ポートフォリオ

HAL名古屋
ゲーム4年制学科　ゲーム制作専攻
木村 優作

目次

作品紹介

Dungeons

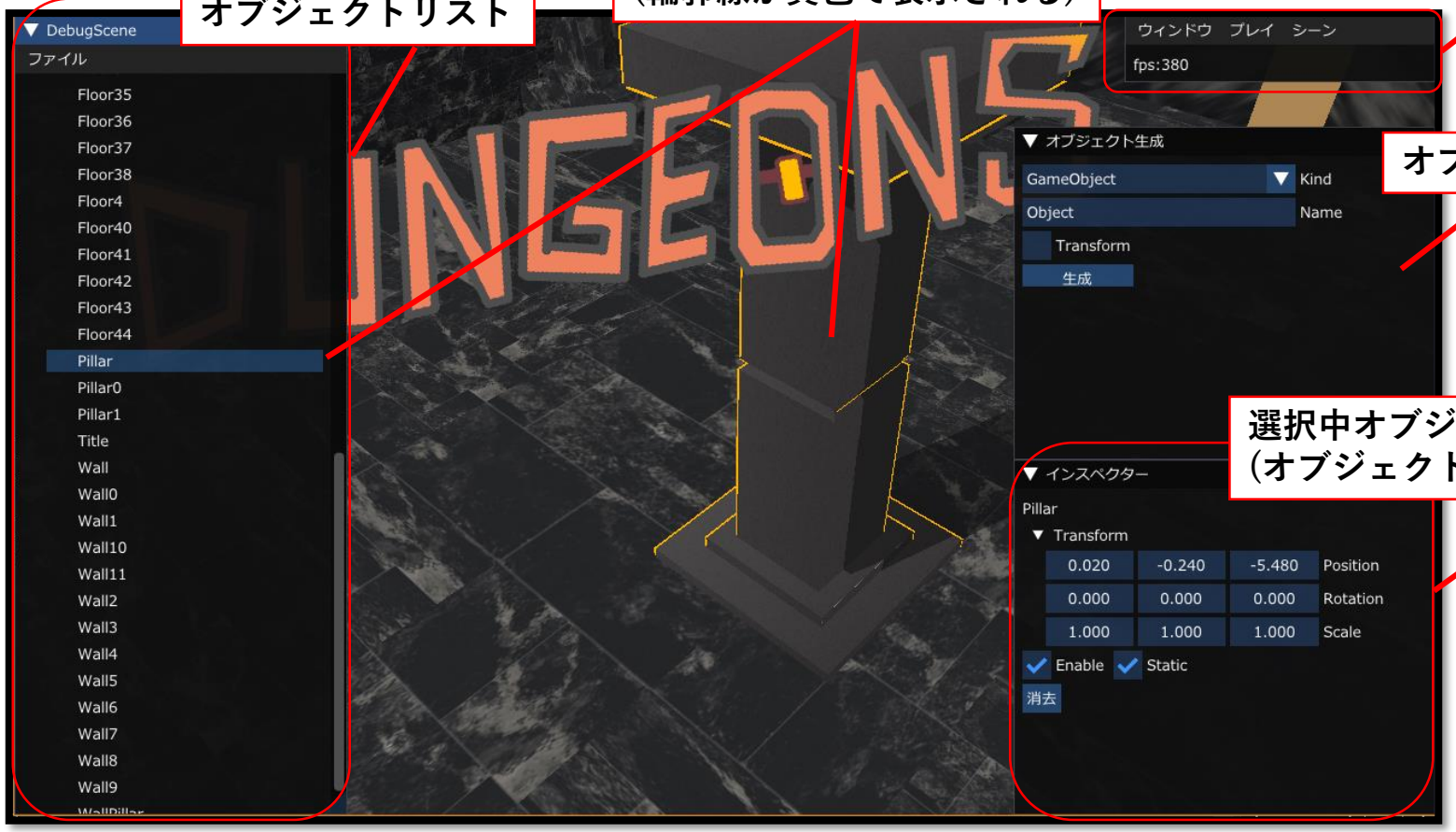
・ジャンル：アクション
・制作期間：ベース部分：2022年 7月～
ゲーム部分：2022年12月～
2023年 4月
・制作人数：1人
・使用言語・ツール：DirectX11 ,C++
IMGUI ,Effekseer
Blender

