技術追究型プログラマー

名前　　：田代　昂汰
出身　　：福岡情報ITクリエイター専門学校
GitHub　：https://github.com/itasi29
趣味　　：ゲーム・ルービックキューブ・釣り
スキル
- C++　　：2年半
- C#　　：2年
- HLSL　　：1年
- Unity　　：2年
- Git Hub　：2年半

自己PR

paizaで**Aランク！**

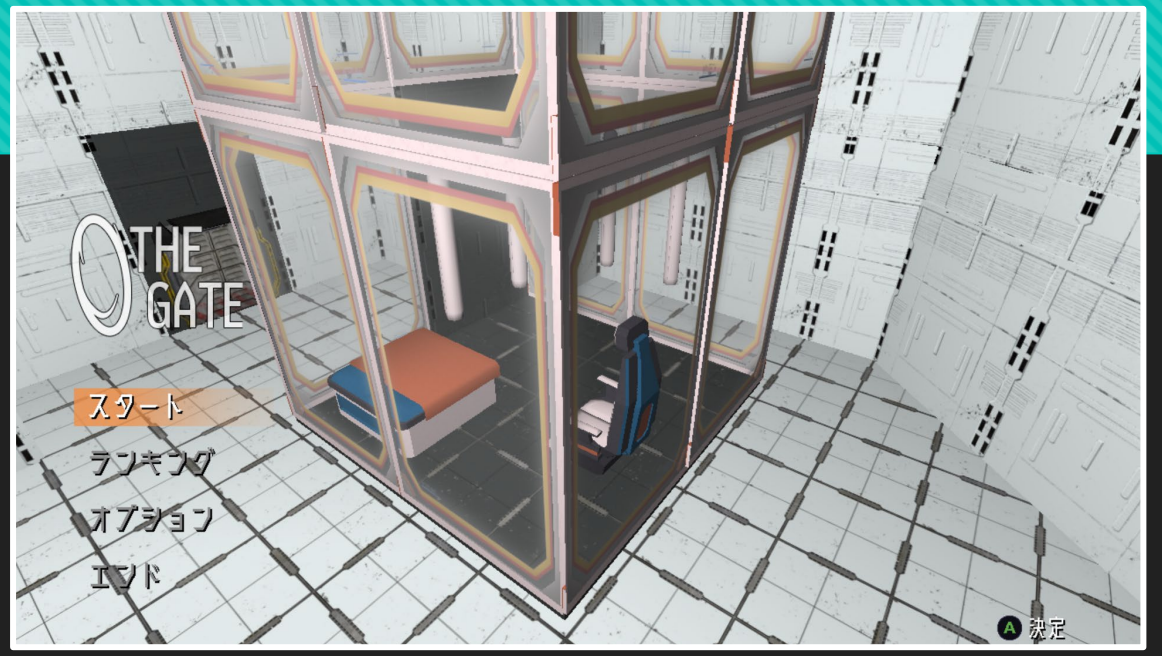
イベントで出展！

休日は釣りにも！

就活作品

