PORTFOLIO

プログラマー

浦山 巧

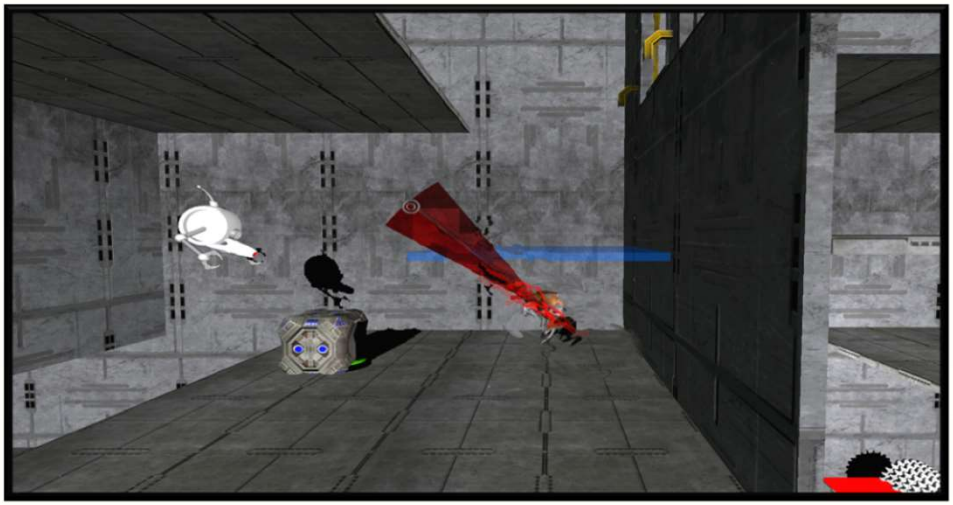
ASOポップカルチャー専門学校
ゲーム・CG・アニメ専攻科　ゲーム専攻

NOT SCRAP

夏の学内コンテストに向けてチーム制作した作品でゲーム甲子園やGFFに応募しました。

方向性の持った磁石を使い引き寄せや反発を使いクリアを目指すアクションゲームです。

私は主にBullet3を使ったギミックやプレイヤーの挙動と当たり判定を行いました。
初めての３D作品で大量のバグを出しながら制作しました。他にも、物理演算エンジンもこの作品で初めて使用して計算してくれるありがたさと物理法則従った動きを制御するのが難しくとても苦戦した作品です。



プラットフォーム	PC
ライブラリ	DXライブラリ
物理演算エンジン	Bullet3
使用言語	C++ / HLSL
制作人数	４人
制作期間	2022年4月～2022年8月 企画1ヵ月　制作3ヵ月
ジャンル	3Dパズルアクション

プロジェクトのURL　:https://drive.google.com/file/d/1OMOe7VAb0FriYxLrFhjCJa9RdtOE_zoB/view?usp=share_link

担当箇所

Player

Bullet3を使ったプレイヤーの移動制御,当たり判定、DXライブラリの機能を使ったライト

Stage

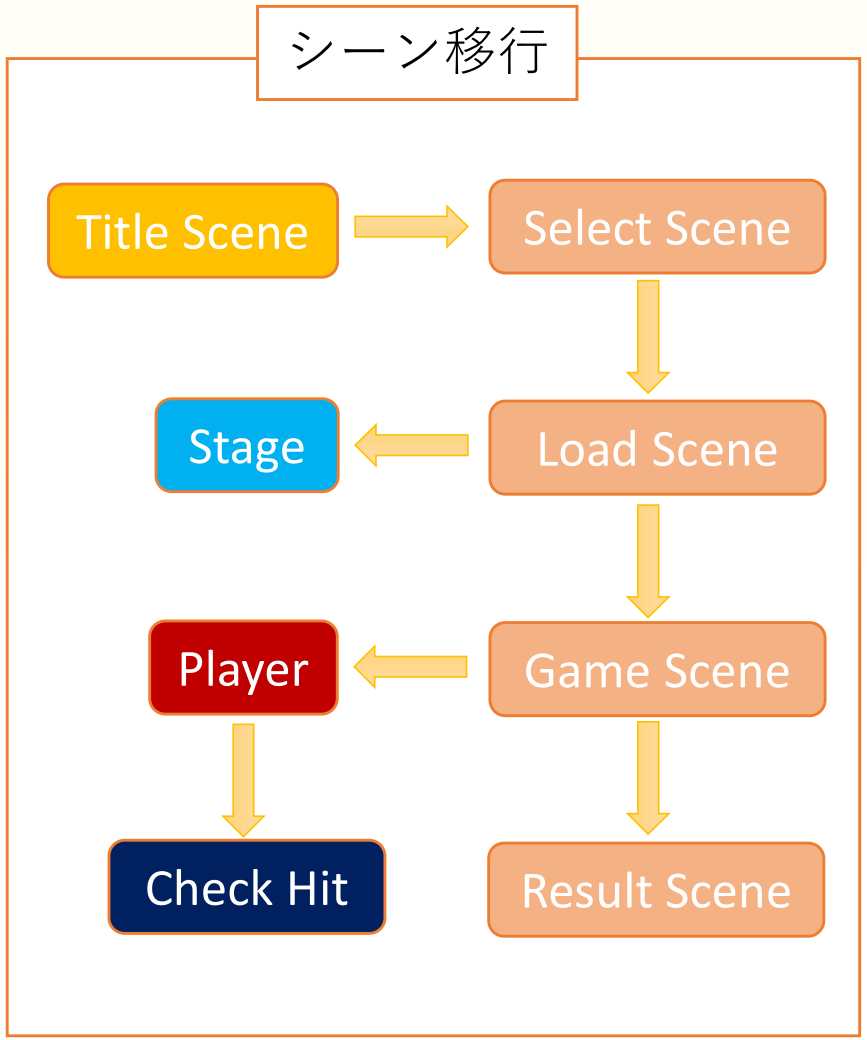
Bullet3用にマップの当たり判定モデルから衝突形状を作成

Title Scene

モデルの作成、スクラップの挙動、DXライブラリの機能を使った影

Check Hit

当たり判定用クラス作成



Player

軸の固定

ゲームの仕様上Z方向の移動をしてはいけないのでBullet 3をそのまま使うと真横に飛んでも壁に当たった時の反動で少しのずれが発生してプレイヤーの動きがおかしくなっていました、なのでset Linear Factor()でZ方向に行かないように固定しています。回転も同じくset Angular Factor()で倒れないようにしています。

```
//移動可能な方向の設定
body->setLinearFactor(LinearVec);
//回転可能な方向の設定
body->setAngularFactor(AngularVec);
```

