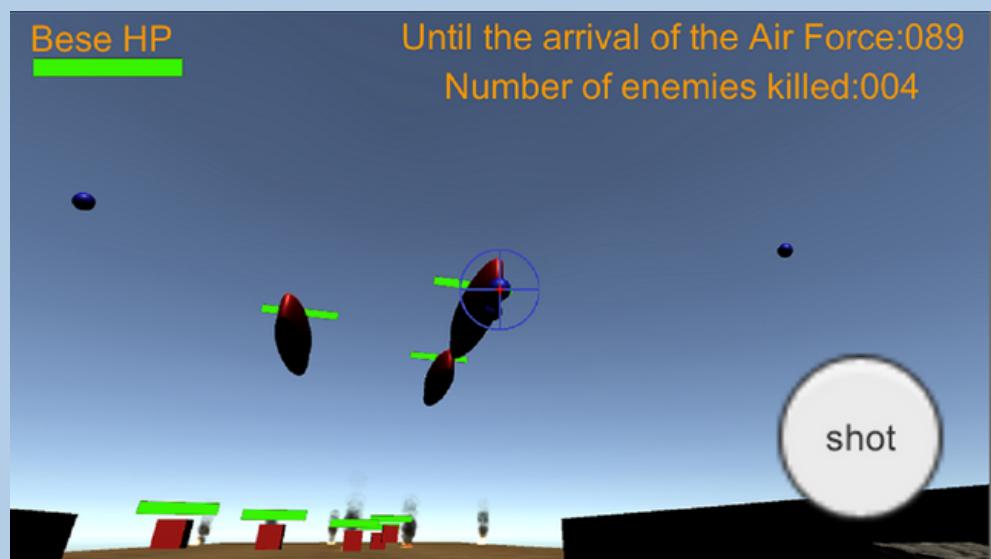


Portforio



総合学園ヒューマンアカデミー仙台校
ゲームカレッジ プログラマー専攻
最終更新日2021年4月7日 **高橋悠夢**

プロフィール

名前	高橋 悠夢
性別	男
年齢	17歳
出身地	宮城県仙台市
希望職種	ゲームプログラマー
得意科目	歴史、国語
趣味	ゲーム、アニメ、戦史



自己紹介

FPS, TPS、戦略, 歴史シミュレーションや
アクションなど幅広くいろいろなジャンルの
ゲームを遊んでいます。（ホラー以外）

使用ツール



Microsoft VisualStudio 2019,Visual StudioCode,
Unity,Github,SourceTree



slack,Backlog,Chatwork,XAMPP

開発言語



C/C++,C#,

少しだけ触ったツール、開発言語



Unreal Engine,JavaScript,Python

作品紹介

一作品目Urban defense

制作ツール Unity

5ページ～8ページ

二作品目DIRTY WORK

制作ツール Visual Studio 2019

9ページ～15ページ

三作品目Base defense

制作ツール Unity

16ページ～20ページ

四作品目Bat destroyer

制作ツール Visual Studio 2019

21ページ～26ページ

五作品目Desert town

制作ツール Unity

?ページ～?ページ

六作品目Wisdom World

制作ツール Unity

27ページ～31ページ

七作品目DIRTY WORK

DirectX 12リメイク

制作ツール Visual Studio 2019

?ページ～?ページ

八作品目ゲーム大賞向け

制作ツール Visual Studio 2019

?ページ～?ページ

Urban defense —

作品目



プレイヤー人数
1人

ジャンル	2Dシューティング
制作期間	一ヶ月
制作人数	自分(個人制作)
開発言語	C++,C#
制作環境	Unity
対象機種	Androidスマートフォン
担当箇所	作品全般

ゲームの概要

戦闘機を上下に操作して、敵の攻撃を避けながら攻撃を当ててBossのHPを削りきって、クリアを目指すゲームです。

ゲームのルール

- ・敵に当たったり、敵からの攻撃を受けると即死。
- ・Bossから出現する敵を倒すとスコアが貢える。
- ・自機の攻撃（ビーム）は一定間隔で自動で発射される。
- ・できるだけスコアを稼いで、BossのHPを削りきるとゲームクリア。

こだわった点



自機の攻撃をビームのように見せるため
Trail Rendererをつけてビームっぽくなるように
演出した。



数字と
バーでHP
←を表示。

←点滅して
いる。

Bossに攻撃(ビーム)が当たったことが分かるように
1、当たった時はBossが点滅するようにした。
2、Bossの上にHPバーをつけた。
3、SEを出した。



スマホで操作しやすい
ように、画面の上側を
タップすると上へ、
下側をタップすると
下へ行くようにした。

苦労した点

初めての作品制作だったのでまだ初心者の自分が、一ヶ月の期間で作れて、遊んで楽しいと思えるゲームのアイデアを出すのに少し手間取りました。制作するゲームの仕様などを決めて、いざ作業に入ると分からないことが多く、さらにバグが多発してしまいバグ修正に苦労しました。

問題点

分からないところなどはインターネットなどで調べて参考にしていたためプログラムの書き方が統一できず修正時にどこが正常に動いていないのか、このプログラムはなにをやっているかなどが分からなくなっていたため、バグの修正に時間が掛かってしまいました。

解決方法

このプログラムが何をやっているのかなどを作成時にコメントを残すこと、インターネットで調べたプログラムは自分の分かりやすいよう、一部を書き換えたり、変数の命名形式を統一させたり、そのプログラムの処理を理解してから書くようにすることによってバグ修正にかかっていた時間を少なくすることが出来ました。

DIRTY WORK

二 作 品 目



プレイヤー人数 1人

ジャンル	2Dアクション	
制作期間	一ヶ月	
制作人数	プランナー	1人
	デザイナー	1人
	プログラマー	1人
開発言語	C++(DirectX)	
制作環境	Visual Studio 2019	
対象機種	Windows PC	
担当箇所	プログラム全般、 ゲームのアイデア出し。	

