Dドット風味のゲームにしたい
- ・Unityを使用して学内コンテストに向けたゲームを制作したい



2Dドット風味を出して、コンテスト向けのゲームとして何か他の個性出すには？



## ○結論

白黒の配色をメインにした、2d下スクロールアクションのゲームを作る。

→色を限定することで独特な雰囲気を出して目に留まりやすくし、さらに下に降りるゲームという個性をつけければ注目度が増えると考えたため

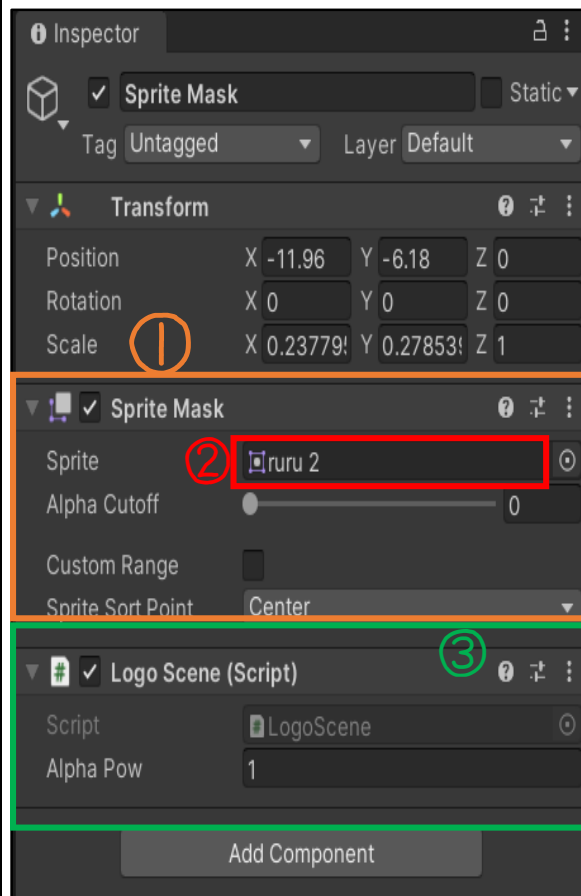
タイトル	Dive Chase Down
キャッチコピー	下へ、下へ
ジャンル	2D下スクロールアクション
想定ハード	PC
操作デバイス	キーボード・マウス/ゲームパッド
コンセプト	<ul style="list-style-type: none"><li>・2Dドット風味のゲームにしたい</li><li>・Unityを使用して学内コンテストに向けたゲームを制作したい</li></ul>
メインターゲット	成人男性、レトロゲームが好きな人 2Dアクションが好きな人
ポイント	レトロゲームのような雰囲気の中に今のゲームのような演出がある。

# 演出

Dive Chase Down:4/7

## Mask画像

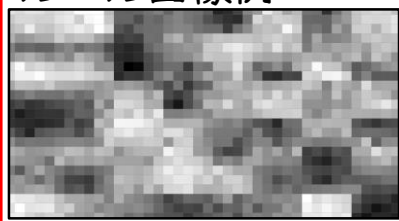
SpriteMaskコンポーネントを使用して、ルール画像を利用し、AlphaCutoffの値をスクリプトから徐々に加算してシーン遷移等の演出しています。



①SpriteMaskコンポーネントをMask処理を行うオブジェクトのInspectorにアタッチ。

②ルール画像をSprite画像として用意し、Spriteで用意した画像を選択。

ルール画像例



③シーン遷移の演出を行うscriptを制作し、Inspectorにアタッチ。

演出画面の動画

<https://youtu.be/UY7xMQHDRdw>



## LogoSceneScript

```
void Start()
{
    //SpriteMaskコンポーネントを取得
    mask_ = GetComponent<SpriteMask>();
}

void Update()
{
    //スプラッシュスクリーンの表示が終わってれば
    if (SplashScreen.isFinished)
    {
        if (!isLogo_)
        {
            isLogo_ = true;

            //コルーチン処理開始
            StartCoroutine(HometoScene());
        }

        if (isMask_)
        {
            if (mask_.alphaCutoff <= 1)
            {
                //alphaCutoffを1になるまで加算
                mask_.alphaCutoff += AlphaPow * Time.deltaTime;
            }

            if (mask_.alphaCutoff == 1)
            {
                isMask_ = false;

                //オブジェクトの表示をオフに
                gameObject.SetActive(false);

                isScene_ = true;
            }

            if (isScene_)
            {
                isScene_ = false;

                // HomeSceneに遷移
                SceneManager.LoadScene("Home", LoadSceneMode.Single);
            }
        }
    }
}
```

ゲーム起動時にコンポーネントの取得

ロゴの表示を待ってからマスク処理の開始フラグをtrueに(遅延処理は8Pに記載)

AlphaPowにdeltaTimeをかけてFPS依存に

画面を暗転させるために表示をオフ

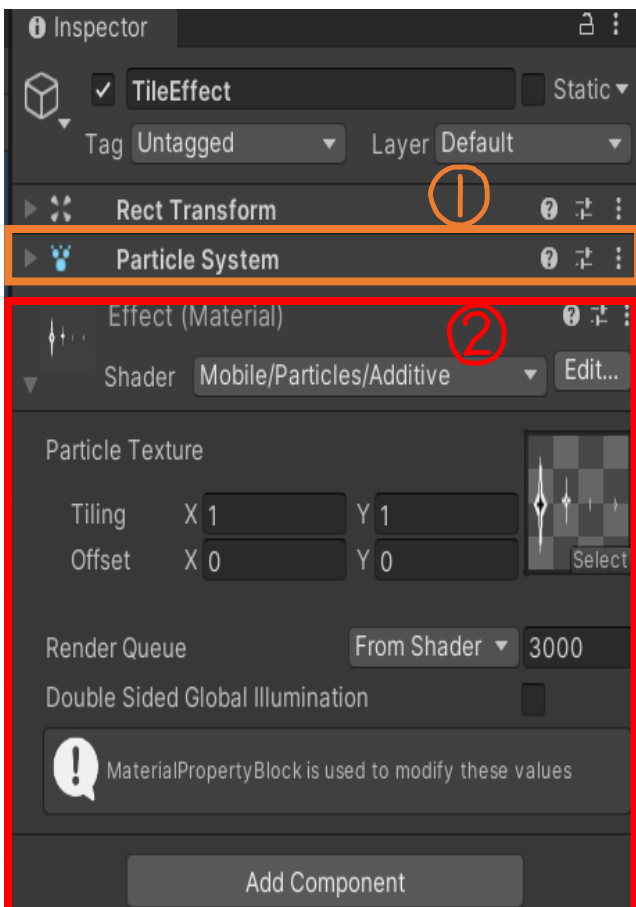
```
public IEnumerator HometoScene()
{
    //2秒待機
    yield return new WaitForSeconds(waitTime);
    isMask_ = true;
}
```

# 演出

Dive Chase Down:5/7

## タイトルロゴ

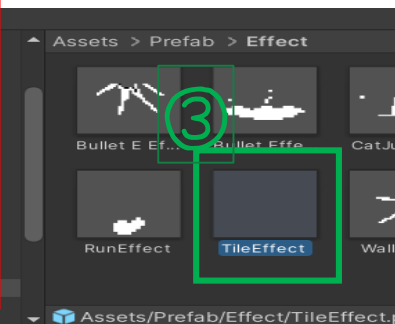
タイトルロゴの出現演出にはマスク処理とParticleSystemを使用したエフェクトも出現させ、演出を行っています。



①ParticleSystemコンポーネントをオブジェクトのInspectorにアタッチ。Particleの調整を行う。

②Particleとして使用したい画像を登録。また、Shaderを変更する。

③ScriptでInstantiateを使用するため、作成したエフェクトオブジェクトをプレハブ化。



演出の動画

[https://youtu.be/0\\_FWRR65FTQ](https://youtu.be/0_FWRR65FTQ)

## Particle処理

```
if (isParticle)
{
    //パーティクルシステムのインスタンス生成
    ParticleSystem newParticle = Instantiate(particle);
    // 発生場所をアタッチしているGameObjectの位置に
    newParticle.transform.position = this.transform.position
    //パーティクルの発生
    newParticle.Play();

    //効果音の調整
    Audio.pitch = 1;
    Audio.volume = 1;
    Audio.PlayOneShot(TitleLogoSE);

    //パーティクルの削除
    Destroy(newParticle.gameObject, ParticledDethTime);

    //パーティクルフラグをfalseに
    isParticle = false;
}
```

