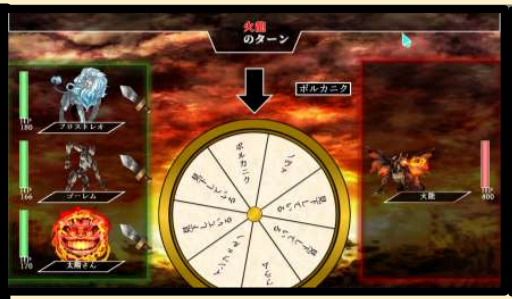
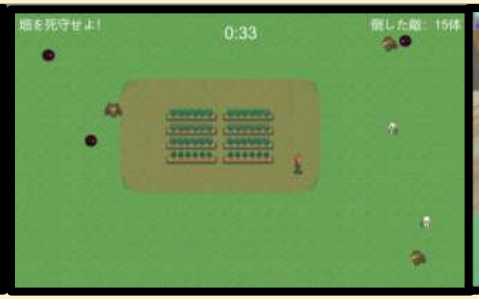






PORTFOLIO

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コース

クライアントエンジニア志望

野田　武道

自主制作作品

縫い包み（ぬいぐるみ）

プラットフォーム	PC
ジャンル	3D脱出ホラーゲーム
使用言語	C＋＋、HLSL
制作環境	DxLib
制作人数	1人
制作時期	2022/10～2023/1

化け物が徘徊する館からギミックを解きながら脱出する3D脱出ホラーゲームです。

シェーダーを使って、こだわったグラフィックを作ろうと思った時、ホラーゲームだとシェーダー演出が映えると思ったので、3Dのホラーゲームを作りました。

・3Dモデルを見栄え良く描画する

・ホラー感を強調できる演出を作る

というのを目標にHLSLシェーダーを使って、様々なグラフィックスの作成をシェーダーで挑戦しました。

動画URL:

https://youtu.be/D1QJutfuqnE

シェーダーで再現した表現一覧

今回のゲームに取り入れたシェーダー演出の一部です。使用頻度が多く、プレイヤーの目を引きやすいだろうと思い、ディゾルブシェーダーに気合を入れて作成しました。

▼ライティング

▼フォグ

▼ノイズ

▼グリッチ

▼ディゾルブ

▼UVスクロール

▼スフィアマップ

シェーダー動画:

https://youtu.be/FE67rZRyq7c

グリッチ、ディゾルブ

-シェーダーによる演出-

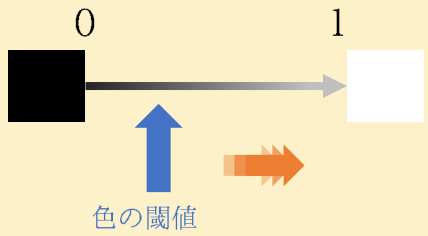
▸グリッチ

▸ディゾルブ

グリッチ（色収差）は、色相を分解してズラすことで表現しています。チューニングのあってないTV画面のような見た目で、ホラー感を増すことができました。

ディゾルブでは、徐々に消えるような演出をノイズ画像を使うことで表現しています。ノイズ画像の黒い部分から、徐々に色が変化する処理を実装しています。燃えるお札やぬいぐるみを描画するために使用しました。

▾ノイズ画像



◇黒い部分を０、白い部分を１として０の部分から、色を消していく。（お札の場合）

リアルな表現を作る

- 法線マッピング

- スフィアマップ

今回、3Dゲームを作るにあたっての課題が、モデルがのっぺりしている、グラフィックがチープ等の問題点がありました。

その課題を解決するために、シェーダーを使って『法線マッピング』と『スフィアマップ』という描画手法を実装しました。

これらを実装することにより、グラフィックをよりリアルに描画することが可能になりました。グラフィックがリアルになったことにより、ゲームのホラー感、恐怖感も増すことができました。