ゲームの概要

車を上下に操作して、右側から出てくる車を規定タイムの間避け切って、追跡してくる警察から逃げ切るゲームです。

ゲームのルール

- ・スコア(\$)がライフの役割もしている。
- ・警察車両に当たるとゲームオーバー。
- ・一般車両に当たるとライフが減る。
- ・ライフが0になるとゲームオーバー。
- ・できるだけスコア（ライフ）を残して、規定タイム間避け切るとゲームクリア。

こだわった点



パトカーのサイレンを3Dサウンドで流すことによってより追跡されている臨場感をプレイヤーに実感できるようにしている。



検問のパトカーの位置が事前に分かるようないいマークと注意表示を出してプレイヤーが検問のパトカーを避けられるように、ぱっと見て分かるようにした。



車の移動はこのゲームの重要なところなので徐々に移動できるようにして、慣性のある動きにすることで車に近い操作感にしたかったので移動キーを押している間徐々に数値を足していくその数値で移動させるようにして別方向へ移動しようとしたときはその数値を徐々に減らしていくことで移動できるようにした。

```

628 void GameMain::プレイヤー処理() {
629
630     if (プレイヤー_state == 0) {
631
632         if (Keystate.IsKeyDown(Keys_W) || pad.Buttons[3] != 0) {
633             スピード_y = スピード_y - 0.2 - (スピード加速 / 1000);
634
635             if (スピード_y < -5) {
636                 スピード_y = -5;
637             }
638
639             if (!上下移動SE->IsPlaying())
640                 上下移動SE->Play();
641
642             プレイヤー向き_state = 1;
643         }
644
645         if (Keystate.IsKeyDown(Keys_S) || pad.Buttons[0] != 0) {
646             スピード_y = スピード_y + 0.2 + (スピード加速 / 1000);
647
648             if (スピード_y > 5) {
649                 スピード_y = 5;
650             }
651
652             if (!上下移動SE2->IsPlaying())
653                 上下移動SE2->Play();
654
655             プレイヤー向き_state = 2;
656         }
657
658         if (Keystate.IsKeyUp(Keys_W) && pad.Buttons[0] == 0 && Keystate.IsKeyUp(Keys_S) && pad.Buttons[3] == 0) {
659             if (スピード_y > 0) {
660                 スピード_y = スピード_y - 0.1;
661             }
662
663             if (スピード_y < 0) {
664                 スピード_y = スピード_y + 0.1;
665             }
666
667         }
668     }
669 }
```

苦労した点

二作目で多少プログラムの知識が増えてきたので、一作品目よりも効率的にプログラムを書くことが出来るようになり、新たに覚えた処理なども書いていたためにプログラムが肥大化し、バグ修正、一部のゲームのアイデアの変更時にそのプログラムがどの処理を担当しているかが分からなくなってしまいました。

問題点

新しいアイデアが追加されるたびにプログラムを書き足していたし、提出の期限などがあったのでスピードを重視してしまっていたので余裕がなくコメントなどを残すことなどがあろそかになっていたのが原因でした。

解決方法

関数を使いプレイヤー関係のコード、タイトルの処理、クリア画面の処理ごとに大雑把ではありましたがあわせてまとめることでコードの解読性がよくなり、改善することに成功しました。

日本語の変数

この作品がVisual Studio 2019を使った最初の作品だったのですがこの頃の自分は日本語の変数を使って作成していました。

そこで日本語での変数にどういうメリット、デメリットがあったのかを書いてみたいと思います。

○メリット

- ・プログラムが理解しやすい。
 - ・コードを流し読みして、なんとなく処理の流れをつかむことができる。
 - ・プログラムの上達がしやすい。
- ・英語の解読、英訳に悩んだり、辞書を引く時間が無くなる。

○デメリット

- ・コードが打ちにくい
 - ・ifやforなどは英語で書かないといけないので日本語にして、これはひらがな、これは漢字、カタカナに変換、英語で書いて...などの入力切替、変換をやらないといけないのでコードが打ちずらい。
- ・英語の変数のほうが慣れている人とチーム制作をすると相手が困惑する。
- ・日本語が分からない人との制作ができない。
- ・日本語を使用することによってバグが発生する可能性がある。

- ・日本語の変数を使っている人は少数派。
- ・個人的な感想ですが日本語を使っている人を見たことがないため。
- ・英語の変数名に慣れている人は何かきっかけがないと日本語に変えようとしない。
- ・そのため、日本語を使う人の数が増えない。

○自分は現在どっちを使っているか？

個人的には日本語を使った方が分かりやすいので授業など、自分しか見ない、書かないときは日本語をつかって、チーム制作などでは英語を使っています。

使用している日本語変数の例

Initialize ↓

```

フォント = GraphicsDevice.CreateSpriteFont(T("MSゴシック"), 50);
攻撃SE = SoundDevice.CreateSoundFromFile(T("攻撃SE.wav"));

自攻撃力      = 200;
敵守備力      = 100;
基礎ダメージ  = 0;
ダメージ_state = 0;
ゆらぎ        = 0;
総合ダメージ  = 0;
攻撃回数      = 0;

//はやぶさの剣-----
一回目ダメージ = 0;
二回目ダメージ = 0;
//-----

/////フィッシャー・イエーツのシャッフル-----
ランダム      = 0;
スペースキー回数 = 0;

for (int i = 0; i < ランダム数; i++) {
    ナンバー[i] = i + 1;
}

for (int i = 0; i < ランダム数; i++) {
    int シャッフル      = MathHelper.Random(0, ランダム数 - 1);
    int シャッフル2     = ナンバー[i];
    ナンバー[i]         = ナンバー[シャッフル];
    ナンバー[シャッフル] = シャッフル2;
}
//-----