Instantiateで作成したプレハブを、このScriptをアタッチしているオブジェクトの位置に生成。その後ParticleSystemを再生モードに設定。

再生時間が終了したら、生成したプレハブを削除し、もう一度再生しないようにパーティクルフラグをfalseに設定



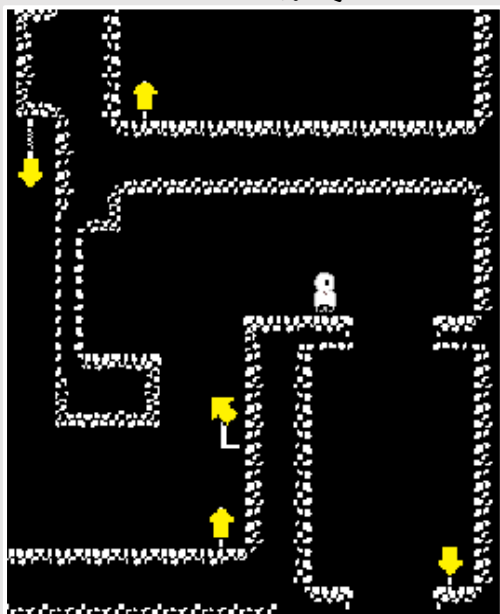
# 表現

Dive Chase Down:6/7

## 視線誘導

進行方向やゲーム目的の提示等を、このゲーム独特の雰囲気表現するために、テキストをなるべく使用せず、ムービーやキャラのアニメーション看板等のオブジェクトで行いました。

### 看板オブジェクトでの誘導



ムービー



<https://youtu.be/9Ph2ZXlouEA>

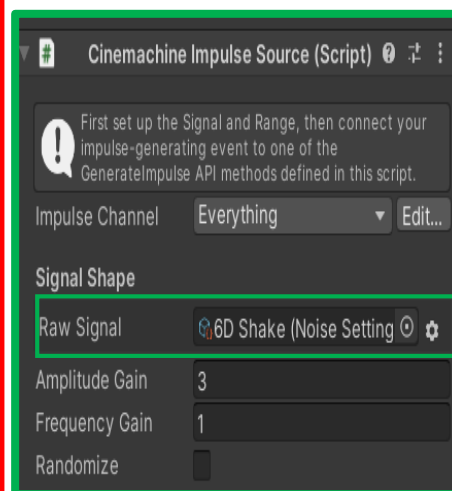
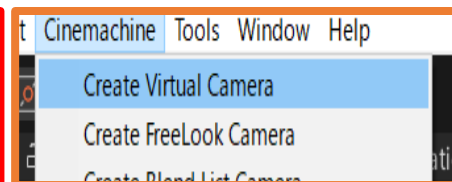
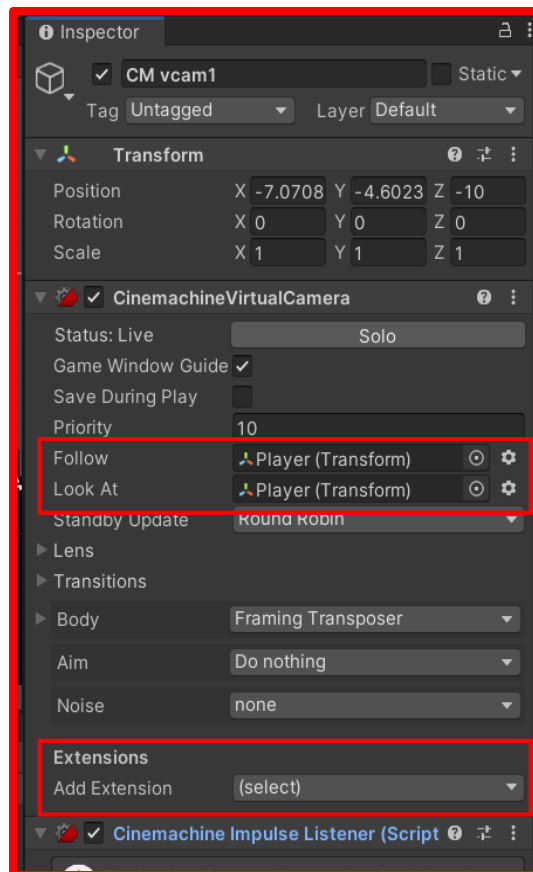
アニメーションでの誘導



<https://youtu.be/ISj6HHeGYGE>

## 画面振動

何か強い衝撃が起きたことを表現するために攻撃、被弾時等に画面振動を「Cinemachine」で実装しています。



```
//画面振動の信号送信
//CinemachineImpulseSource ImpulseSource;
ImpulseSource.GenerateImpulse();
```

画面振動時の動画

[https://youtu.be/Xba0\\_nz4P9g](https://youtu.be/Xba0_nz4P9g)



Cinemachiceをインストール  
メニューバーから仮想カメラ  
を作成。

仮想カメラをプレイヤーに追従  
させるためFollowとLookAtに  
Playerオブジェクトをアタッチ。  
そして画面振動をするために  
AddExtensionで  
CinemachiceImpulseListener  
を選択。

CinemachiceImpulseSourceを  
追加して、衝撃の揺れを表現  
するためRawSignalで6Dshakeを選  
択。

Scriptで画面を振動する時に  
「GnerateImpulse()」を使用。

# コルーチン / アニメーション

Dive Chase Down:7/7

## コルーチン

テキストの出現や穴に降りた演出等で、一拍置いた演出したい時にコルーチン(遅延処理)を使用しています。これにより演出等で一拍置いた独特な動きを実現しています。

```
//ステージ移動判定がtrue  
if (isStage)  
{  
    //遅延処理呼び出し  
    StartCoroutine(CameraMoveY());  
}
```

```
private IEnumerator CameraMoveY()  
{  
    //遅延処理  
    yield return new WaitForSeconds(waitTime);  
  
    //カメラの移動演出  
    this.transform.DOMove(  
        new Vector3(  
            transform.position.x,  
            -moveLenY,  
            transform.position.z),  
        moveSpeedY  
    ).OnComplete(() =>  
    {  
        //移動完了時  
        isCameraMoved = true;  
    });  
}
```

コルーチンを実行するために「StartCoroutine」関数を使用。

IEnumerator型で関数を作成。

時間的に遅延させたいのでWaitForSecondsで時間を指定。

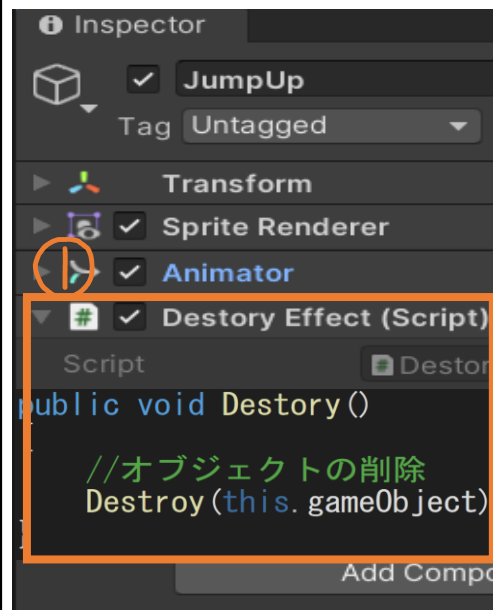
遅延処理後、演出処理を行う。

使用部分の動画  
<https://youtu.be/cXgz0Tq21bw>



## アニメーションイベント

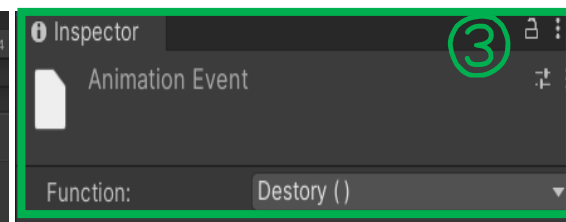
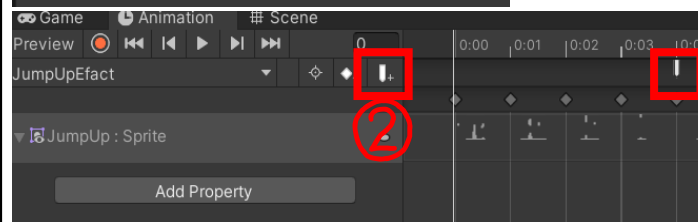
エフェクトを終了次第オブジェクトごと削除する際にUnityのAnimatorでは終了時を取得する機能がありませんでした。そのためアニメーションイベントを使用して終了時に関数を呼んで削除できるように実装しました。