改善前と改善後の比較動画

URL：https://youtu.be/fZ4wzwYG5bw

リアルな表現を作る

-法線マッピングの適用-

▸壁の窪み

Before

After

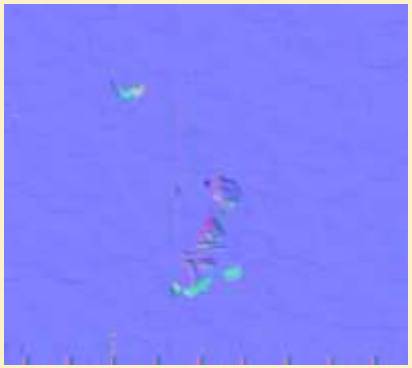
▸法線マップ処理（HLSL）

```
//法線マップから色を取得し、0～1を-1～1に変換
const float3 tanNormal = normalize(normalMapTexture.Sample(normalMapSampler, input.uv).xyz * 2 - 1);

//カメラー自身のベクトル
const float3 ray = normalize(input.viewPos);

//接ベクトル空間→ビュー空間に法線を変換
const float3 normal = ConvertCoordinateSpace(tanNormal, input.viewTan, input.viewBin, input.viewNorm);
```

```
/// <summary>
/// 拡散反射の強さを計算
/// </summary>
/// <param name="lightRay">ライトのレイ</param>
/// <param name="normal">法線</param>
/// <returns></returns>
float CalculateDiffuse(in float3 lightRay, in float3 normal)
{
    return saturate(dot(normal, -lightRay));
}
```

法線マッピングは、法線マップという画像をモデルに反映させることで、モデルの凹凸、陰影を再現することが出来る手法です。この手法により追加のポリゴンを使わずに詳細な見た目を実現することができました。

◇壁の窪みの法線マップ画像



・より詳細なオブジェクトの法線ベクトルのX,Y,Z座標に対応したRGB画像

法線マップを使うことで、壁のくぼみに影が描画されました。これによってモデルに凹凸ができ、ゲームの立体感が感じられるようになり、見た目が大幅に改善されました。

リアルな表現を作る

-金属質な表現-

▸流し台のモデル

Before

After

▸スフィアマップ処理（HLSL）

```
//視線の反射ベクトル
float3 refRay = reflect(ray, normal);
refRay = mul(refRay, vmat);
refRay.xy = refRay.xy * float2(0.4, -0.4) + 0.5;

float b = saturate(dot(normal, -light));

float4 spCol = sphereMapTexture.Sample(diffuseMapSampler, refRay.xy);
float rough = roughnessMapTexture.Sample(diffuseMapSampler, input.uv).r;
float metalic = metallicTexture.Sample(diffuseMapSampler, input.uv);

float3 rgb = saturate(result.rgb);
//カラー
float3 color = lerp(totalSpecular, rgb * b, rough);
//反射カラー
float3 refCol = lerp(lerp(1, spCol.rgb, metalic) * rgb, 0, rough);
result.rgb = color + refCol;
```

グラフィックが微妙という問題で、金属のオブジェクトの質感がイマイチという問題がありました。そこで今回はスフィアマップを用いて、金属質な表現を行いました。

▸スフィアマップで使用した画像

周囲の景色を反射しているようなテクスチャを貼り付けることで、金属特有のツルツル感が作ることができ、金属の質感を表現することができました。これにより、リアリティが薄いという課題を解決しました。

ルーレットリガー

プラットフォーム	PC
ジャンル	コマンドカードバトル
使用言語	C＋＋、HLSL
制作環境	DXライブラリ
制作人数	1人
制作時期	2023/5～2023/7

学内で行われるコンテストに応募するために作った作品です。ルーレットを止めてコマンドを決定し、ユニット同士で戦うルーレット式コマンドカードバトルとなっています。

このゲームでは、グラフィックスプログラミングを学ぶために、初めてHLSLシェーダーに触って制作しました。ユニットの状態演出やポストエフェクトなど、見た目や演出部分に力を入れて取り組みました。

シェーダーでいくつもの描画を行っているため、描画関数をクラス化し、手軽にシェーダーでの描画を行えるクラス設計を実装しました。

動画URL:

https://youtu.be/_ZMxLUDDxoY

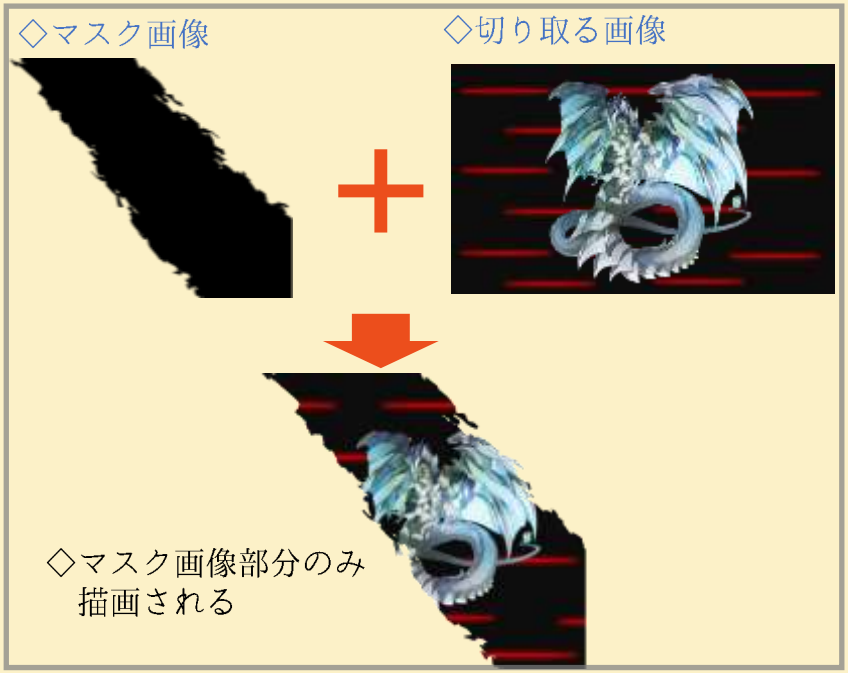
マスク処理

▸ユニット死亡シーン

▸マスク処理（HLSL）

```
float4 main(PS_INPUT PSInput) : SV_TARGET
{
    //UV座標を受け取る
    float2 uv = PSInput.TexCoords0;

    //マスク画像のアルファ値が０以下は描画しない
    float4 maskCol = g_MaskTex.Sample(g_SrcSampler, uv);
    if (maskCol.a <= 0)
    {
        discard;
    }

    //UV座標とテクスチャを参照して、最適な色を取得する
    float4 srcCol =
        g_Tex.Sample(g_SrcSampler, uv);

    return float4(srcCol.rgb,1.0f);
}
```

ユニットが死亡したとき、枠組みの範囲内で死亡ユニットの拡大描画したかったので、マスク処理を自作して使用しました。



黒塗りのマスク画像と切り取る画像を用意し、テクスチャ情報をピクセルシェーダーに転送します。マスク画像部分のみ、ピクセルカラーを返すことでマスク処理を実現しています。

トランジション

▸シーン遷移中のゲーム画面

▸トランジションのコード

```
float4 main(PS_INPUT PSInput) : SV_TARGET
{
    //UV座標を受け取る
    float2 uv = PSInput.TexCoords0;

    //UV座標とテクスチャを参照して、最適な色を取得する
    float4 srcCol =
        g_SrcTexture.Sample(g_SrcSampler, uv);
    float4 fadeCol =
        g_FadeTexture.Sample(g_SrcSampler, uv);

    //アルファ値
    float alpha = 1.0f - (fadeCol.r + fadeCol.g + fadeCol.b) / 3.0f;
    alpha -= 1.0;                   //[0.0f]～[-1.0f]の状態にする
    alpha += (g_pra * 2.0f);        //[0.0f]～[2.0f]の時間割合を加算する

    return float4(srcCol.rgb, srcCol.a * alpha);
}
```

シーン遷移時のカットインの演出をピクセルシェーダーとルール画像を用いて、わかりやすいトランジションを作成しました。

ルール画像▸

ルール画像とゲーム画面をシェーダー側に転送し、ルール画像の明度を利用し、ゲーム画面のアルファ値を操作することで、ルール画像の黒色側から徐々に描画されていきます。

白黒のルール画像であれば、画像を差し替えるだけで他のトランジションに変更できるように柔軟性を意識して作りました。

未定

プラットフォーム ： PC

ジャンル ： 3D脱出ゲーム

使用言語 ： C#

制作環境 ： Unity

制作人数 ： 1人

制作時期 ： 2022/6～2022/8

ほぼまっさらな空間にあるギミックを解いていくことで、新たなオブジェクトと様々なギミックが出現します。それを繰り返し、最終的に部屋から脱出することがこのゲームの目的となっています。