最終制作作品

作品名　：THE　GATE



ジャンル　：謎解き3Dアクションゲーム

開発環境　：DxLib, C++, HLSL

対応機種　：PC

制作期間　：約4カ月

制作時期　：2年次10月～2年次２月頭

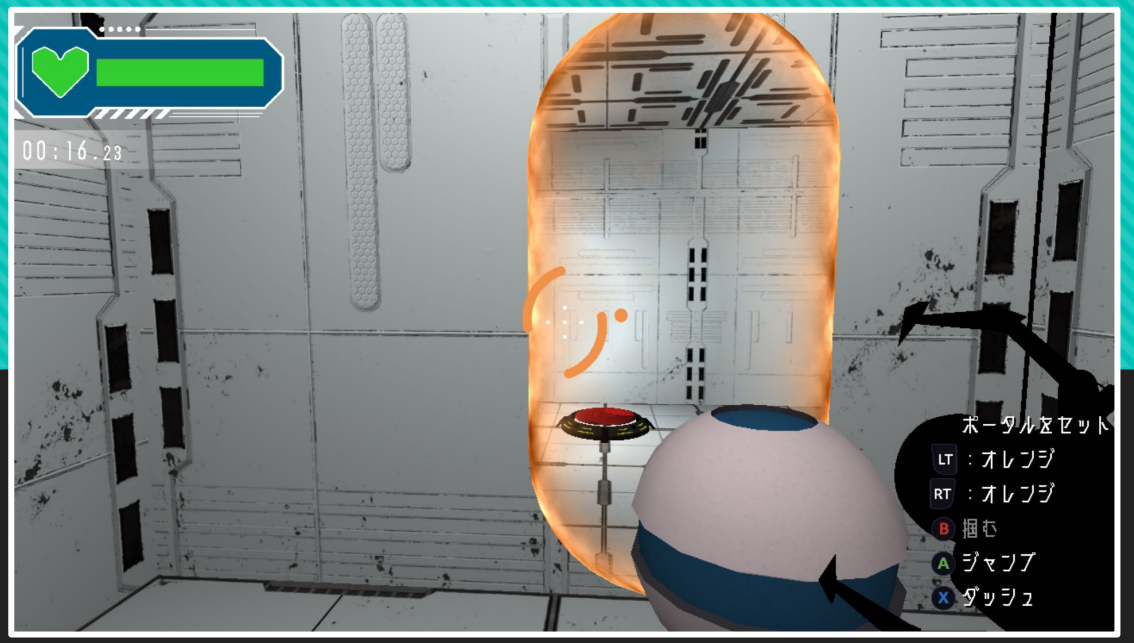
担当　　：モデル・UI・一部ステージ制作以外全て

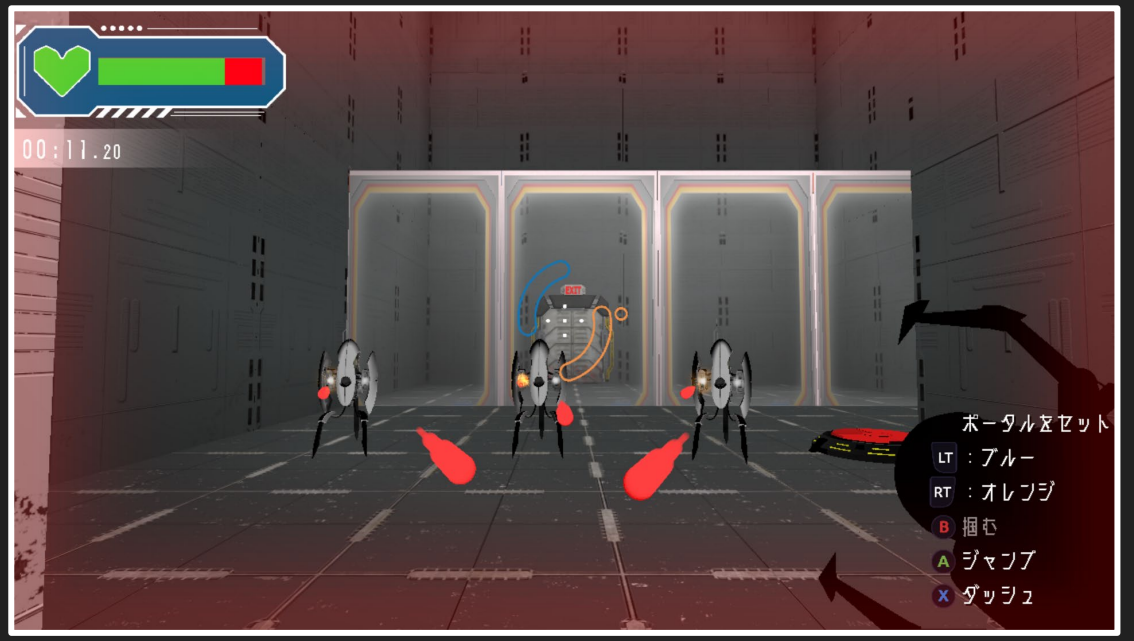
Git　　：https://github.com/itasi29/THE_GATE.git

