ポートフォリオ

総合学園ヒューマンアカデミー　京都校
ゲームカレッジ　プログラマー専攻
福永　駿斗

使用言語　: C/C++,　 C#

使用ツール: DXライブラリ、Unity

開発環境　: Visual Studio 2019

目次

プロフィール

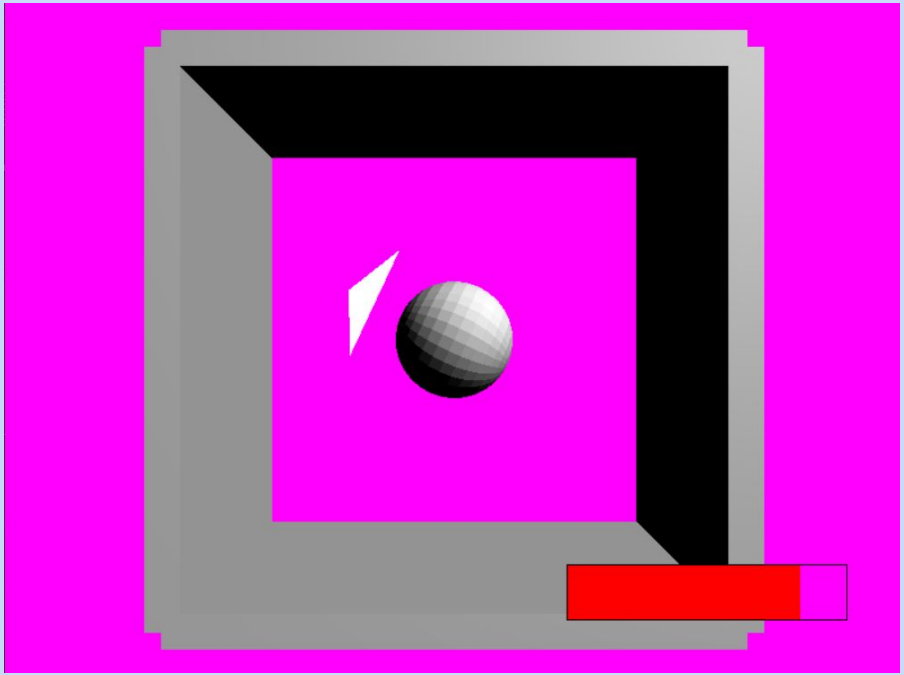
名前：福永駿斗

生年月日：2002年12月05日

好きなこと：ゲーム、動画鑑賞

使用言語　：C/C++,　C#
使用ツール：DXライブラリ、Unity
開発環境　：Visual Studio 2019

2年前期個人制作



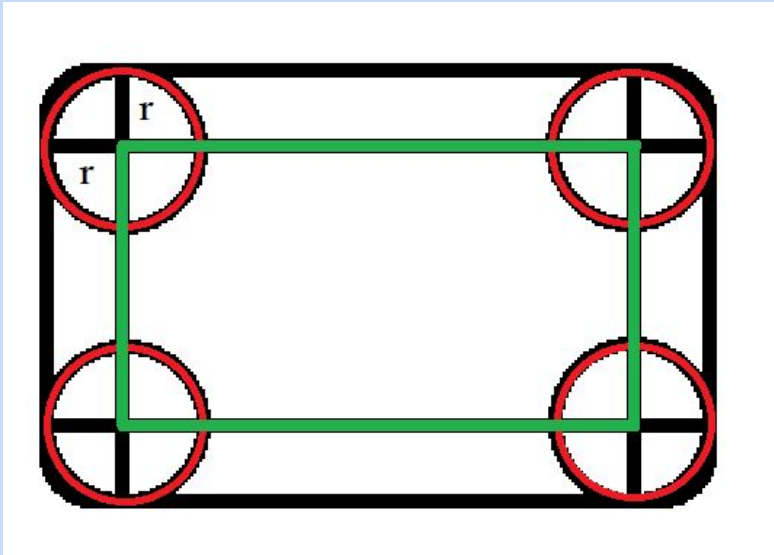
環境
使用ツール：Visual studio2019
DXライブラリ
使用言語：C/C++

