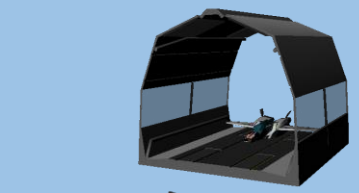
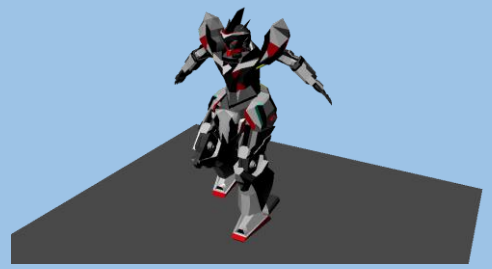
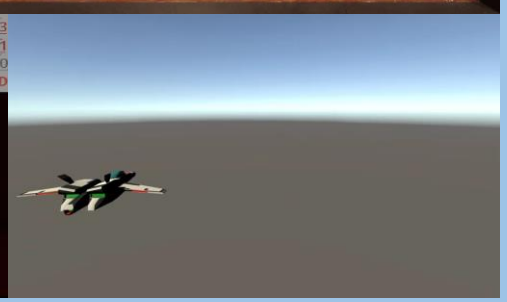
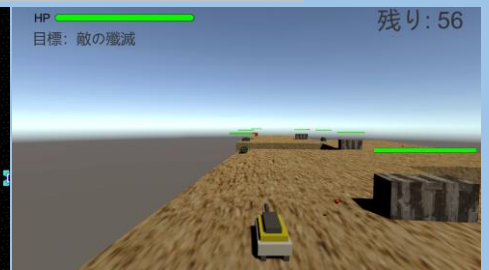
ポートフォリオ









HAL大阪　岩本典志

目次 ※各項目をクリックすると指定の項目まで移動することができます

※ポートフォリオ内のQRコードは**クリック**または**スマホから読み取る**ことでリンク先に移動できます。

プロフィール

岩本(イワモト) 典志(ノリユキ)

兵庫県伊丹市出身
2001年8月20日生まれ
HAL大阪 ゲーム制作学科 プログラム専攻 3年在学中

趣味：サイクリング、プラモデル作りなど
特技：ドット絵制作など

資格

- 基本情報技術者
- CGエンジニア検定 ベーシック
- 情報検定 情報活用試験 1級
- 情報システム試験 プログラマー認定
- 第一種普通自動車免許(AT限定)

作品QRコード

リンク先ではいくつかの作品の
プロジェクト一式と実行環境をアップロードしています。

使用言語

受賞歴

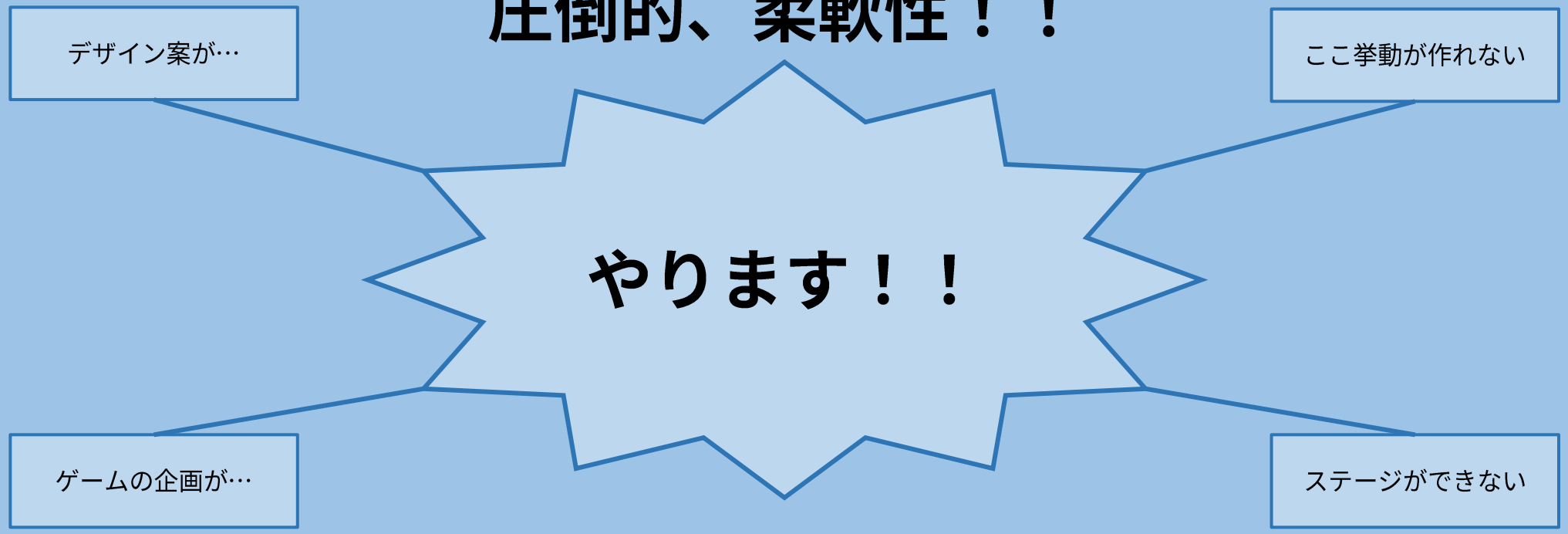
学内コンテスト
- 3校合同コンテスト 独創力賞
- 3校合同コンテスト 構成力賞
- HEW EVENT WEEK 意欲賞

