

vol.32

*Copyright S.R.D.G*

Ra

tio

nale



## 目次

代表挨拶	01	
-プログラミングコンテスト-		
競技部門		
高橋 諒大	電気システム工学科3年	
菊地 輝	電気システム工学科3年 「第29回プログラミングコンテスト」	03
競技アイデア募集企画		
岸 隆佑	機械・エネルギーコース2年	
根地戸 龍生	機械・エネルギーコース2年	
岩佐 佳慧	建築デザインコース2年	05
-活動報告-		
根地戸 龍生	機械・エネルギーコース2年 「CLIP STUDIO」	07
菅原 伶太	マテリアル環境コース2年 「弾幕ゲーム」	09
渡邊 天翔	建築デザインコース2年 「アクションゲーム制作」	11
岩佐 佳慧	建築デザインコース2年 「ibis Paint」	13
村主 太陽	建築デザインコース2年 「音楽制作」	15
レモンを絞る人	電気システム工学科3年 「Unity でターゲットマシン」	17
Nasatame	電気システム工学科4年 「C++ & OpenSiv3D による シューティングゲーム制作」	19
-プロコン旅行記-		
鈴木 雄裕	総合工学科 Ⅲ類 1年	22
編集後記	23	

## 代表挨拶

---

### 会長挨拶

電気システム工学科 3年 高橋 謙大

今年も「Rationale」をお読みいただきありがとうございます。11月中旬には追加稿としてプロコンに関する記事をホームページに公開いたしますので是非そちらもお読みください。

さて、今年は高専祭とプロコンの日程が重なったこともあり、初めての試みとして「Rationale」をWeb上にも公開しました。絵や旅行記などを鮮明な画像でお楽しみください。

私は高専祭当日プロコンに出場するため徳島に行っています。優勝できるよう頑張ってきますので、応援のほどよろしくお願ひいたします。当日はYouTubeにてライブストリーミングも行われるので興味のある方はそちらもご覧ください。

最近のプロコンの競技部門を見ていると人間の力量が問われる要素が多くなってきてているような気がします。プロコン実行委員会の方々も競技テーマを考えるのは大変そうですがプログラムが活躍しにくい競技テーマは非常にプロコンとして残念に感じます(というかそもそもルールに穴が多いのでまずきちんと定義してほしい...)。これからは競技アイデア募集企画など、実際に競技を行う側の目線を取り入れながら、よりプログラミング能力を競い合う競技に改善していくといいなと思いました。

改めて、「Rationale」作成に協力してくださった方々にこの場を借りて深く御礼申し上げます。これからも部員達の活躍を温かく見守っていただけると幸いです。

---

### 副会長挨拶

機械・エネルギーコース 2年 根地戸 龍生

今年も無事「Rationale」が完成しました。部員それぞれが違うテーマを原稿にまとめ上げとても良い「Rationale」を仕上げられたと思います。プログラミングをやってみたい人やパソコンで絵を描いたり、音楽を作つてみたい人はこれを機にはじめてみてはいかがでしょうか。

私たちの部活では、パソコン甲子園、プログラミングコンテストに毎年参加しています。今年のプログラミングコンテストは徳島で開催されます。おそらく今その徳島で部員達が頑張っているでしょう。仙台高専名取キャンパスが優勝することを願います。

私は今年から副会長になりました。色々と慌ただしいとこや人前に出ると緊張して上手く喋れなくなってしまうこと多々あります。ちゃんと副会長になれるのか心配です。色々と不安なことはありますが少しでも副会長っぽくなれたらいいなと思っています。

最後になりましたが、この「Rationale」の作成にかかわった部員の皆さん、顧問の北島先生と佐藤先生、そして今「Rationale」を読んでいる皆様にお礼を申し上げます。ありがとうございました。

第29回

プログラミング  
コンテスト

# 第29回全国高等専門学校 プログラミングコンテスト競技部門

## 1. はじめに

私たちソフトウェア研究部会は毎年、全国高等専門学校プログラミングコンテスト(以下、プロコン)に参加しています。今年は徳島県阿南市で開催されます。

プロコンには競技部門、課題部門、自由部門の3部門があり、今年は競技部門の開催25周年を記念して競技アイデア募集企画も開催されています。今年は競技部門と競技アイデア募集企画に参加しました。競技部門では、運営委員会が用意した競技に対して各自が用意したプログラムで、他校と早さや正確さを競い合います。

## 2. 競技部門 一巡りマスター

今年の競技は2チームによる陣取りゲームです。プログラムの計算結果を正確に伝達する手段が勝利の鍵を握ります。詳しい用語は表1に記載します。

## 3. 競技概要

### 3.1. 試合の進行

下の3ステップを1ターンとして、試合終了までこれが繰り返されます。試合開始時に、司令塔にフィールドの点数情報とエージェントの初期配置が書かれているQRコードが配られ、30秒の作戦ステップが与えられます。

1. 作戦ステップ: 司令塔がエージェントに指示を出します。エージェントはその場で待機しています。
2. 意思表示ステップ: 指示を受け取ったエージェントが指示を審判に伝えます。マスへの移動だけでなく相手のタイルを剥離することもできます。
3. 行動ステップ: エージェントは、意思表示ス

電気システム工学科 3年 菊地 輝  
電気システム工学科 3年 高橋 謙大  
テップで審判に伝えた指示通りにフィールド上を行動します。

最後のターンの行動ステップが終わると試合が終了します。

### 3.2. 時間制限

1ターン10~20秒程度、ターン数は試合ごとに異なり40~80ターンで行われます。

### 3.3. 勝敗判定

基本的に点数で勝敗を判定します。

#### 3.3.1. 点数計算方法

試合終了時点で、自チームのタイルが置かれているマスの点数の合計を計算します。

その次に、自チームのタイルによって囲まれた領域内の全てのマスの点数が加算されます。この場合、領域内の点数が負の数であっても正の整数として加算されます。下の領域点の例だと、-1のマスを囲んでいるので領域点は+1点とされ、合計9点となります。

1	4	3	4	1
4	0	2	0	4
3	2	-1	2	3
4	0	2	0	4
1	4	3	4	1

図1 領域点の例

領域点には相手のタイルが置かれているマスも含まれます。自分のタイルが置かれているマスは領域点には含まれません。

### 3.3.2. 勝敗決定方法

以下の優先順位で勝敗を決定します。

1. タイル得点と領域点の合計が大きい方
2. タイル得点の大きい方
3. サイコロを振り、数の大きい方

### 3.4. 通信手段・意思表示手段

試合中は司令塔・エージェント共に声や音による意思疎通が禁止されます。但し、ハンドサインやジェスチャーによる意思疎通は可能です。

司令塔から指示を出す際に使用できる道具は A4 サイズのトランプ 1 セットです。しかし、モニターをエージェントに見せる等の行為は禁止されています。

エージェントは外部と通信のできる電子機器、暗号を解読するための、いかなるものも持ち込みが禁止されています。

エージェントが指示を審判に伝える際には、2 枚のうちわを使用します。エージェントの向いているつま先の方向で前後左右、片手のうちわを掲げれば掲げた左右方向の斜め前、両手で正面を表します。また、掲げたうちわの表裏で移動と剥離を判断します。エージェントのつま先が別々の方向を向いていた場合、うちわの掲げ方を間違えていた場合、両方のうちわの表裏が異なる場合は全て無効となります。

