

KSS PC BOOK 2023

KSS PC CLUB

KSS PC BOOK 2023

2023-06-03 版 KSS PC Club 発行

目次

はじめに	3
第1章 部員アンケート 2023	4
第2章 茨城県版新型コロナ対策サイトの振り返り	14
第3章 パソコンがカタしてたら大学合格した話 + 大学生活所感	24
第4章 学生のキャッシュレス決済を考える	35
第5章 ソフトウェアシンセサイザを作ってみよう	41
第6章 ニコニコ動画風コメント掲示板2はこうやって作られた	56
第7章 茨城県教育情報ネットワークの脆弱性を報告した話	61
第8章 技術と人間 + pygameについて	64
第9章 AIと一緒にゲームを作ろう！	70
第10章 「文字の自動生成の仕組み」について	84
執筆者紹介	93

はじめに

このたびは本書をお手に取っていただきありがとうございます。本書は令和5年度けやき祭のためにパソコン部の有志によって作成された部誌です。

古河中等教育学校パソコン部（通称：KSS PC Club）は、部員各々が自分の興味のある分野で自由に活動しています。その分野は多岐にわたり、ゲーム開発、音楽制作、動画制作、3DCG制作など多種多様です。この自由な活動の発表の場として、部員に自分の活動について自由に執筆してもらい、一冊の本に仕上げました。

部誌の制作は昨年度に引き続き2度目となりますが、昨年度執筆に参加していない部員も多く参加しており、もちろん内容もバラエティ豊かであるので、昨年度以上の仕上がりになっていると自負しております。

今年度は4年ぶりに文化祭が一般公開となり、この部誌も古河中等内外の多くの方々が手に取られることと存じますが、読者のみなさまが、好きなことを追求して創作をしていく面白さや楽しさに触れ、また、私たち KSS PC Club を象徴する「自由」を感じて頂けることを願っております。

部長 長津颯馬/Nagaso.cpp

お問い合わせ先

本書に関するお問い合わせは部員まで。

免責事項

本書に記載された内容は、情報の提供のみを目的としています。したがって、本書を用いた開発、製作、運用は、必ずご自身の責任及び判断の上で行ってください。これらの情報による開発等の結果について、著者はいかなる責任も負いません。

表紙画像: [Technology illustrations by Storyset](#)

第1章

部員アンケート 2023

eno1220

1.1 はじめに

こんにちは。2022-23年度副部長のeno1220です。

この記事では、2023年4月（1年次生入部前）に実施した部員アンケートの結果をみていこうと思います。今回はパソコン部に在籍している部員20名と卒業生4名の計24名から回答をいただきました。ご協力いただきありがとうございました！なお、都合上一部の質問を省略しています。回答して下さった方々申し訳ありません。

1.2 統計編

回答者の分布

入学期	人数
3期	2
4期	1
5期	1
6期	5
7期	3
8期	2
9期	5
10期	5

なお、古河中等では2013年度入学生が1期生であり、2023年度の6年次が6期生となっています。パソコン部（当初はパソコン科学部だったらしい）として活動していた生徒がいるのは3期生以降です。

1.3 動機編

プログラミングを始めた理由やきっかけは？

- パソコン部に入部しているいろ
- パソコン部に入部して、先輩からC言語を教えてもらった
- パソコン部に入り教わったこと

「パソコン部に入ってプログラミングを始めた」という部員が多くみられました。筆者と同期である6期生の部員も、全員プログラミング未経験で入部しています。

ゲームを作りたいから

- ゲームを作ってみたいと思ったから
- 小学生のころゲームにハマっていて自分でも作ってみたいと思ったからです
- フリーの脱出ゲームにハマっていたときに Parafla というソフトを使えば自分でも似たようなゲームが作れると知ったのがきっかけでした。わけも分からずに ActionScript のコードをコピーしていたのを覚えています。

プログラミングに興味があったから

- 学校などでScratchに興味を持った
- プログラミングを学びたかったから

といった回答がありました。この他、周りの影響や「何かを作りたい」と思いプログラミングに興味を持った部員がいるようです。

使用したことのあるプログラミング言語は？

1位 C++ (12名)