学外コンテスト
- 日本ゲーム大賞2022 アマチュア部門 優秀賞
- ゲームクリエイター甲子園2022 総合大賞

使用ツール

自己PR

どんなことでも挑戦する。
気になったのならやってみる。
圧倒的、柔軟性！！



特にゲーム制作をする際に、発揮されていると感じている。
自身の限界を決めてしまうのではなく、さらに挑んでいくのが
私の強みだと思っています。

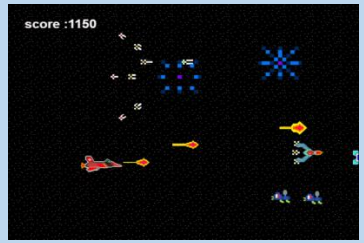
今までのゲーム制作について

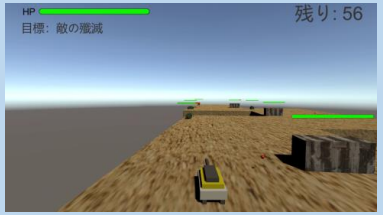
高校
- Unityでのピンボールゲーム制作

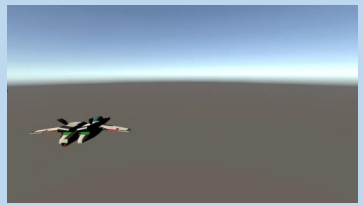
専門学校1年
- Gdevelop5を使ってシューティングゲーム制作
- C言語での防衛アクションゲーム制作
- Unityでタンクシューティングゲーム制作

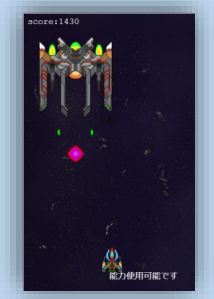
専門学校2年
- Unityでスニーキングゲームをチームで制作
- DirectXを利用して、タワーディフェンスゲームをチームで制作

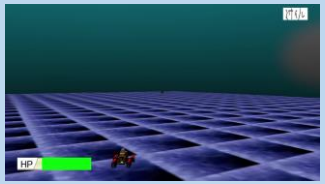
専門学校3年
- Unityでアクションパズルゲームをチームで制作
- Phaser3を利用して、シューティングゲームを制作
- DirectX11を利用して、三人称シューティングゲーム制作中







個人制作ではシューティングゲームを主に作成し、
チーム制作では様々なジャンルのゲームを制作している。

作品：ギアガチャン

チーム作品
ギアガチャン

作品概要

歯車を操作して、宝箱を獲得していくアクションパズルゲームです。
歯車がかみ合う金属的な「感触」を表示した作品。

制作について

- 使用ツール
  Unity
- 制作期間
  4ヶ月
- 制作人数
  12人
- ジャンル
  アクションパズルゲーム
- 担当箇所
  メインゲームのプレイヤー挙動処理