①削除処理をするエフェクトオブジェクトに削除処理を書いたScriptをアタッチ。

②AnimationタブでAnimationEventを関数を使用するアニメーションのフレームに追加。

③配置したAnimationEventを選択してInspectorのFunctionをDestroy()に選択。



# HORRIFIC HOUSE

—ホリフィック ハウス—

Horrific House:1/6

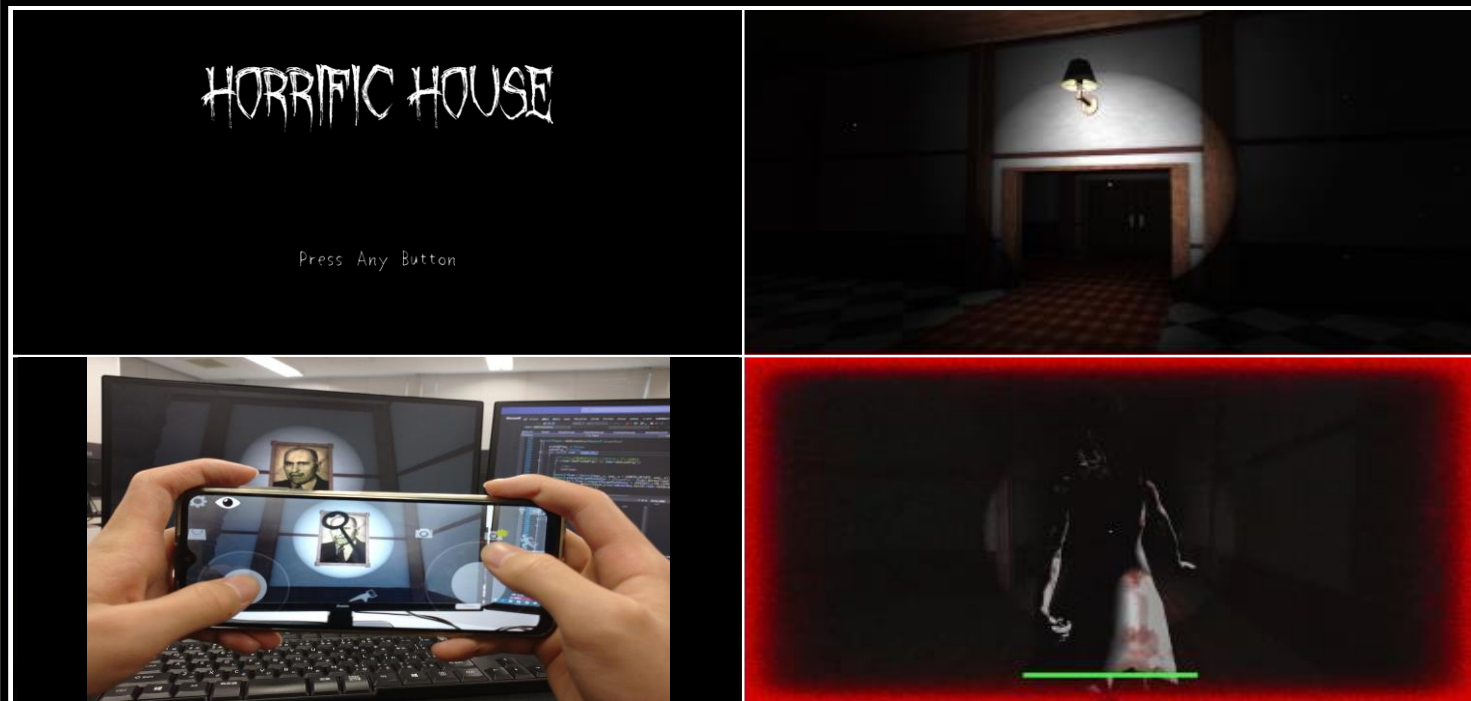
3年生の後期に学内コンテストに向けて制作した作品です。

予選時には約70作品の中から第1位に選ばれ、本選でも7作品の中から1位に入賞しました。

カメラを駆使して幽霊が徘徊している洋館から脱出する3D探索脱出ホラーゲームです。ARモードを搭載しており、実際にスマホを画面に向けてプレイできます。そのためのARアプリとステージはUnityで制作しています。

このゲームではチーム制作になり、ゲームの企画や演出・表現の考案、音声・画像の素材作成と収集、ステージ制作、プレイヤーのアイテム取得を担当しました。

企画の段階から、意見や演出・表現の案出しを積極的に行い、素材等の作成や収集、相談等、円滑にチーム制作ができるように意識して制作に当たりました。



プラットフォーム : PC  
ライブラリ : DXライブラリ / Unity  
使用言語 : C++ / C# / HLSL  
制作人数 : 5人  
制作期間 : 4ヶ月  
企画(1ヶ月)/プログラム(3ヶ月)  
ジャンル : 3D探索脱出ホラー

[プレイ動画] [ P V ]



<https://youtu.be/aXqmEGSkRu8>



<https://www.youtube.com/watch?v=Dh1w9SOcwKw>

# 担当箇所

Horrific House:2/6

○企画  
企画/音声・画像素材の収集と制作

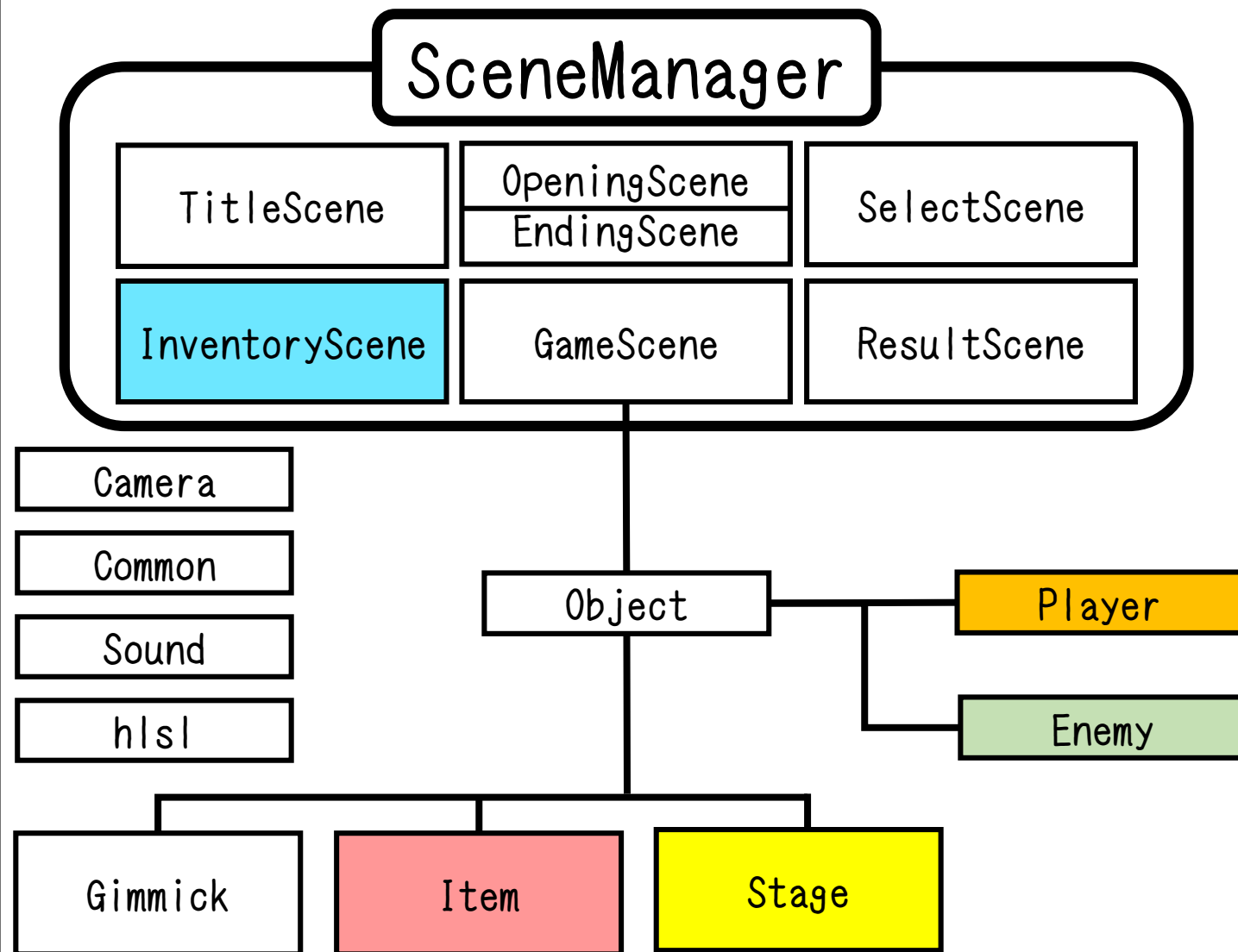
○Player  
アイテムの取得、インベントリに追加

○Item  
モデル・アイテムリストの制作

○Stage  
モデルの制作・描画

○Enemy  
モデル・アニメーションの制作

○InventoryScene  
アイテム説明の制作・描画



## ○コンセプト

- ・カメラを使用した目新しいようなゲーム
- ・チーム全体で3Dゲームのシェーダー等の研究をしたいという意見



カメラを使用し、なおかつチームの意見を取り込んだゲームにするには？



## ○結論

「零」シリーズのようなカメラをメインギミックに据えたホラーゲームにAR要素を追加  
→ホラーならアニメーションやアクションに割く時間を減らし、浮いた時間を利用し、シェーダーやエフェクトの演出・表現に挑戦できるため