磁石の挙動

下方向にレイを飛ばして磁石があった場合Bullet3で作って置いたPlayer bodyに磁石の方向の力を段々と加えてレイが当たらなくなったら加える力を消して磁石っぽい動きを再現しています。

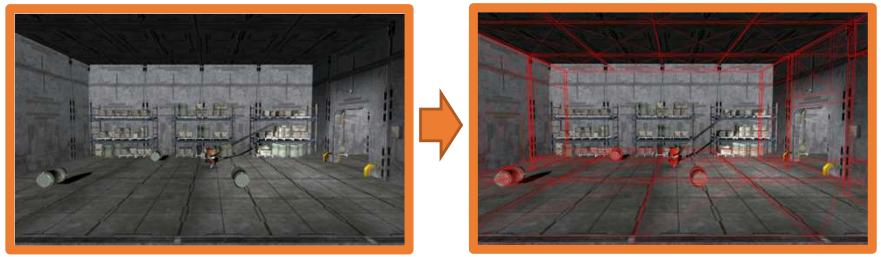
```
//レイが当たってる間上昇する
auto NorthFlag = CollisionRay()(Moved_Pos_,{ 0,-1,0 },{ 0, distance / 2 , 0 },Wnode,
        IpStage.GetNorthStageCollList()).HitFlag;
auto SouthFlag = CollisionRay()(Moved_Pos_,{ 0,-1,0 },{ 0, distance / 2 , 0 },Wnode,
        IpStage.GetSouthStageCollList()).HitFlag;
if (NorthFlag || SouthFlag)     //レイ
{
    //上昇中ブロックを貫通しないように上方向にレイを飛ばす
    if (!CollisionRay()(Moved_Pos_, -NowMagVec_,{ speed_,speed_,0 } , node,
        IpStage.GetALLStageCollList(), true).HitFlag)//レイ
    {
        Playerbody_->applyCentralImpulse(
            -NowMagVec_.tobtVec() * WEAK_MOVE * 2); //上昇
    }
}
```

Stage

衝突形状の作成

Bullet3を使用した際に元々使っていた当たり判定では、Bullet3の判定に関与することが出来ず当たっていてもモデルは床や壁を貫通してしまいました。
なのでBullet3の方で当たり判定を作って当たり判定はBullet3で行うことにしました。

マップの当たり判定



モデルの参照用メッシュ構築と取得

```
// 地面の衝突形状の作成
btBvhTriangleMeshShape* ground_shape = new btBvhTriangleMeshShape(VartexGround, true, true);
btstageBody_ = CreateRigidBody(0.0, trans, ground_shape, 0);

g_collisionshapes.push_back(ground_shape); // 最後に破棄(delete)するために形状データを格納しておく
```

ポリゴン数だけ繰り返してトライアングルリストに格納

```
//地面のメッシュからトライアングルリスト作成
btTriangleMesh* VertexGround = new btTriangleMesh();
Vector3 pos[3];
// ポリゴンの数だけ繰り返し
for (int i = 0; i < Mesh.PolygonNum; i++)
{
    pos[0].x = Mesh.Vertexs[Mesh.Polygons[i].VIndex[0]].Position.x;
    pos[0].y = Mesh.Vertexs[Mesh.Polygons[i].VIndex[0]].Position.y;
    pos[2].y = Mesh.Vertexs[Mesh.Polygons[i].VIndex[2]].Position.y;
    pos[2].z = Mesh.Vertexs[Mesh.Polygons[i].VIndex[2]].Position.z;
    btVector3 pos0(pos[0].x, pos[0].y, pos[0].z);
    btVector3 pos1(pos[1].x, pos[1].y, pos[1].z);
    btVector3 pos2(pos[2].x, pos[2].y, pos[2].z);
    VertexGround->addTriangle(pos0, pos1, pos2, false);
}
```

衝突形状の作成をして格納

```
// 参照用メッシュの構築
MV1SetupReferenceMesh(collisionHandle_, -1, TRUE);
// 参照用メッシュの取得
MV1_REF_POLYGONLIST Mesh = MV1GetReferenceMesh(collisionHandle_, -1, TRUE);
```

HORRIFICHOUSE

冬の学内コンテストに向けてチーム制作した作品です。学内コンテストでは、１位を獲得しました。

徘徊している幽霊に捕まらないようにカメラをうまく使いながら屋敷から脱出するゲームです。カメラを向けている感じを出すために実際の携帯のカメラを使ってARでプレイすることが出来ます。

初めてシェーダーを実装した作品でこれまで作ってきた作品で一番見た目はよくなったと思います。他にも携帯とPCのソケット通信を行った作品です。

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++ / HLSL
制作人数	５人
制作期間	2022年9月～2023年2月 企画１ヵ月　制作3ヵ月
ジャンル	３D ARホラーゲーム

プロジェクトのURL　: https://drive.google.com/file/d/1KRxBJmDSpzY4K6YF2AzJ3-NcJqCOxrHo/view?usp=share_link