### 4. 探索方法

探索方法はプロコンが終了次第記述します。

### 5. おわりに

この記事を読んでくださっているとき、私たちはプロコンの会場で戦っています。普通の陣取りゲームとは違い領域点というルールが追加されており、それぞれの学校ごとに様々な戦術が見られると思います。また今回の大会では PC での探索だけでなく、人間からの人間への伝達が重要な大会になっています。各学校の伝達方法にも注目してみると面白いかもしれません。

## 引用元

高専プロコン公式サイト

<http://www.procon.gr.jp/>

表 1 競技用語

#### ・フィールド

複数のマスからなる、試合の行われる場です。最大  $12 \times 12$  のマスが敷き詰められ、必ず長方形または正方形になります。1 辺の最低マス数は公開されていませんが、最低総マス数は 80 です。配置されるマスの点数は、一方のチームが有利にならないように中央線に対して必ず水平、または垂直方向に線対称になります。エージェントの初期位置も同様に中央線に対して必ず水平、または垂直方向に線対称に配置されます。

#### ・マス

マスにはそれぞれ -16 ~ 16 点が割り合てられており、0 点以下の数は少数に抑えられています。エージェントが移動する際に、そのマスに自分のチームのタイルを置きます。タイルが剥がれる場合、何も置かれていないマスの状態に戻ります。

#### ・タイル

エージェントがマスを通過する際に置かれるものです。自チームのタイルが置かれているマスの点数が自分のチームの得点になります。タイルを剥がし、何も置かれていないマスの状態に戻すのには 1 ターン必要です。

#### ・司令塔

フィールドの外から、PC を操作しエージェントに指示を出す人間です。各チーム内で 1 人が司令塔を担当します。PC を操作することができるのはチーム内で司令塔だけです。司令塔は試合中に、指定された司令塔エリアから出ることができません。

#### ・エージェント

司令塔からの指示を受け、フィールド内を移動する人間です。各チーム 2 人がこれを担当します。うちわ以外の所持は基本的に認められていません。エージェントは試合中に、フィールド外に出ることはできません。

# 競技アイデア募集企画

機械・エネルギーコース2年 岸 隆佑

機械・エネルギーコース2年 根地戸 龍生

建築デザインコース2年 岩佐 佳慧

**Coming soon**

**11/12**

# 活動報告



# CLIP STUDIO

機械・エネルギーコース 2年 根地戸 龍生

## 1. はじめに

私はCLIP STUDIOというデジタルペイントのソフトウェアを使いイラストを描きました。このソフトウェアではイラストはもちろん、漫画やアニメーション制作もできます。

## 2. レイヤーについて

まずレイヤーについて説明します。イラストはレイヤーというものに描きます。レイヤーは何枚もの紙が重なって層になっているイメージです。それぞれのレイヤーには名前を付けることができ、その紙に何を描いたかを示すことにより、後で見返したとき分かりやすくなります(図1)。

## 3. メイキング

今回このイラストでは600dpiの解像度を用いて描きました。解像度とは、画像の密度のことです。dpiとはその解像度の単位です。

### 3.1. ラフ画・下書き

最初にラフ画を描きます。ラフ画では「Gペン」で自分が書きたいものを大まかに描きます。下書きはラフ画より具体的に描きます。

### 3.2. 線画

今回線画で使用したツールは「ざらつきペン」です。線画専用のレイヤーを作りて描きます。描く前に線を見やすくするため、まず下書きをしたレイヤーの不透明度を下げます。また下書きの色を水色などのより線画の色が際立つものにすると線が見やすくなります。

きれいな線を描きたいときは、「手ぶれ補正」を使います。手ぶれ補正を高くすると線にブレが無くなりきれいな線が描けます。線を描く際は1本の線で隙間なく描きます。大きい隙間があると後の塗りつぶしで色がはみ出してしまう場合があります。



図1 レイヤー



図2 下書き



図3 線画

### 3.3. 色塗り

まず色分けをします。どこに何色を塗るかを分かるように「塗りつぶし」で塗ります。線画のどこかに小さい隙間がある場合、塗りつぶしの「隙間閉じ」という機能を使います。隙間閉じの数値は隙間の大きさを表し、設定した数値以下の隙間は閉じられたものと認識され色が塗られます。

色を塗るとき顔のパーツや背景、服などをレイヤーごとに分けておくことで、影やハイライトを塗るとき作業しやすくなります。塗り分けに必要な境界線がない部分を塗るときは、G ペンで境界線を描いてから塗りつぶしツールで塗ります。髪にグラデーションをかけたいときは、まず範囲を選択します。次にサブツールの「描画色から透明色」を選択します。最後にグラデーションの開始位置をクリックし、グラデーションをかけたい方向にドラッグします。

### 3.4. 影、ハイライト

前の工程で部分ごとに分けていたレイヤーを「透明ピクセルをロック」にします。ロックすると透明部分が描画できなくなります。その後そのレイヤーの上に新しい影、ハイライト用のレイヤーを作成し下のレイヤーにクリッピングします。透明ピクセルをロックしたレイヤーにクリッピングすると下のレイヤーで描画した部分しか塗れなくなります。これにより、はみ出して余計な部分まで塗ってしまう心配が無くなります。

影、ハイライトは G ペンで境界線を描き、塗りつぶします。サブツール「色混ぜ」の「水彩なじませ」でハイライトをぼかし、その上からまた細かい影を付け足します(図 4)。

### 4. 終わりに

CLIP STUDIO PAINT を使い始めて 1 年と少し経ちました。いまだに分からぬことのほうが多くまだまだ学ぶべきことがあります。私が今回のイラストを描くうえで一番苦労したところは色塗りです。他の人のイラストを見るとそれぞれ違った塗り方やブラシを使っていて何をしたらいいかと迷いました。次に描くときには光の当たり方を注意し、もっと CLIP STUDIO PAINT の機能を

把握したうえで描きたいです。

絵を描くソフトは CLIP STUDIO PAINT の他にも Adobe Photoshop やペイントツール SAI、GIMP などいろいろあります。是非皆さんも描いてみてはいかがでしょうか。