タイトル	Horrific House
キャッチコピー	カメラが貴方を救う
ジャンル	3D探索脱出ホラー
想定ハード	PC
操作デバイス	キーボード・マウス/ゲームパッド
コンセプト	<ul style="list-style-type: none"><li>・カメラを使用した目新しいようなゲーム</li><li>・チーム全体で3Dゲームのシェーダー等の研究をしたいという意見</li></ul>
メインターゲット	成人男性、ホラーゲームが好きな人 何か目新しい要素をプレイしたい人
ポイント	実際にカメラを構えてプレイをするARモードがあり、新らしいプレイ体験ができる。



# Player / Item

Horrific House:4/6

## アイテムの取得

アイテムの取得と取得したアイテムをInventoryListに追加しています。そして、インベントリシーンでInventoryListをそのまま利用できるように実装しています。

①InventoryListの中身を確認。取得してれば同じアイテムを拾えないようフラグをtrueに変更。

②InventoryListに追加。この時にモデルスケールを初期化して、Scaleを代入することでインベントリシーンで同じ大きさに変更できるように実装。

取得時の挙動

<https://youtu.be/6UpXuggCZE4>



```
if (pickUpFlag_)
{
    if (controller_>CheckInputKey(KeyID::Interact))
    {
        //初めてアイテムを見つけたとき時、Tutorialflgをtrue
        lpTutorialMng.SetTutorialFlg(TutorialProgress::Action, true);
        bool flag = false;
        ① for (auto& inventory : inventoryList_)
        {
            if (inventory.itemID_ == item->GetItemID())
            {
                flag = true;
            }
        }
        if (!flag)
        {
            item->SetPickUpFlg(true);
            lpSoundMng.PlayingSound("../resource/sound/PickUpSE.mp3");
            ② VECTOR Scale = MV1GetScale(
                lpModelMng.GetID(item->GetModelString())
                [item->GetModelNumber()]);
            inventoryList_.emplace_back(
                item->GetModelString(),
                item->GetModelNumber(),
                item->GetItemID(),
                Vector3{ 0,0,0 },
                Vector3{ Scale.x,Scale.y,Scale.z },
                "",
                ItemID::Non
            );
            itemID_ = item->GetItemID();
            break;
        }
    }
}
```

## アイテムの構造体

インベントリシーンで使用するために必要なアイテムデータをまとめた構造体です。構造体にまとめることにより、アイテムデータを管理しやすいようにし、値の受け渡しが容易になりました。InventoryObj構造体を使用することでアイテムモデルやインベントリ内での座標等をインベントリシーンで利用できるようになっています。

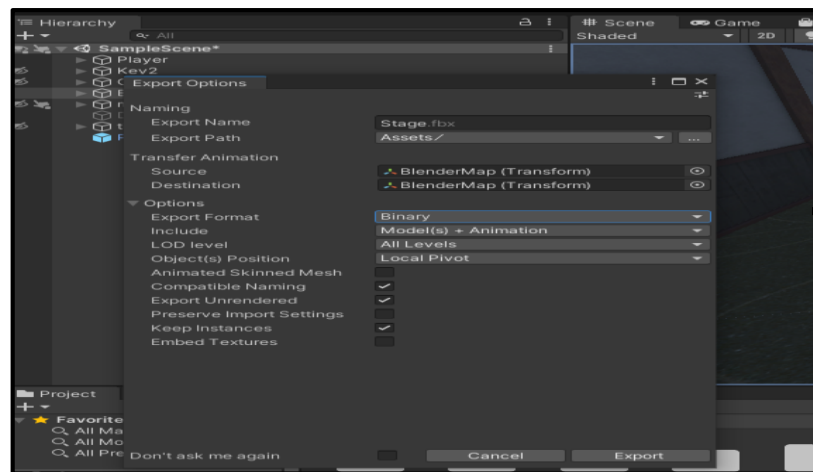
```
struct InventoryObj
{
    std::string modelString_; //モデルハンドル
    int modelNum_; //複製数
    ItemID itemID_; //アイテムID
    Vector3 pos_; //座標
    Vector3 scale_; //拡大値
    std::string inventoryNum_; //インベントリ数
    ItemID usedItemID_; //アイテムID
};
```

Stage / Enemy

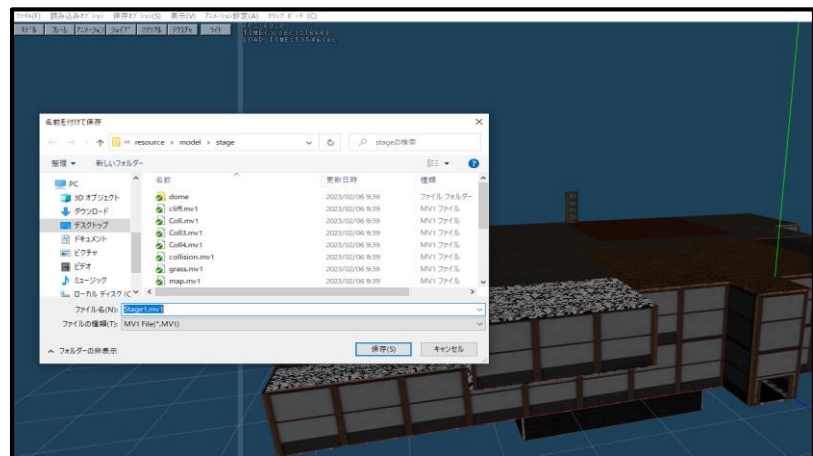
Horrific House:5/6

## Stage制作

ステージの制作にはUnityを使用。  
FBXで出力しDxLibModelViewerで、Mv1形式に変換。



## FBXで出力

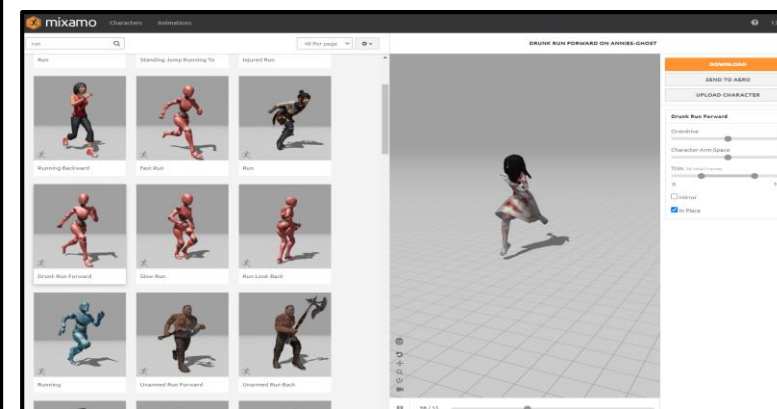


## FBXで入力

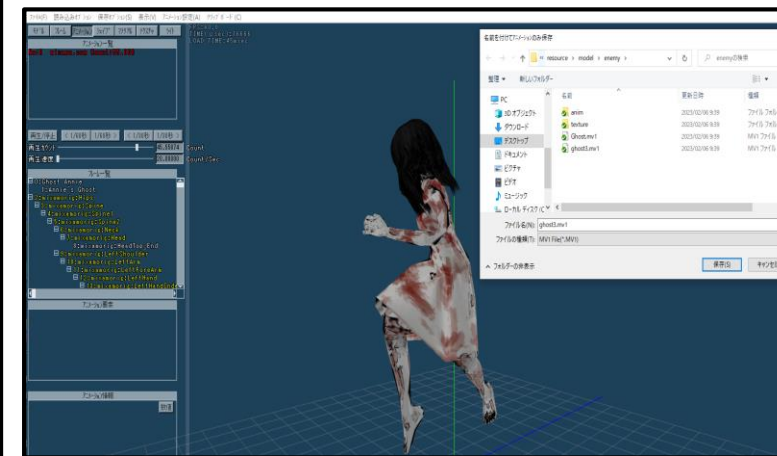
mv|で出力

## Enemy関連

敵のモデルはフリー素材でFBX形式を使用。  
アニメーションは、[mixamo](#)を使用して制作。  
最後に全てDxLibModelViewerで、mvl形式に変換。