担当箇所

SocketServer

携帯とPCを通信するためにwinSockを使用して非同期UDP通信するクラス

UI

画面のUI

Shader

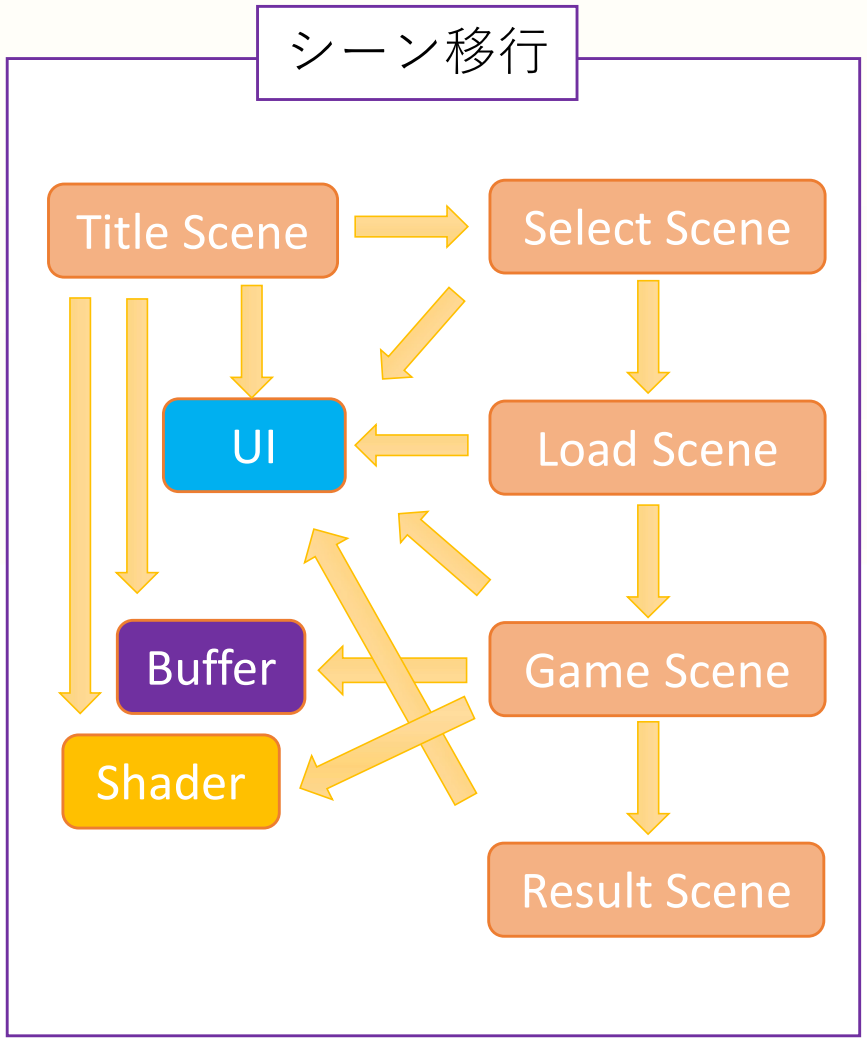
モデルの表示、光の描画、フォグの作成

Buffer

Shaderで使う定数バッファの管理

Common

CBufferMng、Light、OriginalShaderクラスの作成



SocketServer

通信とデータの受け取り

TCP/IP通信では速度が足りず、同期だと止まってしまったため、今回は非同期のUDP通信を行いました。
データの受け取りに関しては、数字の文字を使用して文字と文字の間に「,」を入れて順番で何の情報が来ているか確認して「,」までの数字を取り出してそれを変数に代入しています。

例えば、「0,120,0,1,0,0」のようにもらってきてたら、
x=0 y =120　のように「,」区切りで代入していきます。

文字列から区切りごとに代入する関数

```
void SocketServer::SplitStorage(int& value, std::str
{
    std::string numstr;
    std::getline(dataStream, numstr, ',');
    value = atoi(numstr.c_str());
    TRACE(" data %d ", value);
}
```

受け取り確認を行いデータが来なかったら
データが残らないように変数をリセット

```
//配列の０リセット
memset(buf, 0, sizeof(buf));
//データの受け取り
data = recv(sock, buf, sizeof(buf), 0);
//受け取ったデータの確認
if (data < 1)
{
    if (WSAGetLastError() == WSAEWOULDBLOCK)
    {
        //データが来てない
        //中身をリセット
        cameraValue_.first = 0;
        cameraValue_.second = 0;
```

データが来ていたら、変数に格納

```
        itemId_ = ItemID::Non;
    }
}
else
{
    //データが来た
    std::stringstream dataStream;
    dataStream << buf;
    //データの格納
    TRACE(" x : ");
    SplitStorage(cameraValue_.first, dataStream);
    TRACE(" y : ");
    SplitStorage(cameraValue_.second, dataStream);
```

Shader

スポットライト

DXライブラリにスポットライトの関数があり、それで実装していました。ですが、頂点からライトを計算しているようで思った通りのものが出来なかったためShaderでスポットライトを作成しました。

Dxlibのライト

自作したライト

距離による影響率を計算して小さくしている

```
// 距離による影響率を計算する
// スポットライトとの距離を計算する
float3 distance = length(input.pos - spotLights.position);
// 影響率は距離に比例して小さくなっていく
float affect = 1.0f - 1.0f / spotLights.range * distance;
```

角度を求めてその分の影響率を出す

```
// 入射光と射出方向の角度を求める
// dot()を利用して内積を求める
float angle = dot(ligDir, spotLights.Direction);
// dot()で求めた値をacos()に渡して角度を求める
angle = abs(acos(angle));
// 角度による影響率を求める
// 角度に比例して小さくなっていく影響率を計算する
affect = 1.0f - 1.0f / spotLights.angle * angle;
```

乗算を行い、影響を弱める

```
// 角度による影響率を反射光に乗算して、影響を弱める
diffSpotLight *= affect;
specSpotLight *= affect;
```

Mirroring Cave

冬の学内コンテストに向けた個人制作作品です。

自分と同じ動きをするロボットがいて、自分とロボットをうまく操作しながら松明や扉などのパズルを解いていくゲームです。

初めて一人で制作でこれまで授業で習ったことを活かした作品です。やりたいことが多く時間に追われながら制作していました。

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++
制作期間	2021年9月～2022年2月 企画１ヵ月　制作3ヵ月
ジャンル	パズルゲーム

プロジェクトのURL　: https://drive.google.com/file/d/1ZXRZ_u2HwVt7-teOyG7CNbflZoiH54kf/view?usp=share_link

補正と生成

位置補正

補正を何も書けないとマスとマスの間に入るのが難しく操作性がとても悪くなっていたので移動方向が変わった時にマスごとに位置補正をかけて操作がやりやすようにしました。

マスで分割してずれていたら補正する

```
//位置補正x方向
struct CorrectPosx
{
    bool operator()(Vector2Dbl& pos_, Vector2Dbl& size_)
    {
        int posx = static_cast <int>(pos_.x);
        int sizex = static_cast <int>(size_.x);
        if ((posx % sizex) == 0)
        {
            return false;
        }
        pos_.x = ((posx + sizex / 2) / sizex) * sizex;
        return true;
    }
```

