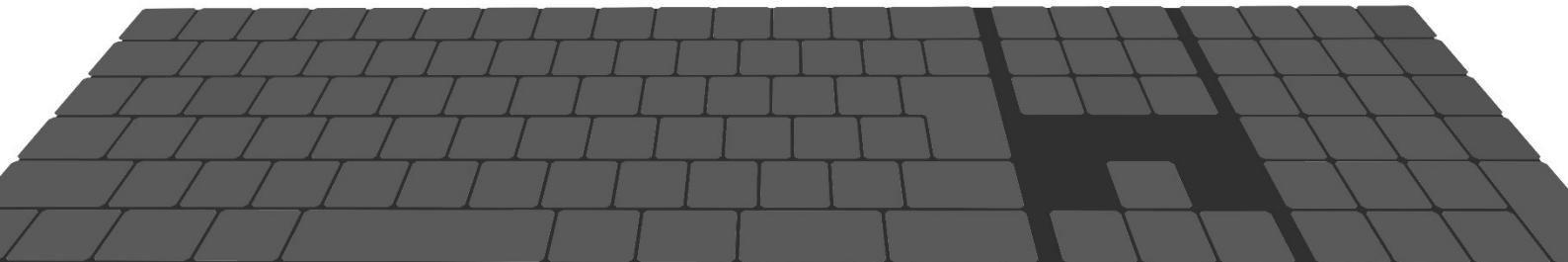


```
#include<iostream>
```

```
main void(){
```

# Rationale

```
}
```



## 目次

顧問挨拶	01		
代表挨拶	02		
<b>-活動報告-</b>			
小形 八雲	総合工学科 II類 1年	「音楽制作」	03
宮原 光敏	総合工学科 II類 1年	「的当てゲーム制作」	05
稻垣 颯起	総合工学科 II類 1年	「Medlyによる作曲」	07
大鷲 寛人	総合工学科 II類 1年	「単語練習プログラム」	09
佐藤 至	総合工学科 II類 1年	「チェスクロック制作」	11
渡辺 龍臣	総合工学科 II類 1年	「じゃんけんプログラム」	13
富山 結都	ロボティクスコース2年	「シンセサイザー操作」	15
鈴木 晴斗	ロボティクスコース2年	「OpenSiv3Dを使ったゲーム制作」	17
日野 綾瀬	機械・エネルギーコース2年	「テニスゲーム制作」	19
岡田 曜	機械・エネルギーコース3年	「オセロ制作」	21
鈴木 雄裕	機械・エネルギーコース3年	「ケーキ集め」	23
編集後記		25	

## オンライン実施について

ソフトウェア研究部会 顧問 北島宏之

先日、第31回全国高等専門学校プログラミングコンテストが、北海道の苫小牧工業高等専門学校主管でオンライン開催されたが、競技部門は開催中止となった。近年、ソフ研からは競技部門での入賞を目指にしていただけに、コロナ禍とはいえ顧問としてもとても残念に感じた。

しかし実際、コロナ禍の影響は大きく、本校でも前期第1Qは学校行事は全て中止、授業は遠隔授業となり、学生も教職員も困惑しながら対応に追われた。部活動も禁止され、夏休みにかけて運動部の東北地区大会や全国大会も一部を除き中止となったことは、ソフ研のプロコン不参加と同じく、致し方ない状況とは言え、特に5年生にとってとても寂しい状況となってしまった。第2Qから後期、本稿執筆時点の11月に入っても授業は対面と遠隔の混在した形態が続いているが、後期に入りようやく各授業の実施形態は落ち着きつつあるように思われる。早くコロナ禍が収束してくれることを祈るばかりである。

他方で、改めて授業や学校生活において、“対面”的重要性を感じられたことも事実である。特に実験や実習については手を使うことにこそ意味があるので、学生は登校のうえで出席し、教員側もコロナ予防を心掛けながら実施している。とは言え、私の授業はその意味では異質で、演習要素を含むプログラミングの授業であるのだが、前期から引き続き同時双方向型の遠隔形態としている。プログラミングの授業であるので、演習室にて学生はそれぞれ演習端末を前にプログラムを入力し実行する対面かつ演習の形態が本来の姿である。当初は、プログラミングの授業を遠隔で実施できるか不安しかなかったが、有難いことにプログラムの入力から実行までブラウザ画面で作業できる環境が無料で利用でき(※1)、学生の自宅における通信環境やデバイス状況の心配はあるものの何とか授業を遠隔実施できている。

出だしに戻るが、今年度のプロコンにおける課題部門と自由部門はオンライン開催された。対して、競技部門は、パソコン甲子園のように問題を出題して制限時間内に解くといった内容(ゆえに、例年の予選不通過者のオンライン参加と同じく今年度は本選もオンライン開催された)ではなく、例年参加学生が競技に実際に触れ、体験する要素もあることから、オンライン開催を見送ったのであろうと推察している。しかし、授業の遠隔実施やテレワークの推進など、いわゆる新しい生活様式を目指そうとする中、プロコン側準備の難しさや、出場各校のオンライン環境の心配など日々不安要素があったであろうことも重々承知の上で、なんとか競技部門もオンライン開催して欲しかったのが正直な気持ちである。昨年の第30回(於宮崎県都城市)でも、当日になって出場辞退チームが出るほど直前の台風の影響が大きかったが、オンライン開催の方法が作られていたのであれば、今年度の開催もまた変わったように思われてならない。プロコンも、各地開催で足を運ぶ楽しさもあるが、そろそろ新しい実施様式を考える時期に来ているように思われる。

(※1)<https://paiza.io/ja>

# 活動報告

会長挨拶

機械・エネルギーコース 3年 鈴木 雄裕

今年は COVIC19の影響により文化祭の中止、新入生の入部が遅れなどイレギュラーなことが重なった年となりましたが無事「Rational」を完成することが出来ました。

私達は全国高等専門学校プログラミングコンテスト(略してプロコン)という大会の競技部門に毎年参加しているのですが競技部門が中止になってしましました。「Rational」には毎年プロコン旅行記や開発したプログラムについての記事を載せているのですが大会が中止となっていましたので今年はそれらの記事はありません。私もこれらの記事を含めて開催を楽しみにしていたので残念に思います。

今年の「Rational」はゲームの制作についての記事が多いように思います。ゲームを作るため少なからずプログラミングに触れているため難しそうで読みにくい、と思うかたもいるかもしれません。それでも出来る限り読みやすくなるよう心掛けたので一回騙されたと思って読んでみて頂ければ嬉しいです。

最後に「Rational」の作成に関わった部員の皆さん、顧問の北島先生と佐藤先生、そしてこれを読んでくださる皆様にお礼申し上げます。ありがとうございました。

---

副会長挨拶

機械・エネルギーコース 2年 日野 綾瀬

「Rationale」を読んでいただきありがとうございます。今年は新型コロナウイルスの影響で、毎年参加していたプロコンの競技部門の中止、ゲームの展示や部誌の配布を行っていた高専祭の中止と、ソフトウェア研究部会にとって厳しい年となりました。そのような中でも、この「Rationale」を完成できたことがとても嬉しく思います。ぜひお楽しみください。

さて、私事になりますが、先月パソコン甲子園というプログラミングの大会にチームで参加してきました。難しくて解けない問題が多く、527 チーム中 178 位と、悔いの残る結果となりました。来年のパソコン甲子園、そしてプロコンでは好成績を残せるよう、より一層プログラミングの勉強に励みたいと思います。

最後になりますが、この「Rationale」を書いてくれた部員の皆様、顧問の北島先生と佐藤先生、今読んでいる皆様にこの場を借りて感謝申し上げます。

# 音楽制作

## 1. はじめに

私は歌声合成エンジン NEUTRINO を使って音楽制作をしました。このソフトは無償で配布されていて、公式ページからダウンロードすることができます。ただしこのソフトは歌声を合成するだけのソフトなので、楽譜を作成できるソフトを別途用意する必要があります。私は公式で推奨されていた Musescore3 を使いました。楽譜を打ち込んで歌詞をつけるだけで人間のような声を作ることができ、初心者でも触りやすいソフトです。

図1は制作途中の画面です。

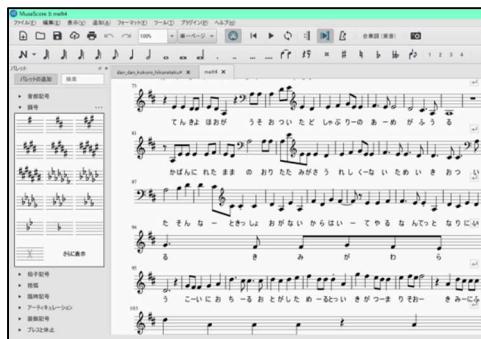


図1 制作途中

## 2. NEUTRINO

ダウンロードしたデータにあるNEUTRINOにはNEUTRINOとWORLDの2つがあります。このNEUTRINOというソフトは楽譜データから发声タイミング・音の高さ・声質・声のかすれ具合などをニューラルネットワークで推定し、実際の歌唱データからなる歌声ライブラリで合成するというものです。WORLDではNEUTRINOで作成されたファイルを元に音声波形を生成します。一方NEUTRINOは生成にかなりの時間を要します。