「Dungeons」はプレイするたびにステージの構造が迷路のように変わるダンジョンを、回避、攻撃、はじく、などのプレイヤーのアクションを駆使しながら攻略する1人プレイ用3Dアクションゲームです。

ステージやタイトル画面を作成するためにIMGUIを使用したエディタの作成も行いました。
キャラクターのアニメーションやオブジェクトのモデルはBlenderで作成しました。

内容紹介(エディタ)

Dungeons



作成したエディタの画面です。
オブジェクトの作成、シーンの読込、保存、テストプレイをすることができます。
作成したシーンデータを読み込むことでゲーム内で使用することができます。

内容紹介(エディタ)

Dungeons

オブジェクト生成ウィンドウ

定義されたオブジェクトの種類から選択

```
// ===オブジェクトのタグ定義===
//  ※ファイルにタグの情報も吐き出すため、オブジェクトを追加する際は下に足していく
enum ObjType {          // オブジェクト型
    OT_OBJ = 0,         // 型無し
    OT_PLAYER,          // プレイヤー
    OT_ENEMY,           // 敵
    OT_GROUND,          // 地面
    OT_WALL,            // 壁
    OT_PILLAR,          // 柱
    BG_3DOBJ,           // 背景オブジェクト
    TEST_2DOBJ,         // テストオブジェクト
    BG_TITLE,           // タイトルロゴ
    BG_ENTER,           // UI
    OT_FENCE,           // フェンス
    PLAYER_ATACK,       // プレイヤーの攻撃当たり判定
    OT_WALLPILLAR,      // 凸凹壁
    ENEMY_ATTACK,       // 敵の攻撃判定
    OT_DOOR,            // ドア
    OT_ENEMY_BULLET,    // 敵の弾
    UI_START_BUTTON,    // ボタンテスト
    UI_TITLE_BUTTON,    // タイトルボタン
    UI_FIN_BUTTON,      // 終了ボタン
    UI_LIFE,            // 体力UI
    UI_STAMINA,         // スタミナUI
    BG_GAMEOVER,        // ゲームオーバー背景
    BG_GAMECLEAR,       // ゲームクリア背景

    MAX_GAMEOBJTYPE
};
```

オブジェクトを現在シーンに生成する

```
// ---オブジェクト生成---
Object* player = CreateGameObject("Player", OT_PLAYER);     // プレイヤー生成
```

オブジェクト生成はオブジェクト生成ウィンドウから行います。
生成するオブジェクトの種類、名前、座標等を選択、入力し生成ボタンを押すと、オブジェクトが生成されます。

オブジェクト生成はエディタ以外で、ソース上から生成できます。

内容紹介(エディタ)

Dungeons

保存時

オブジェクトデータ

```
struct SceneObjData
{
    ObjType id;                 // オブジェクトの型
    Transform transform;        // 座標等
    char name[50];              // 保存した名前
    char pass[100];             // 素材のパス
    bool enable;                // 更新、描画の有無
    bool sta;                   // 静的なオブジェクトかどうか
};
```