図 4 影、ハイライト



図 5 完成絵

### 参考文献

CLIP STUDIO PAINT 使い方講座

<https://howto.clipstudio.com>

/library/categories/view/clipstudiopaint

いちあっぷ

<https://ichi-up.net/>

コンテアニメ工房

<https://conte-anime.jp/>

# 弾幕ゲーム

マテリアル環境コース 2年 菅原 伶太

## 1. はじめに

私はOpenSiv3Dを使用して、弾幕ゲームを作りました。きっかけは、友達が他の弾幕ゲームのクリアに苦戦しているのを見て、攻略の難しいゲームを作つてみたいと思ったからです。

## 2. OpenSiv3D とは

C++を用いてゲームやメディアアート、音声と画像の処理等 3000 以上の機能が無料で使用できるライブラリです。別々の図形同士の接触判定など、一見難しそうな処理も物によっては1行で反映させることができるのが特徴です。

## 3. 制作過程

### 3.1. 準備

ゲーム画面の大きさは 800 \* 800px で作成しました。自由に変更できるので自分の作るゲームのジャンルに応じて変えることができます。

次におおまかにどんなゲームを作りたいのか、どのような順番で制作していくのかということをまとめました。(表 1)各作業を紹介します。

### 3.2. 自、敵キャラクターと弾の宣言

リスト 2 は自、敵キャラクターとそれらが撃つ弾の宣言を示しています。第一引数を画像があるファイルパスに置き換えることで自分の好きなキャラクター等をゲーム内で表示させることができます。

### 3.3. 自、敵の弾が範囲外に出た場合

リスト 3 は自キャラ、敵キャラの弾が画面外に出た場合消去するように書いたコードです。このコードが無いと画面外で永遠に弾が移動し続けてしまいます。発射される弾の数も決まってるので一定数発射されるとそれ以上は発射されなくなってしまいます。

表 1 制作順序

1. 自、敵キャラクターと弾の宣言
2. 画面外の弾の消去
3. 弾の接触判定
4. 敵の弾の出現と移動

### リスト 2 自、敵キャラクターと弾の宣言

```
const Texture textureMe(  
    Emoji("█"), TextureDesc::Mipped);  
const Texture textureEnemy(  
    Emoji(U"enemy"), TextureDesc::Mipped);  
const Texture textureBullet(  
    Emoji(U"bullet"), TextureDesc::Mipped);  
const Texture textureEnemyBullet(  
    Emoji(U"enemybullet"), TextureDesc::Mipped);
```

### リスト 3 自、敵弾が範囲外の場合消去

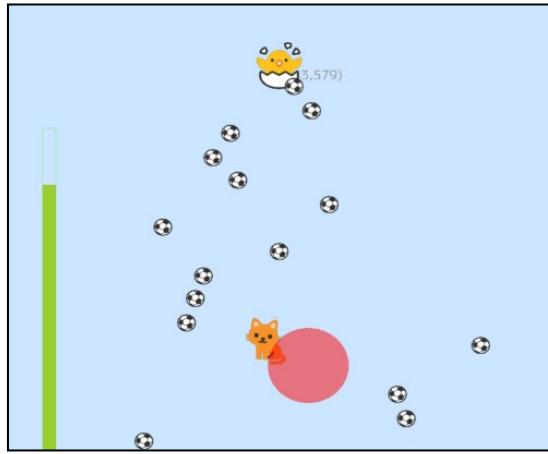
```
for(int i = 0; i < enemyBulletSize; i++) {  
    if (enemyBullets.at(i).exist &&  
        enemyBullets.at(i).y > Window::Size().y){  
        enemyBullets.at(i).exist=false;  
    }  
}
```

### 3.4. 弾の接触判定

リスト 4 は自分の弾と敵本体の接触判定を表しています。キャラクターと弾には見えない円があり、その円が互いに触れたときに接触判定が生じるというイメージです。自分の弾と敵本体の距離が 0 になった(見えない円が触れた)とき、自分の弾が消えゲームが終了するというようになっています。弾同士にも接触判定を加えたので弾同士がぶつかるとどちらも消去されます。

#### リスト4 自分の弾と敵本体の接触判定

```
if(meBullet.exist){  
    if(Point(meBullet.x,meBullet.y)  
        .distanceFrom(  
            Point(enemyX,enemyY))<=hitRange){  
        meBullet.exist=false;  
        endFlag=true;  
    }  
}
```



#### 3.6. 敵の弾の出現と移動

ゲームの基本的な処理が出来上がったので弾幕を作っていきます。今回はランダムな方向にまっすぐ撃ち出されてくる弾と、横に反復しながら近づいてくる種類の弾が発射されるようにしました(図1)。1つ目の弾は、一定の速度で手前側に進み、横方向に弾ごとに違う速度で移動します。2つ目の弾は、250msごとに横方向の速度を反転させながら移動します。弾を発射するには弾の出現と移動の2段階に分けてコードを書きます(リスト5)。どんな軌道になっているか何度も確認しながら修正を加えていきました。弾の動かし方は今回使用したパターン以外にもあるのでアイデア次第では花火のような綺麗な弾幕作ることもできます。

#### 4. まとめ

普段はプレイする側なので、どのようにしてゲームを難しくするか考えて制作していくのは新鮮でした。ゲーム自体制作するのが初めてだったので、先輩に教えて頂きながらなんとか完成させることができて良かったです。接触判定を作る作業が全ての作業の中で一番悩んだ部分でした。今後もC++の理解を深められるよう勉強していきたいです。

#### 参考文献

OpenSiv3D

<https://scrapbox.io/Siv3D/>

C++ 日本語リファレンス

<https://cpprefjp.github.io/>

図1 ゲーム画面

#### リスト5 敵の弾の出現と移動

```
for (int i = 0; i < enemyBulletSize; i++)  
{  
    if (!enemyBullets.at(i).exist)  
    {  
        enemyBullets.at(i).exist = true;  
        enemyBullets.at(i).kind=Random(0, 6);  
        enemyBullets.at(i).y = enemyY;  
        enemyBullets.at(i).x = enemyX;  
        enemyBullets.at(i).power=enemyPower;  
        break;  
    }  
}  
for (int i = 0; i < enemyBulletSize; i++)  
{  
    if (enemyBullets.at(i).exist)  
    {  
        switch (enemyBullets.at(i).kind)  
        {  
            case 5:  
                enemyBullets.at(i).x +=  
                    (sw.ms()%500-250)/  
                    abs(sw.ms()%500-250)*5;  
                enemyBullets.at(i).y += 3;  
                break;  
            case 6:  
                enemyBullets.at(i).y -= 2;  
            default:  
                enemyBullets.at(i).x +=  
                    enemyBullets.at(i).kind - 2;  
                enemyBullets.at(i).y += 3;  
                break;  
        }  
    }  
}
```

# 2D アクションゲーム制作

## 1. はじめに

私は OpenSiv3D を使用してアクションゲームを制作しました。去年も OpenSiv3D でシューティングゲームを制作したのですが、今回は新しいジャンルのゲームに挑戦したくなり、アクションゲームを制作する事にしました。このゲームの処理の内、プレイヤーとブロック、トゲ、ゴールなどとの接触判定について紹介していきたいと思います。