```

は
や
ぶ
さ
の
剣、
フ
イ
イ
ツ
シ
ヤ
エ
ツ
ツ
フ
ツ
シ
フ
ル
の
ヤ
ー

```

KeyboardState Keystate = Keyboard->GetState();
KeyboardBuffer Key_buffer = Keyboard->GetBuffer();

//if (Key_buffer.IsPressed(Keys_Space)) {
//  基礎ダメージ = 自攻撃力 / 2 - 敵守備力 / 4;
//  ゆらぎ = 基礎ダメージ / 16 + 1;
//  //総合ダメージ = 基礎ダメージ + MathHelper_Random(-ゆらぎ, ゆらぎ);

//  //はやぶさの剣-----
//  一回目ダメージ = 基礎ダメージ + MathHelper_Random(-ゆらぎ, ゆらぎ)
//  二回目ダメージ = 基礎ダメージ + MathHelper_Random(-ゆらぎ, ゆらぎ)
//  総合ダメージ = (一回目ダメージ + 二回目ダメージ) * 0.75;
//  //-----

//  ダメージ_state = 1;
//  攻撃回数 += 1;
//  攻撃SE->Play();
//}

//フィッシャー・イエーツのシャッフル-----
if (Key_buffer.IsPressed(Keys_Space)) {
  ランダム = ナンバー[スペースキー回数];
  スペースキー回数++;
}
//-----

```

Update ↑

Draw ↓

```

SpriteBatch.DrawString(フォント, Vector2(0, 0), Color(0, 0, 0), _T("攻撃回数:%d"), 攻撃回数);

if (ダメージ_state == 1) {
  SpriteBatch.DrawString(フォント, Vector2(0, 50), Color(0, 255, 0), _T("総合ダメージ:%d"), 総合ダメージ);
}

SpriteBatch.DrawString(フォント, Vector2(0, 100), Color(0, 0, 255), _T("自攻撃力:%d"), 自攻撃力);
SpriteBatch.DrawString(フォント, Vector2(0, 150), Color(0, 0, 255), _T("基礎ダメージ:%d"), 基礎ダメージ);

//はやぶさの剣-----
SpriteBatch.DrawString(フォント, Vector2(0, 200), Color(0, 0, 255), _T("一回目ダメージ:%d"), 一回目ダメージ);
SpriteBatch.DrawString(フォント, Vector2(0, 250), Color(0, 0, 255), _T("二回目ダメージ:%d"), 二回目ダメージ);
SpriteBatch.DrawString(フォント, Vector2(0, 300), Color(255, 0, 0), _T("敵守備力:%d"), 敵守備力);
//-----

//フィッシャー・イエーツのシャッフル-----
SpriteBatch.DrawString(フォント, Vector2(0, 350), Color(0, 0, 0), _T("ランダム:%d"), ランダム);
SpriteBatch.DrawString(フォント, Vector2(0, 400), Color(0, 0, 0), _T("スペースキーを押した回数:%d"), スペースキー回数);

if (ランダム == 1) {
  SpriteBatch.DrawString(フォント, Vector2(0, 450), Color(255, 255, 0), _T("SR"));
}
else if (ランダム == 77) {
  SpriteBatch.DrawString(フォント, Vector2(0, 450), Color(255, 255, 0), _T("SSR"));
}
else if (ランダム == 100) {
  SpriteBatch.DrawString(フォント, Vector2(0, 450), Color(255, 255, 0), _T("UR"));
}
else {
  SpriteBatch.DrawString(フォント, Vector2(0, 450), Color(128, 128, 128), _T("HR"));
}
//-----

```

作品紹介三作品目



プレイヤー人数
1人

ジャンル	3Dシューティング
制作期間	一か月
制作人数	自分(個人制作)
開発言語	C++,C#
制作環境	Unity
対象機種	Androidスマートフォン
担当箇所	作品全般

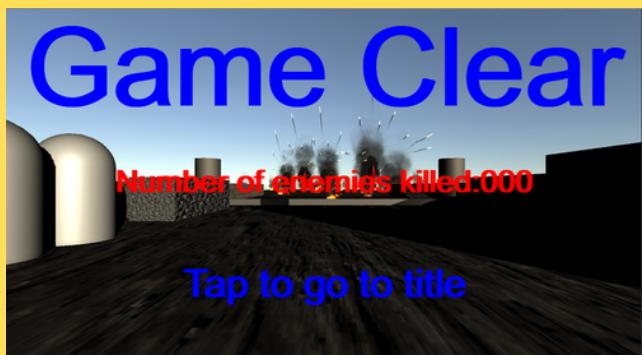
ゲームの概要

奥からくる敵を自機の視点を操作して、
弾を撃って敵を倒し味方と協力しながら
規定タイムの間基地を守り切るゲームです。

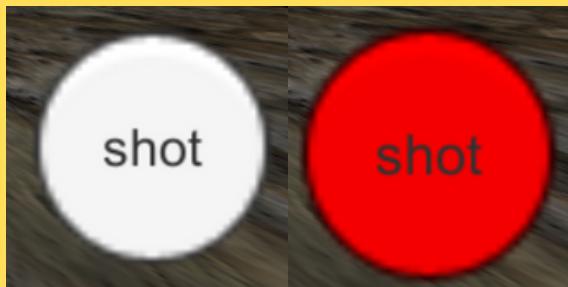
ゲームのルール

- ・ 基地のHPが0になるとゲームオーバー。
- ・ 地上と上空を進んでくる2種類の敵がいる。
- ・ 敵が基地内に侵入すると基地のHPが減る。
- ・ 自機以外に敵を倒してくれる味方が2機いる。
- ・ 味方と協力しながら
規定タイム間避け切るとゲームクリア。