シーン
private:
std::map<名前,オブジェクト>
protected:
std::string シーン名

保存 → シーン名.dat

読込時

シーン名.dat → 読込 → オブジェクトデータ

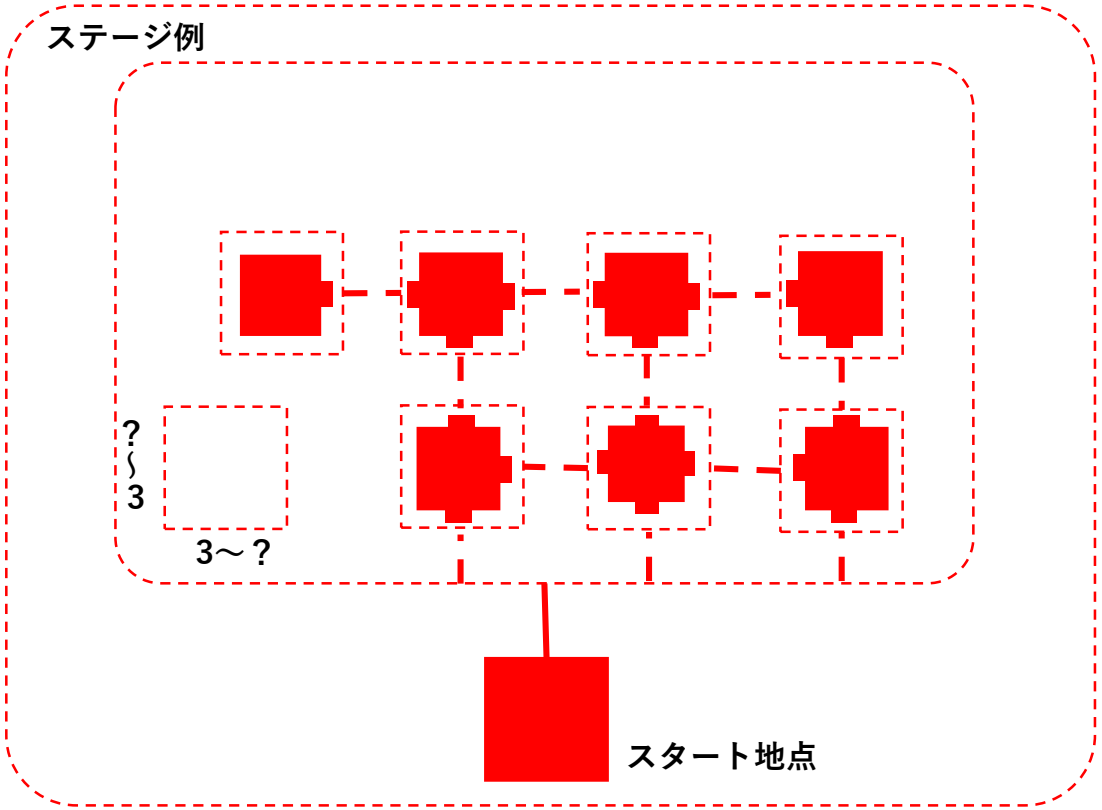
```
struct SceneObjData
{
    ObjType id;                 // オブジェクトの型
    Transform transform;        // 座標等
    char name[50];              // 保存した名前
    char pass[100];             // 素材のパス
    bool enable;                // 更新、描画の有無
    bool sta;                   // 静的なオブジェクトかどうか
};
```

→ 生成・登録 → シーン
private:
std::map<名前,オブジェクト>
protected:
std::string シーン名

保存したシーンデータは上記のように
保存、読込を行っています。

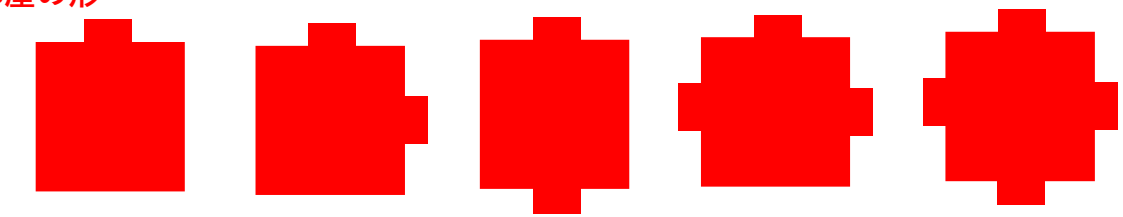
内容紹介(ステージ生成)

Dungeons



ステージの生成は奥の部屋の数、横の部屋の数を乱数で求め、横方向の中央からのずれも乱数で算出し、横、奥の軸方向が一致する隣接する部屋同士すべてを連結します。
連結している数、方向から部屋の形を求め、部屋の形が該当するステージデータからランダムで部屋の情報を読み込みます。

部屋の形



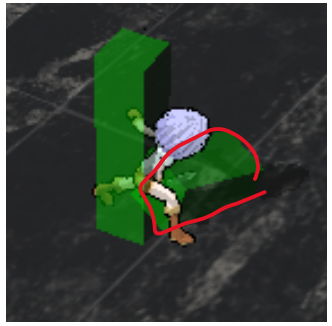
該当するステージデータ例

```
namespace RoomCrossStage
{
    LPCSTR pass[ROOM_CROSS_STAGE_MAX] = {
        "data/stage/Stage5.dat",
        "data/stage/Stage6.dat",
    };
}
```