## 2. OpenSiv3D について

OpenSiv3D とは、C++で簡単にアプリケーションを作れるライブラリです。OpenSiv3D では、コンソール画面上ではなく、ウィンドウを表示してグラフィックスやオーディオを自由自在に操ることができます。

## 3. ゲームの仕様

今回制作したゲームは、プレイヤーに移動とジャンプをさせて、ゴールを目指すゲームです（図1）。このゲームにはトゲや敵、バネ（バネの上でジャンプすると大きくジャンプできる）という要素があります。

トゲや敵に触れたり、ゲーム画面外に落下するとゲームオーバーとなります。また、ゴールに触れるとゲームクリアとなります。

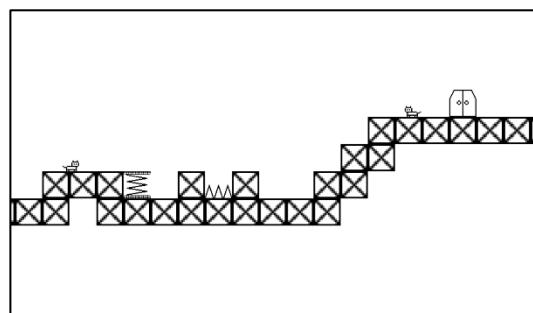


図1 ゲームの全体像

建築デザインコース 2年 渡邊 天翔

## 4. プレイヤーの接触判定

プレイヤーとブロックの接触判定について解説します。接触判定のプログラムを下のリスト1に記載します。

### リスト1 ブロックとの接触判定

```
1. for(size_t i = 0; i < blocks.size(); i++)  
2. {  
3.     if(blocks[i].intersects(m_position_right))  
4.     {  
5.         m_position.x -= 3.0;  
6.     }  
7.     if(blocks[i].intersects(m_position_left))  
8.     {  
9.         m_position.x += 3.0;  
10.    }  
11.    if(blocks[i].intersects(m_position_top))  
12.    {  
13.        m_jumpFrame = 0;  
14.    }  
15. }
```

1行目では接触判定のループさせる回数を決めています。ブロックの数だけ接触判定をしなければならないので、ループする回数に条件をつけています。

3行目から10行目までは、プレイヤーがブロックの左右に衝突したときのプログラムです。プレイヤーが動いているときに進行方向にブロックがあつたら、プレイヤーが1フレームに移動する距離分、その反対の向きにプレイヤーを移動させる処理を行っています。これにより、進む距離が0になります。

11行目から14行目までは、プレイヤーがブロックの下部に衝突したときのプログラムです。プレイヤーがジャンプし、ブロックの下部に衝突したときジャンプフレーム(あと何フレーム分ジャンプできるか判断するための数)を0にし、強制的にジャンプを中断します。プレイヤーには上下左右にブロックとの接触判定をする点があり、それらで接触判定することでブロックにめり込まないようになっています(図2)。

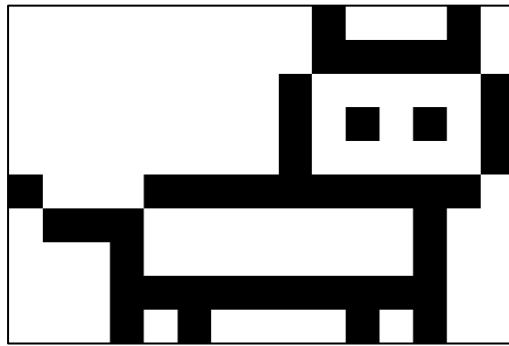


図2 プレイヤーの拡大図

## 5. ゲームの流れ

ゲームの大まかな流れをフローチャートに示します(図3)。ゲームをスタートし、初めにステージの選択をします。その後、選択されたステージを読み込みゲームがスタートします。ゲームがスタートしたら、プレイヤーの移動からオブジェクトとプレイヤーの描画までの動作を繰り返します。プレイヤーが画面外に落下、または敵やトゲに触れるとゲームオーバーの画面が表示されます。ゴールに触れるとゲームクリアの画面が表示されます。その後 Enter キーを押すことで、ステージ選択の画面に戻り、それらの過程を繰り返します。

## 6. おわりに

私がプログラミングを始めて約1年半が経ち、コードの書き方の基本が身についてきたのが実感できました。去年できなかった事ができるようになりました。プログラミングの幅が広がりました。このゲームを制作する際にいろいろな壁がありました。完成させる事ができました。プログラミングの知識はまだまだ乏しいですが、これからもゲーム制作などを続けていこうと思います。

