日本最速プログラマー

名前　　：東愁汰

趣味　　：レース / ゲーム / 音楽 / お酒

好きな串：ズリ / レバー / アスパラ巻 / えのき巻

イチオシ：レーシングカート元ジュニア日本一位

メール　：shpicpkp@gmail.com

X　　：@ShpiChanV

GitHub　：SHUTAHIGASHI (Shpi) (github.com)

所持スキル

経験豊富な男です！

【プログラミングスキル】

C/C++：2年半
DxLib：2年半
Unity/C#：2年
DirectX：半年
Git/GitHub：2年
Effekseer：1年半
Maya：半年
Adobe PhotoShop：3年
Word/Excel/PowerPoint：4年

【ドライビングスキル】

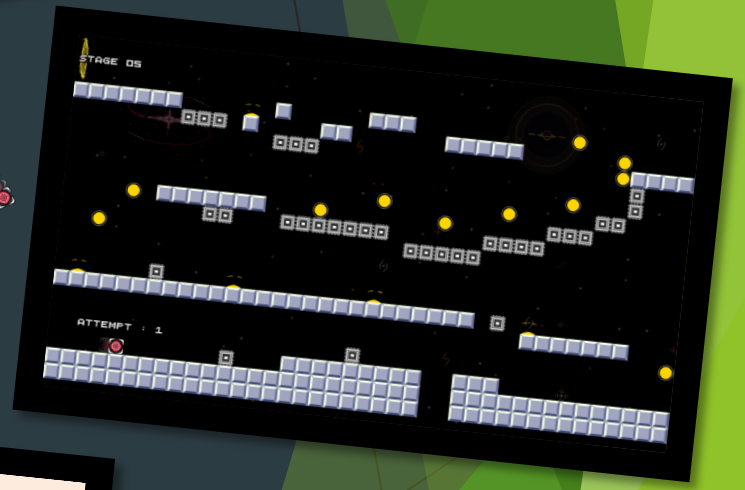
普通自動車免許 MT
ROTAX MAX CHALLENGE 2013
MAXCADET 日本ランキング1位
ROTAX MAX FESTIVAL 2017 JUNIORMAX 予選一位通過
ROTAX MAX FESTIVAL 2018 JUNIORMAX 予選三位通過