動画リンク

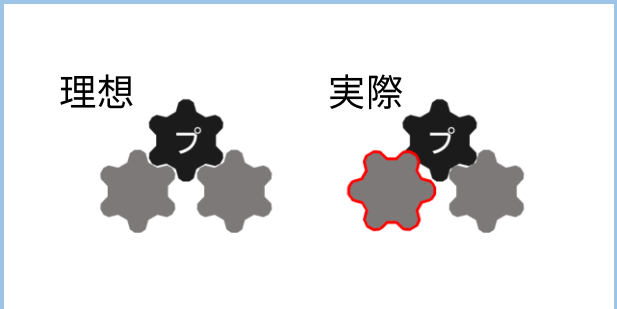
受賞

- 日本ゲーム大賞2022 アマチュア部門 優秀賞
- ゲームクリエイター甲子園2022 総合大賞
- CGC2022 エンターテインメント部門 ノミネート作品

作品：ギアガチャン

１．プレイヤーのかみ合い処理について

このゲームではプレイヤーとゲーム内の一部のオブジェクトがかみ合うと仕様があります。
製作開始時点では接触したオブジェクトに対してかみ合い、くっつくように制作しました。
しかし、二つ以上のギアとの接触時にどうしても見た目の部分で問題が発生してしまった。
下の画像のように条件によってはギアのかみ合いがズレるということがわかった。



解決方法として、ギアや歯付き足場とは1つのみしか接触できないようにすることです。
なんとか2つ以上との接触をできるように処理を組んでみたのですが、かみ合い自体は解決できました。
ただし、かみ合いの修正をすることで意図しない影響をステージに与えてしまった。
結果、相談の末に１つのみしか接触できないようにする仕様に変更してゲームを制作しました。

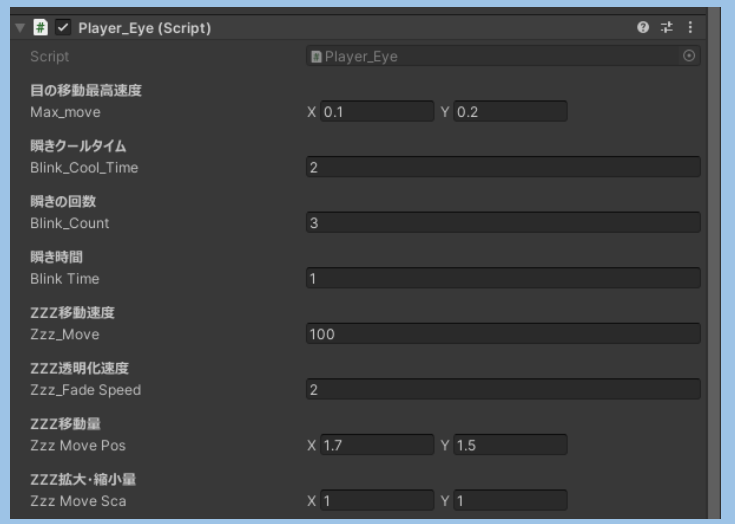
作品：ギアガチャン

2．プレイヤーの移動処理について

移動をするにあたって、Rigidbodyがかみ合い処理と干渉してしまいました。
そのため、移動時はRigidbodyを使わないようにして干渉問題を防ぐことができました。
他にも、移動時の挙動として以下の二つの違いがこのゲームでは発生しました。
・ギア上での移動
・足場上での移動
この二つを実装するにあたって、親子関係を利用してローカル座標を変更しながら実装していました。

移動以外にもジャンプ時の挙動を実装するにあたって、はじめは現存のゲームで使用されているような動きをできるようにしていました。
しかし、ギアがジャンプする際には重みが感じられませんでした。
このことから、ギアの落下処理に変更をかける必要があるのかもしれないと思い、取り組んでみました。
結果、ギアという重みを感じつつ、サウンドにもマッチするかみ合うようなものに仕上げることができました。