1位はC++でした。

C++はたくさんのソフトウェア開発の現場で使用されている他、OSや組み込みシステムの開発にも用いられている人気言語です。

競技プログラミングでもよく使われており、KSS PC Clubで競技プログラミングをやっている人たちのほとんどがC++を使っている印象があります。この他、2022年夏にKSS PC Clubで勉強会を実施した [1]、ゲームやアプリを楽しく簡単に開発するためのフレームワークであるSiv3D [2] もC++を使いますね。

2位 C (11名)

2位はなんとC (言語) でした。

Cは1972年にAT&Tのベル研究所で開発された、初期の高級言語です。当初の開発目的は、UNIXと呼ばれるOSを開発するためでした。1位のC++はCを拡張・改善した言語です。

[1] <https://kss-pc.club/articles/2022/siv3d/>

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

KSS PC Clubでは、6期～7期生が1年次だった頃まで、入部後のチュートリアルのような形でCの講義が行われていたので触れた経験のある部員が多いのではないのでしょうか？近年では部費で購入した「Cの絵本」という本を読んで学習している部員もいるようです。

最近では、後発の高水準な言語が台頭している他、メモリの安全性や記述性の観点からCの利用が少なくなってきていますが、筆者としてはコンピュータの原理や仕様を理解するためにはもってこいの言語だなーと感じています。

3位 Python (8名)

Pythonはインタプリタ言語の一つで、昨今話題の機械学習に使用されるなど、近年人気の言語です。

プログラミングの入門書やwebサイトにPythonが掲載されていることが多い印象があります。また、様々な大学が講義用に作成したPython入門のための教材を無償で公開していることが多いです。

古河中等では、後期課程（高校）における「情報I」の授業でPythonを採用することになっているらしく、7期生以降の生徒 [3] は全員が一度はPythonに触れることになるはずです。

3位 JavaScript (8名)

JavaScriptはChromeやSafariといったWebブラウザで動作するプログラミング言語です。Webサイトをはじめ、モバイルアプリ・デスクトップアプリ、IoTなど幅広い分野で使用されています。

JavaScriptは前述の通り、Web開発の場面では必ずと言っていいほどに使われる言語であり、Web系に関心のある部員の多いKSS PC Clubにおいてこの順位をとっていることは納得です。

5位 C#, TypeScript (5名)

6位 Ruby (4名)

7位 Go, Java, Visual Basic, HTML (言語?) (3名)

8位 Kotlin, CSS (言語?) (2名)

9位 シェルスクリプト, Bash (言語?) , Lua, Verilog HDL, ActionScript, Haskell, ClojureScript, Lean, Julia, Zig (1名)

この他、ここに書けない名前の言語を使っている方が2名ほどいました。

普段の活動について詳細を教えてください(現役生編)

ゲーム制作

- いくつかのゲーム制作
- ScratchやUnityを使ってゲームを作っています

[3] この記事を書いている筆者は6期生なので「情報I」を習わないため、すこし後輩たちがうらやましいです。

- UnityとBlenderを使ってゲームを作っています
- Scratchでゲーム制作

多くの部員が取り組んでいるのがゲーム制作です。ScratchやUnityを中心に様々な方法でゲーム制作を楽しんでいます。

CG

- Blenderを使い、3Dモデリングの勉強
- Blenderを使って幾つかの作品を作った
- 人体やシグマン [4] をBlenderで作った
- Live2DでシグマンのVtuber化に挑戦している

近年人気なのが、Blenderをはじめとしたソフトを使っての3DCG。なぜかシグマンを練習台にする部員が多いような気がします...

WEB開発

- ホームページ制作
- Next.jsを使って学校関連のプロダクトを作っています

KSS PC Clubでは公式Webサイトをはじめとして様々なWeb系の開発を行っています。文化祭でも部員が活躍しています...!!

タイピング

- タイピングの練習
- タイピングの練習などをし、パソコン自体のスキルをあげています！
- 部内タイピング大会優勝、寿司打5000円コースランキング9位
- e-typing Thunderスコア346

とんでもなくタイピングが強い人たちがいますね... KSS PC Clubでは昨年度よりタイピング大会を実施しています。昨年度は卒業生が賞品を差し入れてくださいました。ありがとうございます！

創作活動

- 作詞、作曲
- 簡単な曲を作った
- イラストを書く
- 動画制作

小さいですがペンタブが2台あります。MIDIをはじめとしDTMをやっている部員も多いです。

[4] 古河中等公式マスコットキャラクター

1.4 推し編

推しのプログラミング言語とその理由は？

1位 C++ (5名)

- コードが面白い
- 初めて触った言語
- なんだかんだで高速
- これしか触れたことがない
- C++しか勉強したことがないのでわからないです...