動画URL　：https://youtu.be/fZ-xfAzHb9Y?si=qkCO6bSA03Rvn8dl

THE GATEとは…

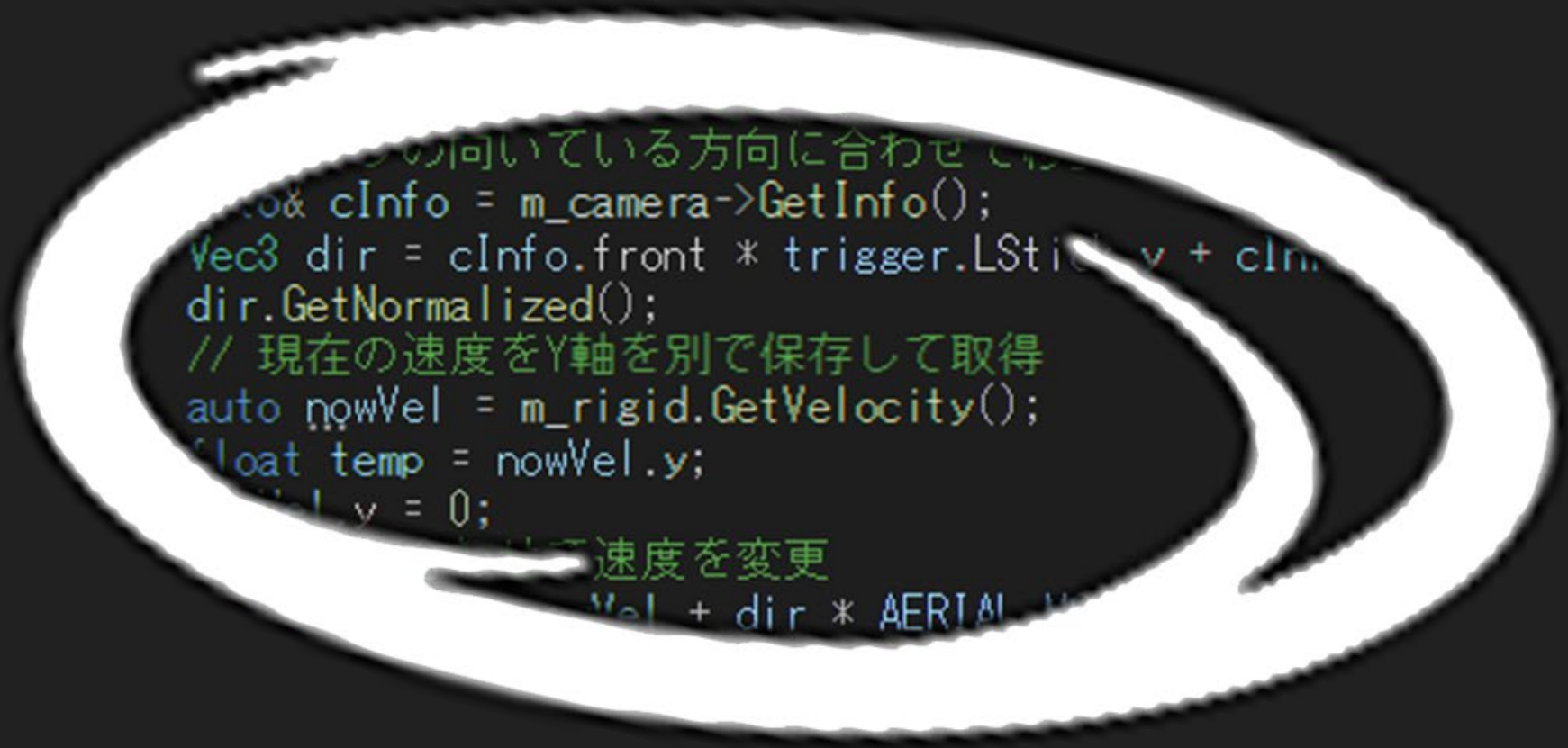
ゲートを駆使して
ギミックを攻略せよ！





※実質ポーt…

技術紹介


の向いている方向に合わせ
&cInfo = m_camera->GetInfo();
Vec3 dir = cInfo.front * trigger.LSti
dir.GetNormalized();
// 現在の速度をY軸を別で保存して取得
auto nowVel = m_rigid.GetVelocity();
float temp = nowVel.y;
y = 0;
速度を変更
+ dir * AERIAL

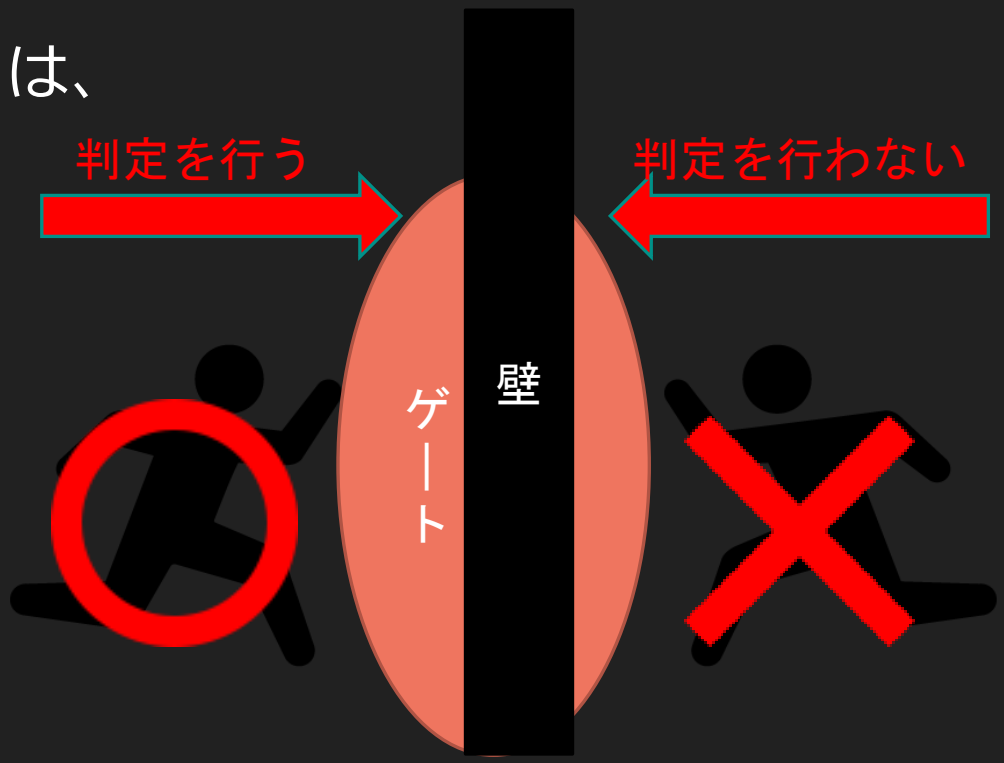
技術紹介①：ゲート

Code：Object/Gate/Gate.cpp

・ゲートのワープ判定にはプレイヤーまでのベクトルと
ゲートの法線の内積を用いて判定を実装
・ゲート判定侵入時から負の場合(反対側)には、
判定を行わないようにし処理不可の軽減

```
bool Gate::CheckWarp(const Vec3& targetPos)
{
    // ゲートから対象に向くベクトルとゲートの法線の内積が-になったらワープ可能
    const auto& gateTotarget = targetPos - (m_rigid.GetPos() + m_collider->center);
    auto dot = Vec3::Dot(m_norm, gateTotarget);
    return dot < 0.0f;
}
```