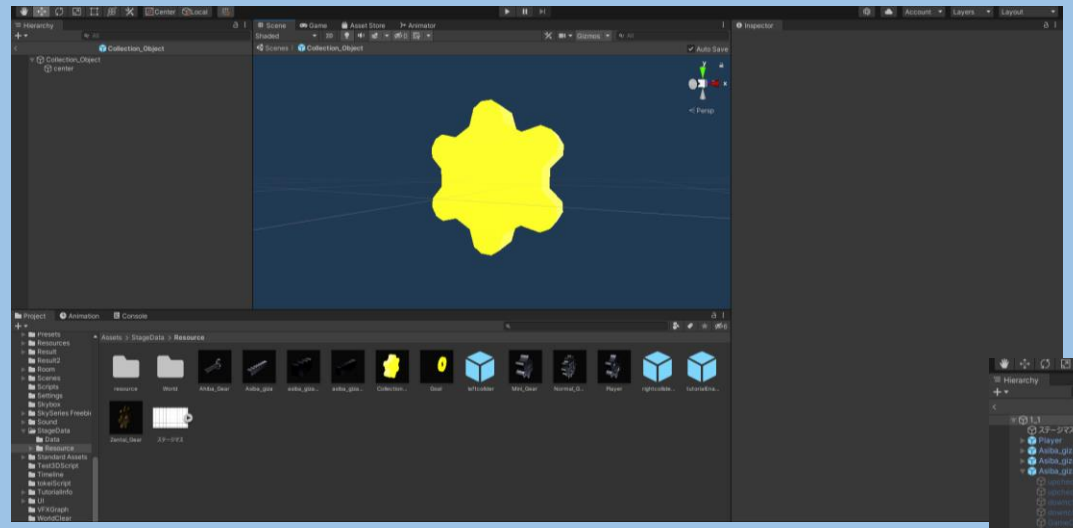
表情の処理については、待機時間や瞬きの速さなど調整が必要になる場所が多くあったので、
Inspectorで誰でも変更ができるようになっています。

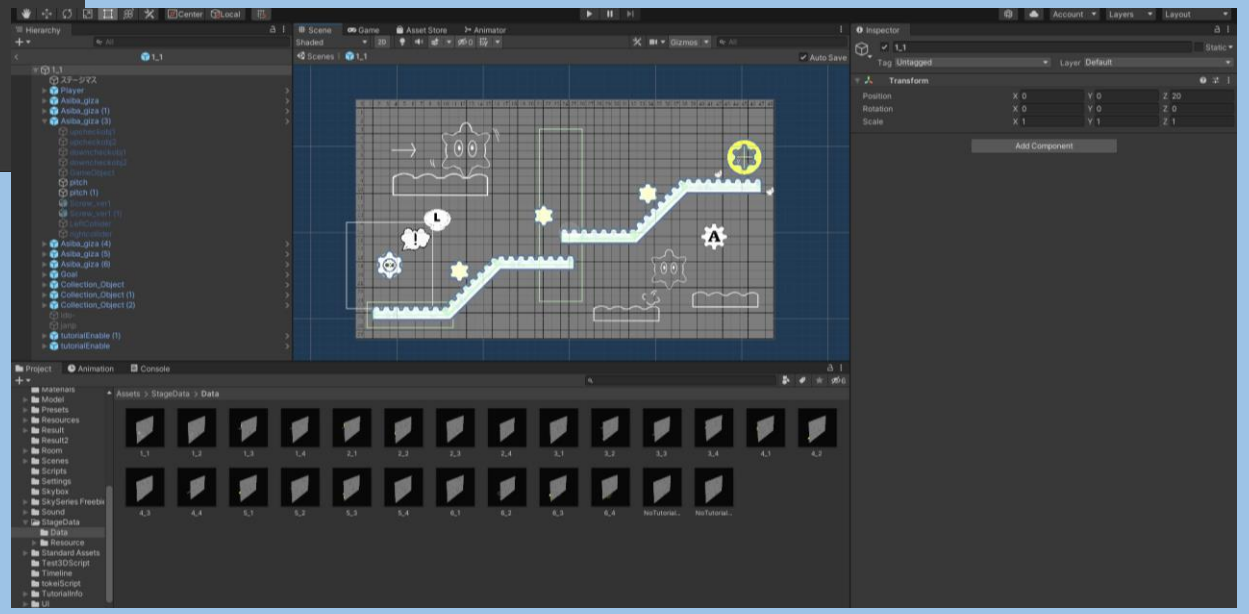


作品：ギアガチャン

３．その他の担当した点について

・セーブデータを実装する予定があったので、csvでのデータの読み書きを行うためにベース部分を制作しました。
・プログラム面以外にも、企画やデザインにも一部関わらせてもらいました。





作品：お兄ちゃん大好き

チーム作品
お兄ちゃん大好き

作品概要

ステージ上を移動する「妹」を守りながら、敵を倒していくゲームとなっています。
フィーバーモードやエフェクトなどで演出面にこだわりを込めた作品です。

制作について

- 使用ツール
  Visual Studio
  DirectX11
  GitHub
- 制作期間
  3ヶ月
- 制作人数
  7人
- ジャンル
  タワーディフェンスゲーム
- 担当箇所
  シーン遷移
  入力処理
  サウンド処理
  ゲーム内演出処理