Tmxからインスタンスの生成

tmxを読み込み、オブジェクト回数分回して、文字列が同じだったら、そのオブジェクトを生成するようにしてtmxを書き換えるだけで変わるようにしています。

tmxの読み込み

```
auto node = objctNode_->first_node("objList");
for (auto ListNode = node->first_node("List");
    ListNode != nullptr; ListNode = ListNode->next_sibling())
{
    std::string listname = ListNode->first_attribute("name")->value();
    auto objx = atof(ListNode->first_attribute("x")->value());
    auto objy = atof(ListNode->first_attribute("y")->value());
    std::string dieValue = ListNode->first_attribute("die")->value();
```

文字列が同じだったら生成

```
if (listname == "Player")
{
    objList_.emplace_back(
        std::make_unique<Player>(objx, objy, die, HintList_));
}
```

earrape

初めてゲーム制作を行った作品です。
チーム制作で行い、クッキークリッカーのようなバカゲーを目指してました。
ゲーム制作もチーム制作やったことなかったので、コミュニケーション不足やプロジェクトの作り方が悪く、マージを大量に引き起こしながら完成させた、とても反省点が多い作品です。

担当箇所

アイテムUI
画像の配置

音
効果音とBGM

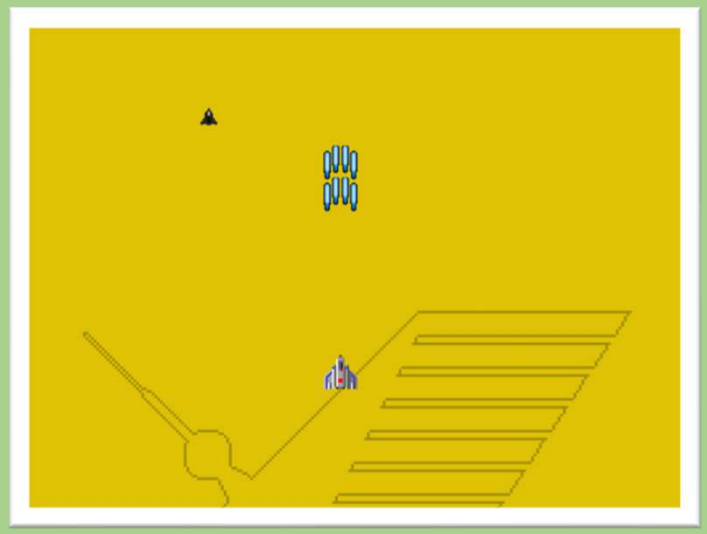
エフェクト
クリック時のエフェクト

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	5人
制作期間	2021年4月～2021年8月 企画１ヵ月　制作3ヵ月
ジャンル	放置ゲーム

プロジェクトのURL:https://drive.google.com/file/d/1UMr4Rnl5Ntv3JEuz0d1RNvZkQ4XD3UQV/view?usp=share_link

授業作品

一年次

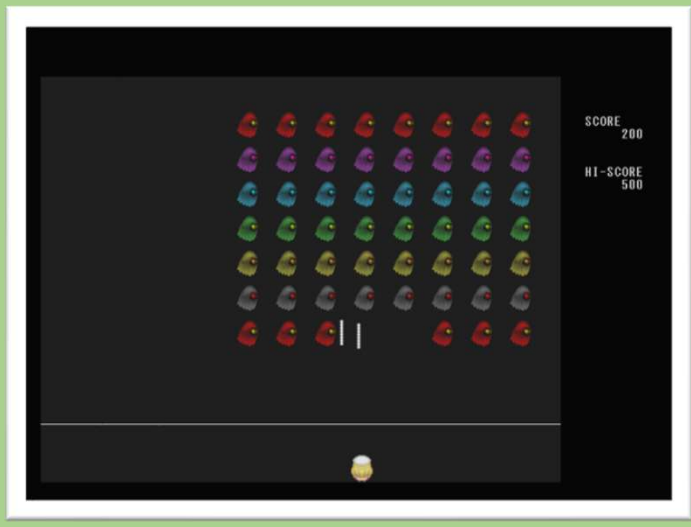


入学して初めてのゲーム制作です。
自機を動かす、弾を発射する、矩形と矩形の当たり判定など、基礎的な事を勉強しました。高校でC言語は少し学んでいましたが実際に作るゲームの難しさを体感しました。

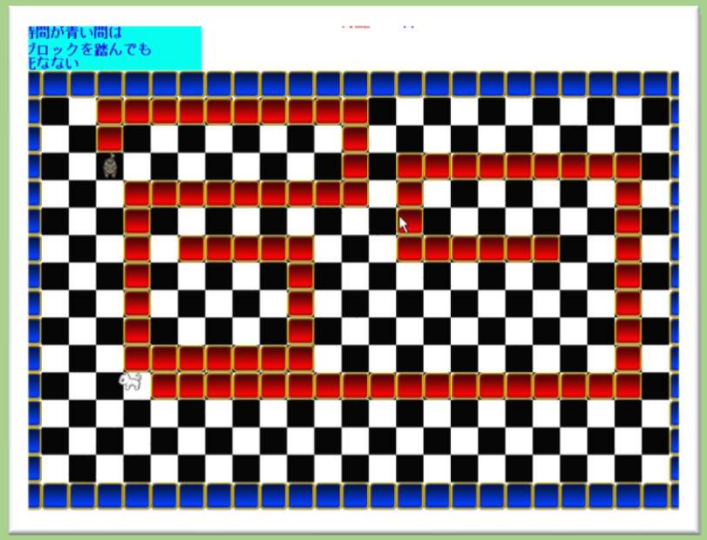
使用ライブラリ　DXライブラリ
使用言語　C言語
制作期間　一年生/前期

大量の敵を配置するために配列を学んで２次元配列を使って移動や当たり判定をやりました。他にも2Dのアニメーションやシーン遷移を勉強しました。初めて画面にフィルターをかけた作品でもあります。