▲判定用コード

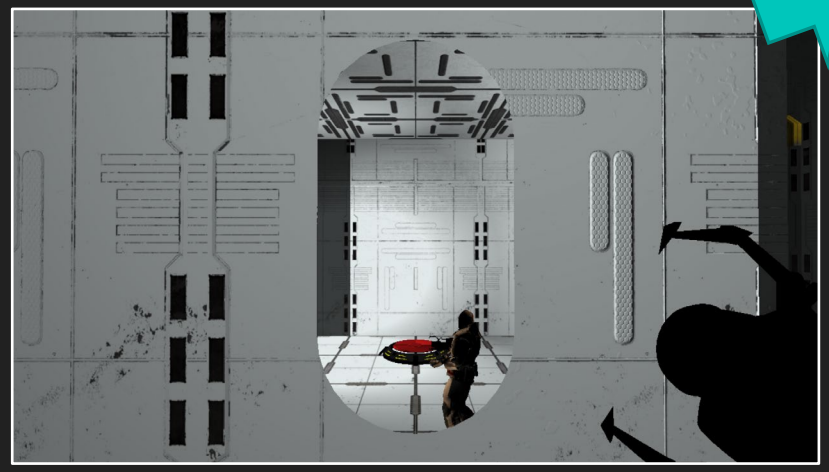


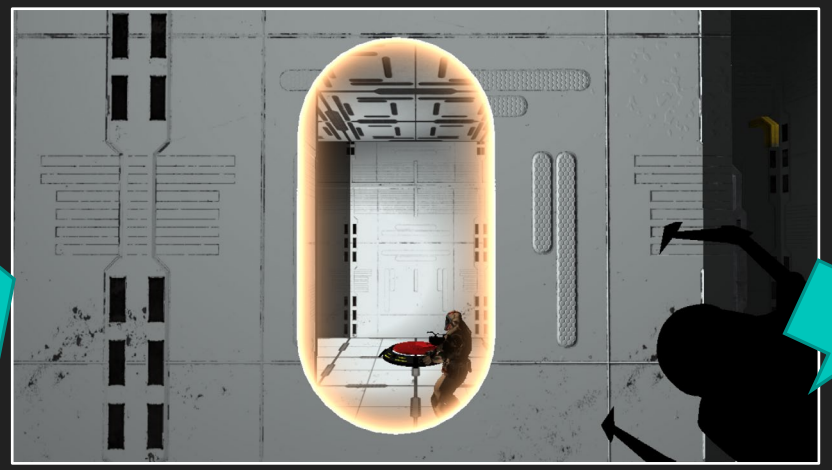
技術紹介②：ゲート

Code：Shader/GatePS.hlsl

・HLSLを用いてゲートの周りの演出を強化

▼何もせずに描画





▲縁を追加

▼Dissolveによる縁のもやもや追加

技術紹介③：プレイヤー

Code：Object/Player.cpp, h

・プレイヤーはステートパターンを用いて　管理することにより
　状態の管理、追加をしやすい構造に

```
//ボタンを押したら攻撃アニメーションを再生する
if (!m_isAttack && !m_isSkill && !m_isDown && !m_isCommand)
{
    //攻撃
    if (Pad::IsPress(PAD_INPUT_3) && !m_isStamina)
    {
        m_isAttack = true;
        ChangeAnim(kAttackAnimIndex);
        m_isMove = true;
        PlaySoundMem(m_axeAttackSeH, DX_PLAYTYPE_BACK, true);//アタック
    }
    else
    {
        if (m_isMove)
        {
            Move();
        }
    }
```

▲ステートパターン無し

ステートパターン
活用

```
// ジャンプに遷移
if (input.IsTriggerd(Command::BTN_JUMP))
{
    OnJump();
    isChange = true;
}

// 走りに遷移
if (input.IsPress(Command::BTN_RUN))
{
    OnRun();
    isChange = true;
}

// 状態を遷移していない場合
if (!isChange)
{
    // 左スティックが入力されている間は歩き継続
    if (Move(WALK_SPEED, false)) return;

    // 入力されていなかったらアイドル状態に遷移
    OnIdle();
}
```

▲ステートパターン有り

技術紹介④：自作ライブラリ

Code：Geometryファイル内

・ゲーム内で計算を楽に行えるよう
ベクトル・行列・クオータニオンのクラスを作成

Vec3
+ x : float + y : float + z : float
+ operator+(val : Vec3) : Vec3 + operator-(val : Vec3) : Vec3 + operator*(val : float) : Vec3 + Length() : float + Normalize() : Vec3 + Dot(val1 : Vec3, val2 : Vec3) : float + Cross(val1 : Vec3, val2 : Vec3) : Vec3 …etc