十字型の部屋は
これらのステージ
からどれかを読込

内容紹介(当たり判定)

Dungeons



プレイヤーモデルのボーン(hand_R)の姿勢を
取得し、剣の当たり判定も同じように動かす。

```
XMMATRIX Bone = m_pParent->GetModel()->GetBoneMatrix("hand_R");
```

剣の当たった位置でエフェクトの発生する位置を変える

```
EFFECT->Play(HIT_EFFECT, GetHitPos());
EFFECT->Play(EXPLOSION_EFFECT, GetHitPos());
EFFECT->Play(SWORD_EFFECT, GetHitPos());
```

当たり判定は主にOBBと境界球で作成しました。
衝突した際にめり込んだ量を戻す計算を行うことで、
壁ずりの動きをするように作成しました。また、
めり込んだ量を算出しないものに関してはフラグで
指定できるようにしました。

特定のボーンの姿勢を取得することで、プレイヤー
のアニメーションに沿うように当たり判定を動かせ
るようにしました。

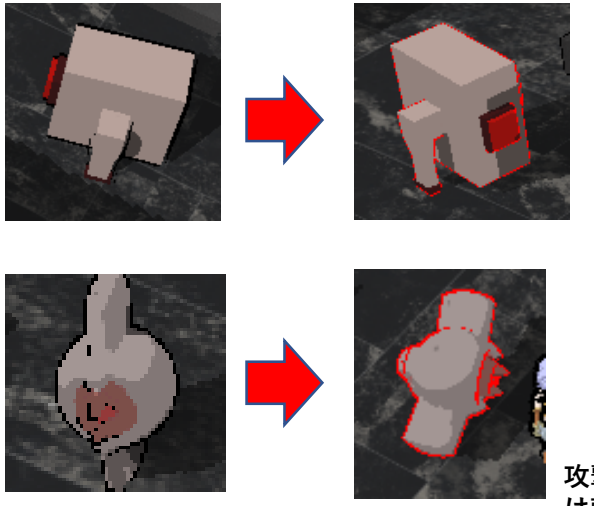
また、衝突した地点も算出することで、衝突した場
所でエフェクトを再生する処理も実装しました。

内容紹介(見た目の工夫)

Dungeons

プレイヤーや敵などの動くオブジェクトが
壁や柱の裏に立った時にシルエットが描画される

遊んでいる最中に、壁や柱の後ろに立った際に
プレイヤーが見えなくなって遊びにくく感じること
があったので、特定のシェーダーを割り当てたオブ
ジェクトの後ろ側に立った際にシルエットが見える
ようにしました。シルエットが単色だと目立ちすぎ
たのでディザパターンで模様をつけています。