使用ライブラリ　DXライブラリ
使用言語　C言語
制作期間　一年生/前期



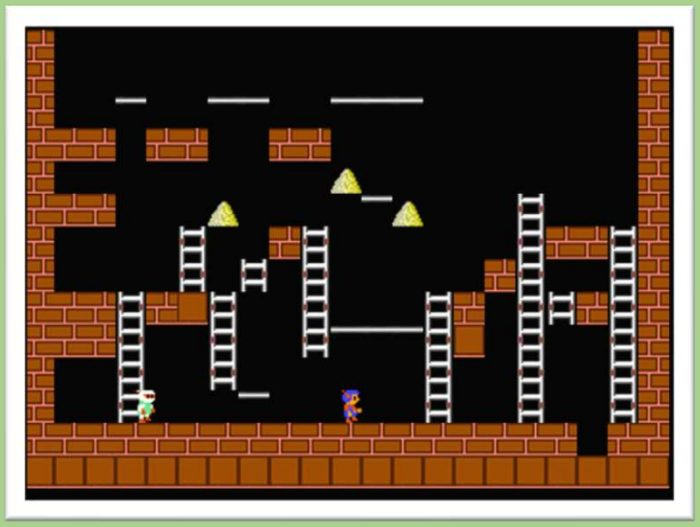
一年次



C++を使って初めての作品です。
２人対戦型ゲームを作成しました。プレイヤーを構造体や配列で制作して、複数のインスタンスが作成しやすくなりました。

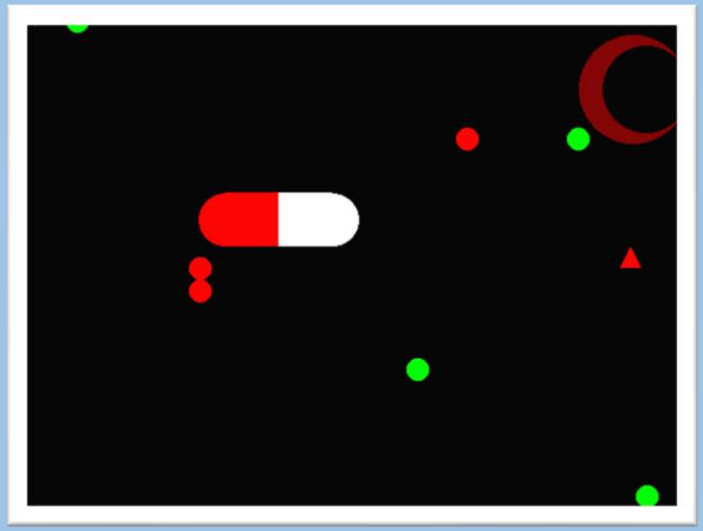
使用ライブラリ　DXライブラリ
使用言語　C++
制作期間　一年生/後期

エディタを作り、マップの作成やテキストにマップ保存を学び、ゲームの中で作って遊ぶことが出来るようになりました。他にも敵の思考ルーチーンなどを作成して、敵が追いかけてくるようになりました。



使用ライブラリ　DXライブラリ
使用言語　C++
制作期間　一年生/後期

二年次

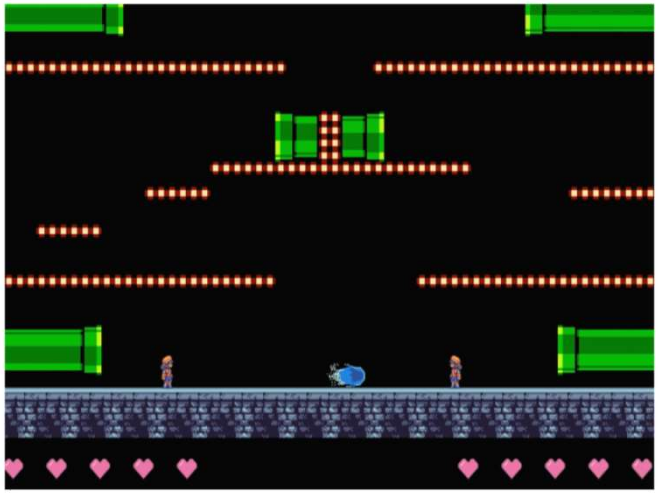


ユニークポインタやvectorなどの配列の使い配列の追加やイテレータを使った削除を学びそれを自分たちで使用してそれぞれに当たり判定つけて動くようにした作品です。

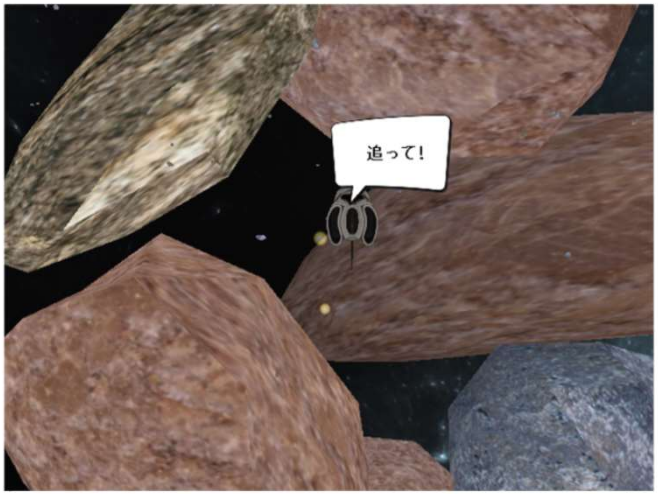
使用ライブラリ　DXライブラリ
使用言語　C++
制作期間　二年生/前期

初めてリングバッファを使いコマンド入力処理や再起処理をや使った行動制御をした作品です。処理の順番がコードを見ただけじゃわからなくなり、理解するのがとても大変でした。