Before

After

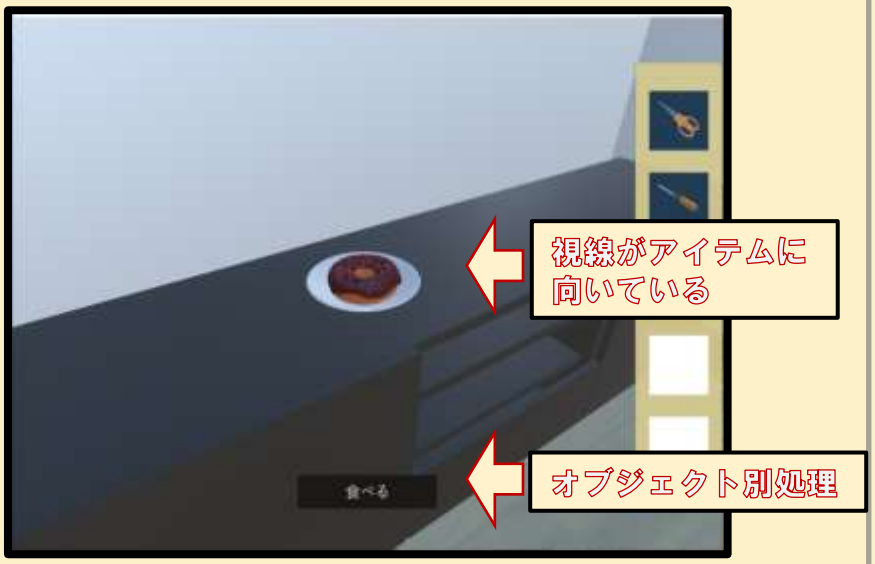
このゲーム制作では、プレイヤーの目線になって考えることを意識して作りました。飽きることないように多彩なギミックを作ったり、画面揺れ等を考えて実装しました。

動画URL:

https://youtu.be/TGeffFnkDZ0

レイの処理

▸レイとオブジェクトの衝突判定



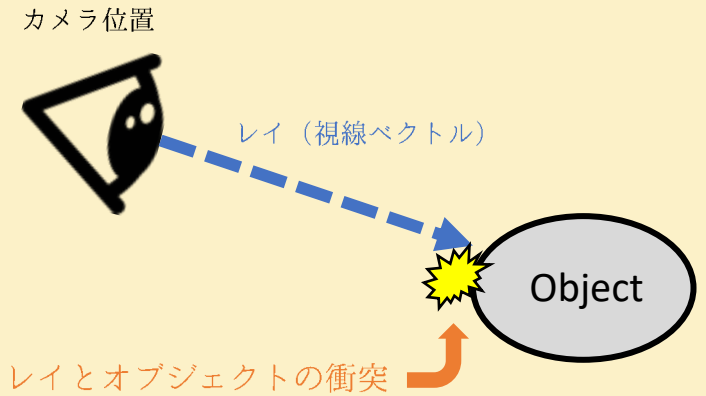
▸レイの発射、衝突判定処理

```
blic void SetTarget()

    //レイの発射地点を画面中央に設定
    Ray ray = Camera.main.ScreenPointToRay(new Vector3(x_MonitorSenter, y_MonitorSenter, 0));
    //レイの情報を取得
    RaycastHit hitInfo;

    //rayの開始地点、（rayの向き）、当たったオブジェクトの情報を格納、rayの発射距離、（レイヤマスクの設定
    //rayとコライダがぶつかった場合、実行する
    if (Physics.Raycast(ray, out hitInfo, chachRange))
    {
        //レイが当たっている間、パネルを表示する
        panel.SetActive(true);
        targetObj = hitInfo.collider.gameObject;

        //アイテムの場合
        if (targetObj.CompareTag("Item")
            )
        {
```