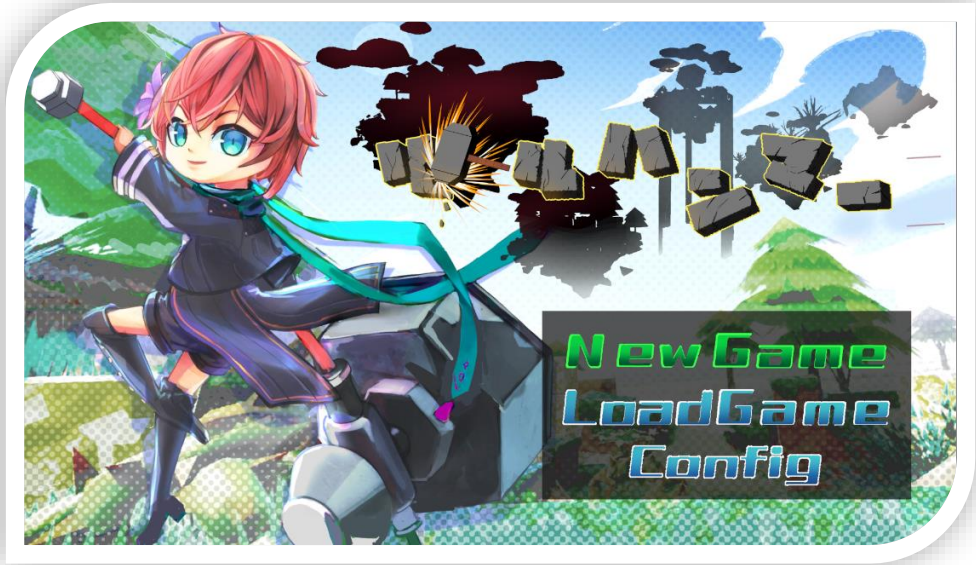


攻撃の直前の敵キャラクター
は輪郭線が赤くなる

オブジェクト全体で輪郭線を描画していますが、
攻撃をしてくる直前の敵キャラクターの輪郭線だけ
赤く描画することで、敵の把握ができ、遊びやすく
なるように意識しました。

作品紹介

リトルハンマー



・ジャンル：パズルアクション
・制作期間：2022年2月~2022年5月末
・制作人数：15人(プランナー2人,
プログラマー9人,
デザイナー3人,動画制作1人
・担当箇所：Git導入、ギミック全般
ミス演出
ゲームクリア演出
・使用言語・ツール：Unity,C#,Sourcetree

「リトルハンマー」はハンマーを使ってブロックを吹き飛ばすパズルアクションゲームです。
吹き飛ばしたブロックで階段を作ったり、橋を架けたりすることでゴールを目指します。
飛ばすブロックの手順、方向を考えることが必要になってきます。
ステージは全部で25ステージ遊ぶことができます。

主に、吹き飛ばすブロックや破壊できる壁などのギミックを担当しました。

担当内容(ギミック)

リトルハンマー

移動ブロック　　ハンマーで殴ると壁や他のブロックに衝突するまで
殴った方向に進み続けます。空中にある場合は
徐々に落下します。

破壊ブロック　　ハンマーで殴ると壊すことができます。

棘ブロック　　プレイヤーが衝突するとミス判定になります。

担当内容(ギミック)

リトルハンマー

左の画面ではプレイヤーを対岸に渡らせるために移動ブロック
を配置しています。

移動ブロックは空中にある場合は徐々に落下し、2マス以上の隙間
を超えることができません

1マス以上の隙間は超えることができるので階段の形に
ブロックを積み上げることができます。

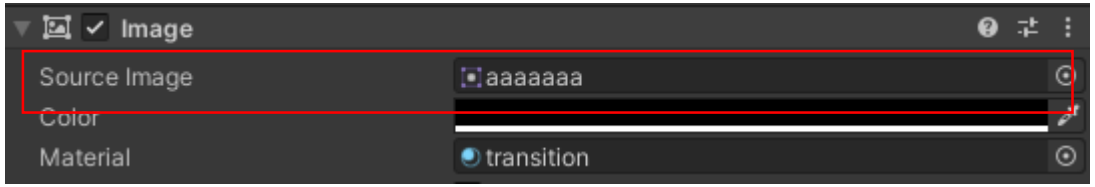
担当内容(ミス演出)

リトルハンマー

マスク画像

棘ブロックに触れると、画面が暗転し、円形にフェードアウトします。
完全にフェードアウトした後にリスタート処理が呼ばれます。

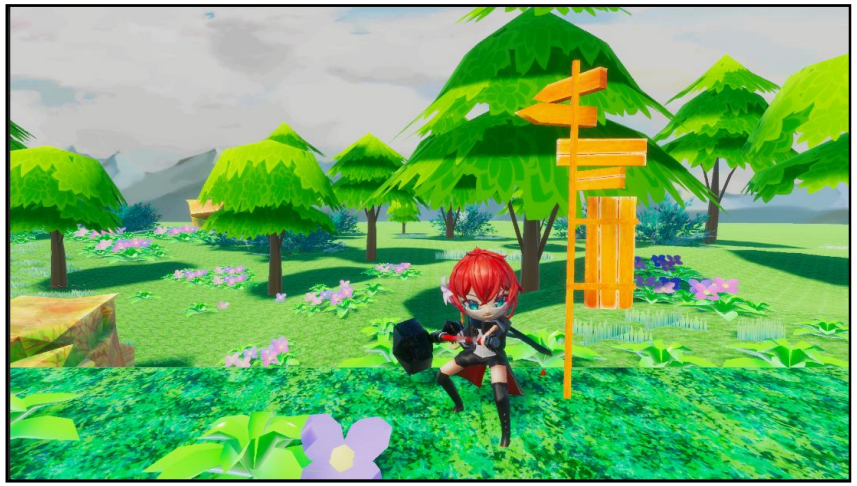
マスク画像のしきい値を徐々に上げることで段々見える範囲が狭まっていくように作成しました。