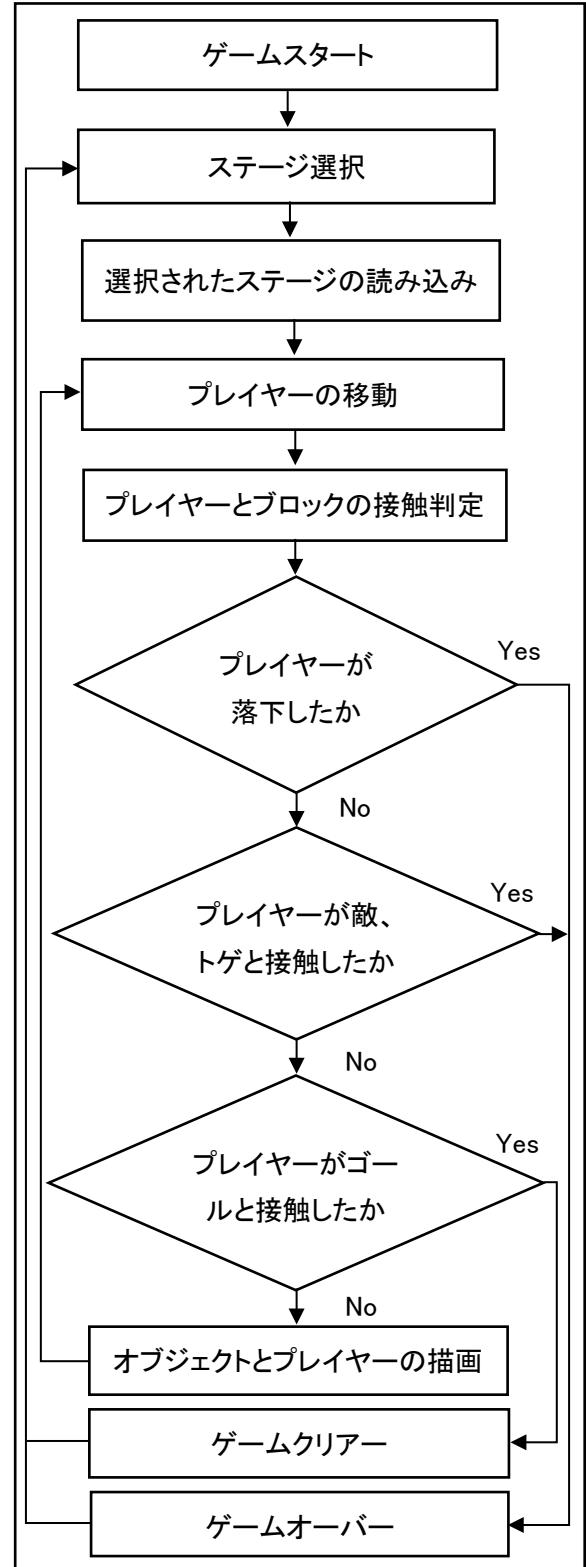


図3 ゲームのフローチャート

## 参考文献

OpenSiv3D

<https://scrapbox.io/Siv3D/>

# ibisPaint X を使ったイラスト制作

## 1. はじめに

私はスマートフォンの無料お絵かきアプリ ibisPaint X を使いデジタルイラスト制作をしました。今回は iPad と専用のデジタルペンを使用し、制作にあたりました。

## 2. ibisPaint X の機能

このアプリの最大の利点は既存の PC 向けのお絵描きツールと遜色無いほど機能が豊富で、プロ、初心者に関わらず、簡単にデジタルイラスト制作をすることが可能である点です。また、オンライン環境であれば他のユーザーによって任意投稿されたイラストがカテゴリ別に閲覧出来ます。それに加え、イラストの作画工程を動画で見ることができ、モチベーションの向上にもつながります。自分で様々な機能を試してみるのも楽しみの一つと言えるでしょう。

## 3. レイヤーとは

レイヤーとは、簡単にいえば透明な紙のようなものです。単体のレイヤーか複数のレイヤーを重ねた物が一つのイラストとなります。重ねたレイヤーは上から順に認識され、上のレイヤーに描画されているイラストを優先的に表示します。

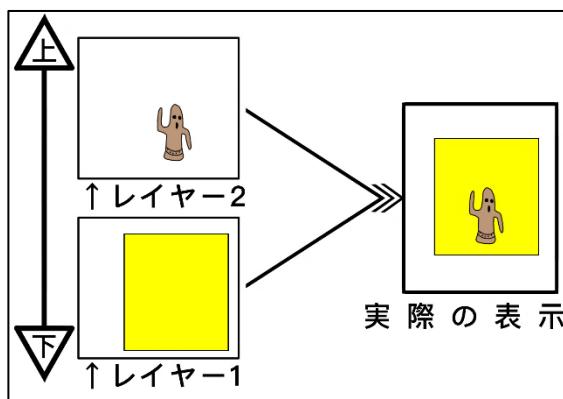


図 1 レイヤーについて

建築デザインコース 2年 岩佐 佳慧

## 4. メイキング

### 4.1. 制作準備

今回の用紙サイズは 1075 × 1518px、解像度は 350dpi で制作しました。

### 4.2. 下書き

まず、自分が描きたいものを簡単に下書きにしました。下書き→線画→色塗りという流れです。



図 2 下書き

### 4.3. 線画

簡単な下書きが終了したら、それを元に新しい線画のレイヤーを作成します。注意することは、元の下書き線よりも細く丁寧に線を描くことです。線画は濃く描いたり、薄く描いたりと使い分けることによって、よりはっきり輪郭がわかるようになります。また、線と線に少しでも空間があると、下塗りの作業で塗りつぶしツールを使った際に意図しない部分まで塗りつぶされてしまうので確認しながら線画を描きます。

#### 4.4. 塗り

線画の作業が終了したら塗りに入ります。私は塗りで部分ごと(ex 目、腕、髪等)に分けて描きました。今回使ったレイヤー数は合計 80 枚で、沢山のレイヤーを使いました。細かく作業がしたかったので今回はレイヤーを多くしました。

まず、図 3 のように簡単にベースとなる色を塗ります。ポイントは塗り残しがないか確認しながら塗っていくことです。毛先や線画が集中している部分は塗り残しが生まれやすいところなので注意しながら描きました。

#### 4.5. 影や光を描く

影や光を描く際のポイントとしては光源の位置を意識しながら描くことです。服は皺を意識しながら描きました。光や影の先をぼかしツールでぼかすことによってより印象が柔らかくなります。また、影や光はできる場所によって色彩が変わるので気をつけながら作業を行いました。

#### 4.6. 描画ツール:ソフトライト

「このイラストは何か足りない…」そう思ったら描画ツールのブレンドモード(※)を使ってみましょう。今回はブレンドモードのソフトライトにしました。ソフトライトは重ねた色に応じて結果が異なります。重ねた色が明るいなら明るく、暗い色であれば暗く表示され、色の部分に重ねず描画した場合は白になります。

※下のレイヤーに対して、特殊な配色の仕方を設定できる機能の事です。

#### 4.7. 仕上げ

最後に背景などを描画すれば完成です。

#### 5. 終わりに

今回描いたイラストはポスターとして掲示されています。印刷版はグレースケールですが、ポスター やオンライン版ではカラーなので是非ご覧ください。

また、私は絵描きを始めてからまだ間もない  
ので、これから勉強を重ねさらに上手く描けるよ  
うに頑張りたいと思います。



図 3 ベース色塗り



図 4 ブレンドモードの使用



図 5 完成図

#### 参考文献

知らなきゃ損する！レイヤーの合成モードって

何？ - CLIP STUDIO PAINT

<https://www.clipstudio.net/>

# 音楽制作

## ～初めての音楽制作～

### 1. はじめに

私はピアノロールをメインとした音楽編集ソフト Domino を使って音楽制作をしました。Windows には無料で使える音源がたくさんあるので、いろいろな音の重なりを作ることができます。使い方が簡単で WEB 上に使い方をまとめたものが多く存在しているので、私のように初めて音楽制作をするという人にお勧めしたいソフトです。作成した音楽は独自形式のほか、SMF 形式で保存可能です。ソフトの容量も小さく、気軽に導入できると思います。図1は音楽制作をしている途中のものです。

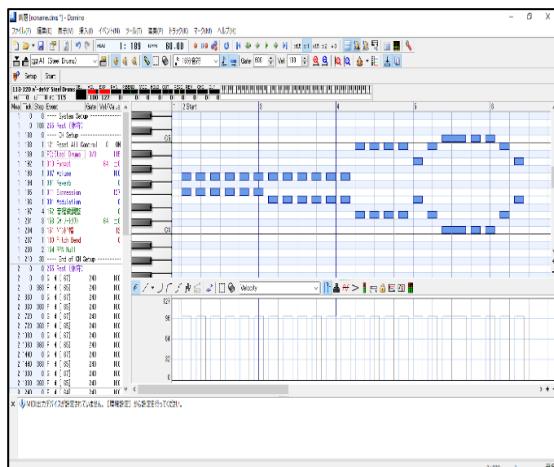


図1 音楽製作途中

### 2. DTM と DAW について

まず、音楽をパソコンで作ることを指す“DTM”(デスクトップミュージック)。そして、“DAW”(デジタルオーディオワークステーション)と呼ばれる DTM をするために必要なソフトが存在します。DAW には様々な種類があり、お金が掛かるものもあれば、フリーで使えるものなど、DAW は非常に多く存在します。