FBXで  
出力



FBXで  
入力

アニメーション  
のみでmv1で出力

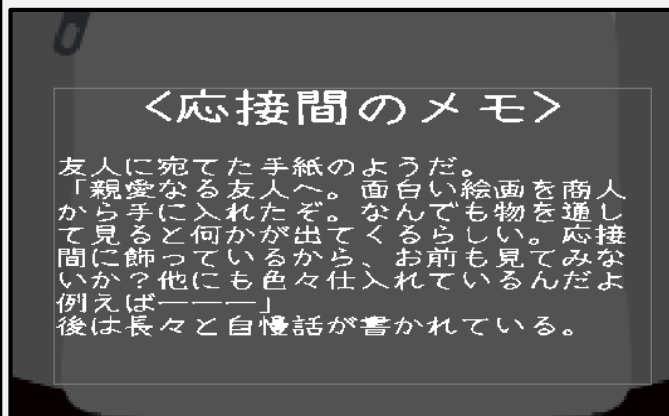
# InventoryScene

Horrific House:6/6

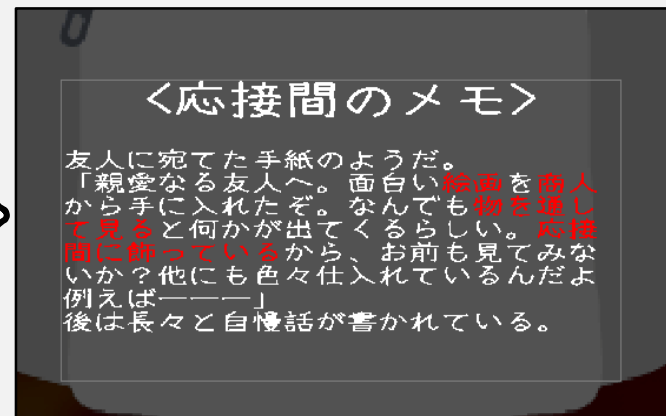
## アイテム説明

アイテム説明が全て一色だと理解するのに時間がかかり、またあまり印象に残らない、という問題がありました。これを改善するため、**重要部分のみを赤色**に変更しました。これにより印象に残りやすくし、またプレイヤーがメモの内容を理解できやすくなりました。

変更前



変更後



## 画像の登録

画像をBaseSceneで登録することによりソースコード見やすくまとめ、可読性を向上させています。

そして、InventorySceneの**テーブル**に、**まとめて**、変更を容易にできるようにしています。

BaseSceneに登録

```
//インベントリ画像
lplImageMng.GetID("resource/image/inventory/kitchenkeydescribe.png", "kitchenkeydescribe"); //キッチンの鍵の説明画像
lplImageMng.GetID("resource/image/inventory/Entrancekeydescribe.png", "Entrancekeydescribe"); //玄関の鍵の説明画像
lplImageMng.GetID("resource/image/inventory/Pantrykeydescribe.png", "Pantrykeydescribe"); //食糧庫の鍵の説明画像
lplImageMng.GetID("resource/image/inventory/musicroomkeydescribe.png", "musicroomkeydescribe"); //音楽室の鍵の説明画像
lplImageMng.GetID("resource/image/inventory/childRoomkeydescribe.png", "childRoomkeydescribe"); //子供部屋の鍵の説明画像
lplImageMng.GetID("resource/image/inventory/basementkeydescribe.png", "basementkeydescribe"); //地下室の鍵の説明画像
lplImageMng.GetID("resource/image/inventory/Photodescide.png", "Photodescide"); //写真の説明画像
lplImageMng.GetID("resource/image/inventory/enemymemo.png", "enemymemo"); //敵の弱点の説明画像
lplImageMng.GetID("resource/image/inventory/storymemo_a.png", "storymemo_a"); //ストーリーメモaの説明画像
lplImageMng.GetID("resource/image/inventory/storymemo_b.png", "storymemo_b"); //ストーリーメモbの説明画像
lplImageMng.GetID("resource/image/inventory/storymemo_c.png", "storymemo_c"); //ストーリーメモc兼壁の説明画像1
lplImageMng.GetID("resource/image/inventory/hint_d.png", "hint_d"); //壁の説明画像2
lplImageMng.GetID("resource/image/inventory/hint_f.png", "hint_f"); //調理器具の説明画像
lplImageMng.GetID("resource/image/inventory/hint_g.png", "hint_g"); //楽譜の説明画像
lplImageMng.GetID("resource/image/inventory/hint_h.png", "hint_h"); //ゲームソフトの説明画像
lplImageMng.GetID("resource/image/inventory/hint_i.png", "hint_i"); //応接間の絵画の説明画像
lplImageMng.GetID("resource/image/inventory/hint_j.png", "hint_j"); //猫の時計と絵画の説明画像
lplImageMng.GetID("resource/image/inventory/hint_t.png", "hint_t"); //警告版の説明画像
lplImageMng.GetID("resource/image/inventory/mapItemdescide.png", "mapItemdescide"); //マップアイテムの説明画像

lplImageMng.GetID("resource/image/Load/gageImage4.png", "fram"); //マスク画像
```

テーブルにまとめる

```
explanationTable = {
    [ItemID::Entrance_Key, "Entrancekeydescribe"], //玄関の鍵の説明画像
    [ItemID::Kitchen_Key, "kitchenkeydescribe"], //キッチンの鍵の説明画像
    [ItemID::MusicRoom_Key, "musicroomkeydescribe"], //音楽室の鍵の説明画像
    [ItemID::FoodBank_Key, "Pantrykeydescribe"], //食糧庫の鍵の説明画像
    [ItemID::ChildRoom_Key, "childRoomkeydescribe"], //子供部屋の鍵の説明画像
    [ItemID::Celler_Key, "basementkeydescribe"], //地下室の鍵の説明画像

    [ItemID::Enemy_Memo, "enemymemo"], //敵の弱点の説明画像

    [ItemID::Story_Book_A, "storymemo_a"], //ストーリーメモaの説明画像
    [ItemID::Story_Memo_B, "storymemo_b"], //ストーリーメモbの説明画像
    [ItemID::Story_Book_C, "storymemo_c"], //ストーリーメモc兼壁の説明画像1

    [ItemID::Wall_Memo, "hint_d"], //壁の説明画像2
    [ItemID::Picture_Memo, "hint_q"], //応接間の絵画の説明画像
    [ItemID::CookWare_Memo, "hint_f"], //調理器具の説明画像
    [ItemID::Score_Memo, "hint_g"], //楽譜の説明画像
    [ItemID::Living_Memo, "hint_h"], //ゲームソフトの説明画像
    [ItemID::Clock_Book, "hint_i"], //猫の時計と絵画の説明画像
    [ItemID::Celler_Memo, "hint_t"], //警告版の説明画像

    [ItemID::Phot, "Photdescide"], //写真の説明画像
    [ItemID::Mapitem, "mapItemdescide"] //マップアイテムの説明画像
}
```



# UNITY DASH!!

◆◆ユニティダッシュ◆◆

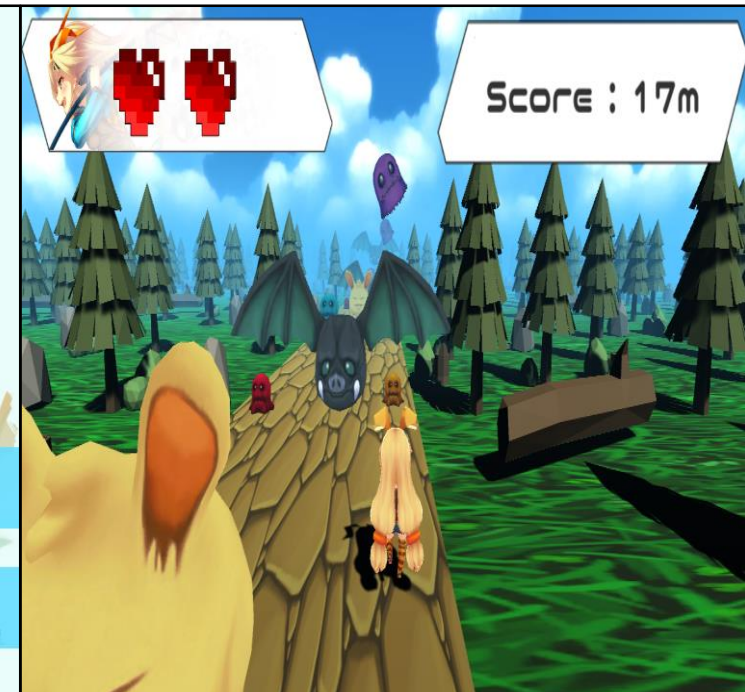
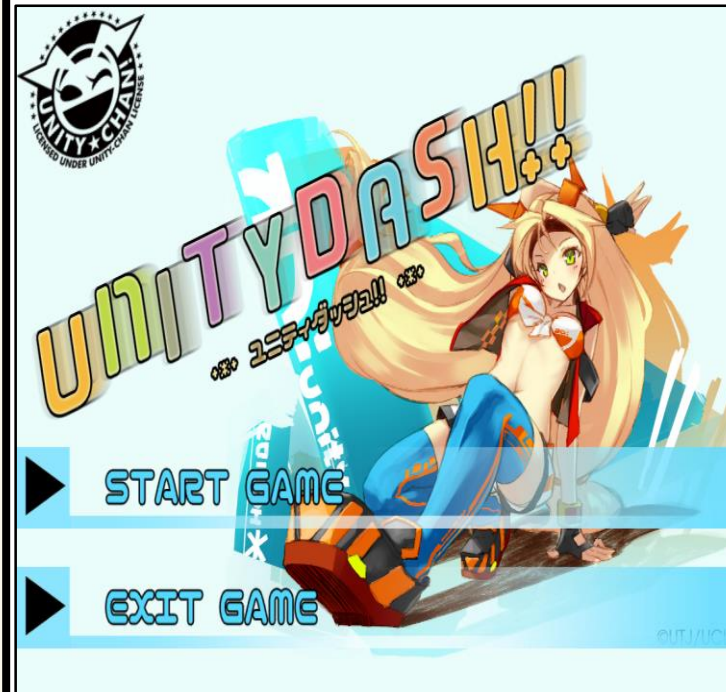
Unity Dash!! : 1/1