こだわった点



ゲームのテーマに合うようにゲームクリア、ゲームオーバーシーンを作成し、エフェクトなども使って演出を豪華にした。さらに文字を複数重ねて表示することで立体的に見えるようにしている。



shotボタンを押しているか押していないかを分かりやすくするために、押していないときは白、押しているときは赤、で表示している。



エフェクトを敵の撃破時に演出することで迫力を上げた。

一
番
近
く
の
敵
を
追
尾
す
る
味
方
が
一
番
近
い
敵
を

```
//初期位置をpositionに格納
position = transform.position;
//rigidbody取得
rigid = this.GetComponent<Rigidbody>();
// 初速をランダムで与える
velocity = new Vector3(Random.Range(-5.0f, 5.0f), Random.Range(-5.0f, 5.0f), 0);
target = GameObject.FindGameObjectWithTag("teki");

Unity メッセージ10 個の参照
void Update()

acceleration = Vector3.zero;

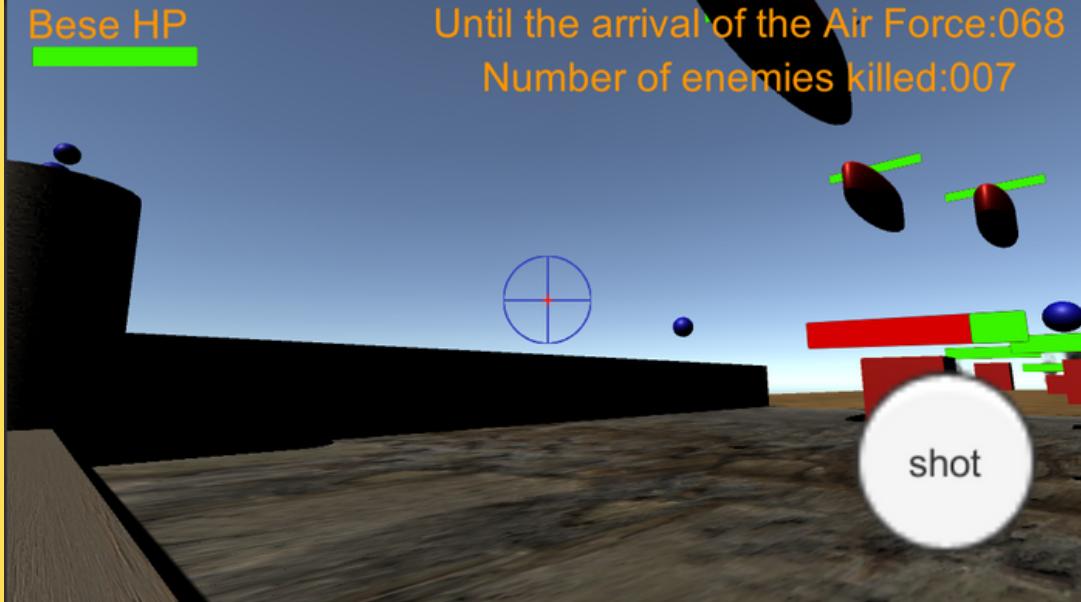
//ターゲットと自分自身の差
var diff = target.transform.position - transform.position;

//加速度を求めてるらしい
acceleration += (diff - velocity * period) * 2f
    / (period * period);

//加速度が一定以上だと追尾を弱くする
if (acceleration.magnitude > 100f)
{
    acceleration = acceleration.normalized * 100f;
}

//着弾時間を徐々に減らしていく
period -= Time.deltaTime;

//速度の計算
velocity += acceleration * Time.deltaTime;
```



基地を守っていることをプレイヤーに
実感させるために、味方のNPCを追加して
射撃する弾に一番近くの敵を追尾し、攻撃する
ソースコードを作成した。
これにより基地を守っているという実感が
より沸くようになった。

苦労した点

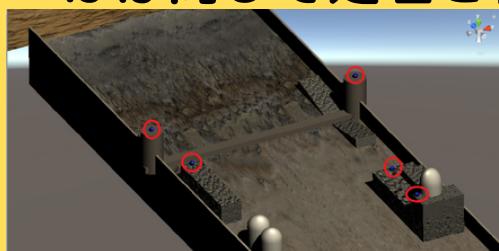
Unity上では正常に動いているものがスマホでは正常に動かないことが多く実機検証にだいぶ苦労しました。特にプレイヤー、味方の射撃スピード、敵の出現スピードがスマホの場合異常に早くなってしましました。そのため今まで数値を直で加算していたところ全てをTime.deltaTimeに変更しましたが、それでも正常に動かず、解決しませんでした。

問題点

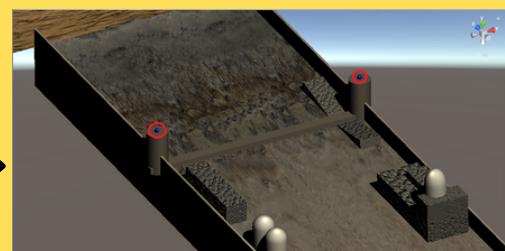
実機検証などのデバックなどをやりながら問題点を調べたところ、スマホの処理能力を超えており処理が重くなってしまっていることが正常に動かない原因だと判明しました。

解決方法

- ・敵の出現数を減らし、敵のHPを上げることで難易度はほぼ同じで処理を軽くした。
- ・味方の数を減らし、弾のダメージを上げることによって味方、弾の数を減らしつつ、難易度はほぼ同じで処理を軽くした。



←改善前
改善後→



四作品目

プレイヤー人數
1人



ジャンル	2Dアクションゲーム	
制作期間	一ヶ月	
制作人数	プランナー	1人
	デザイナー	2人
	プログラマー	2人
開発言語	C++(DirectX)	
制作環境	Visual Studio 2019	
対象機種	Windows PC	
担当箇所	タイトル画面及び リザルト画面の全般。 メイン画面の破壊対象物の 処理全般、 ゲームのアイデア出し。	