使用ライブラリ　DXライブラリ
使用言語　C++
制作期間　二年生/前期



二年次

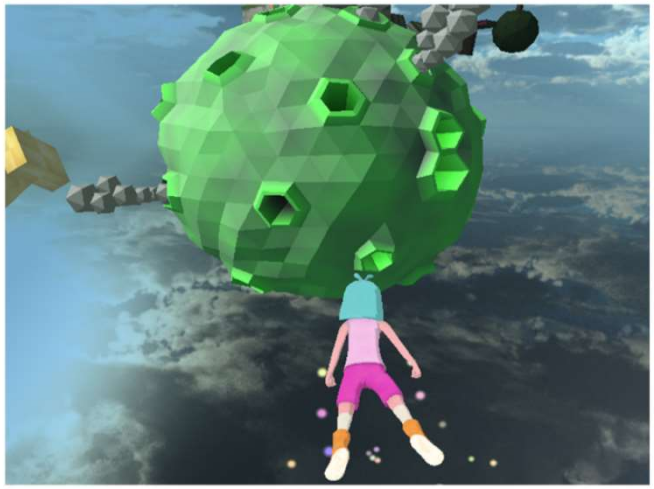


３Dシューティングゲームで導入からイベントシーンまでゲームとしての要素が入ったものを作りました。3Dカメラについて学び、ばね付き追尾カメラを作成しました。

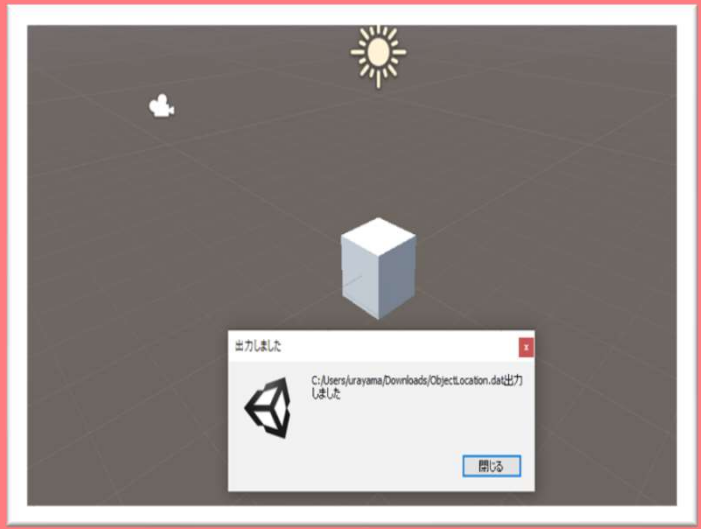
使用ライブラリ　DXライブラリ
使用言語　C++
制作期間　二年生/後期

マリオギャラクシーのようなゲームで状況に応じた重力制御やカメラに合わせたプレイヤーの制御などを学びました。座標だけで死亡などを判別できないので理解するのが時間がかかった作品です。

使用ライブラリ　DXライブラリ
使用言語　C++
制作期間　二年生/後期



三年次



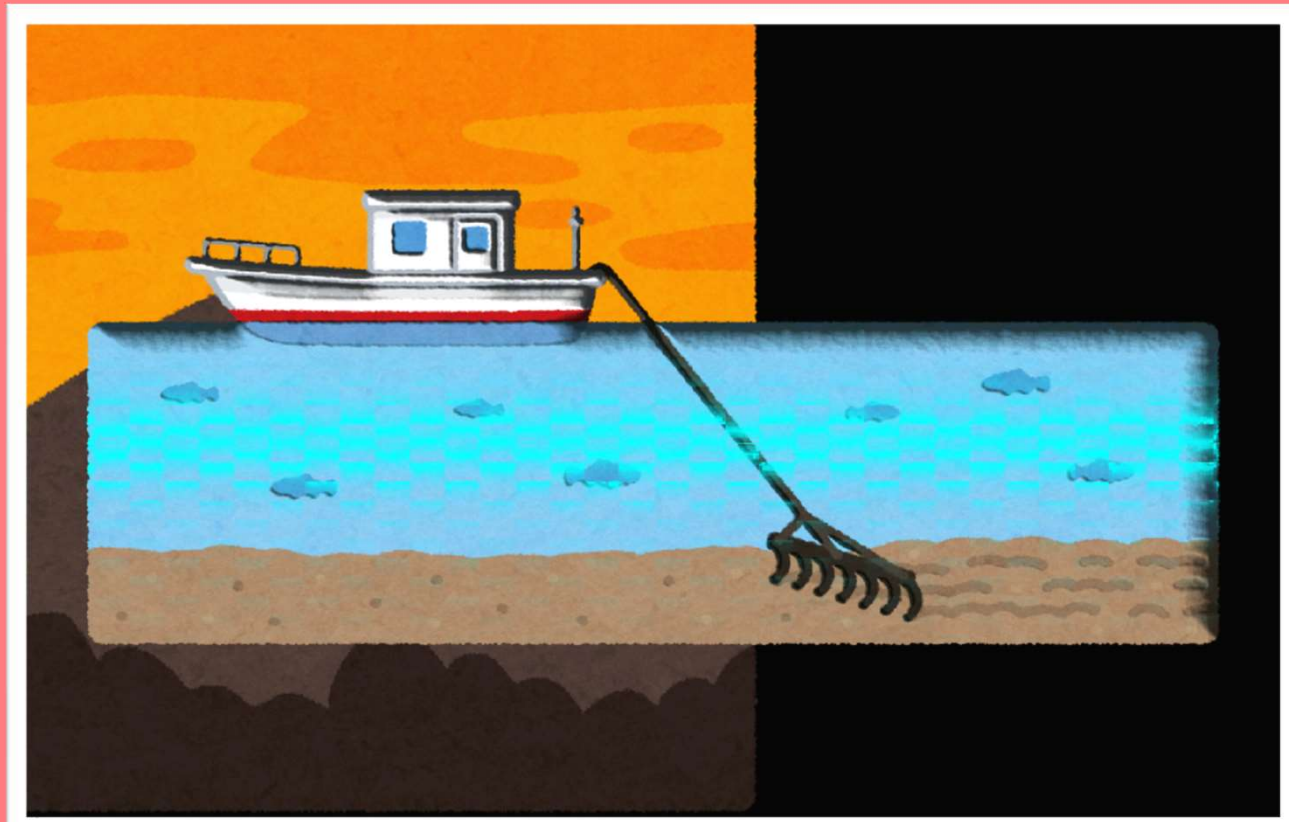
Unityでエディタの作成を行いました。Unityからモデルの座標や文字列をバイナリデータの出力を行いました。