ダウンロードデータに同梱されている声は東北きりたん、謡子の2人です。AIにより人間が歌っているかと思えるほどの歌声を合成することができます。

総合工学科1年 II類 小形 八雲

## 3. 手打ちとmidiデータでのインポート

私は今回、有名な曲である「友～旅立ちの時～」、「メルト」、「いつも何度でも」の3曲を東北きりたんの声でカバーさせていただきました。メロディーを打ち込む際、「友～旅立ちの時～」は手で入力しました。楽譜が手元にあるとはいえ、目で見て1音1音入力するのはとても大変でした。「メルト」と「いつも何度でも」はmidiデータで配布しているサイトがあったので拝借し、Musescore3にインポートしました。midiデータでインポートするほうが圧倒的に時間がかかるず、ミスも少ないのでMusescore3で楽譜を打ち込む際はmidiデータでインポートするほうが良いです。ちなみにPDFファイルからもインポートできますが正しく入力されていることが少ないため、あまりおすすめはしません。手打ちに比べれば圧倒的に速いのですが、ミスに気付かないことが多いので注意しなければいけません。

## 4. 音と歌詞

メロディーを打ち込み終わると歌詞の打ち込みに移りますが、ここで注意するべき点としては発音と歌詞の打ち込みです。例えば「こんにちは」の「は」と「花」の「は」、この二つの発音の違いは日本語を勉強していれば理解できます。ただ機械には難しいので発音をするそのままの音で打ち込むように気をつけます。”わ”や、”を”などがその例です。

促音や拗音を除いて1音に二つの文字は発音できないので発音するタイミングに音を分けるのですが、これがとても大変でした。あとは短い音、たとえば”っ”のような音を入れる場合やメロディーが速い場合は音自身を短く変更したり、直後にプレスを入れたりなどで工夫しました。これによりスムーズな音の移り変わりを再現できました。

## 5. 出力

メロディーと歌詞の打ち込みが終了したらいいよ出力です。ちなみに NEUTRINO にはオンライン版、オフライン版の二つが存在し、オンライン版を使用するには Google アカウントの所持が必要になります。オンライン版、オフライン版ともに1GB ほどの容量を必要とします。オフライン版だと出力に時間はかかりますがこのあと説明する”調声”が楽になります。ただし、NEUTRINO の実行時間は 4 分の曲だと 20 分ほどかかります。なのでまずオンライン版で出力し、そのデータをダウンロードした後 NEUTRINO 調声支援ツールに読み込ませ、そこでオフライン版の NEUTRINO と WORLD を実行するのが最適だと思われます。

## 6. 調声

出力が終わると次は最後の工程である調声になります。この調声に私は NEUTRINO 調声支援ツールを使いました。このソフトではきりたんの声にビブラートをかけたり、音間の移動を滑らかにすることができます。これにより疑似的にスラーを再現することができます。さらにはブレスをつけることで更に人間味を帯びた声にすることができます。子音と母音の発音タイミングまで変更することができます。

ここからは自分の気が収まるまで調声して出力することの繰り返しでした。私は人が歌っているものを聞いて歌い方を参考にしています。音の上がり方や下がり方、ビブラートの音の周期や振幅などを参考にしています。より細かいところまで見るととても人間のように歌ってくれます。

図 2 は調声中の画面です。ここではピッチの編集をしていますがダイナミクス、音量の変更も可能でより感情の込もっているような歌い方も再現できます。

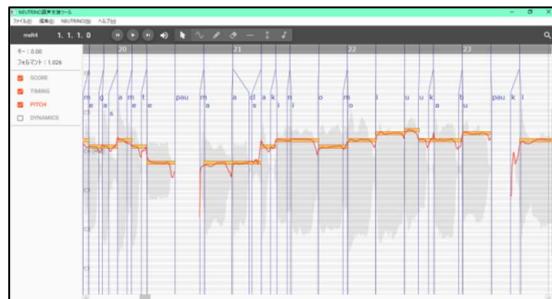


図 2 調声中の画面

## 7. 最後に

最初に言いましたが、楽譜を打ち込んで NEUTRINO、WORLD を実行するだけで人間が歌っているかのようなきれいな歌声が聞けます。AI は進化していると実感できるようなソフトでした。また私が使ったすべてのソフトは無料で配布されています。お金をかけずにここまでクオリティーのものができたときは本当に感動しました。これからもっときりたんに歌わせたいと思います。

### 参考文献

Musescore3

<https://musescore.org/ja>

NEUTRINO

<https://n3utrino.work/>

NEUTRINO 調声支援ツール

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

山の音楽(midi)アルバム

[http://momo1949.hobby-web.net/mm/mount\\_midi.html](http://momo1949.hobby-web.net/mm/mount_midi.html)

「メルト」で使用したカラオケ

<http://www.supercell.sc/music/3melt.zip>

蜜柑丼 --ボカラ口楽譜らしきものを作ってみた。

<http://kiko.fuyu.gs/micandonburi/>

# 的当てゲーム制作

## 1. はじめに

私は OpenSiv3D を使って、的当てゲームを作りました。ゲームを作ることにまだ慣れていないので、練習としてこのゲームを作成してみました。このゲームに使われている最低限の動作を説明していきたいと思います。

## 2. ゲームの動作とプログラム

### 2.1. ゲームの主な流れ

まず的当てゲームとして、基本的な挙動は的を左クリックしたときに、得点が入って的がランダムな座標に移動するという流れにしました。こうすることで、基本的なゲームに必要なプログラムの組み方を覚えました。

### 2.2. 的の表示

的は見やすく、直感的にわかるような赤色で作成しました。またランダムに座標を変えるようにプログラムを作りました。

### 2.3. 的がクリックされた際の挙動

次に的をクリックされた際の挙動を組みました。このプログラムはゲーム内で常時動かしておく必要があるため、while(System::Update())の中に入れました。この中に入れることで、プログラムをゲーム中に常時作動させることができます。その中でも特に的が左クリックされた際という条件が付いているので、while(System::Update())の中にif文を入れました。これによって条件が満たされた際に if 文の中のプログラムを作動させるプログラムを組むことができます。実際のプログラムは表1です。

総合工学科 II類 1年 宮原 光敏

表1 プログラムの一部

```
while (System::Update())
{
    Circle target(Scene::Center() +
        (targetPos * 100), 35);
    target.draw(Palette::Crimson);
    ClearPrint();
    Print << score;
    if (target.leftClicked())
    {
        targetPos = RandomVec2();
        ++score;
    }
}
```

### 2.4. 得点の表示

次に、的をクリックした際に増える得点の表示をします。これもゲーム内で常時動かす必要があるため、while(System::Update())の中に入れます。得点を増やすプログラムはすでに組んであるので、Print<<score;で表示させます。そうすることで、画面の右上に得点が表示されるようになります。

### 3. プログラムの大まかな流れ

ここまでプログラムを図1のフローチャートに示します。まず最初に、背景との得点を表示します。このとき的はランダムな座標にあり、得点は0となっています。また、的が左クリックされたときに、得点を1点増やして、的をランダムな座標に移動させます。

そして、得点が100になったときに得点を0

にして初期状態に戻すようにします。これがこのゲームのプログラムの大まかな流れです。実際のゲーム画面は図2です。

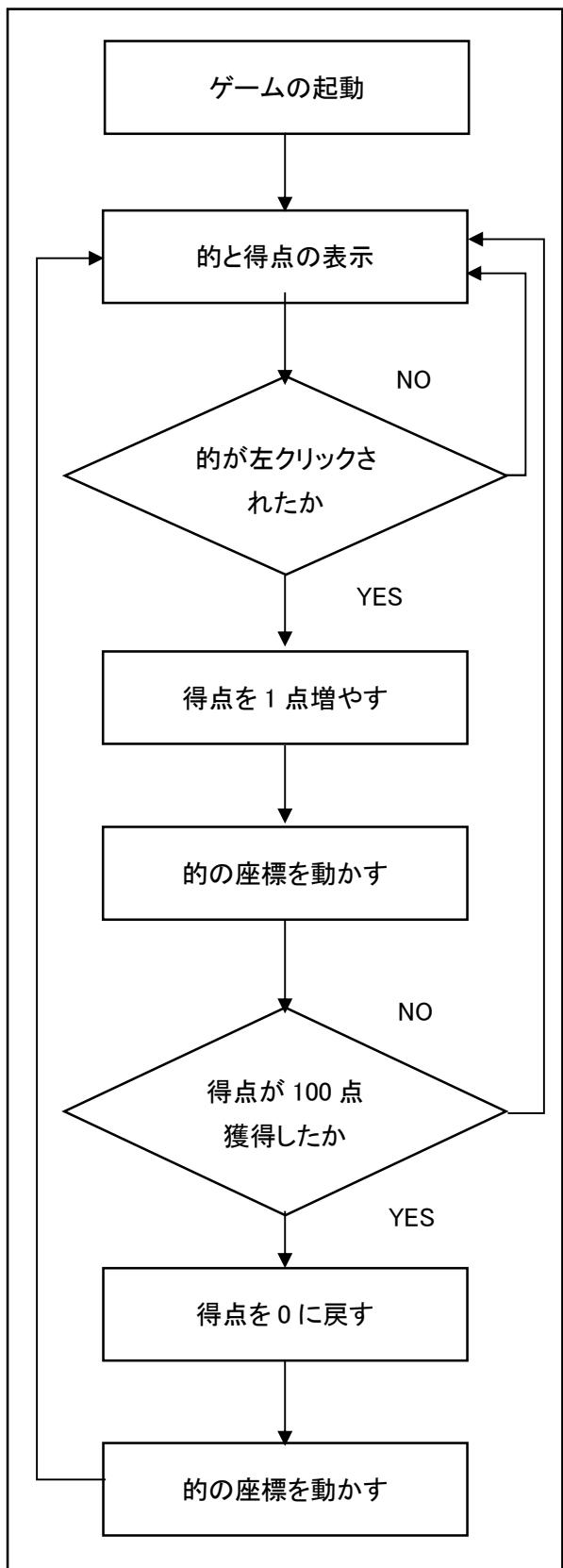


図1 ゲームのフローチャート

#### 4. おわりに

今回は初めてのゲーム制作でした。まだ、プログラミング言語を完全に覚えていないものあって、なかなか思った通りに動かないこともあります。それがあっても自分でゲームの大体の部分を作れたというだけで十分な満足感が得られました。