アイテムなどのオブジェクトごとの処理はレイを使っての判断を行っています。

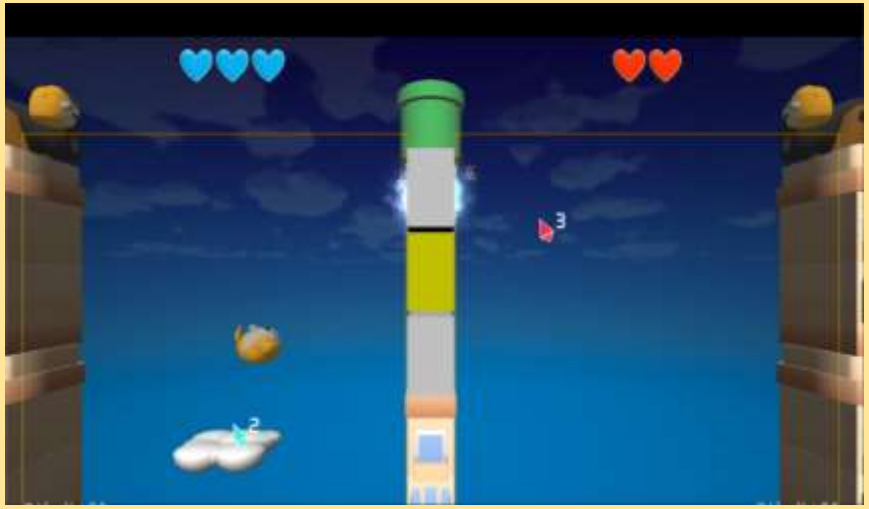
視線座標からレイを飛ばし、レイが当たったオブジェクトの種類別にそれぞれの処理を行っています。

◇レイの衝突判定



このレイを使用して、アイテムの取得、アイテムの使用、ドアの開閉、オブジェクトを調べるなどのプレイヤーの行動を行っています。

アニマリープ
~AnimaLoop~



プラットフォーム	PC
ジャンル	3D対戦ゲーム
使用言語	C++
制作環境	DXライブラリ
制作人数	3人
制作時期	2024/2～2024/3

春の学内コンテスト用に制作した対戦型の3Dバラエティゲームです。初めてチーム制作を行った作品であったため、何よりもチームがわかりやすいプログラムを書くことに注意しました。