使用エンジン　Unity
使用言語　C#script
制作期間　三年生/前期

アンリアルエンジン４で制作した作品です。
アンリアルエンジンの基本的な移動、生成、破棄などを行い、応用として、シェーダーやボーンの使い方などを学びました。

使用エンジン　Unreal Engine4
使用言語　ブループリント
制作期間　三年生/前期

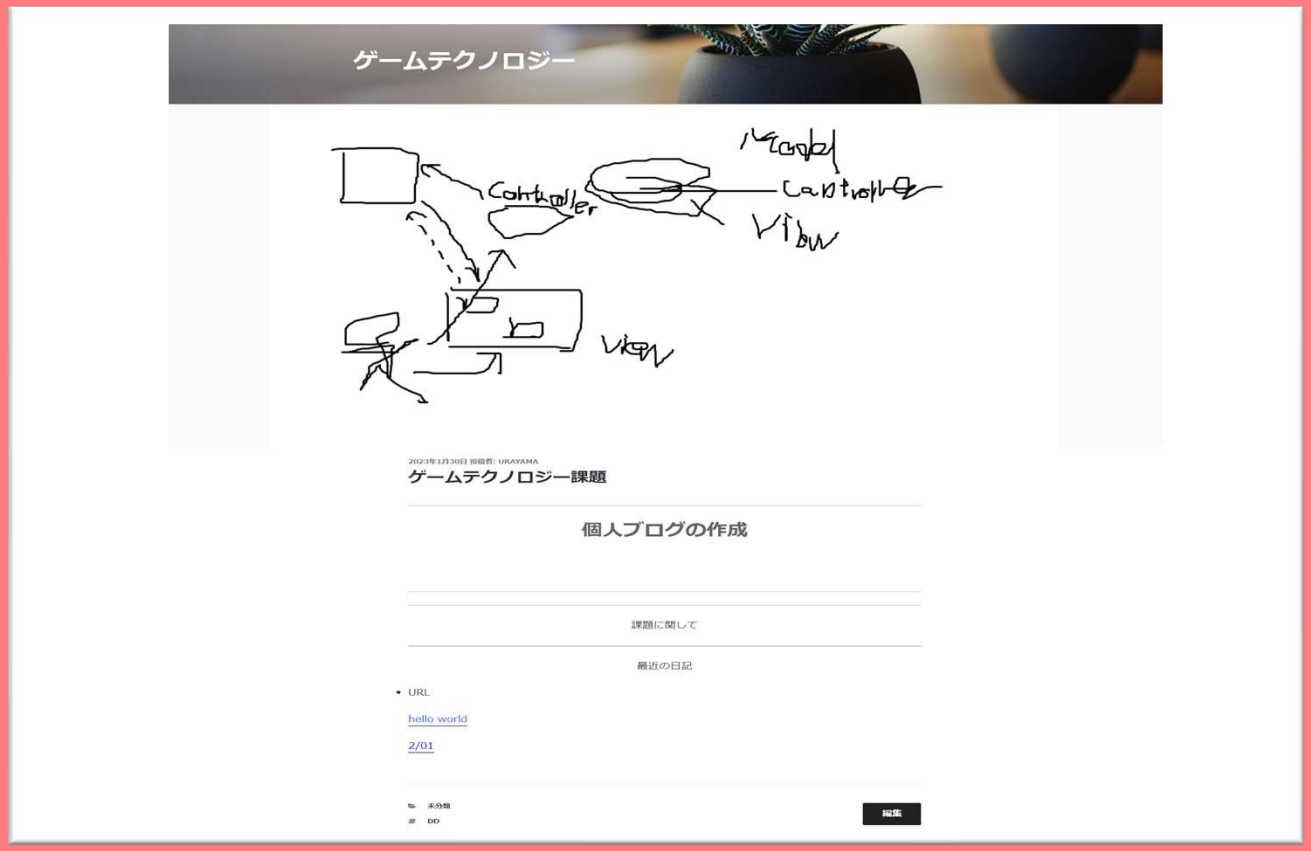
三年次



使用ライブラリ　DXライブラリ
使用言語　C++/HLSL
制作期間　三年生/後期

シェーダーの基本的な使い方を学びました。
DXライブラリを使用した際のテクスチャや定数の
渡し方を学び、色反転やノーマルマップを使用した
演出の仕方を学びました

三年次

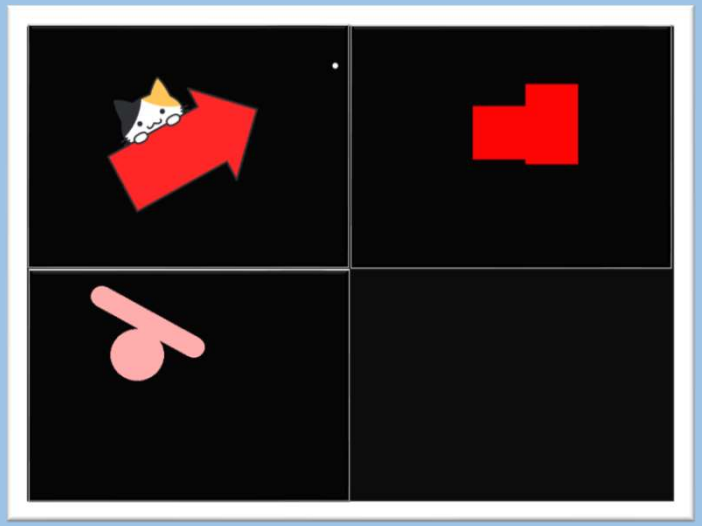


使用環境　Multipass上のLinux
制作期間　三年生/後期

サーバーの基本的な立て方を学び、wordPressを使用して作成しました。
Linuxの仮想マシンはMultipassで作成して、
wordPressはdbeaverを使用しました。

数学作品

二年次

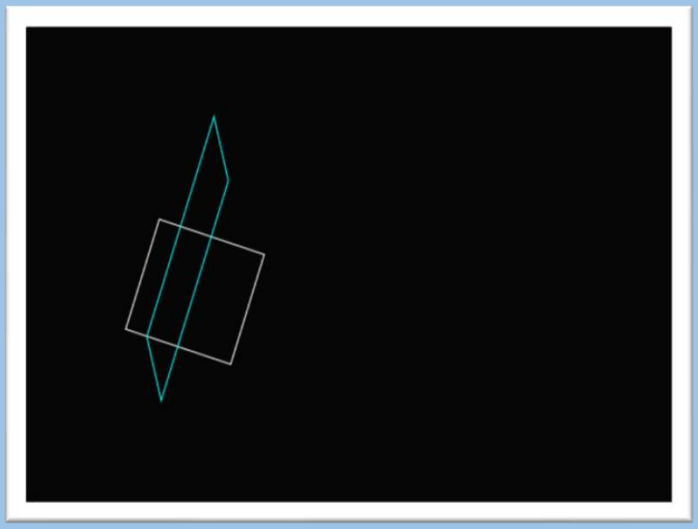


絶対値を使った四角同士の当たり判定、カプセルと円の当たり判定、ベクトルやatan2について、それぞれ学びました。これからの制作でもよく使う判定について基礎的な事を学べた。