動画リンク

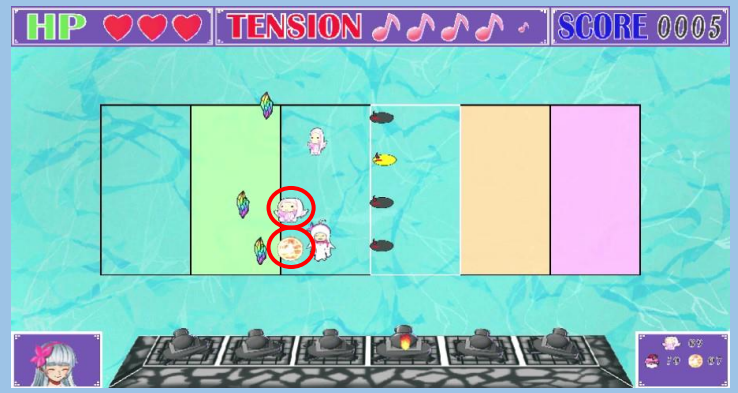
作品：お兄ちゃん大好き

１．入力処理について

制作にあたってDirectInputとXinputのどちらかにするべきか考えました。
このゲームは学内コンテスト向けに制作を開始したため、規約がいくつかありました。
その中に、「審査する際のコントローラーはXboxコントローラーです。」と記述されていたため、
Xinputを使用した入力処理を制作しました。
しかし、コントローラーがない際にデバックができなかったり、審査員にコントローラーを持っていない人もいる等
コントローラーだけでは少し制作や審査に問題が出るのではと思いました。
そこで、**コントローラー入力をXinput**に、**キーボード入力をDirectInput**にするという使い分けを行いました。
これにより、制作時のデバックがやりやすくなり手軽にプレイもできるようになりました。

２．ゲーム内演出処理について

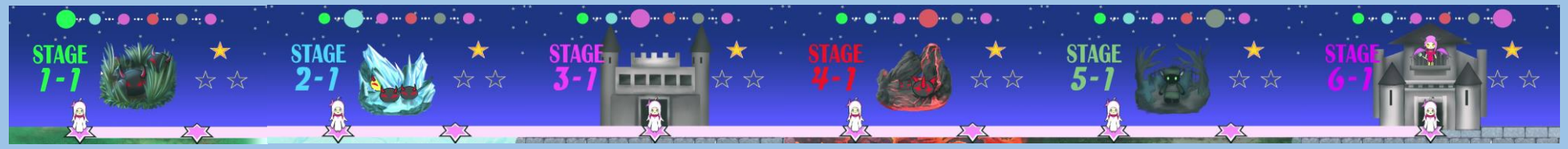
ゲーム内において、弾やスコアを回収する際に演出を加えました。
アイテムがドロップした場所から弾、スコアどちらに対して移動量を計算する。
移動完了時点に到達するまでに消滅したように表現するため、サイズを小さくしていくようにしている。
このサイズ変更をすることでアイテムが、吸収されているような表現もできた。
サイズ変更という要素がアイテムの消滅と吸収の二つの表現をできたので、
このゲームで重要な演出の一つとなっている。



作品：お兄ちゃん大好き

３．ステージ選択について

このゲームではステージが複数存在するため、ステージ選択画面を制作しました。
しかし、このゲームではデータを全て読み込んでしまうとメモリの使用量が1GBを軽く超えていました。
そのため、影響によりステージ選択では全ての画像データを読み込むのが危険なため必要最低限のデータのみを読み込んでいます。



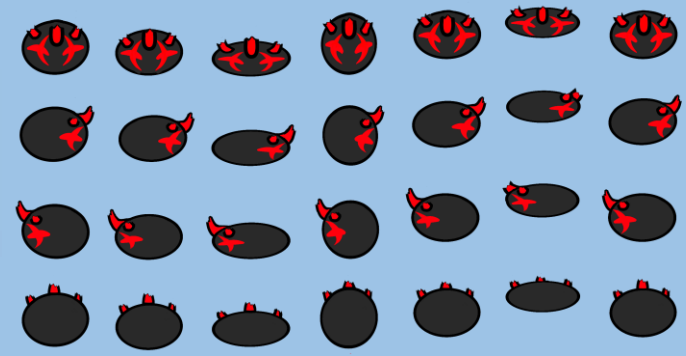
４．シーン遷移について

シーンマネージャーを制作し、シーンを簡単に管理できるようにしました。
シーンは他のメンバーが制作する可能性があった為、事前にベースとなるシートを制作し、それを継承してのシーン制作を行った。
この継承してのシーン管理により、制作終盤で追加された説明用シーンなども問題なく実装することができました。