2位 Scratch (4名)

- 初心者でもやりやすい
- わかりやすい
- 直感で作れるから
- 初心者でもわかりやすい

プログラミング言語かどうか怪しいですが... Sctachはやはりその手軽さとわかりやすさが人気の要因のようです。

3位 C (3名)

- わかりやすい
- 応用すれば数多の可能性を見つけることができる

4位 JavaScript (2名)

- 最高の可搬性と周辺ツールの豊富さ！
- 書きやすくライブラリやツールが豊富！

4位 Python (2名)

- 使いやすい
- AIが作れる

5位 Rust (1名)

- Rustは安全で書きやすくていいぞ！

5位 TypeScript (1名)

- 健全性を犠牲にしても使いやすさを優先した強力な型システム。書いていて楽しい！その上、@ts-ignore するだけで型エラーをもみ消してくれる。どんな黒魔術でも書きたいように書かせてくれる動的型付けな JavaScript を好きなだけ楽しめる。そしてオートコンプリートや型チェックの安心感が欲しくなったら、後から型を書き足せばいい。型を操る機能も豊富で、mapped types や template literal types が使える言語は TypeScript の他にないと思う。やりたいうように型をこねくり回せる。あとは追加で Babel を挟めば、言語機能のもの足りないところだって（ほぼ）思いのままに書き換えられる！衛生的なマクロだって、トランスパイル時計算だって、c# の using だって、golang の defer だって使える！そして毎月のように処理系が更新されていく。高カインド型や nominal な型だって、そのうち言語機能に取り込まれるはず。（実際、既に GitHub に issue があがってる。2014 年からずっと open なままだけど……）
- このほか、JavaやC#などを推す意見も出ました。

推しのコードエディタとその理由は？

Visual Studio Code

なんと11名（13名中）の支持を得る結果となりました。仕組みが簡単でわかりやすく扱いやすい、機能が豊富であるほか、たくさんの拡張機能が公開されていて便利などの意見がありました。

エンジニア業界でコードエディタといえば、たいていvimとEmacsで論争が起こるのですが、KSS PC Clubにおいてはvscodeが圧倒的なシェアを誇っているようです。vimやEmacsはコマンドライン上で操作することやショートカットを覚えなくてはならないなど、初心者からは敬遠されがちな存在となっていることや、vscodeのもつ操作性や拡張性の高さなどがvscodeが人気の要因のようです。

このほかneovimやpaiza IOなどの意見がありました。

推しのフォントとその理由は？

色々なフォントが挙げられました。気になったフォントをいくつか掲載します。

- CascadiaCode: デベロッパ向けでいいぞ！
- Source Han Sans: クセがなくどんなシーンでも読みやすい書体
- Oswald: えも言われぬかっこよさがしびれる
- ましゅまるポップ: どんなに物騒な内容でも不愉快に感じないから
- はちまるポップ: かわいい
- モボ: シンプルで見やすく、色々な場面で使った
- Recursive Sans & Mono: 筆記体に近いカジュアルで遊び心のある見た目。あるいは、スッキリと角ばっていて読みやすいサンセリフや等幅フォント。そのすべてを、たった一つの可変フォントで実現してくれる。しかも見た目のカジュアルさや等幅さや斜体さを自由に組み合わせられる！
- Hack: シンプルでいいし、Nerd Fontもある
- Hackgen: 日本語も綺麗、Nerd Fontもある

- 筑紫A丸ゴシック:文字の丸みがとても好みに合います

推しのプログラミング関連の書籍やWEBサイトとその理由は?