このゲームはプログラムとしても簡単に作られていますが、初めてプログラムで物を作る自分にとってはそれだけでも最初から作るには大変なものでした。これを踏まえてこれからはプログラミング言語の学習をより深く行っていきたいと思います。

来年も作る際には、プログラミング言語の関数などといったものも覚えて、先輩たちが作っているような更に出来のいいゲームを作りたいと思います。

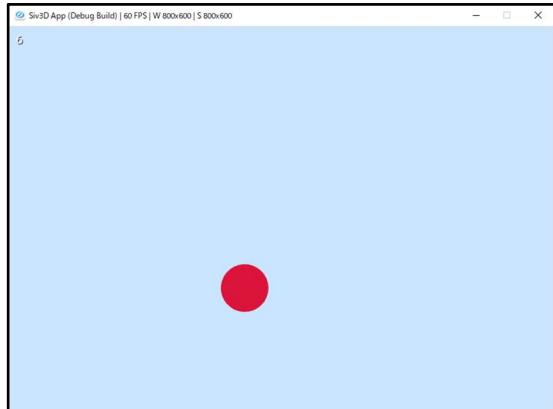


図2 ゲーム画面

#### 参考文献

OpenSiv3D

<https://siv3d.github.io/ja-jp/>

# Medly による作曲

## スマホアプリ Medly ではじめる簡単作曲

### 1. はじめに

この文書は「Medly」というアプリケーションを使った楽曲制作についてのレーショナルである。このアプリケーションは Android、MacOS、windows、iOS で使用可能である。今回説明するダウンロードの方法に関して、参考文献の「Android 用の Medly APK をダウンロード」「Medly pc ダウンロード」を参考にしている。今回は iPhone8 で作成した。図 1 は Medly のアイコンである。



図 1 アプリアイコン

### 2. Medly の使い方

まず Medly のダウンロードについて説明する。iPhone では App Store からダウンロードできる。Android は参考文献に記載したサイトからダウンロードする。PC の場合は nox app または Bluestacks app というエミュレータアプリケーションで使用できる。

### 3. Medly の仕様について

このアプリケーションの仕様について説明する。このアプリケーションを開くと 2 つのアルバムが表示される。「スターターテンプレート」と「開始」である。スターターテンプレートはアリ

総合工学科 II類 1年 稲垣 颯起

側が用意したデモが 8 曲用意されている。このデモは、はじめから使える楽器だけで作られた曲を見ることができる。楽器のセットは無料のスター以外にジャズやロック、クラシックといったポピュラーなものからヒップホップ、ディスコ、チルステップなどマイナーなものもある。他にもメルヘンバウンスや桜など様々な種類がある。BPM はある程度自由に変更でき、キーも変更可能である。また、音階がシンプル(制限なし)・長調・短調選択できるので初心者でも簡単に作曲できる。もちろん音の大きさは楽器ごとに変更可能で、コピーもでき、ストレスを感じることなく作業できる。そして、バーカウントという仕様があり、定めた場所が、決めた回数繰り返されるまで他の場所は同じ動きをするというものである。これを使うと何度も同じフレーズを繰り返したいときにとても便利である。

### 4. 曲作り

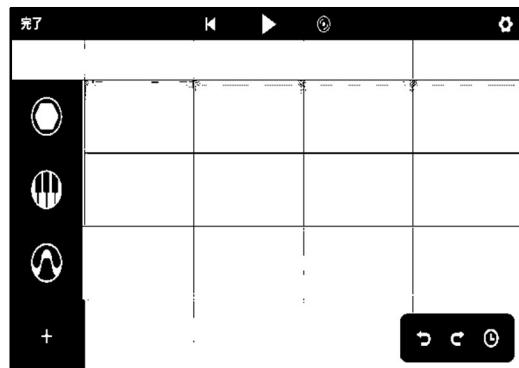


図 2 操作画面

実際に曲を作ったときの手順について説明していく。図 2 は操作画面である。作成した曲はテンポが BPM128 でキーは F。音階はシンプルで作成したものである。

まず適当なアルバムを開き、右上の設定の隣の+のマークをタップし空白を選択する。すると紺色の枠が左側にあり、水色の枠が右側に向かって続いている。紺色の枠が楽器選択で水色の枠が音を入力するセルである。まず右上の設定マークをタップする。すると、テンポ、キー、音階の順に設定欄が並んでおり、上記の設定で入力する。次に楽器の選択をしていく。今回はドラム、ピアノ、そしてウォンプという電子音で作ってみた。そしてAメロ、Bメロ、サビを決めて曲を作る。先ほど説明したバーカウントは、音の入力セルを開いた後に設定から選択してカウント回数を選ぶことが出来る。この時、同じ動きをする側ではなくメロディーなどの同じ動きをしないものを選ぶ。そして、いくつかの工程を終えて曲が完成した。オートセーブがあるので、保存などは不要である。終わったらアルバムに戻り、3点リーダーのところから名前をつけて完成である。この流れは図3に示した。

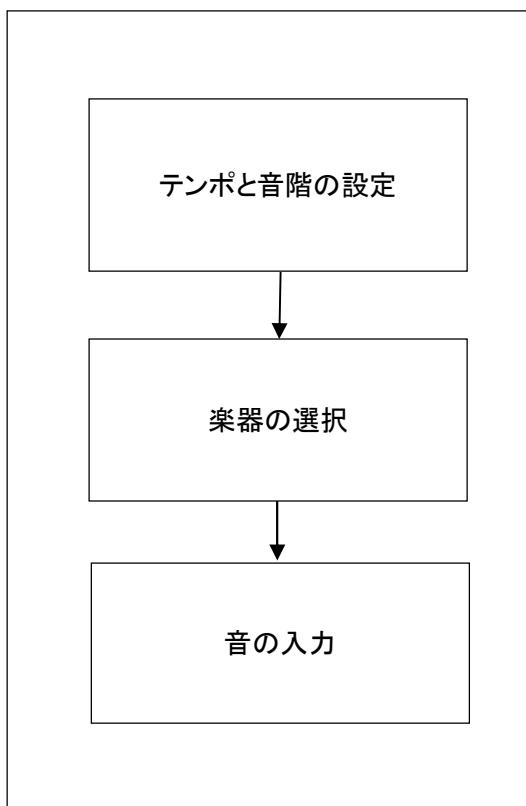


図3 作曲のながれ

## 5. おわりに

今回初めてレーショナルを書いたため、読みにくいところや、分かりづらいところがたくさんあったと思う。今回紹介しなかった楽器でも使えるものはまだたくさんあるので好きな楽器の組み合わせで楽曲を作りたい。今回紹介したこのアプリケーションは、私が中学生の頃から使っていて、予備知識がなくてもはじめることができる。誰でも作曲できると思うので興味があればぜひ挑戦してみてほしい。参考文献の欄にiPhoneとMacでダウンロードできるURLがあるのでそこからダウンロードできる。

## 参考文献

Android用のMedly APKをダウンロード  
<https://androidapp.jp.net/apk/940268124/medly>

「Medly」をApp storeで  
<https://apps.apple.com/jp/app/medly/id940268124>

Medly-Music Maker  
<https://medlylabs.com/>

図1のアイコン  
<https://is3-ssl.mzstatic.com/>

Medly pcダウンロード  
<https://windowsapp.tokyo/app/940268124/medly>

# 単語練習プログラム

～単語 10 問テスト版～

総合工学科 II類 1年 大鷲 寛人

## 1. プログラムの概要

このプログラムは、勉強したい英単語などを自分で追加し、それらの中から1問1答形式でランダムに10問出題されるものです。それを解くことで単語練習をすることを目的としています。

## 2. 制作について

### 2.1. 制作の動機

私がこのプログラムを作ろうと思ったきっかけは2つあります。

1つ目は、この部誌を書くための題材にするためにプログラムを組もうと思ったからです。そして2つ目は、私自身が英単語の暗記がやや苦手であるため、単語を練習できるプログラムが欲しいと思ったからです。

### 2.2. 使用ライブラリ

このプログラムを制作するにあたり、"OpenSiv 3D(0.4.3)"という、ゲームを作ることに長けたライブラリを使用しました。

## 3. デザイン

起動画面は図1に、リザルト画面は図2に示します。ゲーム画面はモノクロで非常にシンプルな構成になっています。そしてこのようなデザインにした理由は2つあります。

1つ目はカラフルにするメリットがあまり無いと思ったからです。2つ目は、単語練習に集中しやすくするためです。

## 4. プログラム本体

### 4.1. フローチャート

このプログラムの大まかな流れを図3に示します。このフローチャートでは、1問1答形式で画面上部に日本語で表示された意味を持つ英単語を入力します。解答するとその場で正誤判定が下されます。10問解き終わると成績が表示されます。入力した答えが合っているかどうかは音で分かれます。