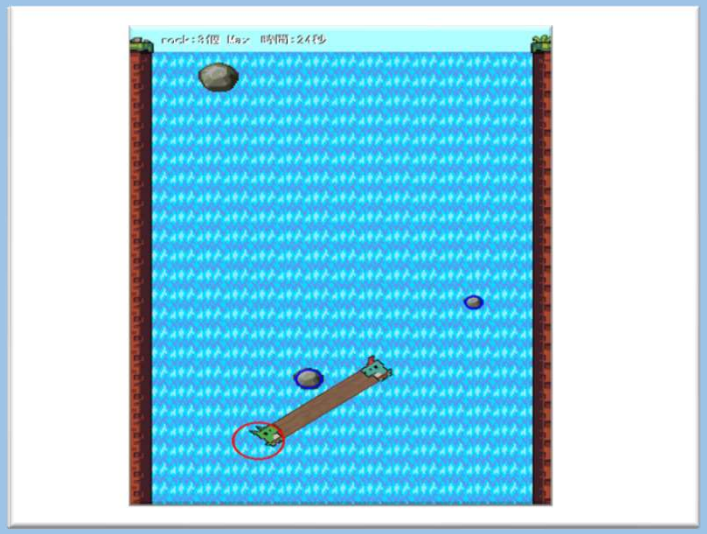
使用ライブラリ　DXライブラリ
使用言語　C＋＋
制作期間　二年生/前期

Matrixについて学んだ作品で、マウスの位置を中心に回転するようになっています。
単位行列、平行移動行列、回転行列を学びました。

使用ライブラリ　DXライブラリ
使用言語　C＋＋
制作期間　二年生/前期



二年次



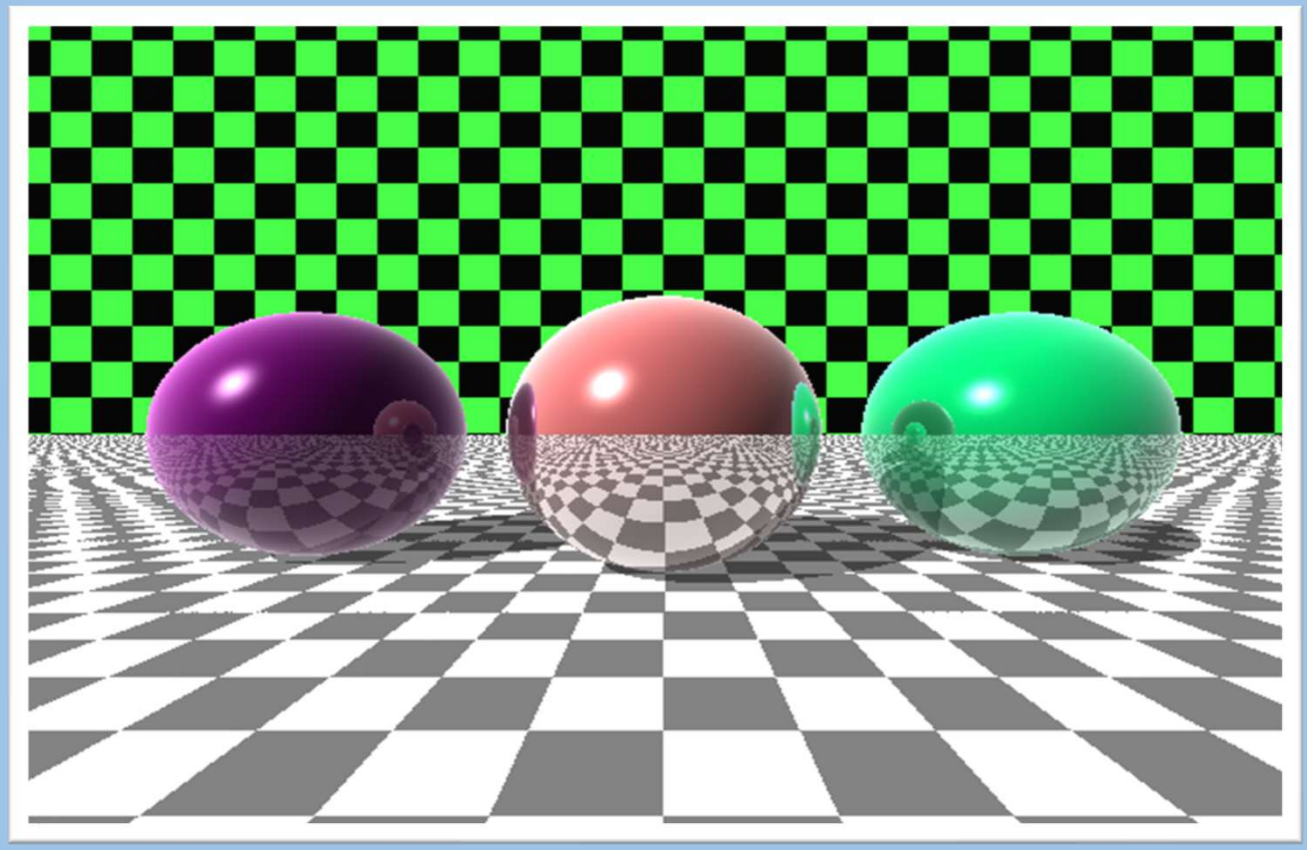
カプセルと円の当たり判定を利用して回転の軸を
右か左にして、90度を超えないように回転させて
一番上まで行くゲームです。
これまで数学で習ったことをゲームにした作品です。

使用ライブラリ　DXライブラリ
使用言語　C++
制作期間　二年生/前期

数学を使ったシュ―ディングを作成しました。
cos、sin、PIを使って弾の角度を計算して周囲
に弾を飛ばしたり、平方根を使った当たり判定を
学びました。

使用ライブラリ　DXライブラリ
使用言語　C++
制作期間　二年生/前期

二年次



古典的レイトレーシングについて学びました。
レイを飛ばして、球体に当たったら反射ベクトル作成を繰り返して、最終的な色を決定された色を塗っています。

使用ライブラリ　DXライブラリ
使用言語　C++
制作期間　二年生/後期

Thank you