書籍編

- 『Thunder本』、かなりわかりやすくおすすめ
「ゲームで学ぶ探索アルゴリズム実践入門」(青木栄太 著) のことです。この記事の筆者は購入したまま積んでしまっています...
- 『苦しんで覚えるC言語』、とても分かりやすくまとめられていて理解しやすいと思います
- 『苦しんで覚えるC言語』、「苦しんで」とあるが、プログラミング初心者にとってとても理解しやすい一冊だと思う。これを読んでプログラミングにハマった、最高の一冊です!
書籍の他にWebサイト版 [5] もあって、こちらを推している部員もいました。
- 『アルゴリズムイントロダクション』
- 『プログラミング言語を作る』、誰もが一度は、ぼくのかんがえたさいきょうのプログラミング言語を作りたいと思ったことがあるはずだが、それを現実にできるのがこの本。あと読みやすくて楽しい。
- 『7つの言語7つの世界』、普段触れるような言語では使うことのできないような言語機能や背景にある理念の存在を知ることができる。いつも使ってるプログラム言語が提供してくれる機能に物足りなくなってくる。そして、ぼくのかんがえたさいきょうのプログラム言語を作りたくなる。
個人的には自作言語(自然言語)で記述する自作言語(プログラミング言語)を作りたいです。

[6]

WEBサイト編

- Qiita (<https://qiita.com>)
- MDN (<https://developer.mozilla.org/ja/docs/Web>)、Web 関連で困ったことがあったら取り敢えずここを読もう!と胸を張っておすすめできるサイト。日本語版はたまに情報が古いことがあるので英語版も参照されたい。
- GitHub Skills (<https://skills.github.com/>)、Git/GitHubの使い方を完全に理解していない頃、高頻度でconflictを派生させたり、うまくpushできなかつたりと困っていたが、このチュートリアルを通してGit/GitHubに関する初歩的な知識を得ることができた。

[5] <https://9cguide.appspot.com/>

[6] インスパイア元:

https://docs.google.com/presentation/d/1ykiqrcc7yUPdyTvULpysPmE3o6WvecJ6Cx_g6O_1vaA

1.5 パソコン編

自分が使用する用途で持っているパソコンの種類は?

OS	人数
デスクトップPC・ノートPC両方持っている	9
デスクトップPCのみ持っている	1
ノートPCのみ持っている	9
持っていない	5

パソコン室で活動しているので一応一応学校のパソコンを使うことはできますが、性能がよくないとかそういうレベルではない状況です... あと4-5年は同じパソコンを使い続けなくてはなりません...

使用しているパソコンのOSは?(複数回答)

OS	人数
Windows	18
ChromeOS	5
macOS	4
Ubuntu	2
ArchLinux	1
Manjaro Linux	1

ChromeOSは、古河中等においてはGIGAスクール構想のもとで前期生に一人一台配布されているPCに搭載されているなど、シェアを広げています。ちなみに、筆者はいつか自作OSを搭載したパソコンで日常生活を送るという夢があります。

1.6 そのほか

今後取り組みたい活動は?(情報系に限らずなんでも構いません)

- AtCoder青になりたいな～～
競技プログラミング頑張っていきたいですね。
- 作曲(作れるジャンルを増やし、コード進行の研究をする)
- 作曲やunityです
- 作曲、c言語の習得

- Unreal Engine あるいは Unity 等に挑戦
- Blenderに触れる
- 3Dモデルを作ったりし始めたので、超簡単なアニメーション作成がしてみたい
作曲や3DCGに関心がある部員が多いみたいですね。仲間がたくさんいると心強いです。
- C++を使用した完全自作のオリジナルゲームを作ること
- ストーリーのあるゲームを作りたい
- スクラッチでゲームを作る
- 様々なジャンルのゲームを作りたい
部の力を合わせて一つの大きなゲームが完成するのを見てみたい!
- 大学の履修登録サイトやニュースをスクレイピングしてスマホに最適化した UI を提供するアプリを作りたい。
- AI、できるならChatGPT APIを使ってウミガメのスープを作りたい
- プログラミングをもっとたくさん出来るようになりたい
- 体力を維持する
かなり大事なことですな。
- 女装
なるほど...?

何か主張したいことがあればご自由にどうぞ!

- パソコン室のパソコン、カスすぎ!(×4名)
- もうちょっとパソコンのスペックが欲しい。
- パソコンが少し重いです
- 急にプロキシが戻って動かなくなることがたまにあってイラッとする。
パソコン室のパソコン、どうにかありませんかね...
- 在校生がんばれ!! 久しぶりに中等行きたいなあ
いつでもお待ちしております～
- 古河中等関係者向けの Mastodon インスタンスとか建ててみたいよね
Mastodonは分散型SNSのひとつです。自宅サーバとかで運用してみたい...
- 睡眠環境を整えてほしい
パソコン室にベッドなどを設置して欲しいということでしょうか...?
- Excelなどのソフトをもっと教えて欲しい
いろいろと教えることができずすみません...一通り教材を揃えたいです...
- プログラミング言語、覚えること多すぎ 頭割れる
わかります
- 楽しめ!
はい!!
- 怠惰に生きててすみません。だけど反省はしない
生きていて偉い!