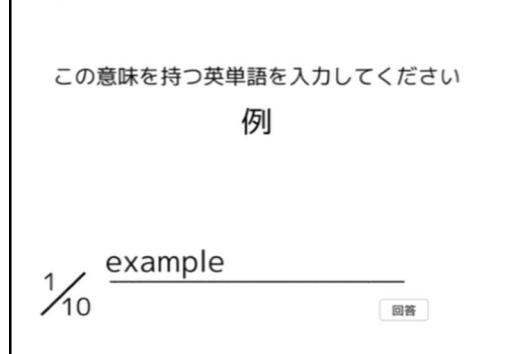


図1 起動画面



図2 リザルト画面

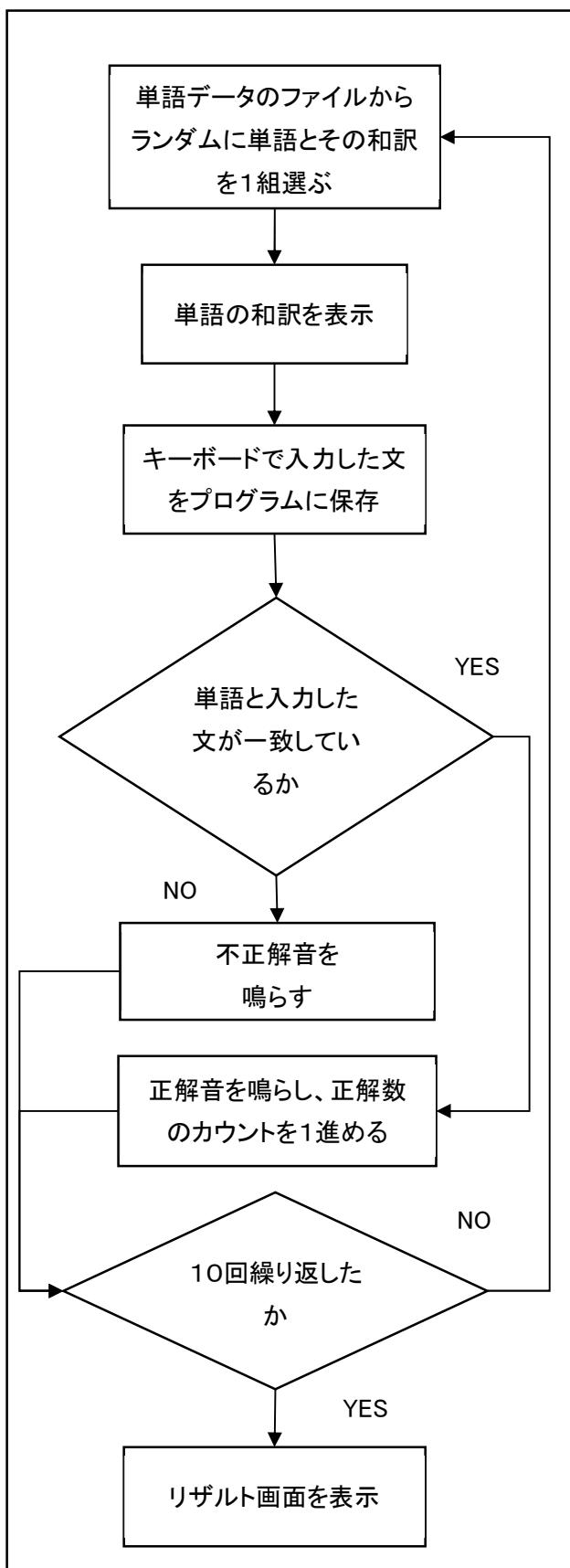


図3 フローチャートの一部

#### 4.2. プログラム

左のフローチャートの一部のプログラムをリスト1に示します。上のフローチャートの1番上の“単語データのファイルからランダムに単語とその和訳を1組選ぶ”という部分と、1番下の“リザルト画面を表示”という部分以外のプログラムです。

#### リスト1 プログラムの一部

```

1. if (i <= 10)
2. {
3.     font2(word2).drawAt(moji2, Palette::Black);
4.     TextInput::UpdateText(text);
5.     if (word3 == text)
6.     {
7.         maru = maru + 1;
8.         seikai.playOneShot();
9.     }
10. else
11. {
12.     fuseikai.playOneShot();
13. }
14. }

```

#### 5. おわりに

今回のプログラミングで苦労したことは、どのようなプログラムを書けば単語のデータを入れるファイルの書き込みと読み取りができるのかという問題を解決することでした。

そして反省している点は、“練習”とタイトルに付けていたのにも関わらず、テストモードしか作っていなかったことです。

これから抱負として、プログラミング技術を向上させつつ、それを活用するためにこのプログラムを更新していくこう思います。最終的には自分の納得する形で完成させることを目標に頑張りたいと思います。

#### 参考文献

Siv3D: クリエイターのための C++ ライブラリ

<https://siv3d.github.io/ja-jp/>

# チェスロック制作

## 1. はじめに

今回私は GitHub というサイトにあった SacuL さんのチェスゲームにチェスロックの機能を追加しました。チェスロックとは自分のターンの制限時間を計測してくれるもので、一般的なチェスの大会では必ずと言っていいほど使用されています。今回がはじめてのプログラミングで手間取ったりしてしまった部分もありますが、無事完成できて嬉しく思います。では次から私の作成したチェスロックのプログラミングの内容について紹介していきます。

## 2. Unity について

Unity について紹介していきます。Unity とはゲームエンジンの一つでプログラミングには C# というプログラミング言語を使用します。Unity では 3D ゲームを作成することができ、様々なゲームで利用されているゲームエンジンです。条件こそありますが、無料で使えるため、興味のある方は是非ダウンロードしてみてください。

## 3. 今回のプログラミングの仕様

今回制作したプログラムはプレイヤーのターン中に 60 秒間操作がなかった場合に時間切れで負けるように設定しました。なぜ制限時間を 60 秒にしたのかというと、一般的な試合では 1 試合中に持ち時間 15 分で持ち時間が切れたたら 1 ターンにつき 30 秒なのですが、それだと難易度が高くなってしまうので、その倍の時間をとつてターンごとに時間をリセットすることでゆっくり考えられるようにしました。なお駒を選択する、駒を移動する、ターンが切り替わることでタイマーがリセットされる仕組みになっています。残り

総合工学科 II類 1年 佐藤 至  
時間は図 1 のように表示されます。

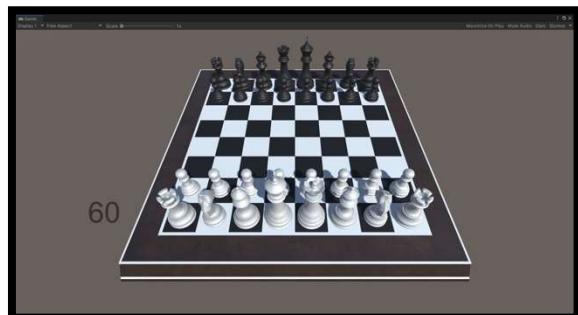


図 1 ゲーム画面

## 4. 制限時間の計測

では次に制限時間の計測についてのプログラムについて解説していきます。図 2 を見てください。まず 2 行目が制限時間を定義しているプログラムです。4~8 行目がカウントダウンのプログラムで、制限時間が 0 になるまで 1 秒ごとに制限時間を 1 秒減らすのを繰り返すプログラムになっています。制限時間が 0 になったら 9 行目の時間切れ負けの処理を行います。

```
1. IEmulator CountDown(){
2.     int Time= 60;
3.     timeCountText.text=Time . ToString();
4.     While (TimeCount>0){
5.         Yield return new WaitForSeconds(1);
6.         TimeCount--;
7.         timeCountText = Time.ToString();
8.     }
9.     TimeOut();
10. }
```

図 2 カウントダウンのプログラム

## 5. 時間切れでの負けのプログラム

今度は時間切れでの負けの処理をするプログラムを紹介していきます。図 3 を見てください。このプログラム全体で時間切れでの負けから盤面のリセットまで全て終わらせるようにできています。このプログラムは 3~6 行目のプログラムで後攻のターンか否かでどちらが負けるか判断する仕組みになっています。後攻のターンなら先攻の勝ち、そうではないなら後攻の勝ちという処理をしています。そして 7~13 行目までが盤面をリセットするプログラムです。仕組みとしては一旦全ての駒を消去した後に並べ直すようにしています。

```
1. private void TimeOut() {  
3.     if (isWhiteTurn)  
4.         Debug.Log("Black wins");  
5.     else  
6.         Debug.Log("White wins");  
7.     foreach(GameObject go in inactiveChessman)  
    {  
8.         Destroy(go);  
9.     }  
10.    isWhiteTurn = true;  
11.    BoardHighlights.Instance.HideHig  
12.    SpawnAllChessmans  
13. }
```

図 3 時間切れでの敗北を処理するプログラム

## 6. 終わりに

今回初めて何かを制作するためのプログラムを書いたのですが、自分で考えた通りに動くプログラムを書くのが意外と難しく、どうすれば思い描いた通りに動くのか調べていたりしたら、かなり時間がかかってしまいました。ですが完成した時の達成感はとても大きく、大変だったという気持ちよりも嬉しい気持ちの方が大きかったです。来年はさらに知識をつけて、1 から自作のゲームを制作できるようになりたいです。