制作期間：2ヶ月

ヒューマンアカデミーのイベントで
他校の作品を見る機会があり、自分も
作ってみたいと思い、1年の後期で3Dをしたので
復習のため制作した個人制作物

球と直方体の当たり判定

```
//矩形の横幅追加領域判定
if (T < x && B > x && L - radius < y && R + radius > y && N < z && F > z)
{
    return true;
}

//矩形の縦幅追加領域判定
if (T + radius < x && B - radius > x &&L < y && R > y && N < z && F > z)
{
    return true;
}

//矩形の厚み追加領域判定
if (T < x && B > x && L < y && R > y && N - radius < z && F + radius > z)
{
    return true;
}

//左上手前の当たり判定
if ((L - y) * (L - y) + (T - x) * (T - x) + (N - z) * (N - z) < radius * radius)
{
    return true;
}

//右上手前の当たり判定
if ((R - y) * (R - y) + (T - x) * (T - x) + (N - z) * (N - z) < radius * radius)
{
    return true;
}
```



円と矩形の当たり判定を元に横、縦、厚みを球の半径分判定部分を増加し、各面に球が触れると判定が発生する。
各頂点に判定を行う球と同じサイズの球を配置し頂点付近の当たり判定を実装

反射処理

```
CenterPos.y += spherecollider.radius;
TopPos.y -= (boxcollider.scale.y / 2);
NearPos.y -= (boxcollider.scale.y / 2);

Normal = CalcLineNormal(CenterPos, TopPos, NearPos);
```

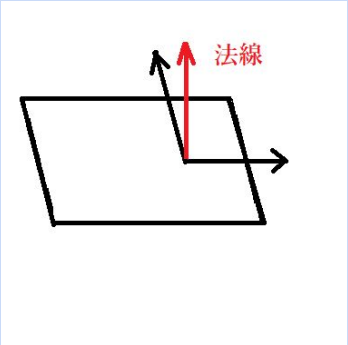
```
VECTOR Collision::CalcLineNormal(VECTOR Vec1, VECTOR Vec2, VECTOR Vec3)
{
    VECTOR Vec01, Vec02;

    Vec01 = VSub(Vec2, Vec1);
    Vec02 = VSub(Vec3, Vec1);

    return VCross(Vec01, Vec02);
}
```

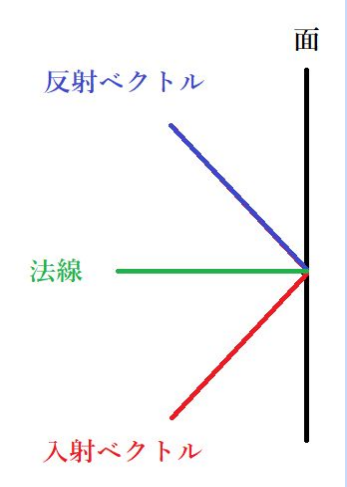
```
VECTOR Collision::CalcReflectVector(VECTOR direction, VECTOR normal)
{
    VECTOR L = direction;
    float LdotNx2 = 2.0 * VDot(VScale(L, -1.0), normal);

    return VAdd(VScale(normal, LdotNx2), L);
}
```