ゲームの概要

リサイクル工場の解体職人である主人公が
飛んでくる破壊対象物をタイミングよくバットを
振って物を粉々にすることが目的のゲーム。

ゲームのルール

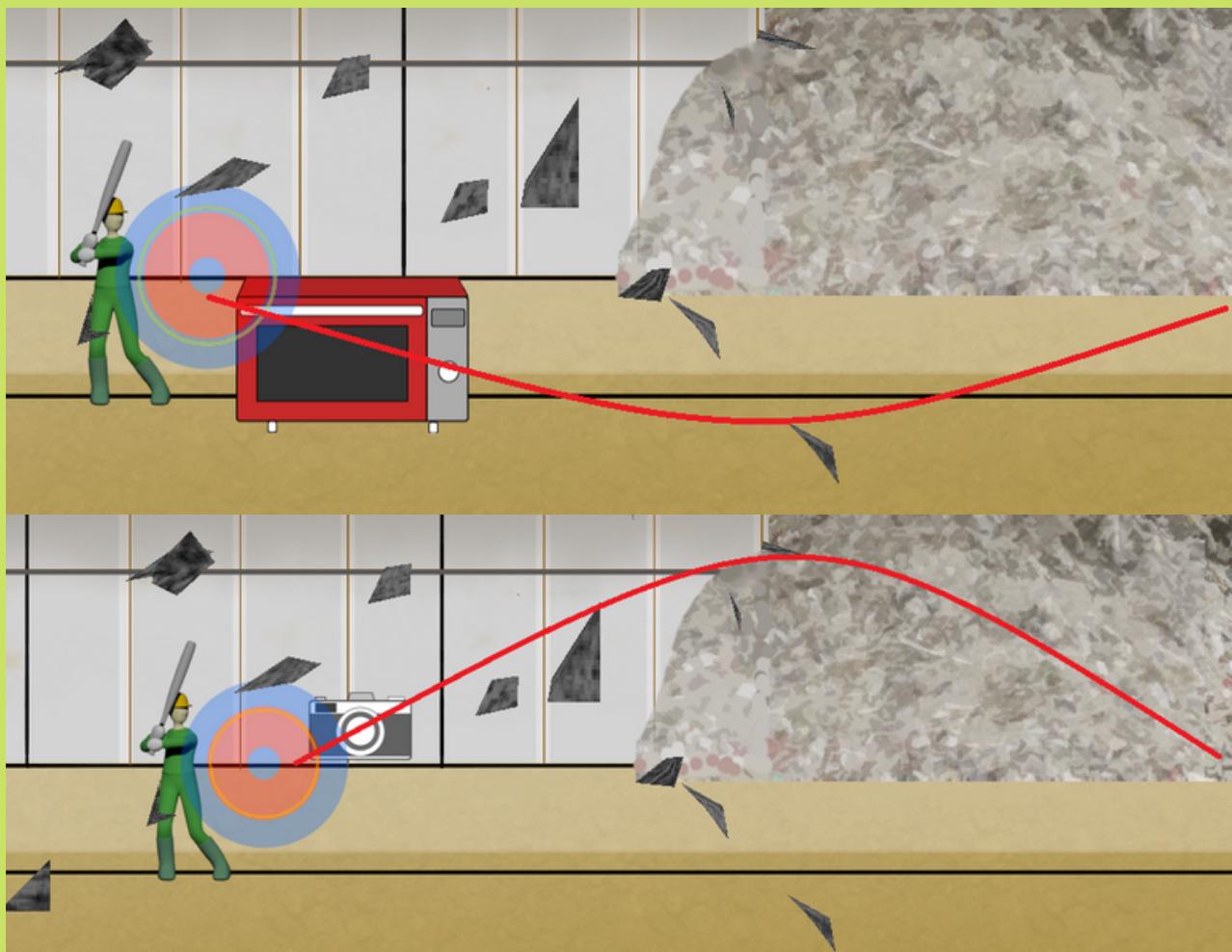
- ・破壊対象物を粉々にするとスコアが増える。
- ・増えるスコアはタイミングが正確なほど高スコア
が貰える。
- ・できるだけコンボをつなげながらスコアを稼ぎ、
クリアしてハイスコアを更新しよう！