## 参考文献

今回使用したチェスゲームのプログラム

3D-Chess-Unity

<https://github.com/SacuL/3D-Chess-Unity>

初心者でも簡単にできる！Unity でカウントダウンタイマーを作成する方法

<https://freesworder.net/unity-countdown-timer/>

Unity: カウントダウンタイマーを作る

<https://dianxnao.com/unity%EF%BC%9A%E3%82%AB%E3%82%A6%E3%83%B3%E3%83%88%E3%83%80%E3%82%A6%E3%83%83%B3%E3%82%BF%E3%82%A4%E3%83%9E%E3%83%BC%E3%82%92%E4%BD%9C%E3%82%8B/>

一定の時間間隔で処理を実行する方法まとめ（時間制御）

<https://qiita.com/Nagitch/items/fb9157b1cb27f3d37696>

Unity 公式サイト

<https://unity3d.com/jp/get-unity/download>

# じゃんけんのプログラム

総合工学科 II類 1年 渡辺 龍臣

## 1. はじめに

僕は簡単なプログラミングから始めようと思い、じゃんけんのプログラムを組んでみました。

理由は、初めて自分で作るプログラムだったので簡潔な内容で、新しく使う要素を盛り込んだプログラムを組みたいと考えたからです。

## 2. プログラム

### 2.1. プログラムの内容

右の図1はプログラムの流れを模したフローチャートです。

このプログラムは、ランダム関数を宣言することによって乱数ライブラリを使っています。これによって擬似の対戦相手を作りじゃんけんを行うことができます。

これで数字を生成し、後述するプレイヤーの数字とコンピュータ側の数字との和で、考えることのできる9つの結果にそれぞれ当てはめられています。

プログラムの全体としては、まずプレイヤー側に変数 `a` に 10、20、30、の 3 つの数字の中から入力させます。この 3 つの数字には、小さい方から「グー」「チョキ」「パー」の意味があり、それを数字の入力画面に提示することでプレイヤーは任意の手を選ぶことができます。

先ほど触れた 9 つの結果ですが、1 人で行うじゃんけんの結果は最大 9 通りであることに由来します。

この 9 通りを 2 枝の数字で定義していて、`switch case` 文によって当てはまる 1 通りを選び対応した文字を出力させます。これが一連の流れです。

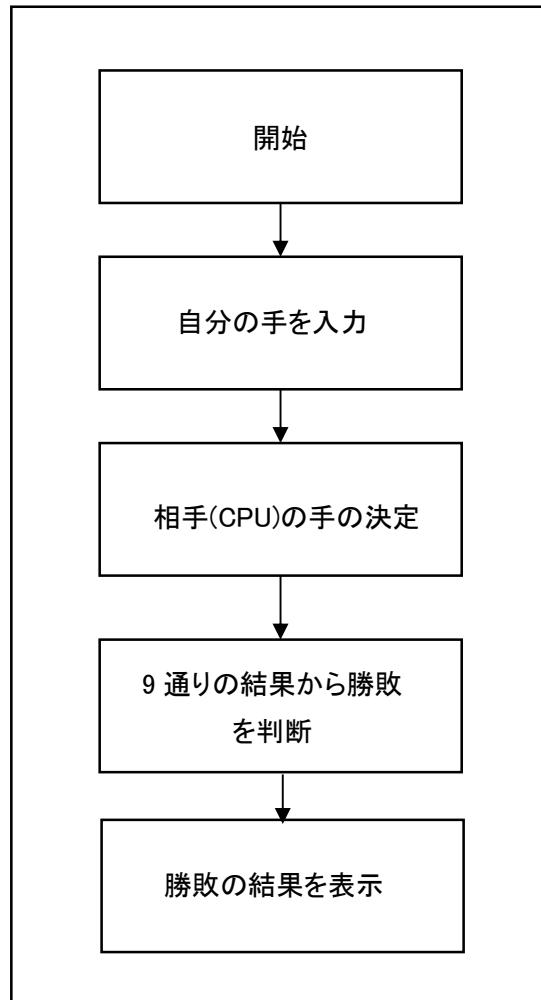


図1 プログラムの流れ

### 2.2. コードを書くにあたって

このコードでは一つ大きな問題があり、それは正確な勝敗の判断がされないという点でした。

原因として、共通の値を持っていると思っていたランダム関数の変数を余分に呼び出していたことで値が変わってしまっていたことが原因でした。これを直すためにランダム関数の変数 `rand100(mt)` を別な変数に入れることで値が変

わらないうようにしました。

その過程でもコードの処理順でうまく保存できずプログラムが動かないこともありました。下のリスト 1 は問題があつた部分を修正したコードです。

### リスト 1 修正後のコード

```
1. random_device rnd;
2.     mt19937 mt(rnd());
3.     uniform_int_distribution<>
4.         rand100(1, 3);
5.     cout << "相手の手" << endl;
6.     cout << "グー,1 チヨキ,2 パー,3"
7.         << endl;
8.     int b = rand100(mt);
9.     cout << b << "\n";
10.    //9 通りの処理
11.    switch (a+b) {
12.        case 11:
13.            cout << "draw" << endl;
14.            break;
15.        case 12:
16.            cout << "win" << endl;
17.            break;
18.        case 13:
19.            cout << "lose" << endl;
20.            break;
21.        case 21:
22.            cout << "lose" << endl;
23.            break;
24.        case 22:
25.            cout << "draw" << endl;
26.            break;
27.        case 23:
28.            cout << "win" << endl;
29.            break;
30.        case 31:
31.            cout << "win" << endl;
32.            break;
```

```
33.        case 33:
34.            cout << "draw" << endl;
35.            break;
36.        default:
37.            cout << "入力された値に誤り
38.                があります。" << endl;
39.            break;
40.    }
```

### 3. おわりに

繰り返しになってしまいますが、このプログラムは簡単で、乱数などまだ自分が使つたことのない関数を利用して、新しいプログラムを組もうと思いました。

途中大きな問題が見つかり先輩方や友人に解決するための方法を聞くこともありましたが、一緒に考えてくれた方々のおかげできちんと動いてくれました。

C++を学び始めて約 4 ヶ月程度のまだ初心者の部誌をここまで読んでいただきありがとうございます。

来年は Unity による制作を視野に入れています。まだまだプログラミングは始めたばかりですが来年以降もよろしくお願ひします。

### 参考文献

乱数の生成、出力について

<http://vivi.dyndns.org/tech/cpp/random.html>  
の指定範囲乱数生成の項目から一部コードを  
引用しています。

# シンセサイザー操作

ロボティクスコース 2年 富山 結都

## 1. はじめに

私はシンセサイザーの TAL NoiseMaker を使って音を作りました。この TAL NoiseMaker はプラグインで、動かすためのソフトに cubase elements を使いました。ここではトランペットの音を作っていく、シンセサイザーの使い方を書ききます。シンセサイザーは他のソフトでも似通った機能が多いので、ここでの知識を他のシンセサイザーでも生かせると思います。

## 2. シンセサイザーとは

シンセサイザーは音を作る楽器です。オシレーター、フィルター、アンプの 3 つを操作して音を作ります。音を作る楽器なので、ピアノや管楽器やドラム他には、シンセサイザーでしか出せない音も作ることができます。

## 3. オシレーターについて

オシレーターとはピッチ、つまり音程を司ります。このシンセサイザーでは、OSC1 と OSC2 の所です。図 2 を参照してください。そして波形も司ります。波形には代表的なものが 3 つあります。もっとも使われているのが、ノコギリ波(saw)です。この波形は、ストリングスやプラスなどに使われます。さらに矩形波(square)と、パルス波(pulse)もあります。矩形波はファミコンなどで使われた音で、印象的な音です。パルス波は上下の幅の違いで音が変わり、長さの違いが大きくなるほど音が細くなります。この調整は PW のつまみで設定できます。他には三角波(triangle)やサイン波(sine)があり、ノイズ(noise)もあります。波の形は図 1 です。

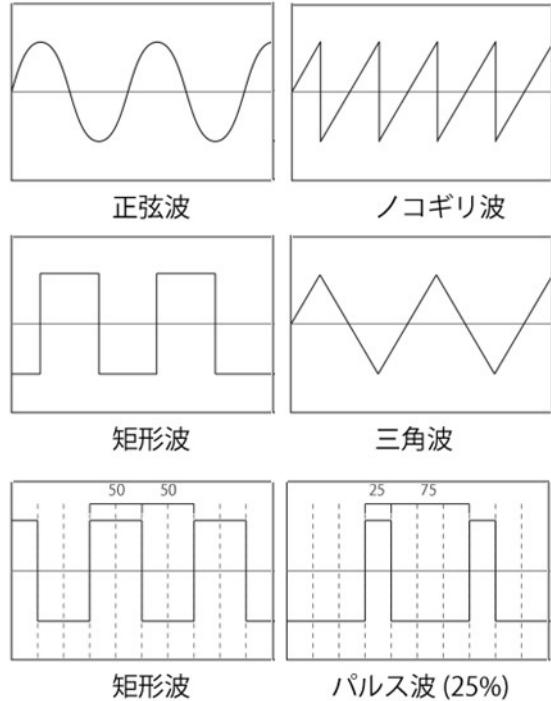


図 1 波形の種類

オシレーター本来のピッチ設定を解説します。設定は OSC1 の TUNE で操作でき FINE で微調整できます。ベースなら -24(-2 オクターブ) エレクトロピアノなら 0(オクターブ変化なし) のように設定できます。