2年生の後期に制作した作品です。  
ゲームエンジンがどういう物か知るために  
初めてUnityを使用した自主制作です。

自動的に前方に走るユニティちゃんを操作  
して、左右移動とジャンプで障害物の敵を  
避けてスコアを延ばすゲームです。  
操作はキーボード・マウスのみになります。

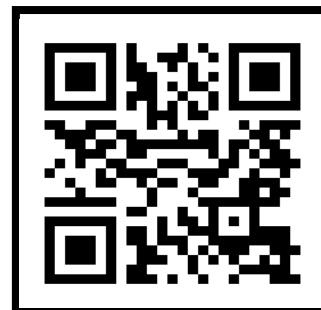
このゲーム制作ではUnityでのゲーム制作を  
知る事や操作方法に慣れる事を目的に制作  
しました。特にオブジェクトの生成・配置等  
の基本操作や、画像・3Dモデルの読み込み  
と使用方法、アニメーション、スクリプト  
の書き方に意識して制作しました。

この制作を通してUnityの使用感、独特の仕  
様等を知る事ができ、同時にDXライブラリ  
での制作とは違った苦労も体験することが  
できて、今後の良い経験になりました。



プラットフォーム : PC  
ゲームエンジン : Unity  
使用言語 : Unityスクリプト (C#)  
制作人数 : 1人  
制作期間 : 4ヶ月  
企画(1ヶ月)/プログラム(3ヶ月)  
ジャンル : ランゲーム

[プレイ動画]



<https://youtu.be/5MvIwUbHSKE>

# LOCK'N'ROLE

◆ロックンロール◆

Lock' n' Role: 1/1

2年生の前期に制作した作品です。  
チーム制作を経験するためにクラス内で、  
3人のチームを組んで制作しました。

HPかSAN値が0にならないように最後まで話を  
読み進めるノベルゲームです

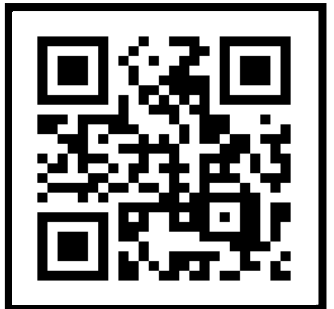
担当した箇所は、ゲームの企画立案や、シナ  
リオ・テキストの作成と実装、立ち絵の描画  
ステータス管理、ダイス処理、素材の収集を  
担当しました。

最初のチーム制作で、制作の進行方法を模索  
しながらの制作になってしまい、またチーム  
内のコミュニケーションが上手くいかず、  
ゲームとしての完成度が低くなってしまいま  
した。ですがこの経験は、後のチーム制作で  
は、反省を生かして制作に当たることができ  
たので、良い経験だったと感じています。



プラットフォーム : PC  
ライブラリ : DXライブラリ  
使用言語 : C++  
制作人数 : 3人  
制作期間 : 4ヶ月  
企画(1ヶ月)/プログラム(3ヶ月)  
ジャンル : TRPG風ノベルゲーム

[プレイ動画]



<https://youtu.be/jLxwwKa3At4>



<授業作品>

# 1年次

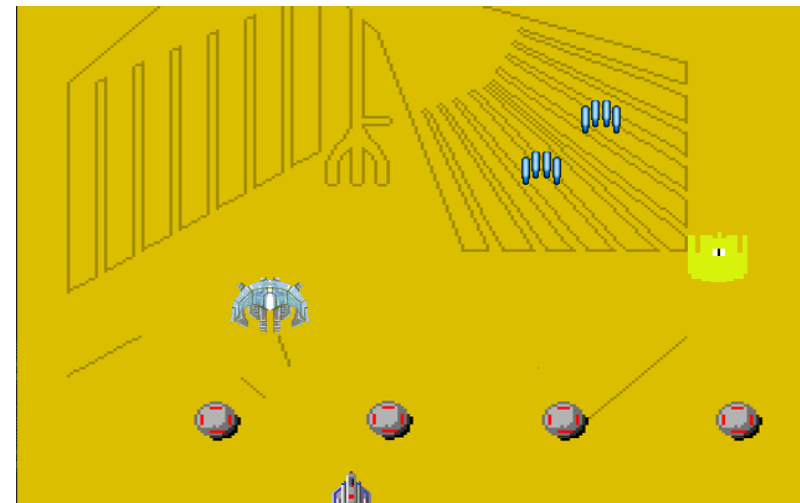
使用ライブラリ：DXライブラリ  
使用言語：C言語  
制作時期：1年生後期

[動画]

<https://youtu.be/LgkyXhfKZQM>

入学して初めて制作したゲームです。  
初めてプログラミングをしたので全てが新鮮でした。  
1枚の画像を繰り返して描画し、スクロールさせる処理や、  
敵が画面外に出た時、ランダムな位置から降りてくる処理  
の実装に苦労しました。処理結果を紙に書いてイメージし  
やすくする等の工夫をして実装しました。

## シューティング



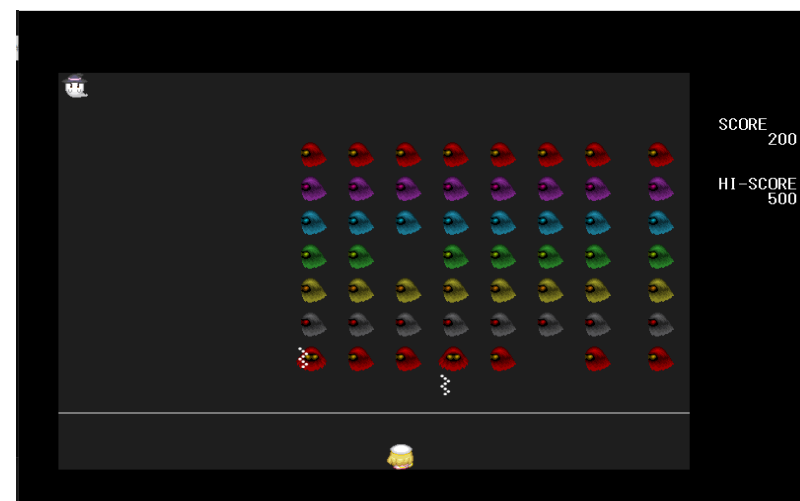
使用ライブラリ：DXライブラリ  
使用言語：C言語  
制作時期：1年生後期

[動画]

<https://youtu.be/LgkyXhfKZQM>

インベーダーゲームを参考にした作品です。  
ここでは配列やハイスコア計算等の実装を学習しました。  
敵を複数出すための配列やスコアの保存に苦労し、プログラムの難しさを改めて実感しました。放課後に先生方や友人に相談し、もう一度コードの説明を聴いたり、処理の内容を見直す等をして、実装することができました。

## インベーダー





# 1 年次

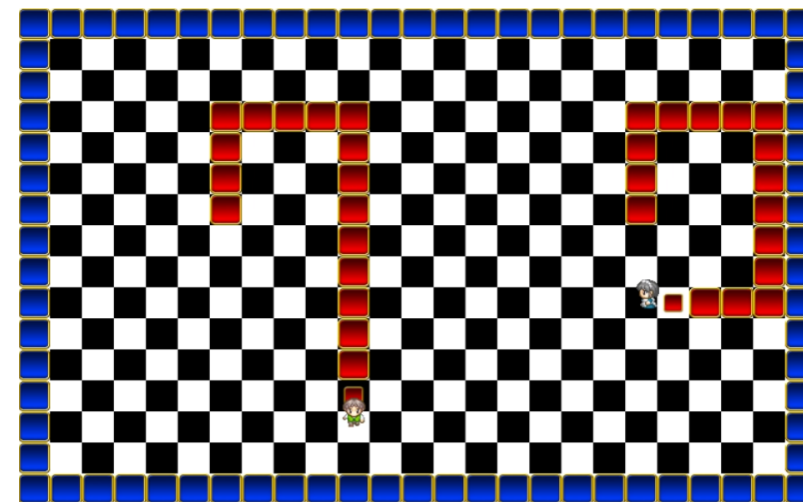
使用ライブラリ：DXライブラリ  
使用言語：C++  
制作時期：1年生後期

[動画]

<https://youtu.be/JJXmwRfEFNO>

C++を初めて使用して制作したゲームです。二人対戦型のゲームで移動した後にブロックが配置され、そのブロックに当たると敗北になります。シーンやオブジェクトごとにクラス分けして、プレイヤーを配列で制作しています。歩いた後にブロックを描画する処理に苦労しました。放課後に友人等に質問し、実装することができました。

スネークゲーム



使用ライブラリ：DXライブラリ  
使用言語：C++  
制作時期：1年生後期

[動画]