就活作品

最終作品：1本　2年次：2本　1年次：1本

SPACE Fighting

SQUAREJUMPER



最終作品

作品名：SPACE FIGHTING

ジャンル：3DSPACEシューティング

開発環境：C++ / DxLib
VisualStudio2022
Effekseer

対応機種：Windows

制作期間：約400時間

GitHub：

https://github.com/SHUTAHIGASHI/HIGASHI_SpaceFighting

ゲーム内容①

宇宙空間を漂う岩を避けつつ、

襲い来るインベーダーたちを倒し、僕らの地球を防衛しよう！

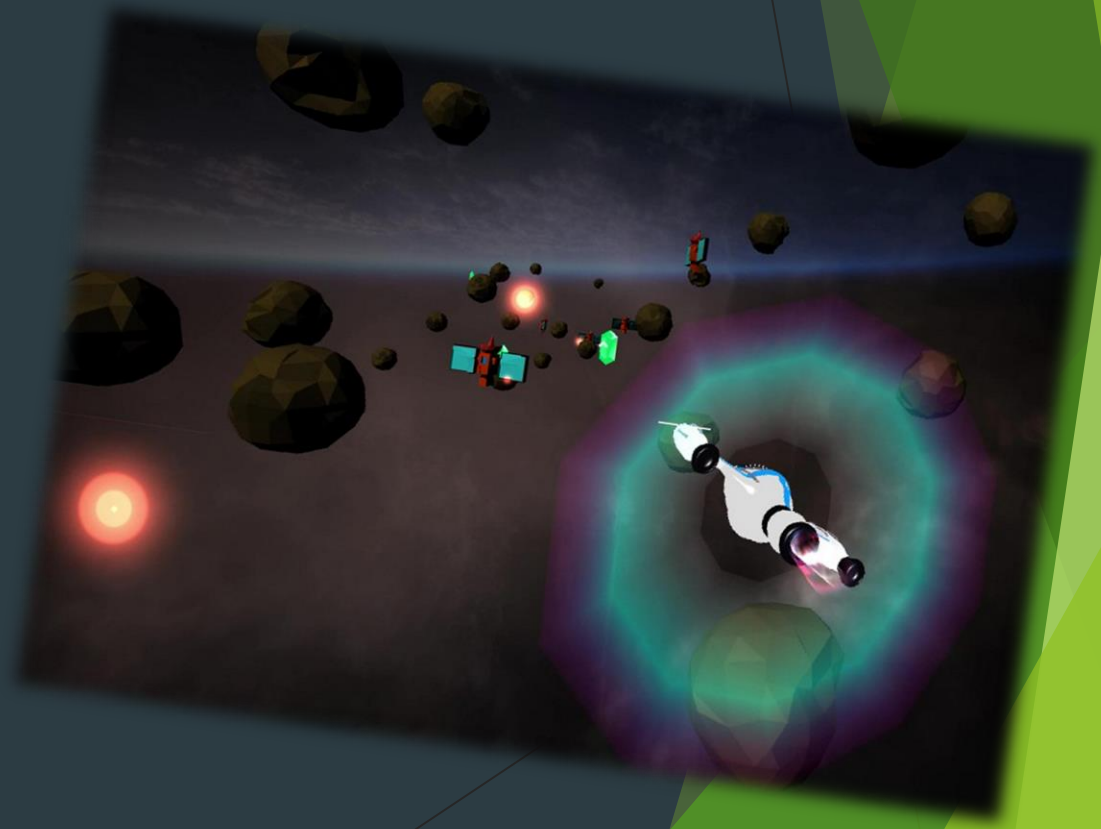
ゲーム内容②

敵の弾幕は回避してうまくかわそう！

ボスにはチャージショットをぶつけて大ダメージを与えよう！

ゲーム内容③

爽快感あふれる**ド派手な戦闘！**



技術紹介①

ステージ読み込みのCSV 外部ファイル化

A1 ボス生成判定

	A	B	C	D	E
1	ボス生成判定	ボス体力	岩オブジェクト数	敵撃破数	ステージサイズ
2	0	0	30	30	50000
3	1	100	30	0	0
4	1	150	30	50	100000
5	1	250	40	100	200000

```
{
    // ファイルから行を読み込む
    while (getline(file, line))
    {
        // 1行目はスキップ
        if (lineCount != 0)
        {
            // 区切り文字で文字列を分割
            strFirst = 0, strLast = 0;
            // 一時ステージデータリセット
            tempStageData.clear();
            for (int i = 0; i < line.size(); i++)
            {
                if (line[i] == ',')
                {
                    // 区切り文字で文字列を分割
                    strLast = i;
                    // 文字列を取得
                    std::string str;
                    for (int j = strFirst; j < strLast; j++)
                    {
                        // 文字列を取得
                        str += line[j];
                    }
                    // 一時ステージデータに追加
                    tempStageData.push_back(str);
                    // 次の文字へ
                    strFirst = strLast + 1;
                }
            }

            // ステージデータに変換
            StageData stageData;
            stageData.isBoss = static_cast<bool>(std::stoi(tempStageData[0]));
            stageData.bossHp = std::stoi(tempStageData[1]);
            stageData.rockNum = std::stoi(tempStageData[2]);
            stageData.enemyNum = std::stoi(tempStageData[3]);
            stageData.length = static_cast<float>(std::stoi(tempStageData[4]));
            // ステージデータに追加
            m_stageData[lineCount - 1] = stageData;
        }
        // 行数をカウント
        lineCount++;
    }
    // ファイルを閉じる
    file.close();
}
```

問題は見つかりませんでした

これにより

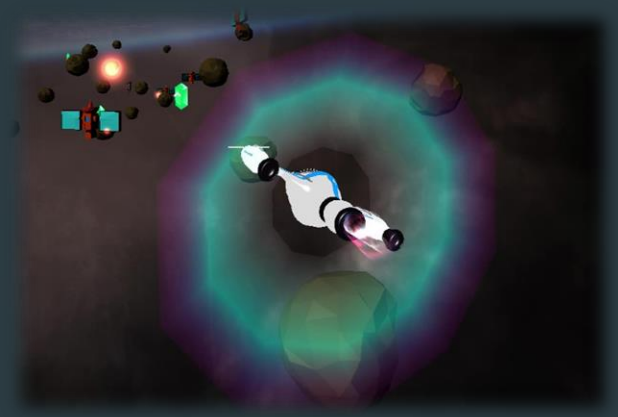
・ステージの難易度調整をよりスムーズに行える
・プログラマ以外の人も簡単にステージの変更