## 4. フィルターについて

フィルターでは、音色を変化させます。3 つ有名なフィルターがあり、よく使うのはローパス(LP)です。これは CUTOFF を使って高音域から音を削れます。そしてハイパス(HP)は低音域から削り、バンドパス(BP)は残すところを指定しそれ以外を削ります。そしてフィルターにはとても大事な RESONANCE(RESONO)のつまみがあります。共振させるつまみで、“ビヨーン”という、これでしか出せない音を出すことが出来ます。

RESONANCE を半分ほどに設定してから CUTOFF をいじると、いかにもシンセサイザーという音を出せるのでとても面白いです。

## 5. アンプについて

アンプは音量を司り、ここでは ADSR の所で操作できます。A(アタック)は音が出てからの最大音量に達するまでの時間を設定します。D(ディケイ)では最大音量から減衰していく時間を決め、S(サステイン)はディケイ後の音量を決めます。R(リリース)は音が 0 になるまでの時間を設定します。

## 6. 実際に作ろう

作るときに基本となる型があると作りやすいので図 2 の通りになるよう設定します。MASTER の OSC1 と VOLUME は音量設定なので好みで設定します。そして MASTER の VOICES で同時に鳴らせる音の数を決めるので mono ではなく最大の 6 にした方がいいです。この型が作れるようになれば音を作るのも簡単です。

図 3 にトランペットの作り方があるので設定を施しましょう。ここでは今まで使ってこなかった特殊なつまみ destination(ADSR の DEST)を使います。このつまみは音が出た後にピッチを変化させるものです。0 で変わらず、-1 に近づくほど



図 3 トランペットレシピ

音のピッチが下がってから設定した音になり、1 に近づくほど音のピッチが上がってから設定した音になります。今回の設定は最初音が下がってから上がり、現実のトランペットに近づくようになっています。

## 7. おわりに

私は昔シンセサイザーを触りましたが、第一印象はとっつきにくいと感じました。大量のつまみが並んで気圧されてしまい触るのをやめしていました。しかし、今回一つ一つのつまみの意味を調べていたら難しさはなく、画面がすっきりとして無駄がないと感じました。今回使わなかった ENVELOPE EDITOR と CONTROL といった機能を使えばよりいろんな音を作れます。シンセサイザーを触るだけでもおもしろいので、ぜひ軽い気持ちで使ってみて下さい。



図 2 スタートレシピ

## 参考文献

### DTM 超初心者講座

[https://dtm-hyper.com/synthesizer/synth\\_3.html](https://dtm-hyper.com/synthesizer/synth_3.html)

クリエイターが教えるシンセサイザーテクニック

野崎貴郎

# OpenSiv3D を使ったゲーム制作

## 1. はじめに

私は今回ゲームを制作しました。使用した環境は以下の2つです。

- Visual Studio2019
- OpenSiv3D(v0.4.3)

## 2. 制作したゲーム

まず初めに制作したゲームについて説明します。プレイ中の画面は図1に示します。このゲームがスタートすると、四方から手が流れてきます。この手と一緒に表示されている円がバーに接しているときに、その手が指さしている方向に対応した矢印キー(例えば下方向に指さしているなら、下矢印キー)を押すと、得点が得られます。プレイヤーは高得点を目指します。得点は中央にいくほど高得点ですが、手が中央に表示されている顔に接してしまうとゲームオーバーとなってしまいます。

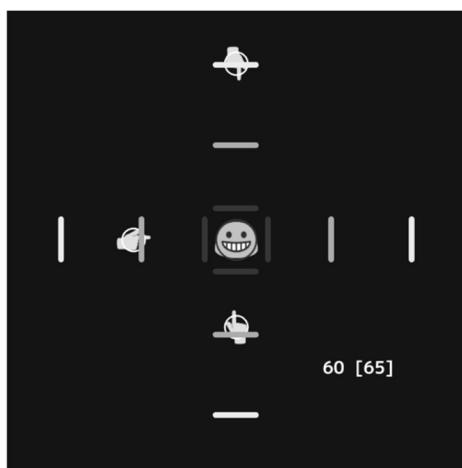


図1 プレイ画面

## 3. 制作過程

次に制作過程について説明します。ここでは、下方向に向かって流れる手に着目します。

ロボティクスコース 2年 鈴木 晴斗

### 3.1 生成と描画

リスト1は手を生成するコードです。まず生成間隔の初期値 spawntime と、経過時間を加算していくための変数 time、そして座標を格納する配列 hands を定義します。time には Scene::DeltaTime() を使い、直前のフレームからの経過時間を加算していきます。time が spawntime を上回ったとき、hands に手の x 座標と y 座標が格納され、time から spawntime が減算されます。この一連の流れがメインループ内で繰り返されます。

#### リスト1 手を生成するコード

```
Array<Data> hands;
const double spawntime = 4.0;
double time = 0.0;
while (System::Update())
{
    const double deltaTime =
        Scene::DeltaTime();
    {
        time += deltaTime;
        while (time >= spawntime)
        {
            Data data;
            data.pos.x = 400;
            data.pos.y = -50;
            hands << data;
            time -= deltaTime;
        }
    }
}
```

描画ではまず手のテクスチャと流れてくるスピードを定義します。テクスチャは絵文字を使用しました。次に座標が格納された配列を for 文を使い、ひとつずつ参照します。その y 座標(左右から流れてくる場合は x 座標)に経過時間と流れてくるスピードを掛けたものを加算していくことで、下方向に向かって流れる描画ができます。またこのとき座標を中心として、手と一緒に表示される円も描画します。

### 3.2 得点とゲームオーバー処理

得点の取得には、まず最初に得点バーの定義(色、形、表示位置)と描画を行います。今回のゲーム制作でこの得点バーの定義が一番大変でした。for 文を使い、配列をひとつずつ参照していきます。次に 3.1 で描画した円と得点バーが交差しているかを調べます。if 文を使い、交差していくて対応したキーが押されている場合には、スコアの変数に得点を加算します。そして、その手の座標を配列から削除します。これと同じようにして、顔と手の円との交差判定も行います。bool 型を使い、交差したときにゲームオーバー処理(リスト 2)を実行します。この処理では、clear() 関数を使った配列の全要素の削除、また max() 関数を使い現在のスコアとハイスコアを比較し、上回っていればハイスコアを更新します。

#### リスト 2 ゲームオーバー処理

```
if (gameover)
{
    hands.clear();
    highscore = Max(highscore,nowscore);
    nowscore = 0;
}
```

生成される毎に少しずつ、一定の値を加算していきます。こうすることですっと同じスピードの単調な得点回収作業ゲームではなくなりました。ただずっと加算され続けるのは速くなりすぎてプレイできないので、初期値をいくらか越えたときと、ゲームオーバーになったときは値を元に戻すようにします。

次にエフェクトの再生です。まずキーを押したときに音が鳴るようにします。Audio でピアノを使用し、得点の低いほうからド、ミ、ソとなるようにしました。また画面左下に表示されているスコアの上に、取得した得点に応じた数字が一瞬表示されるようにしました(図 2)。



図 2 スコアエフェクト

### 5. まとめ

今回は OpenSiv3D を使った 2 度目のゲーム制作でした。前回の制作時に比べて、より理解を深めることができたと感じます。制作途中でゲーム画面のサイズ変更をしたことで、座標を変更しなければならないということがあったので、次回からはしっかりと内容や外観を決めてから制作に取り掛かろうと思います。

### 参考文献

OpenSiv3D リファレンス

<https://siv3d.github.io/ja-jp/>

### 4. ひと手間加える

以上がこのゲームの大まかな構造です。しかしこのままだと、少しつまらないと私は感じたので、いくつか手を加えたいと思います。

まず次第にスピードが速くなるようにします。スピードの初期値を別の変数にコピーし、手が

# テニスゲーム制作

## 1. はじめに

私は OpenSiv3D を使用して、テニスのゲームを制作しました。テニスといっても本格的なものではなく、「PONG」を元にした、簡単なゲームを制作しました。

## 2. 「PONG」について

「PONG」は、ATARI 社が 1972 年に発売した世界初のビデオゲームです。ゲーム内容としては、2 人のプレイヤーがダイヤルを回すことでパドルを操作し、ボールを打ち合うという、非常にシンプルな内容です。このゲームは当時大流行し、現在ではリメイク版をゲームセンターなどで遊ぶことができます。

## 3. ボールと壁の接触判定

ボールと壁の接触判定のプログラムを、リスト 1 に記載します。

### リスト 1 ボールと壁の接触判定

```
1. if ((ball.y < 0 && ballVelocity.y < 0)){  
2.     ballVelocity.y *= -1;  
3. }  
4. if ((ball.y > 600 && ballVelocity.y < 0)){  
5.     ballVelocity.y *= -1;  
6. }
```

1 行目はボールが上側の壁に当たったかどうかを判定しています。当たった場合、2 行目でボールの速度の y 成分に-1 をかけることで、ボールを反射させています。4 行目からは、ボールが下側の壁に当たったかどうかの判定と、実行する処理です。