<https://youtu.be/6Jq2UPtkZSg>

初めてenum classやswitch文を用いたプレイヤーの移動・敵パラメータ・敵のスポン・マップの影響・入力状態のキーチェック・マップの切り替え等を学習しました。敵がこちらの位置に対して壁に当たらないように追いかけてくる処理に苦労し、一度紙に計算や図を書いてイメージがつくようにする等、工夫して実装しました。

ASOUL QUEST



# 1年次

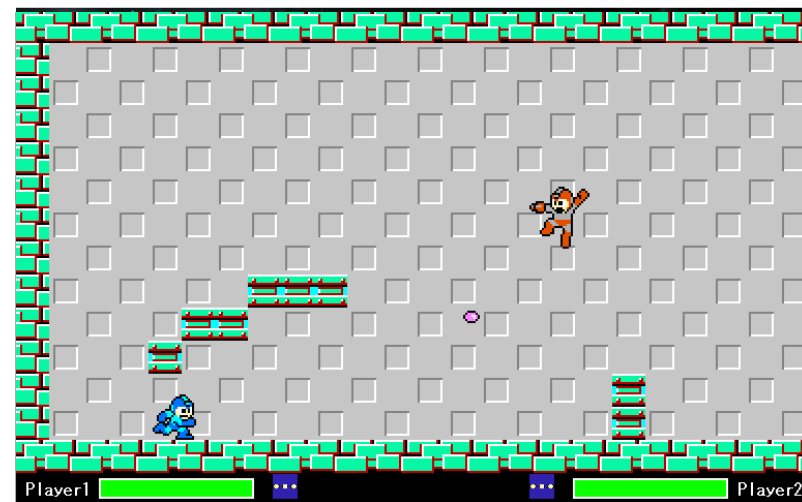
使用ライブラリ：DXライブラリ  
使用言語：C++  
制作時期：1年生後期

[動画]

<https://youtu.be/Zy5pE5Vmt00>

二人対戦の作品になります。ロックマンをイメージしており発射した弾を他のプレイヤーに当ててHPを無くせば勝利になります。この作品では自然な動きを表現するため、重力を考慮したジャンプや移動時の慣性等を学習して実装することができました。マップは画像を分割し配列を使用して、ソースコード内で変更できるよう実装しています。

## ロックマン



## 2年次

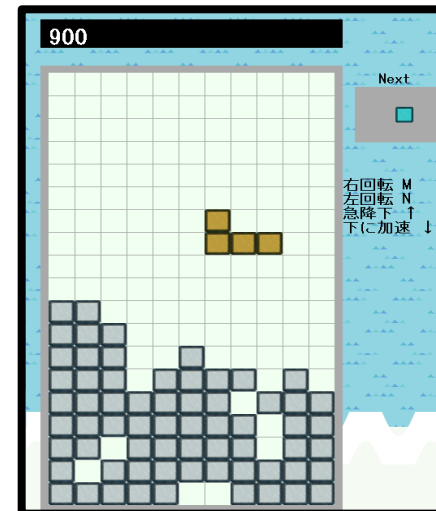
使用ライブラリ：DXライブラリ  
使用言語：C++  
制作時期：2年生前期

[動画]

<https://youtu.be/E06pGjrkTME>

テトリスのような落ちもののパズルゲームです。テトリミノのランダム生成や、形が崩れない回転、次のテトリミノの表示等の実装に苦労しました。その中でもテトリミノの消去の実装に苦労していて、消す（揃っている分）→演出→落とす、という処理が上手くいかず苦労しました。先生や友人に聞いたり再度説明を受けることで実装できました。

### 4Blocks



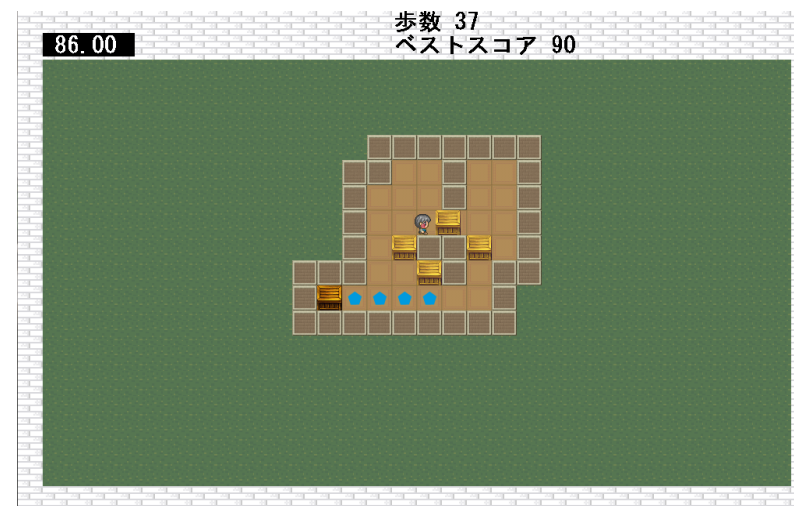
使用ライブラリ：DXライブラリ  
使用言語：C++  
制作時期：2年生前期

[動画]

<https://youtu.be/n4gFM8qsJPA>

荷物を押して時間内に指定の場所に移動させるパズルゲームです。この作品では線形補間による移動や、動的配列クラスの実装、外部ファイルの読み込み等を学習しました。他にもゲーム中のメニューの実装や、色の合成も行うなど学習する物が多く大変な作品でした。理解できなかった箇所等は、放課後先生や友人に聞き制作にあたりました。

### ASOBase



## 2年次

使用エンジン:Unity  
使用言語:C#script  
制作時期:2年生後期

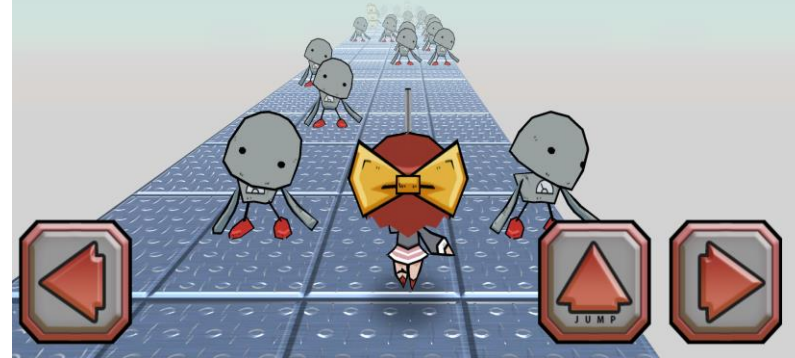
[動画]

[https://youtu.be/3\\_tMyW3FX6o](https://youtu.be/3_tMyW3FX6o)

Unityを初めて使用し、資料を基に、制作した作品です。勝手に進むプレイヤーを操作し、敵を避けるゲームです。Unityの3Dゲームの基本的な操作やオブジェクトの制作と配置方法、スクリプトの組み方を学習しました。初めてのゲームエンジンを触り、Unity独特なシステムに慣れるのが大変でしたが、なんとか制作できました。

### NejikoRun

Score: 31m



使用ライブラリ:Unity  
使用言語:C#script  
制作時期:2年生後期

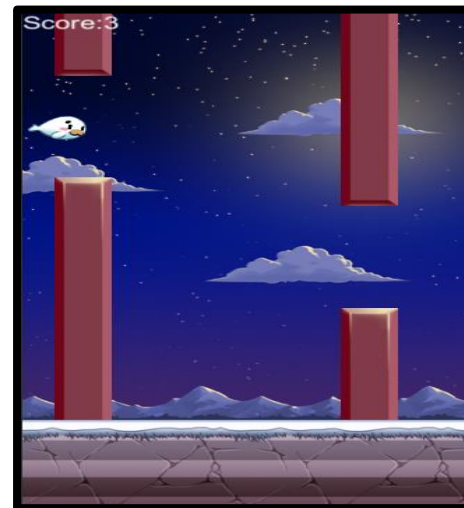
[動画]

<https://youtu.be/JYGDxKGUKTM>

NejikoRunの時と同じく、資料を基に制作した作品です。画面をクリックしアザラシを壁に衝突させずに進むゲームです。3Dの時にはしなかった画像のアニメーションやオブジェクトのランダム生成、ビルド設定を学習しました。ランダム生成される時の位置調整に苦戦しましたが、無事実装することができました。

### FlappyAzarashi

Score: 3





## 2年次

使用ライブラリ：DXライブラリ  
使用言語：C++  
制作時期：2年生後期

[動画]

<https://youtu.be/5ScV0AhsHRM>

初めての3Dで制作したシューティングゲームです。イベントシーンの制作や3Dカメラについて学びました。他にもパーティクルの生成・大砲の配置を外部データとして実装しています。ばね付きの追従カメラとイベントシーンの実装に苦労しました。放課後に先生方や友人に相談・質問を積極的に行い、どうにか実装することができました。