このゲームは端から出てくる動物を雲を作ってゴールに跳ね返すのが目的です。ゴールに入れると相手にお邪魔球を送り、動物を落下させるとライフが減少します。

私の担当箇所は

・プレイヤーの動作、操作関連

・跳ねる動物や雲などのオブジェクト

・物理挙動

などのゲームのメインとなる部分を担当しました。

自主製作

制作時期 ： ２年後期

開発環境 ： DxLib

制作人数 ： １人

制作期間 ： ２か月

剣と魔法でモンスターと戦うアクションゲームです。魔法の発動エリアをポリゴンの上に描画をして、見やすさを工夫して製作しました。

制作時期 ： １年後期

開発環境 ： Unity

制作人数 ： １人

制作期間 ： １週間

初めてUnityで制作したゲームです。パンが弾を発射するシューティングゲームとなっています。弾を発射する間隔、方向を自分で考えて制作しました。

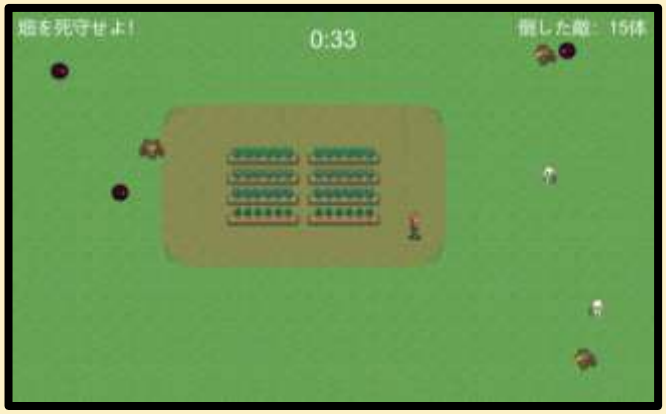
自主製作

制作時期 ： ２年前期

開発環境 ： Unity

制作人数 ： １人

制作期間 ： １週間



画面中央にある畑を周囲から襲ってくるモンスターたちから守るゲームです。 2D画像のキャラクター達をアニメーション制御を作りました。

制作時期 ： ２年前期

開発環境 ： Unity

制作人数 ： １人

制作期間 ： １週間

一人用のエアホッケーのゲームです。障害物を避けながらゴールに入れるのが目的です。パックの動きをベクトルで制御して、不自然なく動くようにこだわりました。

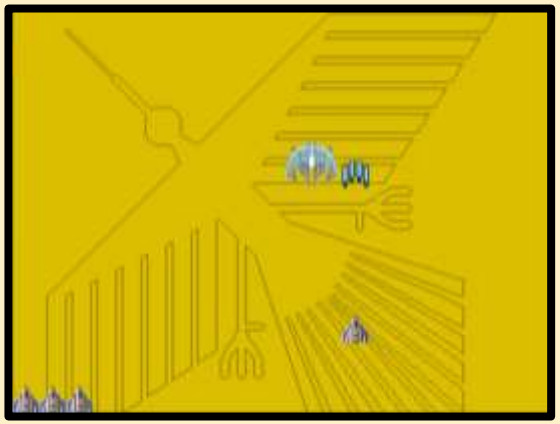
授業作品

1年次

制作時期 ： 1年前期

使用言語 ： C言語

開発環境 ： DxLib

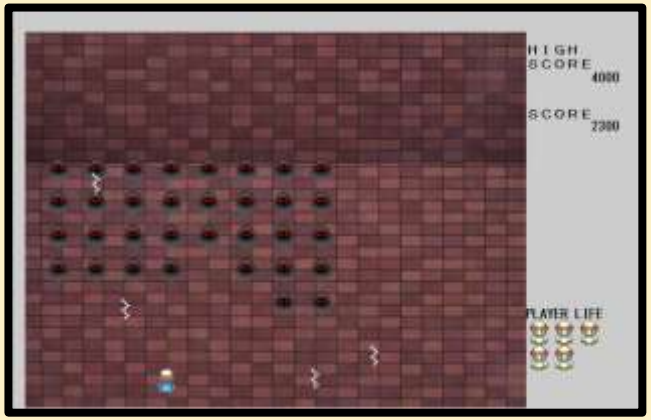


入学して初めて授業で制作した作品です。この作品では、画像の描画方法、当たり判定などゲーム制作の基本的な事について学びました。

制作時期 ： 1年前期

使用言語 ： C言語

開発環境 ： DxLib



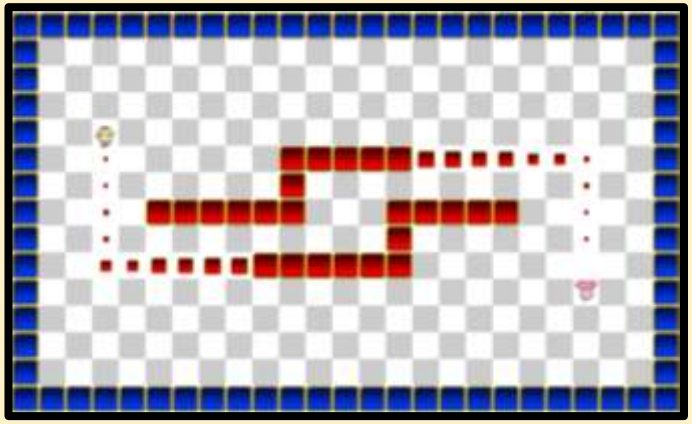
2次元配列による敵の制御を行ったシューティングゲームです。シーンの遷移やスコアの加算などを行っており、配列を使用した処理の考え方を学べました。