作品：お兄ちゃん大好き

５．その他の担当した点について

・ゲーム全体のデザイン系の世界観管理
・敵キャラクターのデザイン(一部)
・UI面のデザイン(一部)
・セーブデータ系の制作
・フェード処理の制作
・ノートシーンの制作
・サウンド再生などのサウンド関連の処理制作



```cpp
#pragma once
#include "GameObject.h"
enum FADE_TYPE {
    FADE_FADINGorDARKENING,//暗転明転タイプ
    FADE_SCALE,           //拡大縮小タイプ
    FADE_IN,
    FADE_OUT,
    FADE_SCALE_UP,
    FADE_SCALE_DOWN,
    FADE_NULL,
};
class Fade :public GameObject
{
public:
    Fade();
    void Update();
    void Release();
    float Get_Color_a() { return this->mColor.a; }
    FADE_TYPE Get_NowFade() { return this->now_fade; }
    void Set_NowFade(FADE_TYPE type) { this->now_fade = type; }
    bool Get_Switching_flag() { return Switching_flag; }
    void Set_Switching(bool n) { this->Switching_flag = n; }
    bool Get_Finish_flag() { return this->Finish_flag; }
    void Set_Finish_flag(bool n) { this->Finish_flag = n; }
    //こいつらがメインで動いてもらう
    void Fade_Process();
    void Scale_Process();
private:
    void Fade_Intialize();  //暗転明転初期化
    void Scale_Initialize();//拡大縮小初期化
    void Fade_In();         //明転処理
    void Fade_Out();        //暗転処理
    void Scale_Up();        //スケール拡大処理
    void Scale_Down();      //スケール縮小処理
    float Fade_Speed;       //スピード変数
    FADE_TYPE now_fade;
    std::string Tex_Data[2];
    bool Initialize_flag = true;//初期化フラグ(Fadeの画像データ更新)
    bool Switching_flag;        //更新フラグ(シーンの画像データ更新)
    bool Finish_flag;           //終了フラグ
};
```

```cpp
DX22_00                                                    Sound
// パラメータ構造体
typedef struct
{
    LPCSTR filename;    // 音声ファイルまでのパスを設定
    bool bLoop;         // trueでループ。通常BGMはture、SEはfalse。
} PARAM;

class Sound:public singleton<Sound>
{
public:
    explicit Sound(token);
    ~Sound();
    void Set_Scene(Scene_Transition scenename);
    void Play(SOUND label);  //音声の開始(0から)
    void Stop(SOUND label);   //音声の停止
    void Pause(SOUND label);  //音声の一時停止
    void Restart(SOUND label);//音声の途中再生
    void Set_Volume(SOUND label,float volume);
    void Release(Scene_Transition scenename);//解放
private:
    HRESULT FindChunk(HANDLE, DWORD, DWORD&, DWORD&);
    HRESULT ReadChunkData(HANDLE, void*, DWORD, DWORD);

    //PARAM g_param[SOUND_LABEL_MAX];

    IXAudio2* g_pXAudio2 = NULL;
    IXAudio2MasteringVoice* g_pMasteringVoice = NULL;
    std::unordered_map<SOUND,IXAudio2SourceVoice*> g_pSourceVoice;

    std::vector<PARAM> TitleSound;
    std::vector<PARAM> StageSelectSound;
    std::vector<PARAM> GameSceneSound;
    std::vector<PARAM> ResultSound;
    std::vector<PARAM> StorySound;
    std::vector<PARAM> OptionSound;

    std::unordered_map<SOUND,WAVEFORMATEXTENSIBLE>g_wfx;          // WAVフォーマット
    std::unordered_map<SOUND,XAUDIO2_BUFFER>    g_buffer;
    std::unordered_map<SOUND, BYTE*> g_DataBuffer;
};

#endif
```