こだわった点



フォントを使わずにデザイナーさんに作成して
もらった1～9の画像を使ってパーカクト数や、
スコアを表示することでかっこよくしている。
Rectで表示する画像の場所を指定して
表示させている↓

```
257 //score
258     if (score_state == 0) {
259         int ten_thousand = GameMain::score / 10000;
260         time_x = (ten_thousand % 5) * figure_x;
261         spriteBatch.Draw(#figure, Vector3(score_x, score_y, -1.0f), RectWH(time_x, 0, figure_x, figure_x), 1.0f, Vector3(0, 0, roll), Vector3(50, 50, 0), Vector2(b_big, b_big));// 万の位
262
263         int fifteen_hundred = GameMain::score / 1000;
264         time_x = (fifteen_hundred % 5) * figure_x;
265         spriteBatch.Draw(#figure, Vector3(score_x2, score_y2, -1.0f), RectWH(time_x, 0, figure_x, figure_x), 1.0f, Vector3(0, 0, roll), Vector3(50, 50, 0), Vector2(b_big, b_big));// 千の位
266
267         int hundreds_place = GameMain::score / 100;
268         time_x = (hundreds_place % 5) * figure_x;
269         spriteBatch.Draw(#figure, Vector3(score_x3, score_y3, -1.0f), RectWH(time_x, 0, figure_x, figure_x), 1.0f, Vector3(0, 0, roll), Vector3(50, 50, 0), Vector2(b_big, b_big));// 百の位
270
271         int tens_place = GameMain::score / 10;
272         time_x = (tens_place % 5) * figure_x;
273         spriteBatch.Draw(#figure, Vector3(score_x4, score_y4, -1.0f), RectWH(time_x, 0, figure_x, figure_x), 1.0f, Vector3(0, 0, roll), Vector3(50, 50, 0), Vector2(b_big, b_big));// 十の位
274
275         int ones_place = GameMain::score % 10;
276         time_x = (ones_place % 5) * figure_x;
277         spriteBatch.Draw(#figure, Vector3(score_x5, score_y5, -1.0f), RectWH(time_x, 0, figure_x, figure_x), 1.0f, Vector3(0, 0, roll), Vector3(50, 50, 0), Vector2(b_big, b_big));// 一の位
278
279     //perfect
280     int time_x2;
281     int tens_place2 = GameMain::perfect_ / 10;
282     time_x2 = (tens_place2 % 5) * figure_x;
283     spriteBatch.Draw(#figure, Vector3(450.0f, 95.0f, -1.0f), RectWH(time_x2, 0, figure_x, figure_x), 1.0f, Vector3(0, 0, 0), Vector3(50, 50, 0), Vector2(b_big, b_big));// 十の位
284
285     int ones_place2 = GameMain::perfect_ % 10;
286     time_x2 = (ones_place2 % 5) * figure_x;
287     spriteBatch.Draw(#figure, Vector3(500.0f, 95.0f, -1.0f), RectWH(time_x2, 0, figure_x, figure_x), 1.0f, Vector3(0, 0, 0), Vector3(50, 50, 0), Vector2(b_big, b_big));// 一の位
288
289     //good_
290     int time_x3;
291     int tens_place3 = GameMain::good_ / 10;
292     time_x3 = (tens_place3 % 5) * figure_x;
293     spriteBatch.Draw(#figure, Vector3(400.0f, 180.0f, -1.0f), RectWH(time_x3, 0, figure_x, figure_x), 1.0f, Vector3(0, 0, 0), Vector3(50, 50, 0), Vector2(b_big, b_big));// 十の位
294
295     int ones_place3 = GameMain::good_ % 10;
296     time_x3 = (ones_place3 % 5) * figure_x;
297     spriteBatch.Draw(#figure, Vector3(450.0f, 180.0f, -1.0f), RectWH(time_x3, 0, figure_x, figure_x), 1.0f, Vector3(0, 0, 0), Vector3(50, 50, 0), Vector2(b_big, b_big));// 一の位
298
299     //miss_
300     int time_x4;
301     int tens_place4 = GameMain::miss_ / 10;
302     time_x4 = (tens_place4 % 5) * figure_x;
303     spriteBatch.Draw(#figure, Vector3(400.0f, 280.0f, -1.0f), RectWH(time_x4, 0, figure_x, figure_x), 1.0f, Vector3(0, 0, 0), Vector3(50, 50, 0), Vector2(b_big, b_big));// 十の位
304
305     int ones_place4 = GameMain::miss_ % 10;
306     time_x4 = (ones_place4 % 5) * figure_x;
307     spriteBatch.Draw(#figure, Vector3(450.0f, 280.0f, -1.0f), RectWH(time_x4, 0, figure_x, figure_x), 1.0f, Vector3(0, 0, 0), Vector3(50, 50, 0), Vector2(b_big, b_big));// 一の位
308
309 }
```



電子レンジとカメラの動きを変えることでゲームが
単調にならないようにすることと、目で
判断できるように物の動きをそれぞれ別々の動きに
なるようにしている。
カメラはベジェ曲線で、電子レンジは波移動で
動かしている↓

```

if (camera_x[i] < 1280) {
    Vector3 bezier = Vector3_Bezier(point[0], point[1], point[2], point[3], t[i]);
    camera_x[i] = bezier.x;
    camera_y[i] = bezier.y;
    t[i] = t[i] + 0.02f * titleScene::hard;
    camera_collision[i] = Rect(bezier.x, bezier.y, bezier.x + 60, bezier.y + 64);
    big[i] -= 0.013;
    circle_alpha[i] = 0.5;
}

if (microwave_x[i] < 1280) {
    big[i] -= 0.0085;
    circle_alpha[i] = 0.5;
    microwave_x[i] = microwave_x[i] - 2;
    /////
    microwave_y[i] = MathHelper_Sin(theta[i]) * 90 * titleScene::hard + 400;
    theta[i] = theta[i] + 1;
}

```

↓最初の描画位置 ↓動かしたいドット数↓

Bat destroyer

Normal

Hard

Space to Start

Enter to Operation

Bat destroyer

Hard

Space to Start

Enter to Operation

ゲームを開始する際にバットで打ったかのような演出があったり、リザルト画面でも評価が前から飛んでくる演出などの細かな演出などを入れることで少しでもゲームの質を上げようとしている。

苦労した点

初めてプログラマー2人で作成するプロジェクトで
Git hubを使ったコミットの際に競合が大量に
発生してしまったり、コミュニケーションが足らず
ゲームの完成予想図があつていなかつたり
したことが苦労しました。

問題点

変数がかぶっていたり、命名規則が
あつていなかつたりしたためにプログラムが
こんがらがり、同じ処理が二回掛かれていたりする
ことが原因でした。

解決方法

コミュニケーションを積極的にとりながら
関数などで区切ったり、変数を新しく作るときには
相手のプログラムの変数を見てコミットの事前に
確認すること、ゲームのイメージを
フェイスブックなどで書いたり、話し合って
イメージの統一をはかることで解決しました。

作品紹介 五作品目

Desert town



プレイヤー人数
1人

ジャンル	VRシューティング
制作期間	三か月(Wisdom Worldを優先して制作していた為、実際は制作していた期間は一か月程度)
制作人数	自分(個人制作)
開発言語	C++, C#
制作環境	Unity
対象機種	Androidスマートフォン
担当箇所	作品全般