上の式により

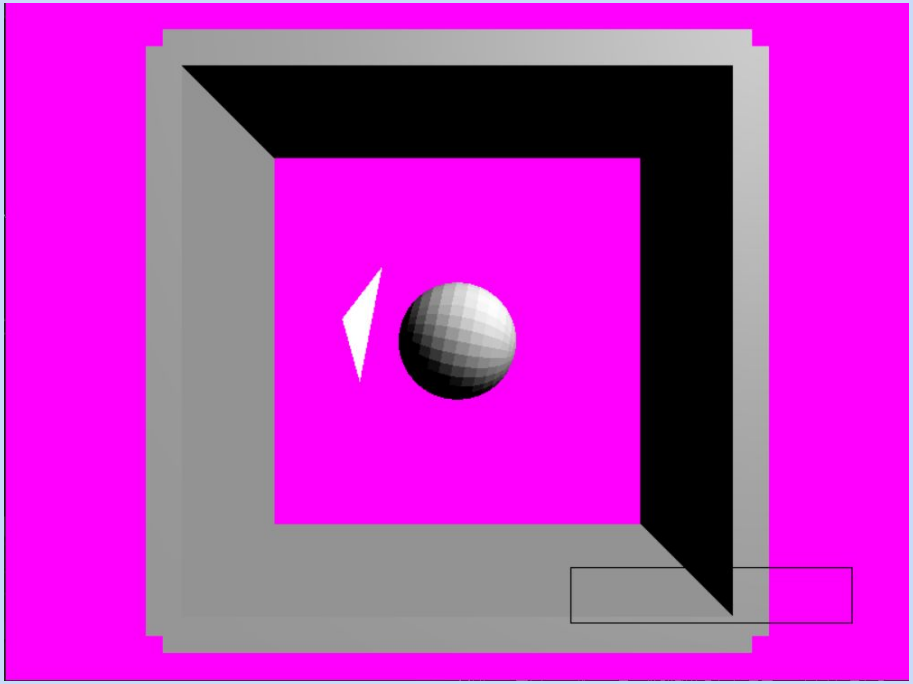
各面に平行で面と球が接触した座標を始点としたベクトルを二つ用意し、そのベクトルを外積したものを法線ベクトルとして扱い

法線ベクトルと入射ベクトルを計算
（2.0 ＊（-L・normal）＊ normal）+ L
反射ベクトルを入手





3Dに挑戦



1から個人で3D作品を作るのは初めてだったのでDxlibの関数を使い様々な表現をするように意識しました。
三角形を描画する関数を利用し矢印を生成し球の向きをわかりやすくしました。
行列を使用して回転の処理を実装しました。

```
float rad = DegToRad(degree);
VECTOR pos;
MATRIX result;

result = MGetRotX(rad);

pos = VAdd(centerpos, VTransform(arrowpos, result));
return pos;
```

1年後期チーム制作

環境
使用ツール：Visual studio2019
DXライブラリ
使用言語：C/C++

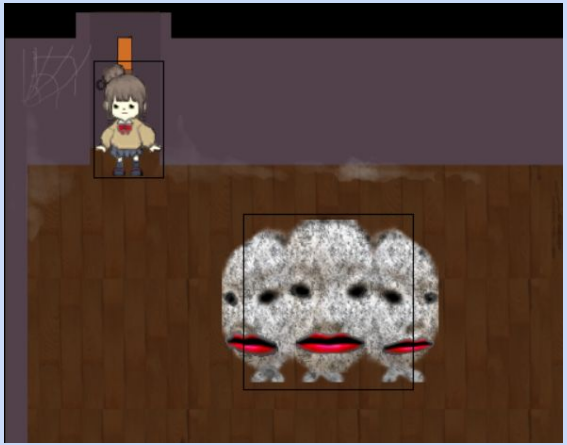
制作期間：3ヶ月

担当箇所：エネミー制御、画面の遷移
当たり判定、UI

制作人数：プログラマー　2人
デザイナー　3人

矩形同士の当たり判定

```
Collision                                   BoxandBox(const BoxCollider collider01, co
bool Collision::BoxandBox(const BoxCollider collider01, const BoxCollider collider02)
{
    VECTOR left_top_01, right_bottom_01, left_top_02, right_bottom_02;

    left_top_01.x = collider01.Posx;
    left_top_01.y = collider01.Posy;
    right_bottom_01.x = collider01.Posx + collider01.Width;
    right_bottom_01.y = collider01.Posy + collider01.Height;

    left_top_02.x = collider02.Posx;
    left_top_02.y = collider02.Posy;
    right_bottom_02.x = collider02.Posx + collider02.Width;
    right_bottom_02.y = collider02.Posy + collider02.Height;

    if (left_top_01.x < right_bottom_02.x &&
        right_bottom_01.x > left_top_02.x &&
        left_top_01.y < right_bottom_02.y &&
        right_bottom_01.y > left_top_02.y)
    {
        return true;
    }
    return false;
}
```