建築デザインコース 2年 村主 太陽

### 3. MIDI キーボード

MIDI キーボードとは音を打ち込むためのコントローラーのことです。MIDI キーボードは音を打ち込む際、押した鍵盤の情報が同時に入力でき、音の長さや強さといった情報が反映されるので、マウスを動かして一つ一つの音をクリックして打ち込むやり方と比べ効率的に進めることができるので、もし音楽制作をする場合は MIDI キーボードを使うことをお勧めします。

今回は部室にある KORG microKEY25をお借りしました。この microKEY25の25は鍵盤の数で、鍵盤数が最も少ないキーボードです。多いものだと88鍵あり、用途によって使い分けられるという点も MIDI キーボードを勧める1つのポイントです。

### 4. コードについて

コード(和音)とは二音以上の音の重なりのことです。例えば、三つの音が重なったコードの基本形をトライアドまたは三和音といいます。さらに音を一つ足したりすると、セブンスコード、テンションコードと呼ばれます。音の組み合わせについてですが、組み合わせ自体は非常に多くの数が存在します。雰囲気がコードによって違うので万人受けするようなコードから、少数の人だけに受けけるようなコードまで、たくさんあります。コードの種類として、major(メジャー)と呼ばれる明るい響きのものがあり、例として“ド・ミ・ソ”や“ファ・ラ・ド”などのコードがあります。また、少し暗い雰囲気の minor(マイナー)というコードでは“レ・ファ・ラ”や“ミ・ソ・シ”などがあります。最後に、dim.(ディミニッシュ)というものは不安定な響きをもつコードで、例としては“シ・レ・ミ”

などがあります。

今回の制作では major(ド・ミ・ソ)を使いました。これらの音を重ねることで、明るく、おしゃれな感じを部分的に出すことができたと思います。

### 5. コード進行

コード進行とはコードを組み合わせてできる音の流れのことです。J-POPで代表的なコード進行は“カノン進行”と呼ばれます。カノンという曲に由来するもので、感動を演出できるようなコードになっています。使われている曲として「壊れかけの RADIO」や「終わりなき旅」などがあります。“王道進行”はインパクトが強くサビでよく使われるもので、非常に多くの曲で使われています。例としては「ロビンソン」「瞳を閉じて」などが挙げられます。“小室進行”は作曲家小室哲哉氏が多用したことから小室進行と呼ばれたコード進行で、「残酷な天使のテーゼ」「WOW WAR TONIGHT」などに用いられています。

### 6. 音楽制作の主な流れ

制作をする上で、参考にさせていただいたサイトの「dn-voice.info」に記載されていた音楽制作の工程順序を参考に私が行った作業を図式化しました(図2)。今回、コンセプトとして音楽制作に慣れるために、既にある曲に自分なりに手を加えて曲を制作しました。選んだ曲は楽譜 자체が簡単で、明るい雰囲気の「Chopsticks」という曲です。制作の初めは、音符とその長さを打ち込んでいく作業で、音の長さや、先ほど紹介したコードも少し考えながら入れてみました。楽器種類として、クラビネット(電子式のキーボード)を選びました。ここまでをメインメロディを作る工程とします。次に BPM(1分間の拍の数)決めです。今回制作した全体の BPM は 120.0 とし、効果音等を入れる作業では、ザイロフォンという鍵盤打楽器と、エレキギターの音を伴奏として入れてみました。これらの楽器を選んだ理由は楽器の音が面白くて、以前から使ってみたいと思っていたからです。

### 7. 終わりに

今回の音楽制作では、専門用語を調べるところから始まり、音楽制作のソフトの使い方を調べ、コードの種類や組み合わせを調べるといった基本的なところからのスタートでした。Domino はシンプルな造りで、初めてでも分かりやすかったです。制作するうえで、難しいところが多々ありました。音楽制作について様々なことを知ることができて良かったです。次回の制作ではもう少し知識を付けてオリジナルの楽曲を制作できるようになりたいです。

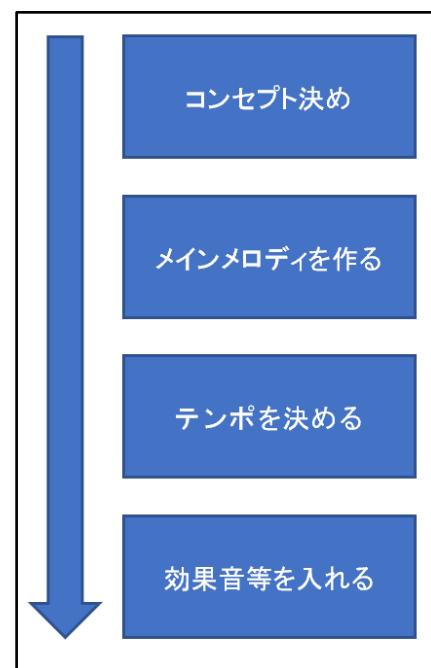


図2 作業工程

### 参考文献

「本格カラオケ制作」なら野田公房♪

<https://dtm-nodakoubou.net/>

藤本健の“DTMステーション”

<https://www.dtmstation.com>

dn-voice.info

<https://dn-voice.info>

サイタ作曲レッスン

<https://cyta.jp/>

# Unity でターゲットマシン

## 1. はじめに

今回私は Unity2018 と SketchUp2017 を使用してターゲットマシンを作りました。

ターゲットマシンとは硬貨を入れるとガムボールが排出され、それを使用したピンボールなどをすることができるアミューズメント機器です。これを作った理由は、店で見かけて Unity で作ろうと思ったからです。

## 2. 使用したアプリケーションについて

### 2.1. Unity

Unity は複数のプラットホーム(OS やゲーム機)に対応するゲーム開発プラットホームで、ゲーム開発会社でも使用されています。これを使うことで物理演算やオブジェクトの配置が簡単になります。このアプリケーションは資金調達が 10 万 \$ 以下などの条件を満たしていれば無償で使用することができます。

### 2.2. SketchUp

SketchUp とはパソコン用 3D モデリングアプリケーションで無償版(Make)と商用版(Pro)があります。SketchUp 内にある 3DWarehouse を使用すれば 3D モデルをダウンロードすることができます。無償版には商用使用の禁止、入力・出力可能な形式の制限、プレゼンテーション機能の制限など、使用できない機能があります。