Matrix4x4
+ m : float[4][4]
+ operator+ (mat : Matrix) : Matrix + operator* (mat : Matrix) : Matrix + operator* (vec : Vec3) : Matrix + Identity() : Matrix + Pos(pos : Vec3) : Matrix + Scale(scale : Vec3) : Matrix + Rot(rot : Quat) : Matrix …etc ※Matrix4x4はMatrixに略してます

Quaternion
+ x : float + y : float + z : float + w : float
+ operator* (q : Quat) : Quat + operator* (vec : Vec3) : Vec3 + Conjugated() : Quat + AngleAxis(angle : float, axis : Vec3) : Quat + GetQuat(v1 : Vec3, v2 : Vec3) : Quat …etc ※QuaternionはQuatに略してます

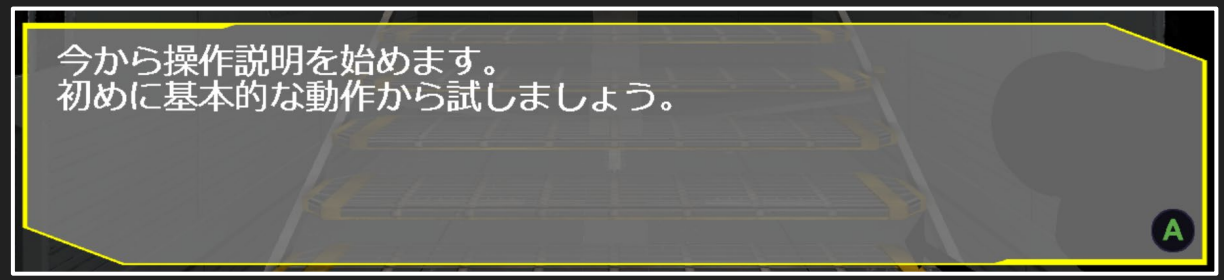
技術紹介⑤：外部ファイル化

Code：StageDataManager.cpp

・メッセージテキストなど変更が多いものをExcelファイルで記入し、
CSVファイル化したものをゲーム内で読み込むことで変更に強く

ID	連続フラグ	連続ID	画像描画フラク	使用画像ID	文字描画間隔	文字列
N_1_1	1	N_1_2	0		5	今から操作説明を始めま
N_1_2	0		1	I_EXPO_CAMERA_0	5	初めに視点を動かしてみ
N_2_1	1	N_2_2	0		5	OK！視点操作は完璧です
N_2_2	1	N_2_3	1	I_EXPO_CAMERA_1	5	もし、カメラの動きが早い
N_2_3	0		1	I_EXPO_MOVE	5	それでは、次に移動をして
N_3_1	1	N_3_2	0		5	OK！次はダッシュをして
N_3_2	0		1	I_EXPO_DASH	5	移動をしながらXボタンを
N_4_1	1	N_4_2	0		5	OK！次はジャンプをして
N_4_2	0		1	I_EXPO_JUMP	5	Aボタンを押すことでジャ