インスペクター上でマスク画像を差し替えることで、他の演出に差し替えることもできるようにしました。

担当内容(ゲームクリア演出)

リトルハンマー



クリア演出では画面をハンマーで殴りつけて、画面が割れるような演出を作成しました。

ハンマーを振るアニメーション後にその時点の画面をレンダーテクスチャに書き込み、メッシュをいくつかに分割したスクリーン形のモデルに貼り付け、そのモデルを最前面に表示しています。

分割したメッシュそれぞれにランダムな移動量を加えることで、バラバラになるように作成しました。

作品紹介

RUN

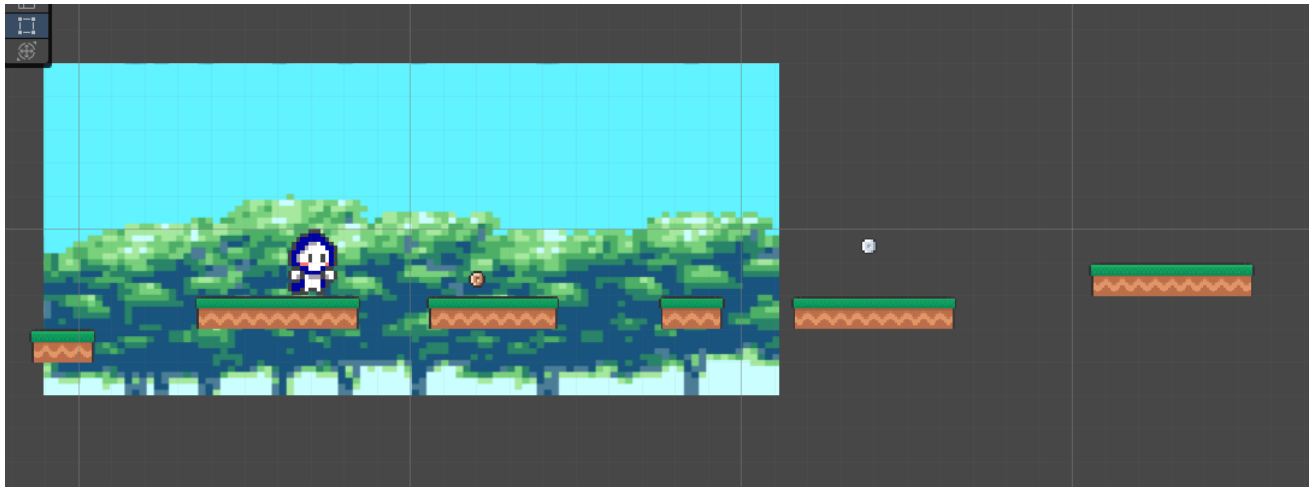
・ジャンル：2Dランゲーム(スマートフォン)
・制作期間：2022年8月中旬~2022年8月末
・制作人数：1人
・使用言語・ツール：Unity,C#

「RUN」はシンプル操作な2Dランゲームです。
道中に現れるアイテムやジャンプを駆使して、できるだけ高いスコアを獲得することを目指します。

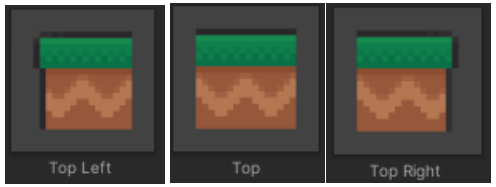
ステージの地形がランダムに生成されるように工夫をしました。
短い期間での制作だったので、シンプルに繰り返し遊べるゲームを目指しました。