- 文化祭、忙しいです...助けて～
助けて～

```
#include <bits/stdc++.h>
using namespace std;

int main() {
    srand((unsigned int)time(NULL));
    if (rand() % 2 == 0) {
        cout << ":send_money:" << endl;
    } else {
        cout << ":send_money:"
             << ":is_all_scam:" << endl;
    }
}
```

もう少し学校から降りる部費が増えるといいな
この他ここに掲載することのできない主張をいただきました。ありがとうございます。

第2章

茨城県版新型コロナ対策サイトの振り返り

a01sa01to

2.1 前置き

こんにちは、古河中等3期生の a01sa01to です。誰やねんお前！については、プロフィール(別ページ)をご覧ください。[1]

今年度は久しぶりのけやき祭一般公開、おめでとうございます 🎉 たぶん私も一般参加していることでしょう。一般公開は私が6年次のときに一時停止してしまいました。生徒としての最終学年での文化祭ができず、かなしかった。この度、2023年3月中旬あたりにマスク着用が自由化され、さらに5月には新型コロナに対する感染症法上の扱いが5類相当に移行したおかげか、復活してよかった〜！文化祭に限らず、新型コロナに関しての規制がかなり緩和されて、どこでも人の外出が多くなったように感じます。[2]

[1] アンケートで「青になりたいな〜」と書いた人だったりするんですが、先日のABC302で入青しました！わーい！

[2] Q. なんでこんな雑談書いたの？ / A. 次ページに画像があって勝手に改ページされちゃうので、スペースを少しでも埋めたかった

ところで、
 新型コロナといえば、茨城県民なら一度はこちらのサイトを見たことがあるのではないのでしょうか？
 (誇張表現)



▲ 図 2.1：茨城県新型コロナウイルス対策サイト <https://ibaraki.stopcovid19.jp/>

左下をよくご覧ください。

このサイトの内容は
クリエイティブ・コモンズ 表示 4.0 ライセンス
 されています (ただし商標等の他団体が権利を
 与えています)。
 © 2020 - 2023 Asa (@a01sa01to)

▲ 図 2.2：© 2020 - 2023 Asa (@a01sa01to)

はい、私が作りました。

コロナも落ち着いてきて、対策サイトも終盤期(?) になってきた [3] ので、このサイトの振り返りをしてみようと思います。

2.2 やったこと

大雑把に書きます。

開発までの流れと初動

2020年1月か2月、日本でクルーズ船の乗客で新型コロナ感染者が集団発生したというニュースが流れていました。某先生が「こりゃ日本でも感染が広がって休校になっちゃうなあ」と冗談っぽく言っていたのですが、これが現実になります。 [4]

3月から [5] 休校になり、3月17日ついに茨城県で初の感染者が確認されました。そのころ、東京都がオープンソースで「新型コロナ対策サイト」を公開したことをニュースで聞いていましたが、調べてみると茨城県版の類似サイトはまだありませんでした。こりゃ茨城県版も作ってみるかーと興味本位で、2020年3月20日 Fork して1日かけて (逆に言えば、たった1日で) 作り終え公開しました。

初めて見た Nuxt.js (+ Vue.js)、GitHub Actions などの技術が使われていましたが、何とか試行錯誤しながら作っていた覚えがあります。当時の自分すごいな…。たぶん最初のコミットは [3550a20](#)。うまくデプロイできず、リポジトリを作り直してファイルだけコピーみたいなことをしてました。そのせいで、上のコミットは一括変更だし [6]、Fork 情報 [7] も消えています。ここに最初のバージョンの画像を載せようとしたものの、Web Archive の読み込みがうまくいかず、諦めました。興味があればビルドしてみてください。

そして初回 Release をします。当時 GitHub の Release 機能や Git の Tag 機能について無知だったの、初回 Release は3月24日でした。 [8]

また、いろいろな方に開発を手伝っていただきました。Issue でのアイデア出し、それにかかわる Pull Request などしていただきました！ [9] 初動にいろいろご意見いただき、より良いサイトになったと実感しています。