▲Excelファイルにてデータの書き込み

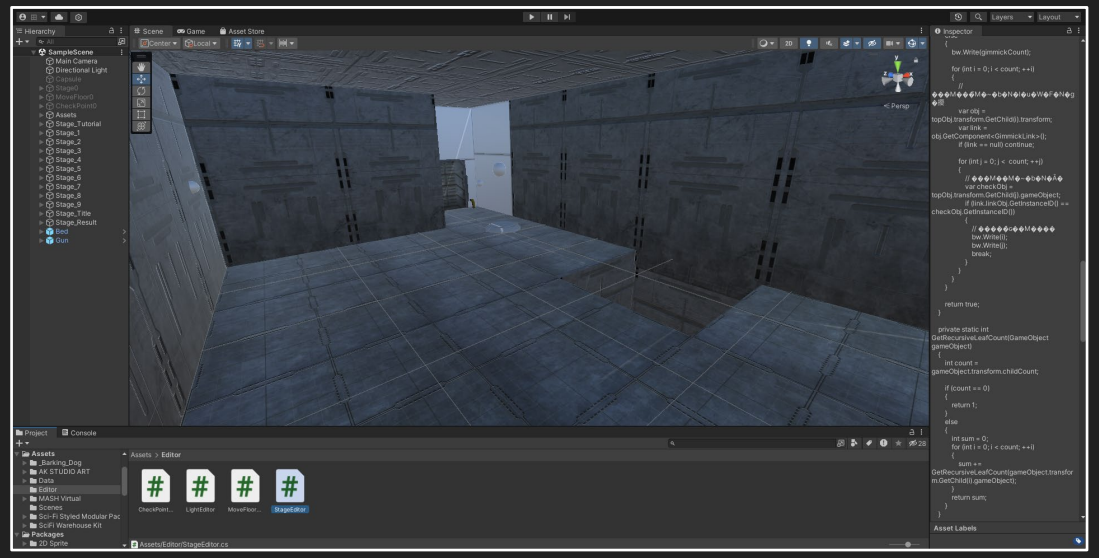


▲実際にゲーム内で取得

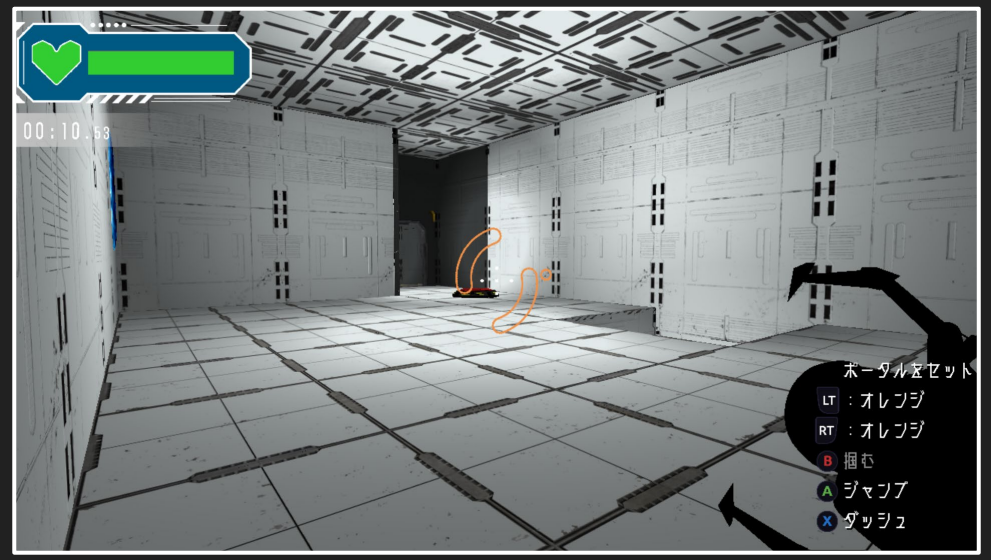
技術紹介⑥：ステージ制作

Code：StageDataManager.cpp

・UnityをEditorとして使用
・ステージに必要な情報を出力、ゲーム内で読み込み



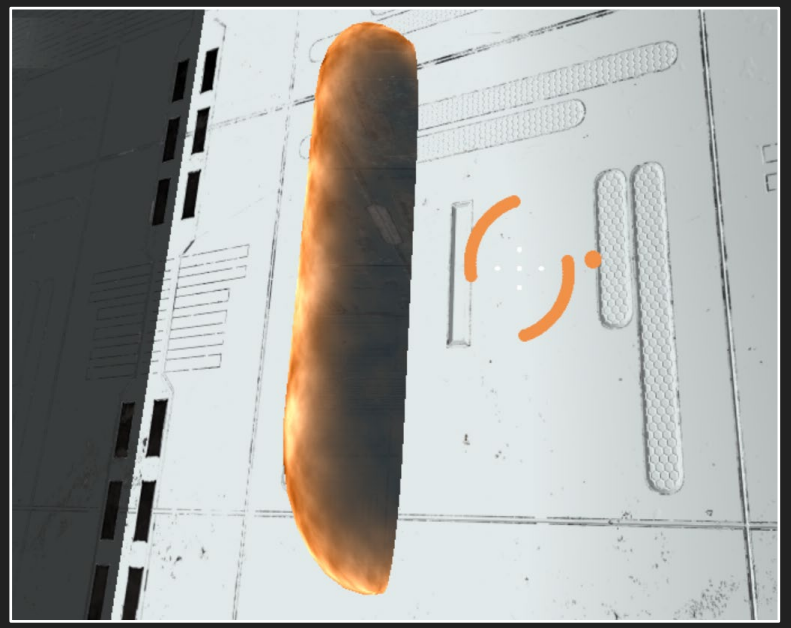
▲Unityにてステージを作成



▲ゲーム内で読み込む

課題：当たり判定

・自作の当たり判定では、
面から法線を取得できない



▲上図のように正しく配置されない
というバグが発生する

解決のため！

・現在行っている自作の当たり判定ではなく
ライブラリに存在するメッシュとの判定で
法線方向を正しく取得できるようにする

※現在修正中ですが、深いところいじっているため
連鎖的にバグ、エラーが多発し解決に至っていません。

その他作品


サーキュックへ
Circle Room
キツネたろう二世の大冒険
GameStart
Option
B：戻る A：決定
Start
Guide
Quit