機械・エネルギーコース 2年 日野 綾瀬

## 4. ボールとパドルの接触判定

ボールとパドルの接触判定のプログラムを、リスト 2 に記載します。

### リスト 2 ボールとパドルの接触判定

```
1. if (ballVelocity.x < 0 &&  
2.     paddle1.intersects(ball)){  
3.     ballVelocity =  
4.         Vec2(-ballVelocity.x,(ball.y -  
5.             paddle1.center().y)*10)  
6.         .setLength(speed);  
7. }  
8. if (ballVelocity.x < 0 &&  
9.     paddle2.intersects(ball)){  
10.    ballVelocity =  
11.        Vec2(-ballVelocity.x,(ball.y -  
12.            paddle2.center().y)*10)  
13.            .setLength(speed);  
14. }
```

1 行目、2 行目はボールが 1P 側のパドルに当たったかどうかを判定しています。当たった場合、まず、4 行目と 5 行目でボールの速度の x 成分をマイナスに、y 成分をパドルの中心の y 座標とボールの y 座標との差を 10 倍した値にします。次に、6 行目で x 成分と y 成分の値を、あらかじめ定義しておいた変数 speed の値だけ倍にします。そうすることで、パドルにボールが当たった場所によって、ボールが反射する向きが変わるようにしています。今回の場合、ボールが当たった位置が、パドルの中央に近いほど、まっすぐ飛ぶようになっています。8 行目からは、ボールが 2P 側のパドルに当たったかどうかの判定と、実行する処理です。

## 5. ゲームの流れ

フローチャートを図 1 に示します。タイトル画面からゲームをスタートすると、画面上部からボールが発射されます。その後、どちらかのプレイヤーに得点が入るまで、パドルの移動からボールとパドルの描画までの処理を繰り返します。得点が入った場合、パドルの位置が中央に戻されて、再びボールが発射されます。どちらかのプレイヤーが 7 点獲得した時点で、勝敗を表示します。その後 Enter キーを押すことで、タイトル画面に戻ることができます。

実際のゲーム画面を図 2 に示します。

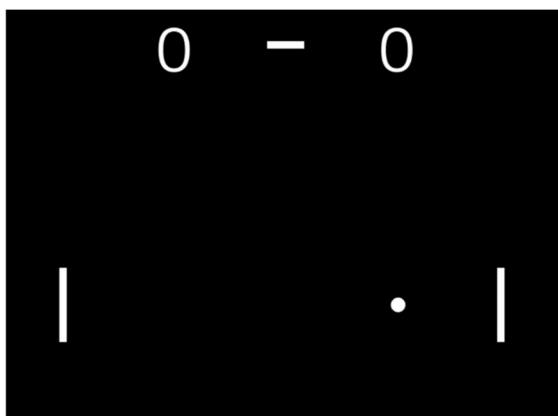


図 2 実際のゲーム画面

## 6. おわりに

私は去年のレーショナルでプロコン旅行記を書いたため、今年初めて活動報告を書きました。最初は格闘ゲームやカードゲームを作りましたが、制作期間や己の実力を考えた結果、「PONG」のようなゲームを作ることにしました。締め切りに追われながらのゲーム制作は、なかなか大変でしたが、ゲームを作る楽しさが分かった気がしました。来年はもっと面白いゲームを作れるようプログラミングの勉強を続けていきたいと思います。この文章を締め切りギリギリに書いていることは反省しています。

最後まで読んでいただき、本当にありがとうございました。

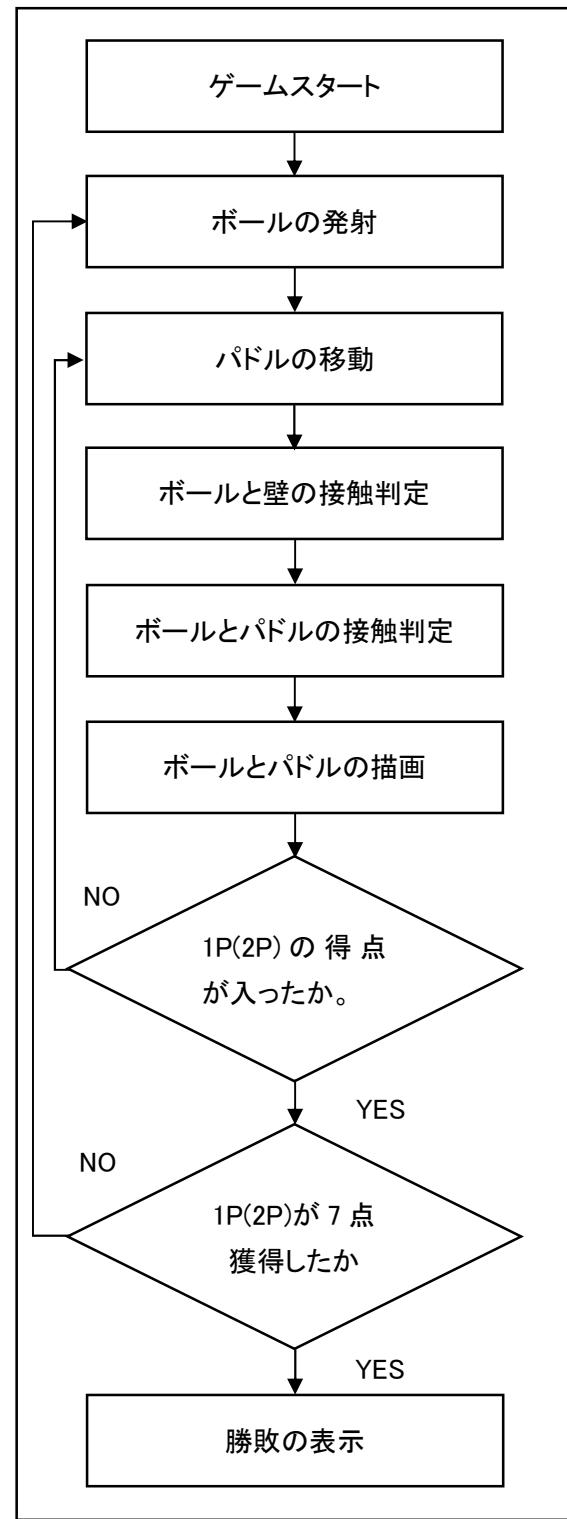


図1 フローチャート

## 参考文献

OpenSiv3D リファレンス

<https://siv3d.github.io/ja-jp/>

# オセロ制作

## 1. はじめに

私は OpenSiv3D を使ってオセロゲームを制作しました。今回は C++の練習も兼ねてこのフレームワークを使用しました。ここでは制作したゲームのうち裏返す判定をピックアップして解説していくこうと思います。

## 2. OpenSiv3D とは

OpenSiv3D とは、“C++で楽しく簡単にゲームやメディアアートを作れるフレームワーク”です。以前は Siv3D と呼ばれていたフレームワークが改良され、現在の OpenSiv3D が誕生しました。

## 3. オセロ

今回制作したオセロは基本的に2つの判定が行われます。

1つ目の判定は相手を裏返せるマスを調べる判定です。もし相手を裏返せるマスがなかったときには相手のターンにするかゲームを終了させます。この判定の一部をリスト1に示します。

2つ目の判定は置いたマスからどこまで裏返せるのかを決める判定です。この判定の一部をリスト2に示します。

### 3.1. コードの説明

リスト1の3行目から23行目で、どこのマスに置けるかを探す判定を行い、置けるマスがあれば true を返します。

リスト2の3行目から18行目では置いたマスからどこまで裏返せるのか数え、20行目から22行目で、数えた分だけ相手マスを裏返します。リスト1とリスト2のコードでは、MASU[i][k]に自分マスの数字が入り、白が1、黒が2、空白が0

機械・エネルギーコース 3年 岡田 曜としています。また、どちらも右側の判定のみなので、この判定を全方向で行います。

このようにして完成したゲームは図1のようになります。Rational では分かりませんが、置けるマスは赤色で塗りつぶしています。

### リスト1 右側の置けるマスを探す判定

```
1. bool r_mark = false, r_true = false;
2. int j = 1 + i, mark=turn % 2 + 1;
3. while (r_mark == false) {
4.     //自分マスだった時
5.     if (MASU[j][k] == mark) {
6.         r_true = true;
7.         r_mark = true;
8.     } else if (j == i + 1) {
9.         r_mark = true;
10.    }
11. }
12. }
13. //ボード越えだった時
14. else if (MASU[j][k] == mark % 2 + 1) {
15.     if (j > 8) {
16.         r_mark = true;
17.     }
18.     j++;
19. }
20. //その他
21. else {
22.     r_mark = true;
23. }
24. }
```

## リスト2 右側の裏返し判定

```
1. //右側判定が終わるまで
2. while (check_r == false) {
3.     if (MASU[x + 1][y] == MASU[x][y] %
2 + 1) {
4.         r++,x++;
5.         if (x >= 8) {
6.             r = 0;
7.             check_r = true; //終了
8.         }
9.     }
10.    else if (MASU[x + 1][y] ==
MASU[x][y]) {
11.        check_r = true; //終了
12.    }
13.    else {
14.        check_r = true; //終了
15.        r = 0;
16.    }
17. }
18. }
19. x = i;
20. for (int j = 0; j <= r; j++) {
21.     MASU[x + j][y] = MASU[i][k];
22. }
```