## 3. 制作したゲームについて

このゲームは A、S、D キーを使用してステージ(茶色の板)を傾けてガム(球)をゴールまで転がすゲームです。操作割り当て A: 左回転 D: 右回転 S: 水平に戻す Space: リセット

電気システム工学科 3年 レモンを絞る人

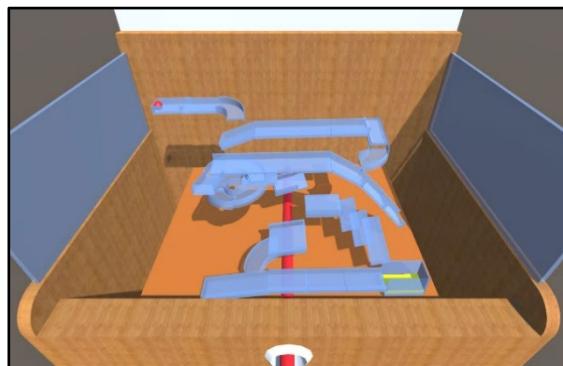


図 1 ゲーム画面

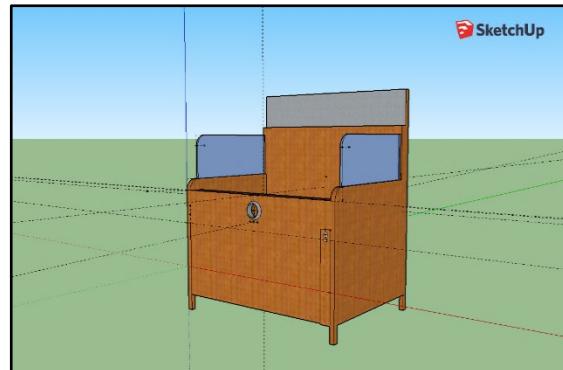


図 2 使用した素材

## 4. ゲームの作成

### 4.1. ステージ作成

Unity 上では簡単な形状(立方体や円柱)の 3D オブジェクトの作成ができます。しかし、単純な形状しか作成できないので SketchUp であらかじめ素材(3D モデル)を作成し、Unity 上に取り込みました。その後衝突判定を生成してオブジェクトがすり抜けないようにしてから配置しました。素材の例を図 2 に示します。

### 4.2. 主なスクリプト

Unity ではオブジェクト等にスクリプト(プログラム)を取り付けることができ、取り付けたオブジェクトを動かしたり他のオブジェクトに衝突した時に指定した動作をさせることができます。こ

のゲームでは主にステージを動かす、ゲームのリセット、テキストの表示・非表示を切り替えるスクリプトがあります。

#### 4.2.1. ステージを動かす

このゲームでは設定した角度までステージを傾けることができます。

これには子オブジェクトは親オブジェクトの影響を受ける、という仕様を使用しています。実際の動作で表すと、ステージ(親)に通路(子)を取り付けているのでステージが傾いたときに通路も同時に傾きます。しかし、通路のみの動作はステージの動きには影響しません。

ステージの傾きを制限するため、どれだけ傾いているかをカウントする変数を用意しました。この変数が決めた傾きの量の上限または下限(例: ±20)であれば、上限下限を無視する方向には傾ける動作をしないようにスクリプトを書きました。ステージを動かすスクリプトの傾きを制限している部分をリスト1の2行目に示します。

#### リスト1 角度制限スクリプト

```
1. //plus 側制限>カウンターならば
2. if (plus_limit > rotation) {
3.     //a キーが押されているならば
4.     if (Input.GetKey("a")) {
5.         //カウンターに 0.4 足す
6.         rotation = rotation + 0.4;
7.         //0.4 だけ Z 軸回転する
8.         this.gameObject.transform.Rotate
9.             ( 0, 0, 0.4f);
10.    }
11. }
```

また、水平に戻す動作はキー入力がある間、カウンターが0になるまでステージを回転するように書きました。

回転している通路は1フレーム当たり決めた数値分だけ回転すると書くことで、常に回転し続ける動作をさせています。

#### 4.2.2. ゲームのリセット

リセット時にする動作はステージを水平にすることと、球の位置を初期位置に戻しテキストが表示されている場合は非表示にすることです。テキストの表示・非表示については後述します。

球を初期位置に戻す動作はSpaceキーを押したら、初期位置に移動させステージが水平に戻るよう書きました。

#### 4.2.3. テキストの表示・非表示

テキストは場外に落下した時とゴールした時に表示されます。ゲーム開始時ではテキストの表示が無効になっています。場外には透明なテクスチャーのオブジェクトを配置しました。このオブジェクトには球と衝突した場合ゲームオーバー時のテキストの表示を有効にするというスクリプトが取り付けられています。また、Spaceキーが押された場合、ゲームオーバー時のテキストを無効にするように書きました。ゴール時のテキストの表示・非表示も同じ方法で実装しました。

### 5. おわりに

テストプレイを頼んだところ、球がどこにあるのか見辛いという意見や、カメラを動かせるようにしてほしいという意見があり、カメラ周りの改良が必要だと感じました。また、処理が重くなってしまうことがあるという問題も解決したいです。

今回の作業中、一度プロジェクトのファイルが破損した事があったので、バックアップを常に取りながら作業をするようにしたいです。

### 参考文献

Unity スクリプトリファレンス

<https://docs.unity3d.com/ja/2017.4/ScriptReference/index.html>

はじめての Unity

<https://unity3d.com/jp/learn/tutorials/projects/hajiuni-jp>

# C++ & OpenSiv3D による シューティングゲーム制作

## 1. はじめに

シューティングゲームは古くから非常に人気があるゲームジャンルのひとつです。東方シリーズやインベーダーゲームは、遊んだことがある人も多いと思います。

そこで私は来場した方全員に、楽しんでもらえるようなインベーダーゲームを高専祭へ向けて開発することにしました。

シューティングゲームを簡単に説明すると、飛んでくる敵に自分が操る機体から弾丸を発射して撃ち落とし、スコアを競うゲームです。しかし、シンプルなシューティングゲームは、開発し尽くされネタが切れてしまっています。そこで私は、AI 関係の研究室に所属していることもあり、顔認識を使いまウスもキーボードも使わず遊べるシューティングゲームを作ることにしました。“スペース 顔ベーダー”というアプリです。ブースに展示しているので是非遊んでみてください。



図 1 “スペース 顔ベーダー”タイトル

## 2. シューティングゲーム開発について

### 2.1. 顔認識の方法及びその速度による問題

ゲーム制作用のライブラリである OpenSiv3D (詳細な説明は 2 年渡邊天翔が行っています) には顔認識用の関数が定義されていて、今回のゲームではその関数を使いました。リスト 1 に