ゲームの概要

プレイヤーは主人公となり、砂漠の町に乗り込んで敵を銃で倒しながら敵が占領している砂漠の町の解放を目指すシューティングゲーム。

ゲームのルール

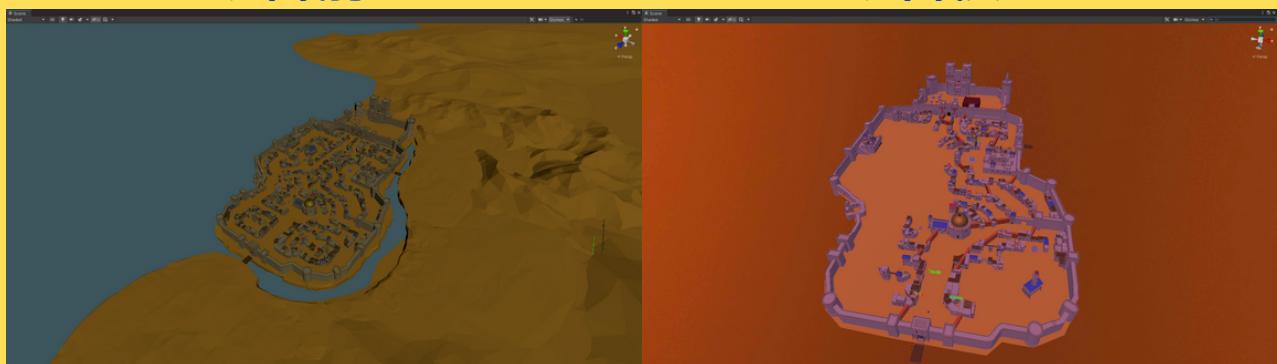
- ・敵を倒す、敵の弾を打ち落とすとスコアが貢える。
- ・プレイヤーのHPが無く、ゲームオーバーがない。
- ・できるだけ敵を倒したりしてスコアを稼ぎながらクリアしてハイスコアを更新しよう！

こだわった点

VR機種でのデバッグ作業が徐々に重たくなってきたため処理を軽くするためにプレイヤーから見えない位置などのオブジェクトを消したりボスステージに行くと通常ステージの描画をしない、影を一部オブジェクトからなくすなど工夫をして処理を軽くしている。

改善前 ↓

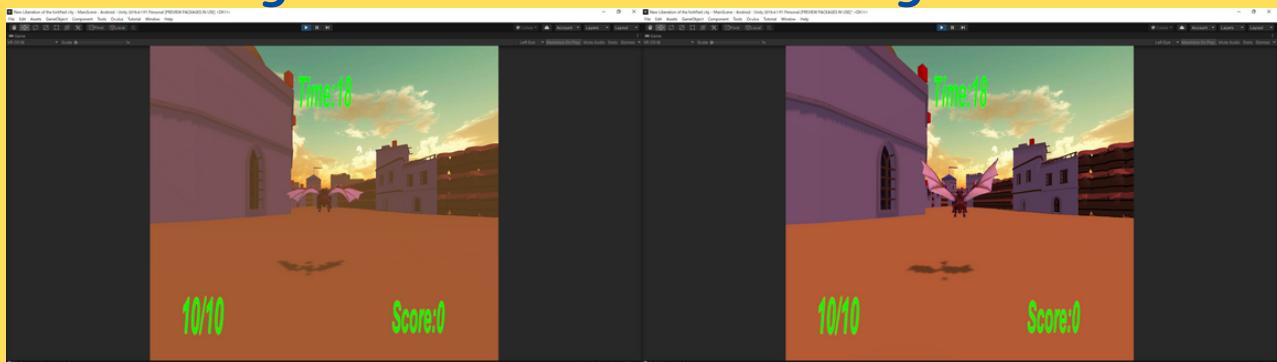
改善後 ↓



Fogをすることで砂が舞って見通しが悪くなるようにしたり、砂漠のイメージと合った建物を設置することで、このゲームの舞台が砂漠だということをプレイヤーが認識しやすいようにしている。

Fogあり ↓

Fogなし ↓



苦労した点

VRゲーム制作は初めてでしたが基本的な部分は今までの知識で製作できていましたがこの制作で苦労した点はプレイするとVR酔いをしてしまうことでフィードバックなどでプレイしてくれた人も酔う人が多かったです。

問題点

このゲームではプレイヤーは銃の操作しかできず、移動は指定したポイントを通るようになっていたのでプレイしている人が次の移動を予測できなかったり、急に視点が動いたりしてしまうことが問題点でした。

解決方法

視点がなめらかに動くように調整したり、進行方向、次の移動が分かるようにプレイヤーの前方にドラゴンを置いて示すようにするなどのVR酔い対策をすることで解決しました。

改善前 ↓



改善後 ↓



Wisdom World 作

品 紹 介 六 作 品 目



プレイ人数
1人

ジャンル アトラクション・
ディフェンスVRゲーム。

制作期間 三ヶ月

制作人数 プランナー 1人

デザイナー 1人

プログラマー 1人

二年生補助

プログラマー 1人

開発言語 C++,C#

制作環境 Unity

対象機種 Oculus Quest

Oculus Quest2

担当箇所 プログラム全般、
ゲームのアイデア出し。

ゲームの概要

ドラゴンを連れた魔法使いとなり、敵からの攻撃や障害物を縮小拡大魔法を使ってVRの世界をともに駆け抜けてクリアを目指すゲーム。

ゲームのルール

- ・破壊対象物を粉々にするとスコアが増える。
- ・増えるスコアはタイミングが正確なほど高スコアが貰える。
- ・できるだけコンボをつなげながらスコアを稼ぎ、クリアしてハイスコアを更新しよう！