制作実績① 個人製作

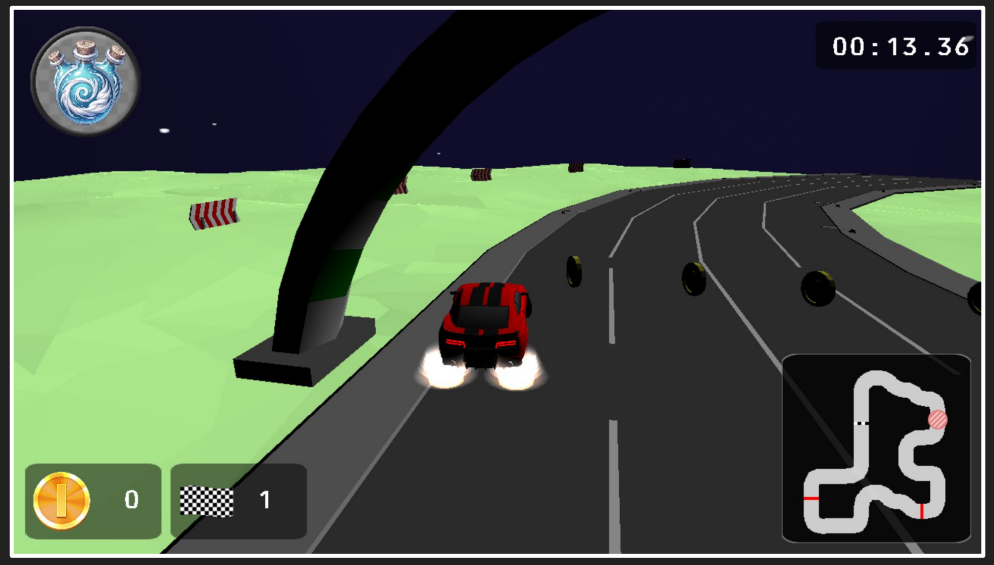
作品名　：サージック！

ジャンル　　：レースゲーム

制作環境　　：DxLib, C++

制作期間　　：3カ月

制作時期　　：2024年6月～2024年9月



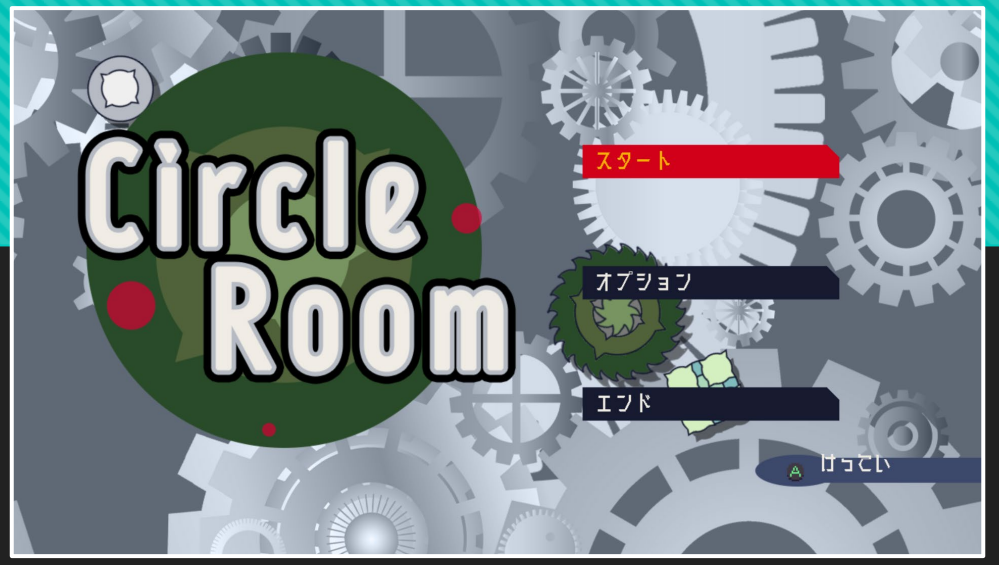
目的、学んだ事　：

- 3D空間での当たり判定
- 簡易的な物理挙動
- オブジェクト指向

Git：https://github.com/itasi29/Cirgic.git

制作実績② 個人製作

作品名　：Circle Room



ジャンル　　：2Dアクション

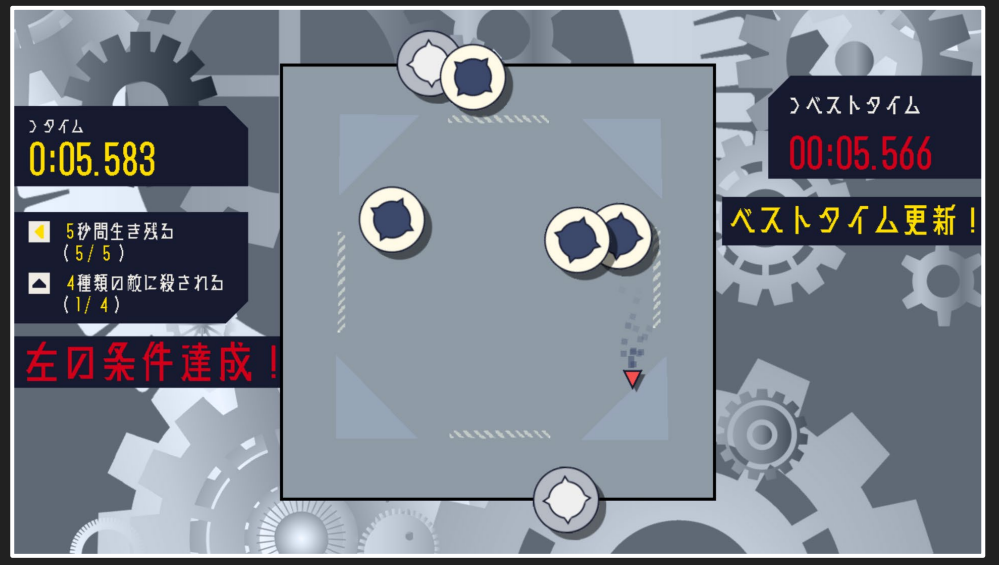
制作環境　　：DxLib, C++

制作期間　　：4カ月

制作時期　　：2023年11月～2024年2月

目的、学んだ事　：

- ゲーム制作の流れ
- csvファイル使った外部データ化
- HLSLを使ったシェーダ



Git：https://github.com/itasi29/CircleRoom.git

制作実績③ チーム製作

作品名　：キツネたろう二世の大冒険