内容紹介(ステージの生成)

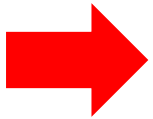
RUN



ステージはプレイヤーが一定距離以上進んだ際に
足場の高さ(プレイヤーのジャンプが届く距離)と
足場の幅をランダムで求め、prefab化したブロック
を生成します。その上にコイン生成の有無、足場
をとの距離を決め、コインも生成しています。



足場ブロックプレハブ



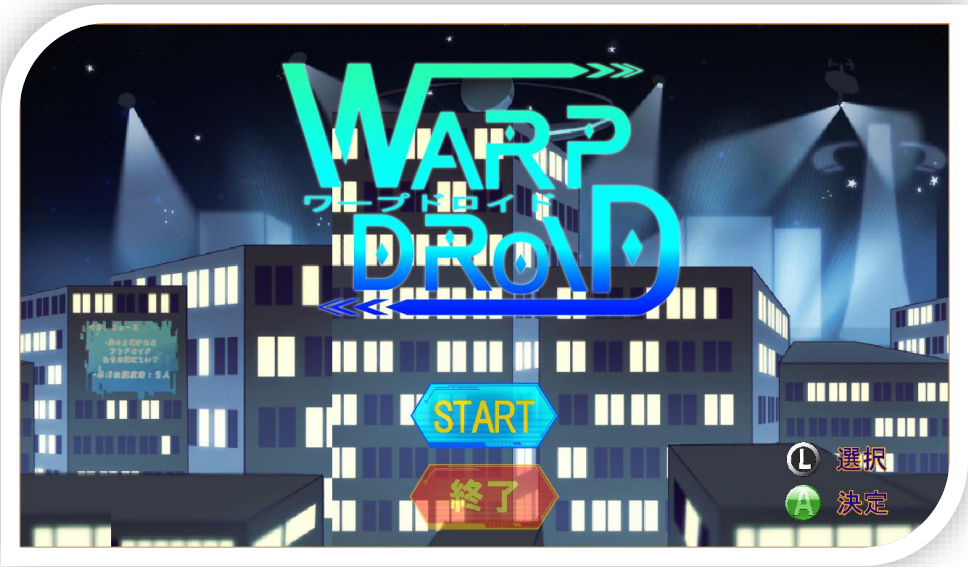
真ん中のブロック(ランダム数)と
左端、右端ブロックを組み合わせて
生成する

道中アイテム

道中に落ちているアイテムは、一定以上のスコアを
獲得した際に、生成した足場の上に出現するように
調整しました。

作品紹介

ワープドロイド



・ジャンル：パズルアクション
・制作期間：ベース部分：2021年 11月～2022年1月
・制作人数：9人(プランナー1人,プログラマー6人,デザイナー2人
・担当箇所：ステージセレクト処理,プレイヤー,ワープの挙動,一部の敵の処理,UIの配置
・使用言語・ツール：C++,DirectX11,GitHub,Effekseer

「ワープドロイド」は一筆書きのように敵を殲滅するパズルアクションゲームです。
プレイヤーは、ワープ範囲内の敵の目前に瞬間移動し、一直線に切り抜けるので
敵をうまく殲滅させられるルートを考える必要があります。
ステージは全部で12ステージ遊ぶことができます。

初めてのチーム制作だったので、勝手がわからない部分もありましたが、手広く担当しました。

内容紹介(ステージセレクト)