[3] 終盤期というのは、5類相当移行によってデータの毎日公表が終了するせいで、更新が頻繁に行われなくなるよというイメージです。少なくとも今のところサイト閉鎖のつもりはありません。

[4] 彼はその後しばらく (一部) 生徒の間で **予言者** と呼ばれていた

[5] 3年前の記憶、あまり定かではないです

[6] 大量の変更でのコミットは好ましくないとわれます。皆さんは一部変更するごとに (もちろん動作確認もして) Commit しましょう！！

[7] GitHub 上のリポジトリ名の下に出てくる「forked from ...」みたいなやつ

[8] <https://github.com/a01sa01to/covid19-ibaraki/releases/tag/v1.0>

[9] <https://github.com/a01sa01to/covid19-ibaraki/issues?q=is%3Aissue+is%3Aclosed>

ドメイン変更

しばらく運用していると、一部の地域版が `stopcovid19.jp` のサブドメインを用いていることに気づきました。調べてみると、東京都版対策サイトの開発元 `Code for Japan` が提供しているようでした。 `covid19-ibaraki.netlify.com` だとダサいので (?)、Slack に参加し [10] `ibaraki.stopcovid19.jp` ドメインを申請しました。v2.0 [11] で反映し、東京都版の派生サイトのファイルを変更する PR を提出しました。 [12] ついでに、開発用サイトのドメインも <https://covid19-dev.a01sa01to.com/> に変更しました。

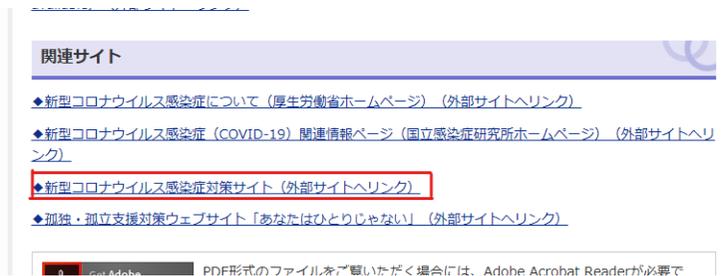
このせいかコロナの波が来たせいか微妙ですが、PV の最初の波はこの時期です。

自動ツイート機能

やっぱり更新情報があったほうがいいよね、と思い実装してみました。実装が楽そうな Python を使っています。いろいろ文献があったので、参考にしながら (ほぼコピペ) 実装しました。

もともと私の個人 Twitter アカウントでツイートしていましたが、さすがに専用アカウントを作りました。 [13] GitHub Actions でデプロイ完了時に「サイトを更新しました!」といった文をツイートするようにしました。さまざまな要望に応えるためにツイート文に、今日の感染者数や重症者数などの情報を加えることも後日から行いました。

県公式サイトへの掲載



▲ 図 2.3 : 県公式サイト <https://www.pref.ibaraki.jp/1saigai/2019-ncov/>

なんと、県公式サイトに掲載していただきました! しかしここまでが長かった...

[10] 幅広くいろいろな活動の情報が入ってきます。無料なので興味のある方はぜひ参加しましょ
う!! <https://www.code4japan.org/activity/community>

[11] <https://github.com/a01sa01to/covid19-ibaraki/releases/tag/v2.0>

[12] <https://github.com/tokyo-metropolitan-gov/covid19/pull/3260>

[13] https://twitter.com/covid19_ibaraki

やっぱりこういった流行っている病の情報サイトは、多くの人に見てもらったほうが良いです [14]。そのためには、県公式サイトに掲載してもらうのが一番です。そこで、まず 2020 年 4 月中旬に県の問い合わせフォームに「掲載してください! [15]」と書き込みました。返信が来ません。電話もしてみます。「部署が違う」ということで無限ループに陥りました。学校の先生にも愚痴ります。県立とはいえ、何も起きません。もう一回県の別の問い合わせフォームに別件付きで (後述) 書きます。すると、メールで返信が来ました。 [16]

そこから県の職員さんとメールでやりとり。無断で県章を使っていた [17] ので変えてくれということで、茨城県の形をしたアイコンに変更しました。そして、ついに掲載していただけることになりました。2020 年 10 月 1 日の出来事です。およそ半年かかりました。ご対応いただいた職員の方に感謝です。