ジャンル　　：3Dアクション

制作環境　　：Unity, C#

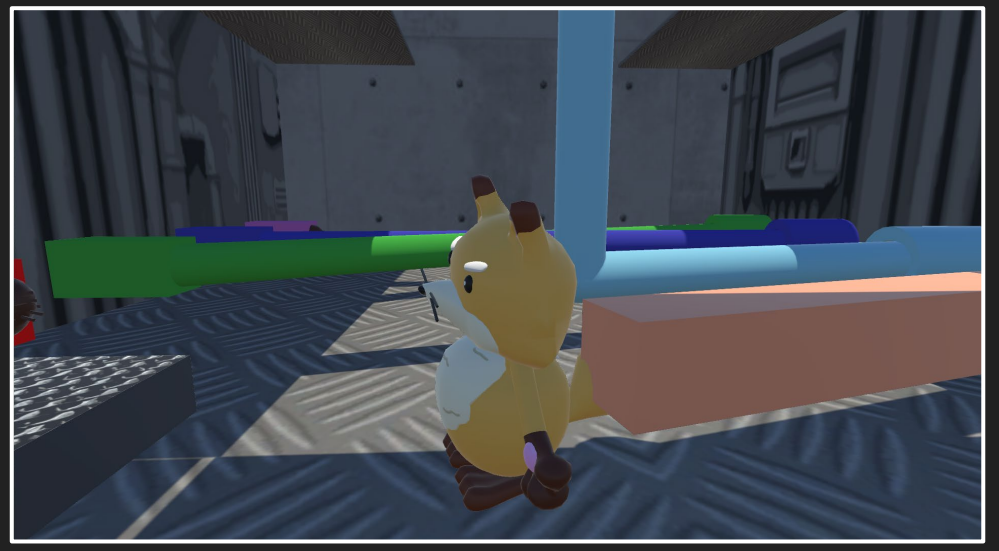
制作期間　　：4カ月

制作時期　　：2024年9月～2024年12月

担当　：
- 2D, 3Dパズルギミック
- タイトル、クレジット

目的、学んだ事　：
- 多職種とのチーム制作
- 同年代以外とのチーム制作

Git：https://github.com/itasi29/FoxProject.git



制作実績④ チーム製作

作品名　：Falling Coin

ジャンル　：2Dアクション

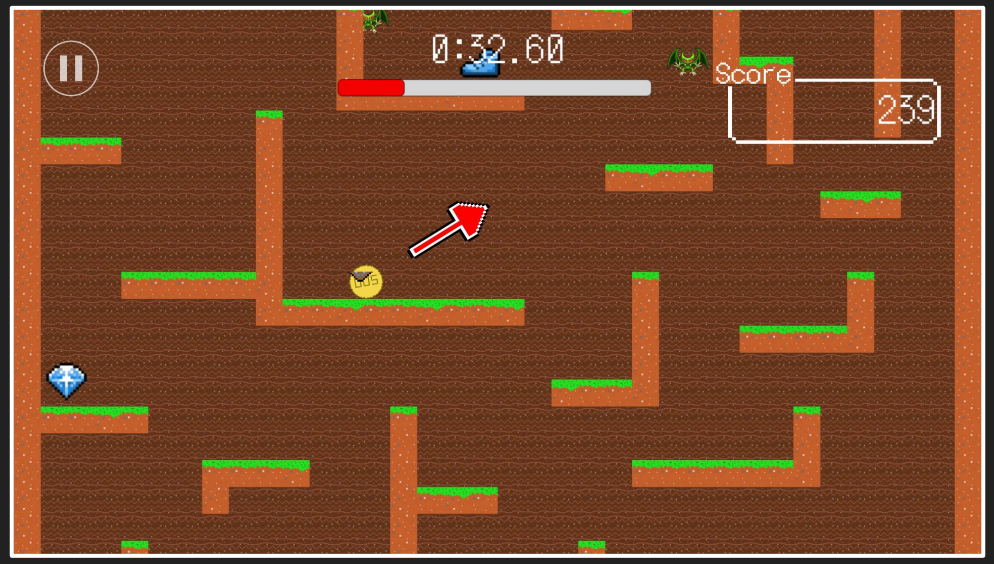
制作環境　：Unity, C#

制作期間　：3カ月

制作時期　：2023年6月～2023年8月

担当　：
- プレイヤー、カメラ
- アイテム

目的、学んだ事　：
- ゲーム制作の大変さ
- ゲーム制作の基礎



Git：https://github.com/itasi29/FallingCoin.git

今後の目標

キャラクター制御に強いプログラマーに
なりたいと考えています。
そのため、よりゲームらしい挙動をできるよう
数学や物理の知識を学び続けていきます。