ワープドロイド

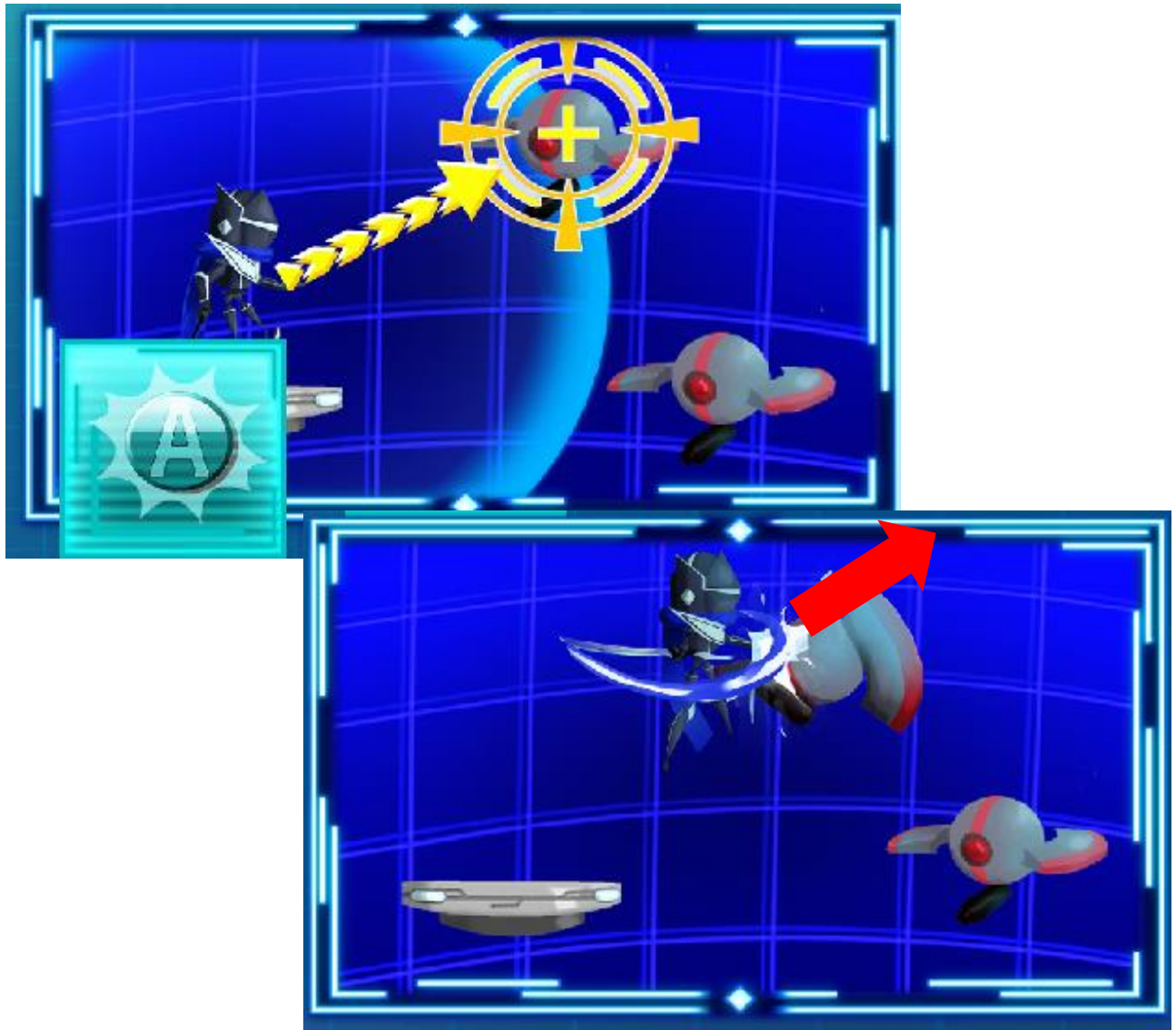
ステージ選択シーンでは、現在のステージのクリア状況に応じて徐々にステージが解放される処理を作成しました。解放されていないステージは選択されないようになっています。

```
private:
    static StageSelectManager* m_pInstance;     // マネージャ
    std::vector<bool> m_pSelect;                // ステージクリア情報
    int m_num;                                  // 現在ステージ番号
```

ステージのクリア状況は単一の管理クラスにbool型の動的配列で管理することで、ステージ数の変更があっても正常に動作するようにしました。

内容紹介(ワープ挙動について)

ワープドロイド



ワープ範囲内に敵キャラクターが入っていると、
ターゲットが表示されます。範囲内に複数の敵がいる
際には、左スティックの入力方向に応じて、
ターゲットを切り替えることもできます。

ターゲットがいる状態でAボタンを押すと、敵の目前
に瞬間移動し、矢印の方向に切り抜けて移動します。
切り抜ける方向は、現在位置からターゲットへの
ベクトルを求め、一定距離だけ進行するように処理を
作成しました。