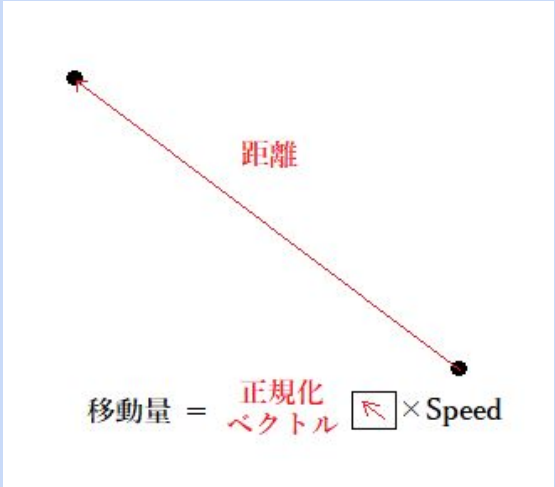


画像サイズのままだと大きすぎると判断したためモンスターの当たり判定を小さめに設定し、他のプログラマーの人にわかりやすくするために枠を描画し、視認性を向上しました。

1年後期チーム制作

敵がプレイヤーを追跡する処理

```
void Enemy::Chase()
{
    distance.x = GameManager::Instance()->GetPlayerPosX() - State.Pos_x;
    distance.y = GameManager::Instance()->GetPlayerPosY() - State.Pos_y;

    distance = VNorm(distance);

    State.Pos_x += distance.x * State.Speed;
    State.Pos_y += distance.y * State.Speed;

    boxcollider.Posx = State.Pos_x + 40;
    boxcollider.Posy = State.Pos_y;
}
```



相互参照を防ぐためにGameManager経由でプレイヤーの座標を取得し、プレイヤーと敵の距離を単位ベクトル化し、最後に当たり判定の更新をする

1年前期個人制作

環境
使用ツール：Visual Studio2019
教師から配布されたエンジン

使用言語：C/C++

制作期間：1ヶ月

ゲームの概要
ロボットが敵を倒してスコアを貯める
アイテムを取ることでロボットを強化していく

1年前期個人制作

```
void RunGameScene()
{
    UpdateBg();
    UpdateRobot();
    UpdateRobotHp();
    UpdateEnemies();
    UpdateBullets();
    UpdateBg();
    UpdateItems();
    RunCollision();

    CreateEnemy();

    if (GetRobotHp() <= 0)
    {
        g_CurrentStep = SceneStep::SceneStepFinish;
    }

}
```

```
if (Engine::IsKeyboardKeyPushed(DIK_A) == true)
{
    CreateBullet(g_Robot.PosX + g_Robot.SpwanOffsetX,
                 g_Robot.PosY + g_Robot.SpwanOffsetY);
    if (g_Robot.BulletPower == true)
    {
        CreateBullet(g_Robot.PosX + g_Robot.SpwanOffsetX,
            g_Robot.PosY + g_Robot.SpwanOffsetY + 25);

        CreateBullet(g_Robot.PosX + g_Robot.SpwanOffsetX,
            g_Robot.PosY + g_Robot.SpwanOffsetY - 25);
    }
}
```

必要な処理はなるべく関数で作り、可読性を向上させることを意識した。
関数内でほかの関数を呼び出す処理を書き呼び出す関数の数を減らした

1年前期個人制作

環境
使用ツール：Unity

使用言語：C#

制作期間：1ヶ月

ゲームの概要
ロボットが敵を倒してスコアを貯める
ボスはダメージを追うごとに難しくなっていく

1年前期個人制作

```
if (Step == 0)
{
    speed = 2.0f * Time.deltaTime;
    transform.Translate(-speed, 0.0f, 0.0f);
}
else if(Step == 1)
{
    if (transform.position.y >= 4.0f)
    {
        speed = -1.5f * Time.deltaTime;
    }
    else if (transform.position.y <= -4.0f)
    {
        speed = 1.5f * Time.deltaTime;
    }

    gameObject.transform.Translate(0.0f, speed, 0.0f);
}
else if(Step == finish_step)
{
    Timer += Time.deltaTime;

    if (Timer > 2.0f)
    {
        Vector3 new_pos = transform.position;
        new_pos.z -= 1.0f;
        Instantiate(ExplositionPrefab, new_pos, Quaternion.identity);
        Timer = 0.0f;
    }
}
```

ボスの行動に力を入れ、常にEnemyを出し続け、Hpが減ると小型機を出し、小型機からもEnemyを出すため徐々に難しくなるようになっている

お目通しありがとうございました。