### 3Dシューティング



使用ライブラリ：DXライブラリ  
使用言語：C++  
制作時期：2年生後期

[動画]

<https://youtu.be/ldi2nZ06SY8>

3Dシューティングで学習した3Dゲーム制作の技術や知識を元に、それに加えマリオギャラクシーのような重力制御やカメラ・プレイヤー制御を学習しました。重力の切り替えや発射台の実装に苦労し、実装するのに時間がかかりました。放課後に残り時間をかけることで何とか授業の進行に追いつくようにしていました。

### 3Dアクション

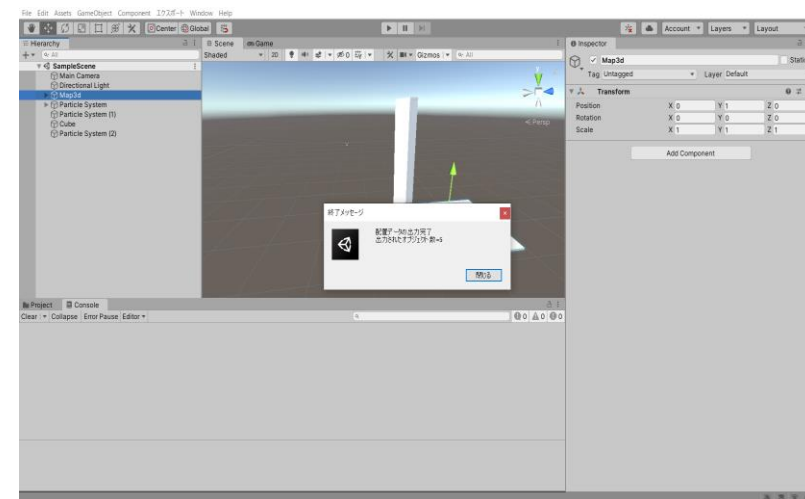


# 3年次

使用エンジン:Unity  
使用言語:C#script  
制作時期:3年生前期

ゲームではなくUnityのメニュー拡張を学習した作品です。  
ヒエラルキータブのオブジェクトを選択し、エクスポートすることで「.dat」形式で外部に配置データを出力します。  
Unityのウィンドウ拡張は初めてで、ゲームエンジンの拡張性に驚きました。

## メニュー拡張



使用ライブラリ:UnrealEngine4  
使用言語:ブループリント  
制作時期:3年生前期

[動画]

<https://youtu.be/3XzKybgFT2I>

初めてアンリアルエンジン4で制作した作品です。  
基本的な移動や生成、破棄などを実装し、応用でボーンやシェーダーの使用方法を学びました。  
ブループリントを使用し、プレイヤーの体力や制限時間、死亡時の演出等を追加で実装し、ゲームとして遊べる程度には仕上げました。

## UnrealEngine4



< 数学作品 >

## 2年次

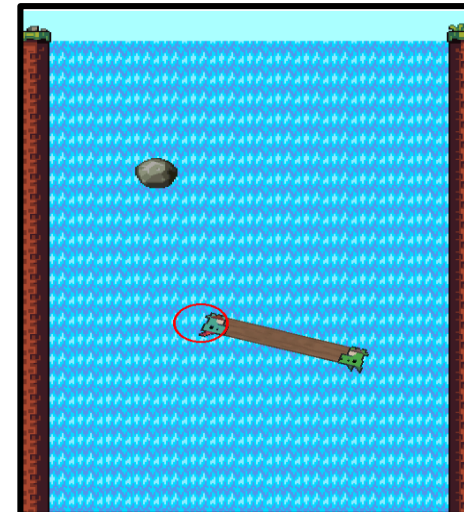
使用ライブラリ：DXライブラリ  
使用言語：C++  
制作時期：2年生前期

[動画]

<https://youtu.be/LgkyXhfKZQM>

カプセルと円の当たり判定を利用して回転の軸を左右選択し、回転させ一番上まで登っていくゲームです。  
これまで数学等で習った部分を復習としてゲームに落とし込むことになり、苦戦しました。

丸太運び



使用ライブラリ：DXライブラリ  
使用言語：C++  
制作時期：2年生前期

[動画]

<https://youtu.be/ogyIkHThPIk>

数学を応用したシューティングゲームの弾幕を制作しました。 $\sin \cdot \cos \cdot \pi$ を使って弾の角度を計算して弾幕の打ち方を変化させ、追尾してくる弾や周囲に弾をばらまいたり、噴水のような発射方法を実装しました。  
他にも平方根を使用した当たり判定を学習しました。

シューティングの弾幕



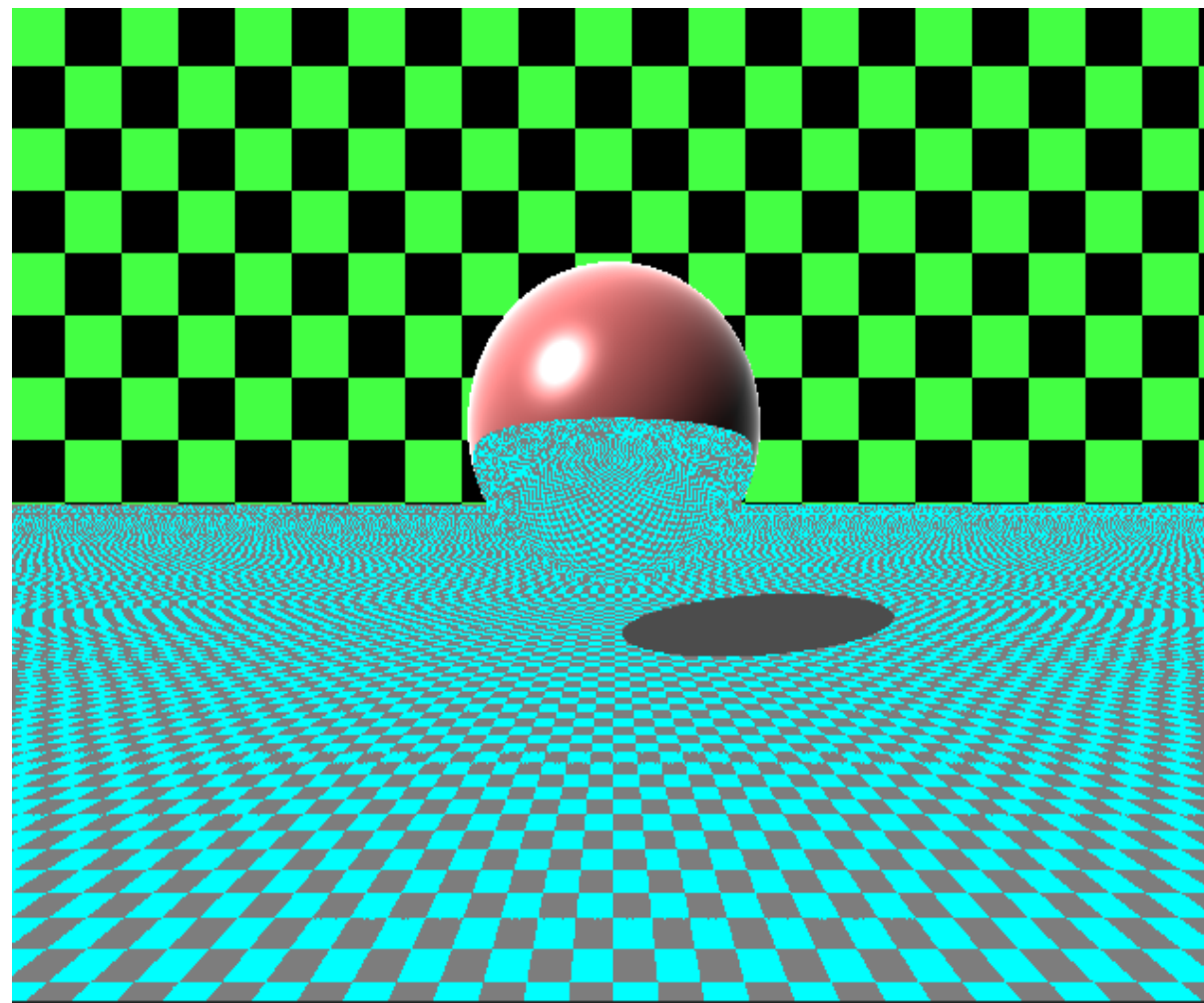


## 2年次

使用ライブラリ：DXライブラリ  
使用言語：C++  
制作時期：2年生後期

古典的なレイトレーシングについて学習し、球体の明るさ・スペキュラ・地面からの反射・影を実装した作品です。床と背景の模様もプログラムで描画しています。レイを飛ばして、球体に当たったら反射ベクトルを作成、という処理を繰り返して、球体に地面の模様を反射させています。数学があまり得意ではないということもあり大苦戦した作品ですが、友人に質問等をしてどうにか実装することができました。特に苦戦したポイントとして地面の反射の処理部分で、反射ベクトルの計算等に多くの時間をかけました。

レイトレーシング



THANK YOU.