という点を実現することが可能になった

技術紹介②

シングルトンクラスによる エフェクトの管理

シングルトンクラスを使い
エフェクシアを管理することで
どんな場面でも使いやすく



```
class EffekseerManager
{
public:
    // デストラクタ
    ~EffekseerManager();
    // EffekseerManager使用者はGetInstance()を通した参照からしか利用できない
    static EffekseerManager& GetInstance()
    {
        // 唯一の実態
        static EffekseerManager instance;
        // それの参照を返す
        return instance;
    }

    // 更新
    void Update();
    // 描画
    void Draw();
    // 各種エフェクトの生成
    void CreateEffect(EffectType type, bool loop, class ObjectBase* obj);
    void CreateEffectAndSetScale(EffectType type, bool loop, class ObjectBase* obj, float scale);
    void CreateEffect(EffectType type, bool loop, VECTOR pos);
    // すべてのエフェクトの停止
    void StopAllEffect();
    // 選んだエフェクトの停止
    void StopEffect(EffectType type);
    void StopEffectTargetObj(class ObjectBase* obj);
    // 無効化されているエフェクトの削除
    void DeleteDisableEffect();
    // 指定したエフェクトの座標を設定
    void SetEffectPosition(EffectType type, VECTOR pos);
    // 指定したエフェクトの回転を設定
    void SetEffectRota(EffectType type, VECTOR rota);
    // 指定したエフェクトが再生中かどうか
    bool IsPlayingEffect(EffectType type);
    // 指定したエフェクト取得
    EffectBase* GetEffect(EffectType type);
```

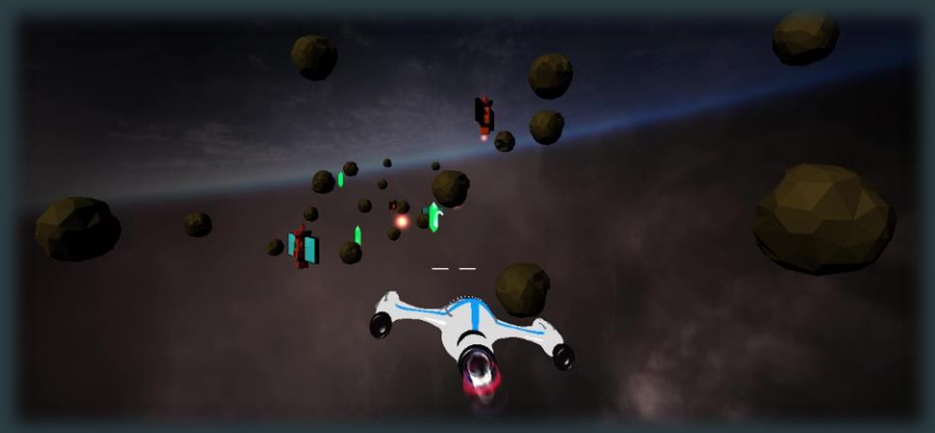
技術紹介③

処理の軽量化

```
void Load::DeleteAllData()
{
    // データを削除する
    for (auto& data : m_data)
    {
        MV1DeleteModel(data.second);
    }
    m_data.clear();
}
```

```
void StageObjectManager::DeleteDisabledObject()
{
    // 無効になったオブジェクトは排除
    auto rmIt = std::remove_if(m_pObjects.begin(), m_pObjects.end(),
        [](const std::shared_ptr<StageObjectBase>& obj)
        {
            return !obj->IsEnabled();
        });
    // 実際に範囲を指定して削除
    m_pObjects.erase(rmIt, m_pObjects.end());
}
```