電気システム工学科 4 年 Nasatame  
顔認識関数の使い方を示します。

#### リスト 1 顔認識関数の使い方

```
Array<Rect> faces = image.detectObjects(  
    HaarCascade::Face,  
    5, Size(20, 20), Size(300,300));
```

顔認識関数は、Haar-like 特徴量を用いたカスケード分類器というアルゴリズムを用いて顔を認識しています。詳しい説明は本筋から外れるため参考文献の 2 を参考にしてください。

この関数は 4 つ要素を指定することにより画像から顔を検出することができます。1 つ目は認識する対象、2 つ目はどのくらい精密に確認するか、3 つ目は認識する最小のサイズ、4 つ目は認識する最大のサイズです。今回この関数は画面の端から端までを 5px ずつずらしながら、最小 20px\*20px から最大 300px\*300px サイズまで一つ一つの領域において顔の有無を確認していきます。

この関数は一度の実行につき、50000 回近く一定の領域に顔が存在するかを確認するため、膨大な時間がかかります。テストした際は関数を一度実行するのに約 4ms もの時間がかかりました。これは fps(1 秒間に何回画面を更新できるか)に換算すると 25fps になります。現在の一般的なゲームのほとんどが 60fps で動いていることを考えると、このままでは快適に遊べないと考えました。

私はその問題を解決するため非同期処理を使用することにしました。最近の PC は CPU という計算を司る部分が複数あります。基本的には同じゲームであれば一つの CPU のみが処理を行います。それは他の CPU がその間、処理を行わず待機していることを示しています。

その待機している CPU に、顔認識の処理を依頼することにしました。リスト 2 が変更後の顔認識の処理です。

### リスト 2 並列化した顔認識処理

```
Array<Rect> futureFaces =  
std::async(std::launch::async,  
[imagePtr] () {return imagePtr->  
detectObjects(HaarCascade::Face, 5,  
Size(20, 20), Size(300, 300));});
```

これにより、25fps だった更新速度は 40fps に改善しました。今回のことから、動作が重く途中の経過を見る必要がない処理は並列化することで高速化できるという大きな知見を得ることができました。またメンバ関数をスレッドで実行するにはポインタを使用する必要があることも分かりました。詳細については参考文献 3 が分かりやすいです。

## 2.2. 敵キャラクターの実装

今回のゲームでは複数種類の敵キャラクターを実装しました。その中でそれぞれの敵キャラクターに描画処理や更新処理など非常に似通った処理が必要になる場合が多数ありました。ほんの少し動きが違うだけのキャラクターを実装するのに if 文をいくつも並べ、コピー＆ペーストを繰り返すコードは読みづらく、書き換えづらくなってしまいます。そこで私は継承を使うことにしました。

継承とは親クラスのメンバ変数やメンバ関数を子クラスからも使用できるようにするものです。詳細については参考文献 4 を読んでください。

今回は Invader クラスを親クラスとして、ShakeInvader や StraightInvader などといったクラスを定義しました。これらの描画関数や更新関数などをまとめて Invader クラスとして実行するため、Array<Invader> に ShakeInvader などのクラスを入れたところ親クラスの描画関数や更新関数が実行されてしまいました。調べたところ、もし子クラスの関数を実行したいのであればポインタから実行しなければならないようです。

そこで、単なるポインタは扱いが難しいため扱いが容易なスマートポインタの一つである shared\_ptr を使用することにしました。定義は Array<std::shared\_ptr<Invader>>となりました。この工夫によりすべての敵キャラクターは、きちんと定められたキャラクター独自の振る舞いをしつつ、プログラム的には同じ方法で取り扱うことが可能になりました。



図 2 複数種類の敵キャラクター

## 3. おわりに

今回は高専祭への展示という目的があったためゲームルールが複雑になるものや操作が煩雑になるものについては導入しませんでした。しかしインターネットで配布する際は、ボスキャラやアイテムを導入したいと思っています。

## 参考文献

1. OpenSiv3D  
<https://scrapbox.io/Siv3D/>
2. 【入門者向け解説】OpenCV 顔検出の仕組と実践(detectMultiScale)  
<https://qiita.com/FukuharaYohei/items/ec6dc e7cc5ea21a51a82>
3. C++でメンバ関数をスレッドで実行する  
<http://tadaoyamaoka.hatenablog.com/entry/2018/02/20/223237>
4. 一週間で身につく C++言語の基本・継承  
<http://cpp-lang.sevendays-study.com/day6.html>

# プロコン旅行記



# プロコン旅行記

総合工学科Ⅲ類 1年 鈴木 雄裕

**Coming soon**

**11/12**

## 編集後記

Web 版 Rationale、いかがでしたか？初めての試みですが、色鮮やかな Rationale になったことで、より部員の活動が伝わりやすくなったと感じています。特に絵師の方々の記事をグレースケールで掲載するのは申し訳ない気持ちでいっぱいだったので、このような形で実現できたのは非常に喜ばしいです。

また、色が付くだけでなく後から記事の追加ができます。今年のように、プロコン旅行記を掲載できない場合でもみなさんに記事を届けることができます。11月12日に記事を追加するのでご期待ください！

最後になりますが、記事を仕上げてくれた部員のみなさん、文章の添削をしてくださった先輩方、忙しい中寄稿してくださった顧問の先生、そしてこの Rationale を読んでくださっているあなたに心からの感謝をこめて、編集後記とさせていただきます。

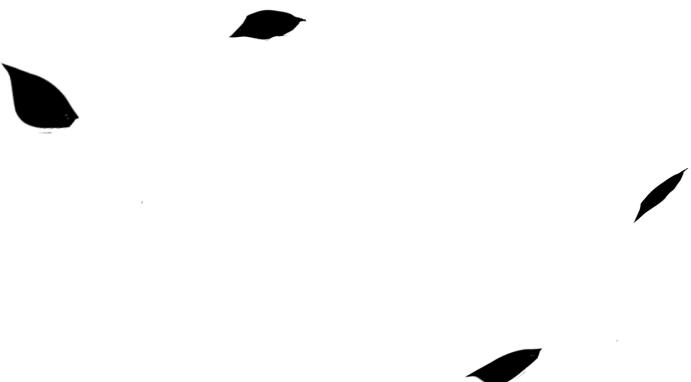
編集長 菊地 輝

## Rationale Volume32

発行日	10月27日
発行団体	仙台高専(名取) ソフトウェア研究部会
印刷所	仙台高専(名取) 事務棟複写室

### STAFF

代表	高橋 諒大
編集	菊地 輝
校正	高橋 諒大 Nasatame 大村 良多 佐藤 朋彦 匿名(猫好き) 木村 竜輔 hysk レモンを絞る人
	菊地 輝
表紙	根地戸 龍生
執筆	ソフトウェア研究部員
製本	ソフトウェア研究部員
プロコン旅行記	鈴木 雄裕



copyright S.R.D.G  
Software Research and  
Development Group