オープンデータの公開

さっきの「別件」です。オープンデータがあったら自動化できるな~と思って、公開しないのか聞いてみました。回答は、主に「CSV ファイルで管理してないから」「個人情報が入ってるから」無理らしいです。まあつまり面倒くさいってことなんでしょうね...。そこで、私が作ったオープンデータなら公開しても構わないか訊ねてみたところ、OK が出ました。そして作成に取り掛かります。

それまでは完全に手動更新でした。VSCode を開いて、リポジトリ内にある `data.json` にその日の感染者情報やら回復者数やらを書き込んで、Commit する日々。 [18] 当時は感染者数が Max でも 20 人くらいだったのでなんとか人力で耐えてました。2020 年 11 月中旬から 60 人を超える波が来たので、タイミングが良かったなと思ったり。

ここからオープンデータにするために、データを移行します。まず、この `data.json` から CSV 互換の形式に変換します。どこにアップするかいろいろ踏まえた結果、Google スプレッドシートに書き込むことにしました。変換用に JavaScript で書いたスクリプトを実行し、コピペで完了しました。 [19]

[14] 個人の意見です。嘘の情報もあるので、一応信頼できるサイトだよ! という肩書きも得たかったのはあるかも。

[15] 実際にはもっと丁寧に詳細な文面ですが、概要はまさにこれです

[16] 今思えばやってることヤバイクレーマーじゃん...

[17] 作成当時は「東京都が都章を使ってたし、茨城県版も県章使っちゃえ!」になってて、その後そのままにしました。ちゃんと許可は取りましょう。

[18] 例: 2020-10-28 の更新 <https://github.com/a01sa01to/covid19-ibaraki/commit/1ac96b1ad3e4b22dda35d7e08f6a8d2bf35b4f54>

[19] まあ「コピペで完了」と書いているわけですが、実際にはスプレッドシート作成が 8 月下旬、そこからどう保存するか、どう変換するか、...などいろいろ考えて、本格的に移行開始したのが 10 月くらいからなので、割と時間がかかってます。

とりあえず、スプレッドシートへの移行は完了したので、これをどうダウンロードするかを考えます。毎回 CSV でダウンロードして、それを `data.json` に変換するのは面倒なので、Google Apps Script でスプレッドシートの内容を CSV としてダウンロードできるようにしました。これで、`data.json` の内容は自動取得できるようになりました。

あとは、自動更新のスクリプトを書くだけです。これは GitHub Actions で定期実行するようにしました。ファイルをダウンロードし、Python で CSV から JSON にして、`data.json` を更新するという流れです。これで、`data.json` 直書きからの脱却が完了しました。[20] とはいえ、スプレッドシートの書き込みが手動です。今後スクレイピングで自動化できるようになりますが、それまではまだほぼ手動更新です。[21]

そして、一般公開のオープンデータの整備をします。とりあえず自分のサイトで、ページを作って公開することにしました。

これでヨシ！と思った矢先、東京都版で高速化のために API を用いていることに気づきました。具体的には、「陽性者の属性」を取得する API です。これまでの陽性者のデータを埋め込んでいた `data.json` が日々肥大化していくので、読み込みが遅くなっていました。私も茨城県版の少し様子を見ていましたが、Loading にかなり時間がかかっていたので、茨城県版も API に移行することにしました。

いろいろ調べてみると、東京都版が用いている API では、GraphQL を用いていました。GraphQL は、クライアント側で必要なデータを指定して、サーバー側でそのデータを返す仕組みです。まあ同じような感じにしておいたほうがいいか...ということで、GraphQL で API を書きました。これが 2020 年 12 月くらいの出来事。当初はサーバーサイドを JavaScript で書いていましたが、さすがに速度が遅すぎるので、少しずつ Golang に移行しました (完全移行は 2022 年 10 月ごろ)。速度が速いとはいえ、めちゃくちゃナイーブな実装なので大量アクセスが来たら死にます 😊