1年次

制作時期 ： 1年後期

使用言語 ： C言語

開発環境 ： DxLib

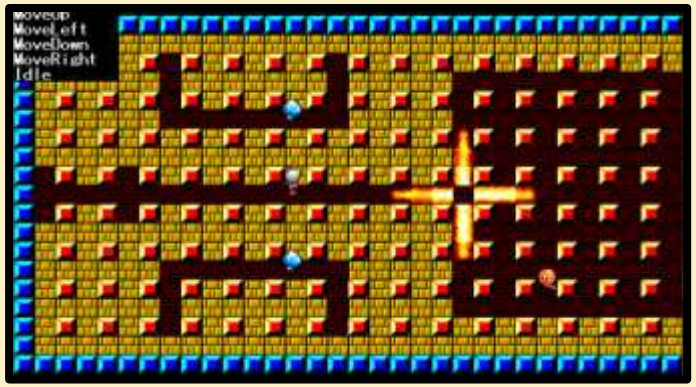


二人で対戦するスネークゲームを制作しました。このゲームではチップごとのマップ制御や列挙型でのシーン管理などを行い、シーン別の処理方法などを学びました。

制作時期 ： 1年後期

使用言語 ： C言語

開発環境 ： DxLib



マップデータをcsvファイルから読み込み、マップを作成しています。爆弾の爆風や当たり判定などを実装し、自分なりに考えた敵の思考ルーチンを作成しました。

2年次

制作時期 ： ２年前期

使用言語 ： C＋＋

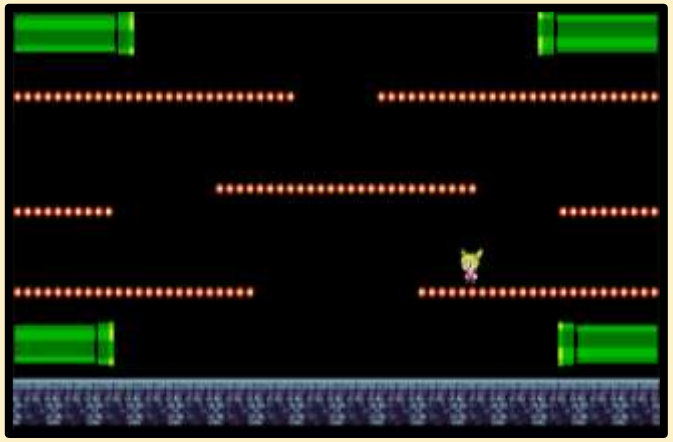
開発環境 ： DxLib

3D空間の制御方法を理解するために制作した3Dシューティングゲームです。3D空間における移動やクォータニオンでの回転を行い、3D制御の基礎を学びました。

制作時期 ： ２年前期

使用言語 ： C＋＋

開発環境 ： DxLib



レイで制御を行うキャラクターを作成しました。レイを使用しての当たり判定を行っています。またリングバッファで管理したコマンドでの行動処理を実装しました。

2年次

制作時期 ： ２年後期

使用言語 ： C＋＋

開発環境 ： DxLib

重力操作やカメラ操作を学ぶために制作した3Dゲームです。回転するカメラの座標、角度を制御するために、行列や球面補完を使用しました。

制作時期 ： 3年前期

使用言語 ： C＋＋

開発環境 ： DxLib

3Dモデルのアニメーションとシェーダーを勉強するために作成しました。右上にある敵キャラのHPをシェーダーで描画して、光り方や減少の仕方をこだわりました。

数学

制作時期：　２年後期

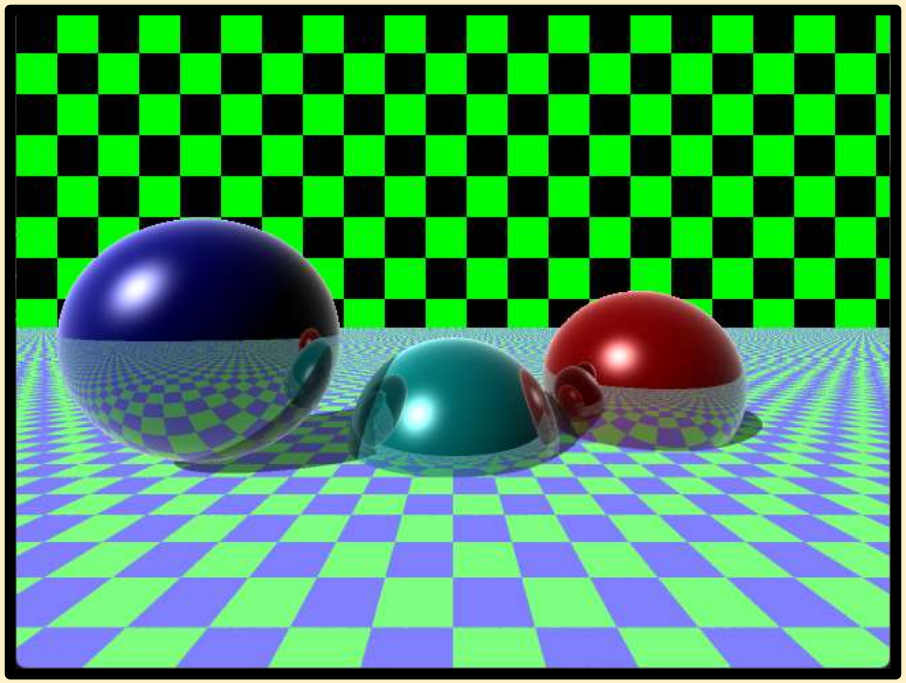
使用言語：　C＋＋

開発環境：　DxLib

数学の授業で行った古典的レイトレーシングを用いた、球、床、影の描画です。

光をベクトルとして計算することで、球の色、リムライト、ハイライト、球の反射、床の模様、影の描画をしています。

球の反射のみでなく、ジャギーを消すためのアンチエイリアシングや、環境光を考えた描画実装などを行っています。



ありがとうございました