とあります。これを防ぐためには、そもそも置けないマスはクリックできないようにすることと、間違えた時だけ盤面を1つ戻すことの2通りが考えられます。

私は後者の方法を使いました。この方法では、ターンが正常に終了した時の盤面をその都度コピーし、エラーが発生したらその盤面に置き換えるコードを使っています。

## 4. おわりに

私は今回で3度目の OpenSiv3D を使ったゲーム作成でした。前回のゲームは完成度が低く、お世辞にも良い作品とは言えない物でしたが、今回の作品はかなりうまく作れて満足しました。

オセロは判定数が多く何度も挫折しかけましたが、先輩方の助言のおかげで最後までやり遂げることができました。来年は別の言語やフレームワークを使って、より完成度の高いゲームを作ろうと思います。

## 参考文献

OpenSiv3D

<https://siv3d.github.io/ja-jp/>

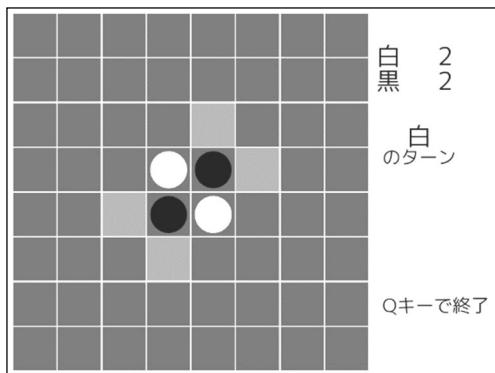


図1 完成したゲーム

### 3.2. 置けないマスに置いたとき

置けるマスは色を塗りつぶしてありますが、間違えて置いてはいけないマスをクリックするこ

# ケーキ集め

機械・エネルギーコース 3年 鈴木 雄裕

## 1. はじめに

今回私はOpenSiv3Dを使ってゲームを作りました。OpenSiv3DとはC++というプログラミング言語を用いて、ゲームやアプリを作ることが出来るライブラリです。

## 2. 制作過程

### 2.1. ゲーム概要

このゲームは自機を十字キーで左右に移動させ、画面上部から落ちてくるケーキを取り多くの点数を取ることを目指します。ランダムに湧くGに当たつたらゲームオーバーです。出来る限り長く生き残り点数を稼ぐのが目的となります。Gの画像はMediBang Paint Proで書きました。

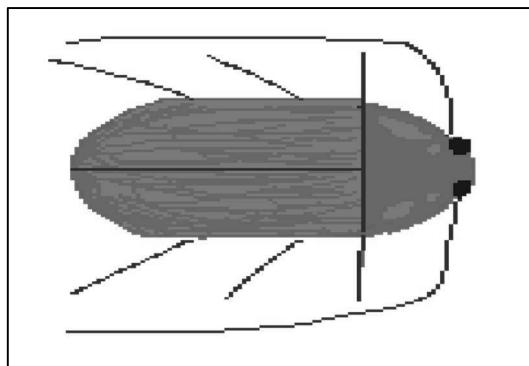


図1 Gの画像

### 2.2. ゲームの動作

ゲームの全体の流れを図2に示します。まず必要な数字である時計の宣言や、ケーキの落下スピードの設定などをここで決めます。次に自機の左右移動を設定します。ケーキを落とす判定、アイテムの当たり判定を設定、Gの生成と接触判定をします。最後に画面全体の描画をします。これを繰り返してゲームが進行します。

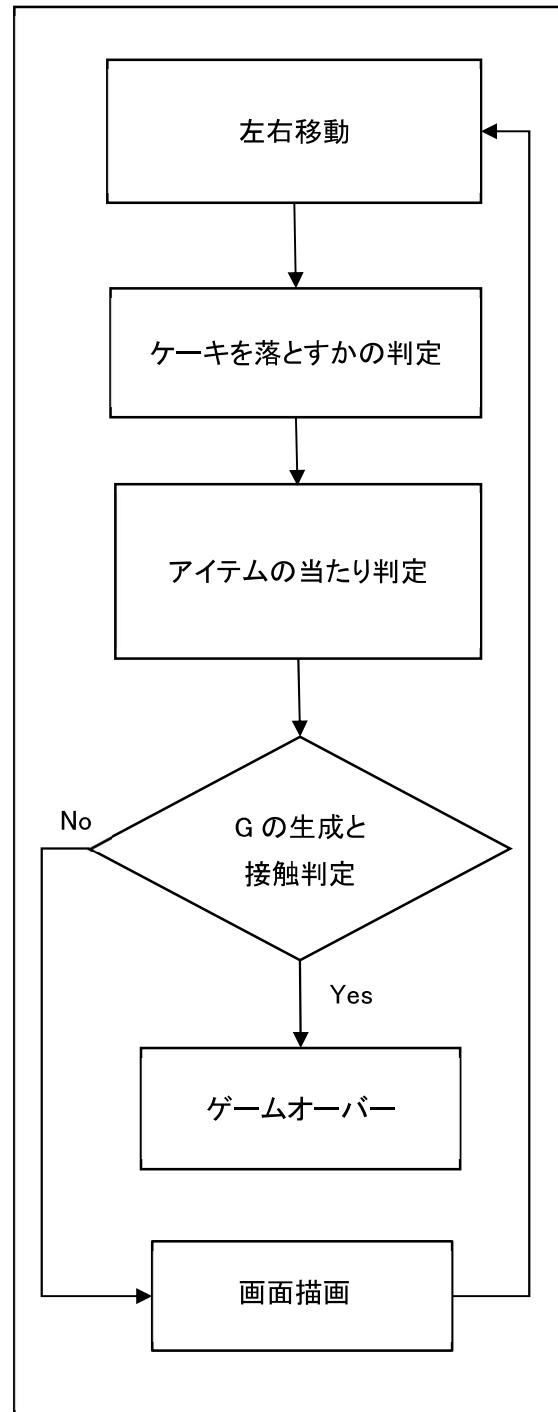


図2 フローチャート

### 3. プログラムの説明

#### 3.1. G の生成と接触

このコードは G を生成し自機と接触したらゲームオーバーにするコードです。

##### リスト1 G の生成

```
1. Rect G = Rect((p * 100), 500, 80, 40);
2. G(h).draw();
3. if (G.intersects(playerCircle)) {
4.     changeScene(State::Title);
5. }
```

Rect((p \* 100), 500, 80, 40)はGを表示する箱の位置と大きさを表します。左から数えて1、2番目の数字で位置を、3、4番目の数字で箱の大きさを指定します。1番目の数字は画面左端からの距離を指定します。p は1から7までの値を取るランダムな数です。これは G の位置がランダムで変わるようにするためのものです。2番目は画面上部からの距離を指定します。3番目で横の長さを、4番目で縦の長さを指定します。2行目のコードは1行目で書いた箱に G の画像を入れて描画しています。h は G の画像データを読み取っています。G(h)で Rect G に G の画像を表示させます。if (G.intersects(playerCircle))で自機との接触判定をします。playerCircle は自機の接触判定を取っている円です。もし G と接触していたらゲームオーバーになります。

#### 3.2. ケーキと自機が接触した場合

このコードはケーキの座標や点数等を生成するコードです。

##### リスト2 ケーキと自機の接触

```
1. while (time >= spawnTime){
2.     Item item;
3.     item.pos.x = Random(100, 700);
4.     item.pos.y = -100;
5.     item.score = 10;
6.     items << item;
7.     time -= spawnTime;
8. }
```

1行目は間隔を開けてケーキを発生させるためのコードです。time が spawnTime より大きい間、2から7行目のコードが動きます。time はゲーム開始から測った秒数が、spawnTime には間隔を開けたい秒数が入っています。2行目で配列を呼び出し item と名前を付けます。3、4行目で item の初期座標を設定します。この座標からケーキが落ちてきます。5行目はアイテムに当たったときの点数です。6行目で item を配列に入れます。ここで設定したケーキを落とすのは別のコードでやります。

最後に time から spawnTime を引き time の値を小さくすることで、2から6行目までのコードが延々と繰り返されることを防ぎ、一定の間隔を空けて動くようにします。

#### 4. おわりに

プログラミングをしばらく触っていない中作つたので忘れたことが多く、もう一度基礎から学ぶ必要性を感じました。使わない知識は忘れてしまっても忘れないようにプログラミングを続け技術の向上を目指し続けていきたいです。

#### 参考文献

OpenSiv3D

<https://siv3d.github.io/ja-jp/>

MediBangPaint

<https://medibangpaint.com/>

ドット絵を描いてみよう！～実践編～

<https://medibangpaint.com/use/2020/05/draw-pixel-art-practice/>

## 編集後記

日増しに寒さが身にしみるようになりました。

さて、今年も Rational が無事に仕上りました。ただ今年はコロナでプロコンに参加できなかつたり Rational の発行が遅れたりと例外的な年でしたが新入部員が数多く入ってください、来年は希望に満ち溢れた年になりそうです。

最後になりますが記事を執筆してくださった部員の皆様、校正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北島先生、佐藤先生、そしてここまで読んでくださったあなたに感謝を申し上げます。ありがとうございました。

編集長 岡田 曜

## Rationale Volume34

発行日 11月11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	鈴木 雄裕
編集	岡田 曜
校正	根地戸 龍生 鈴木 雄裕 岡田 曜 鈴木 晴斗
	富山 結都 日野 綾瀬
表紙	根地戸 龍生 佐藤 至
執筆	ソフトウェア研究部員
製本	ソフトウェア研究部員



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