こだわった点



物の大きさを拡大、縮小させる魔法の機能を
コントローラーでの操作にすることで魔法の杖の
ように使えるようにし、操作はスティックを
倒すだけ、というシンプルな操作にしました。
ですがそれだけだと魔法を使っている感が
出ないので、ポインターでホールドされている際に
二重でエフェクトを出して魔法を使っている感じを
プレイヤーに実感させるようにしました。





ルートの設定も簡単にできてある程度プレイヤーの移動に制限をかけられるようにするために CinemachineDollyCartなどのスクリプトを使用してプレイヤーの進行ルートをPathに沿った移動方法にしました。この移動方法にすることで様々なところを通ったり、急降下するなどのアトラクション要素を入れることが容易になりました。



相棒と共に障害を切り抜ける！

アトラクション性のある迫力のステージ

VRを活用した上下左右移動を使いたかったので
Oculus QuestのHMDの傾きでできるように
することで臨場感がでてなおかつプレイヤーが
直感的に操作ができる、なおかつ慣性がある動きで
ほうきに乗っている感を出すことなど様々な工夫を
することでプレイヤーの没入感をより高めようと
しました。

HMDの傾きを取得してその傾きに応じて移動する↓

```
69 //ヘッドセットでのX座標移動処理↓(インスペクター上のセンターAIの数値は偽なので注意(マイナスの数値は存在しない))
70 if (OVR CameraRig.localRotation.eulerAngles.z >= 10.0f && OVR CameraRig.localRotation.eulerAngles.z <= 100.0f)
71 {
72     if (inertia_speed_time >= 0.1f)
73     {
74         speed_x -= inertia_x;
75     }
76 }
77 else if (OVR CameraRig.localRotation.eulerAngles.z >= 260.0f && OVR CameraRig.localRotation.eulerAngles.z <= 350.0f)
78 {
79     if (inertia_speed_time >= 0.1f)
80     {
81         speed_x += inertia_x;
82     }
83 }
84 else
85 {
86     if (speed_x != 0)
87     {
88         if (speed_x > 0)
89         {
90             if (inertia_speed_time >= 0.1f)
91             {
92                 speed_x -= brake_x;
93             }
94         }
95         else if (speed_x < 0)
96         {
97             if (inertia_speed_time >= 0.1f)
98             {
99                 speed_x += brake_x;
100            }
101        }
102    }
103 }
104 //ヘッドセットでのY座標移動処理↓
105 if (OVR CameraRig.localRotation.eulerAngles.x >= 10.0f && OVR CameraRig.localRotation.eulerAngles.x <= 100.0f)
106 {
107     if (inertia_speed_time >= 0.1f)
108     {
109         speed_y -= inertia_y;
110     }
111 }
112 else if (OVR CameraRig.localRotation.eulerAngles.x >= 260.0f && OVR CameraRig.localRotation.eulerAngles.x <= 350.0f)
113 {
114     if (inertia_speed_time >= 0.1f)
115     {
116         speed_y += inertia_y;
117     }
118 }
119 else
120 {
121     if (speed_y != 0)
122     {
123         if (speed_y > 0)
124         {
125             if (inertia_speed_time >= 0.1f)
126             {
127                 speed_y -= brake_y;
128             }
129         }
130         else if (speed_y < 0)
131         {
132             if (inertia_speed_time >= 0.1f)
133             {
134                 speed_y += brake_y;
135             }
136         }
137     }
138 }
139
140
141
142
143
144
145
146
147
148
149
```

苦労した点

制作の期限に間に合わなくなることがあり、期限に出せたとしても期限ギリギリまで制作することが多くなり、スケジュール管理ができにくくなってしまったりして、苦労しました。

問題点

制作メンバーは最初、プログラマー3人でしたが自分以外のプログラマー2名が体調不良で学校にこれなくなり、プログラマー3人分の作業を自分一人でやらなければいけない状況になってしまったことが原因。

解決方法

制作の効率化のために新しいスクリプトを作るのでなく、Oculus Integrationのスクリプトを改造することで時間を短縮し、短縮した時間で別の作業をすることで製作の期限に間に合わせられるようにした。

七 作 品 目

DIRTY WORK DX12リメイク



プレイ人
数
1人

ジャンル	2Dアクション	
制作期間	一ヶ月	
制作人数	プランナー	1人
	デザイナー	1人
	プログラマー	1人
開発言語	C++(DirectX12)	
制作環境	Visual Studio 2019	
対象機種	Windows PC	
担当箇所	プログラム全般、 ゲームのアイデア出し。	

ゲームの概要

車を上下に操作して、右側から出てくる車を規定タイムの間避け切って、追跡してくる警察から逃げ切るゲームです。

ゲームのルール

- ・スコア(\$)がライフの役割もしている。
- ・警察車両に当たるとゲームオーバー。
- ・一般車両に当たるとライフが減る。
- ・ライフが0になるとゲームオーバー。
- ・できるだけスコア（ライフ）を残して、規定タイム間避け切るとゲームクリア。

Git hub URL

二作品目DIRTY WORK

https://github.com/takahasiyuma/source_code/tree/main/DIRTY%20WORK

四作品目Bat destroyer

https://github.com/takahasiyuma/source_code/tree/main/Bat%20destroyer

六作品目Wisdom World

https://github.com/takahasiyuma/source_code/tree/main/WisdomWorld

上記URLにソースコードがアップロードされていますのでご覧いただければ幸いです。

YouTube URL

六作品目 Wisdom World

[https://www.youtube.com/
watch?v=SU6MMYZvnyA](https://www.youtube.com/watch?v=SU6MMYZvnyA)

上記URLにプレイ動画、紹介動画が
アップロードされていますので
ご覧いただければ幸いです。