・不要になったオブジェクトはすぐに削除
・必要以上の読込、生成、削除は行わない



一年次作品紹介

作品名：Square Jumper

ジャンル：2Dアクション

開発環境：C++ / DxLib
VisualStudio 2022

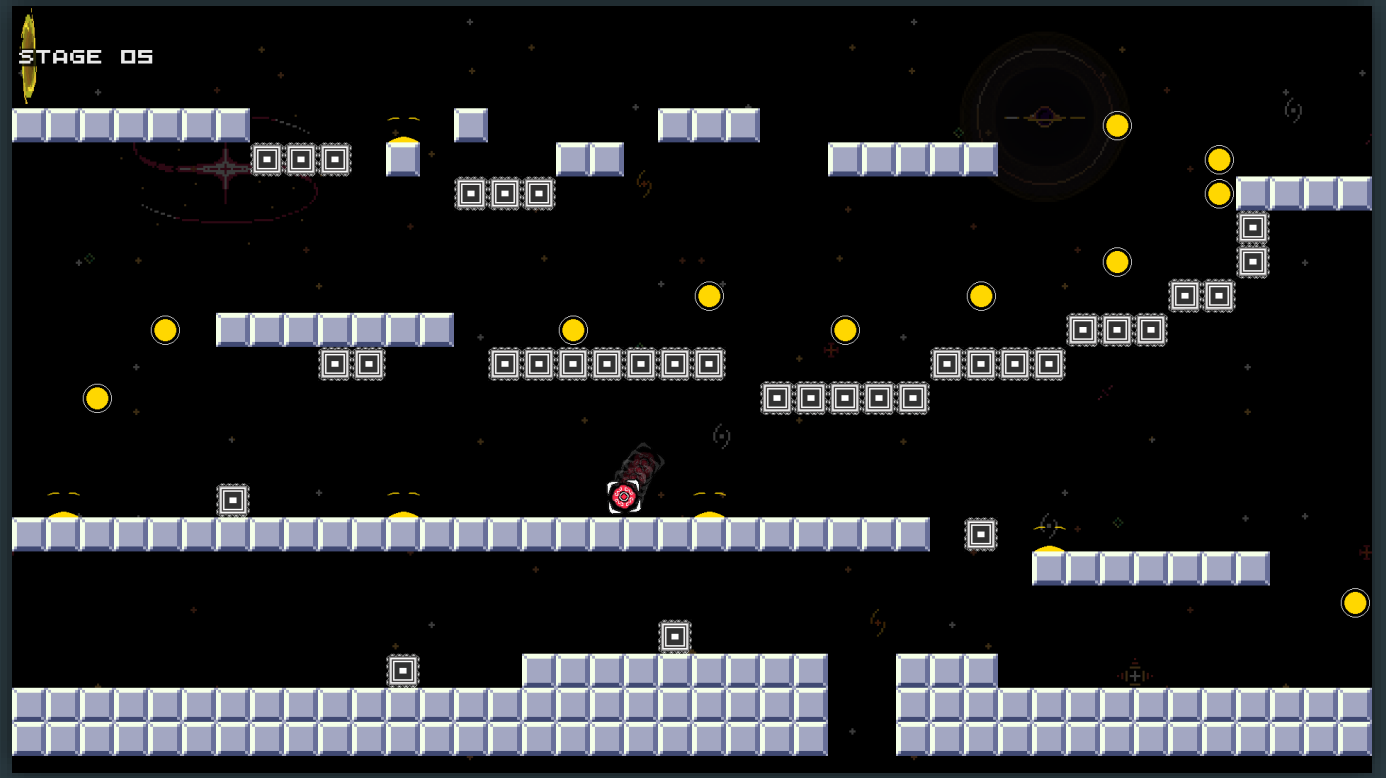
対応機種：Windows

制作期間：約300時間

GitHub：

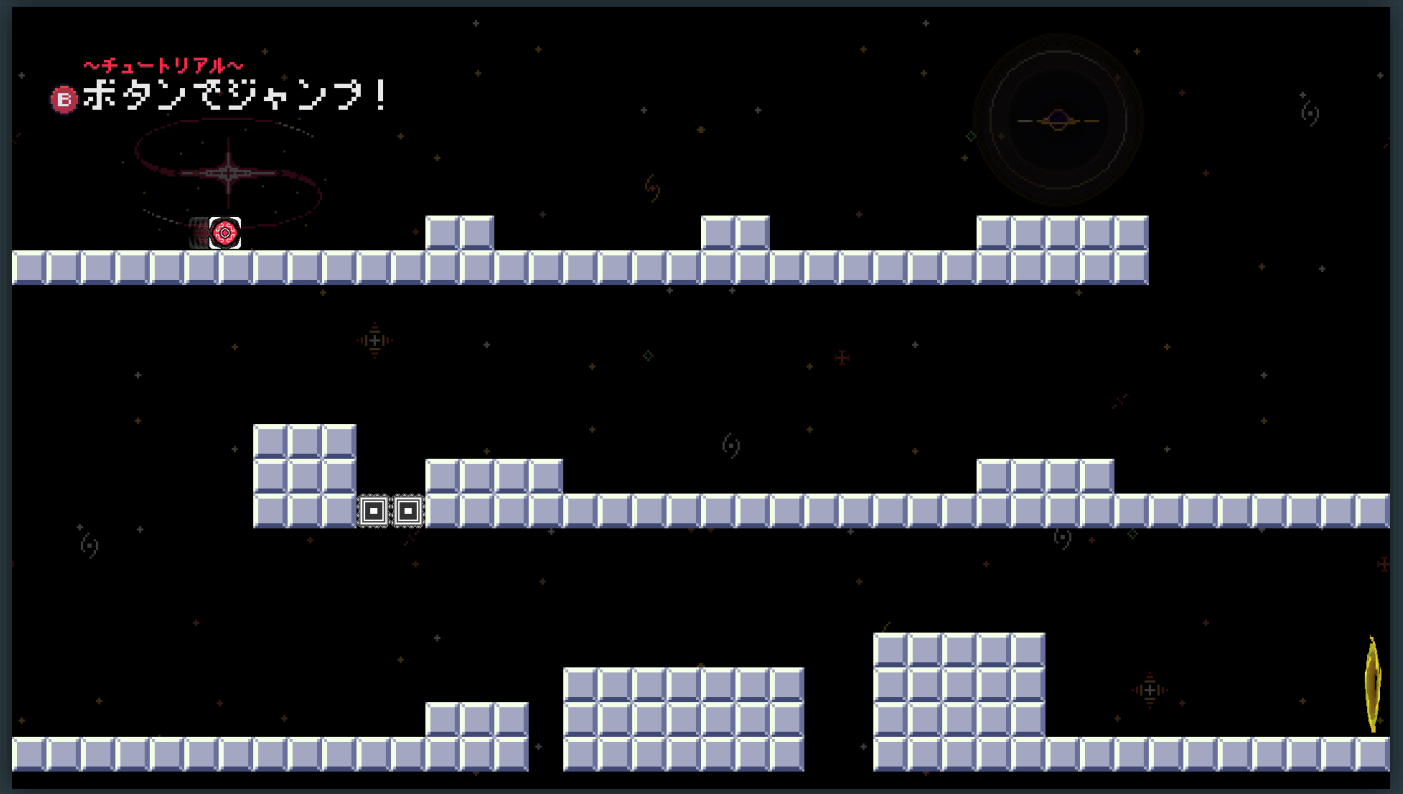
https://github.com/SHUTAHIGASHI/HIGASHI_SquareJumper

ゲーム内容①



ジャンプとギミックをうまく利用して
ステージのゴールを目指そう！

ゲーム内容②



基本はボタン一つの簡単操作で
誰でも遊びやすいゲームに

ゲーム内容③

ステージの挑戦回数に応じた
ローカルランキング

その他作品紹介①

作品名：チノ先生の抜き打ちテスト

ジャンル：2Dクイズゲーム

開発環境：C++ / DxLib
VisualStudio 2022

対応機種：Windows

制作期間：約30時間

概要：約5日間で企画から制作
ドット素材のみ外注

GitHub：

https://github.com/SHUTAHIGASHI/HIGASHI_CHINOTEST

その他作品紹介②

作品名：猫好的ネフィ=サーンへの挑戦状

ジャンル：2Dアクション

開発環境：C++ / DxLib
VisualStudio2022

対応機種：Windows

制作期間：約60時間

GitHub：

https://github.com/SHUTAHIGASHI/HIGASHI_NEFFYBD

今後について

今まで以上に演出部分へのこだわりを深め
実際に触れる遊びの部分だけではなく
見た目でも楽しんで満足していただく

そして、ゲームとして
”伝えたい内容を
しっかり伝えることのできる作品”
を目指して制作に励んでいきます

ご覧いただきありがとうございました