作品：タクシディ

個人制作
タクシディ

作品概要

機体を選択し、特殊能力を使いながら敵を倒すシューティングゲームです。
使用言語がJavaScriptと今までに触ったことのない言語で制作しました。

制作について

- 使用ツール
  Phaser3
  JavaScript
  HTML5
  Git
  Visual Studio Code
  Heroku
- 制作期間
  1ヶ月
- ジャンル
  縦スクロールシューティングゲーム

動画リンク

作品：タクシディ

１．制作について

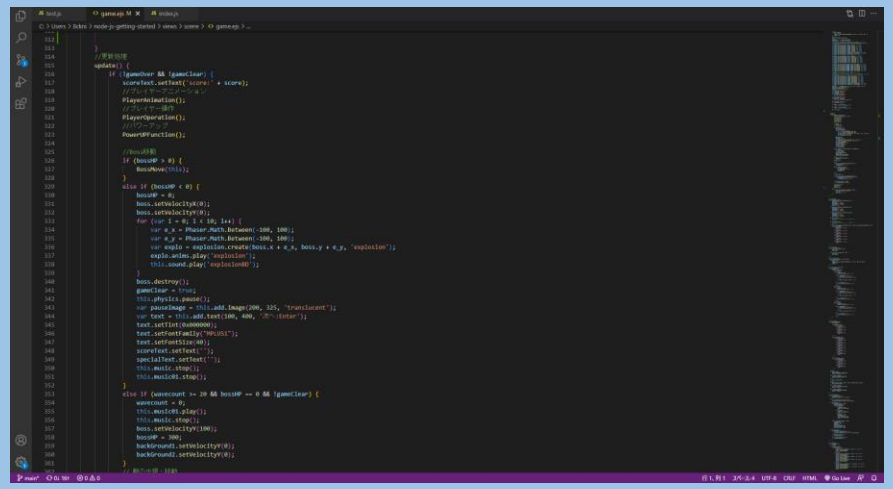
この作品は、過去に制作してきたシューティングゲームを新しい言語で制作したものです。
前回制作したシューティングゲームは完成しなかったため、この作品ではその点をクリアするために制作を開始しました。

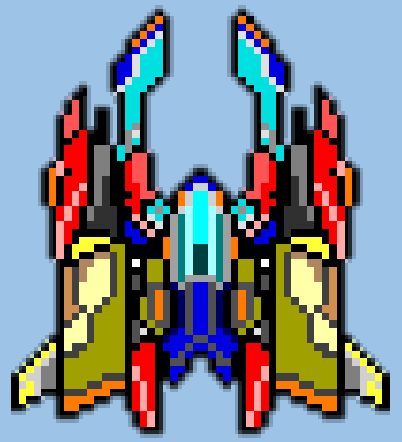
２．画像について

画像は前回のものを再利用しようと思っていましたが、新しい要素などを追加していくことが多くなり
再利用するのが難しくなってしまいました。そこで新たに画像を制作し、前回以上の画像データを使用することとなりました。
敵についても種類が必要になるため困っていましたが、色違いを増やすことで種類を増やすことができました。

３．プログラムについて

今回の制作では、Phaser3というフレームワークを利用して制作を行いました。
さらにこのPhaser3はJavaScriptを使用するため、JavaScriptを学ぶ必要がありました。
わからないことしかない中での制作となりましたが、それに見合った知識を得ることができました。





作品：タクシディ

4．機体選択について

このゲームの独自要素の一つ、機体選択はゲームプレイ前に事前に自分が使用する機体を選ぶことができるものです。
この要素は過去の作品からの流用したものとなっていますが、一部変更した箇所がありました。
それは、機体選択の幅を広げたことです。
元々は、ベース機体を変更することしかできず、さらには最終機体が同じになってしまうという個性のないものとなっていました。
そこで各ベース機体からの派生機体を作成し、機体選択の幅を持たせることに成功しました。

5．弾について

ゲーム内でプレイヤー・敵両者が使用するものが攻撃時の弾です。
処理的には、使用者の座標に生成し、移動するだけのシンプルなものとなっています。
しかし、画像データは一つのものをRBG値を変更することで弾に違いを持たせることができました。
敵の弾に関してはRBG値を変更したものの、大きな違いを持たせるため、新規で画像を制作することとなりました。

作品：タクシディ

５．その他

作品の細かい仕様を説明については下記の資料を参照していただけると幸いです。

最後まで見ていただき、

ありがとうございました。