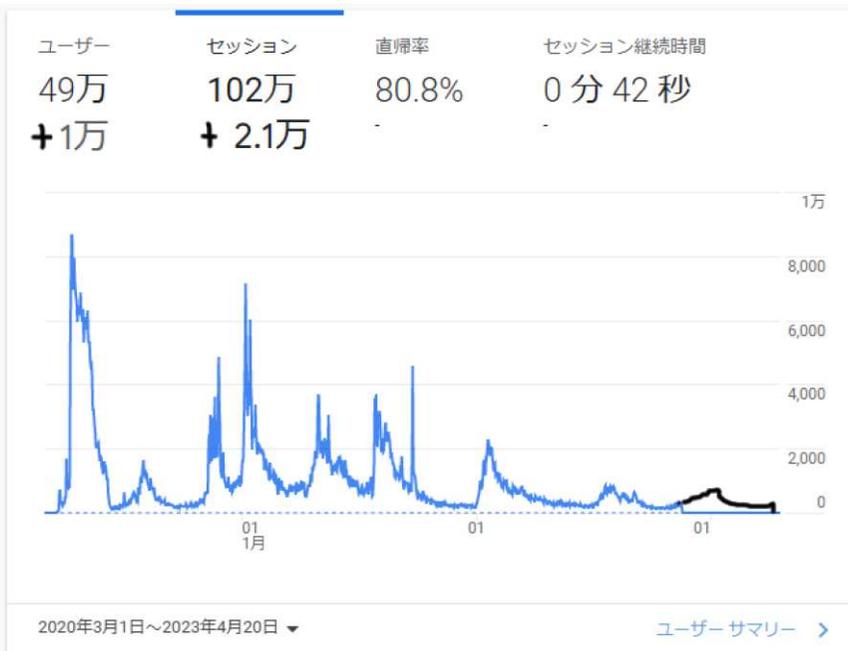
2.3 反応

やはり需要があったようで、Google Analytics で見たところ、ピーク時には約 1 万 PV/日も！ 2023 年 4 月 20 日現在、累計 100 万 PV を超えていたようです。ありがとうございます！面白かったのは、感染者数の波と PV の波がきれいに対応していたこと。2020 年から 2023 年で感染者数がだいぶ変わってしまったので、 $\log_{10}(\text{感染者数} + 1)$ のグラフと重ねてみました。少し合成をミスったのでずれているように見えますが、この図を見ると、PV の増加と感染者数の増加はほぼ同じタイミングで起こっているように見えます。

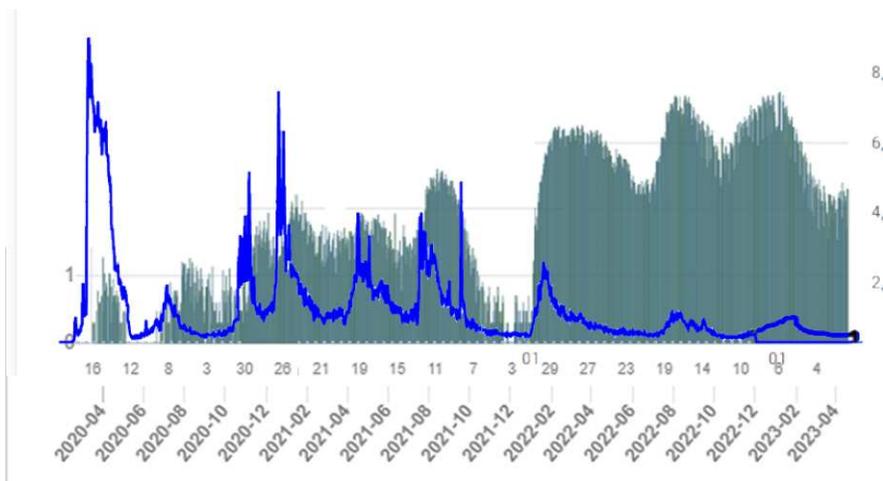
[20] <https://github.com/a01sa01to/covid19->

<ibaraki/commit/babf65b980a880b6e2a59f69bc72d1ffd3e34292>

[21] <https://colab.research.google.com/drive/1WG6GD62sPMY-wkm457hFQf0wVuhUXUpL>



▲ 図 2.4 : Google Analyticsの画面



▲ 図 2.5 : 感染者数 (log10) と重ねた図 (緑が感染者数、青がPV数)

2022年9月下旬から Google Analytics のプロパティを変えたためグラフが別になってしまったんですが、画像を重ねるのがめんどくさかったのでグラフは手書きです。グラフはほとんど合っているはずですが、「ユーザー +1万」「セッション +2.1万」も 2022年9月下旬から 2023年4月20日までのものです。

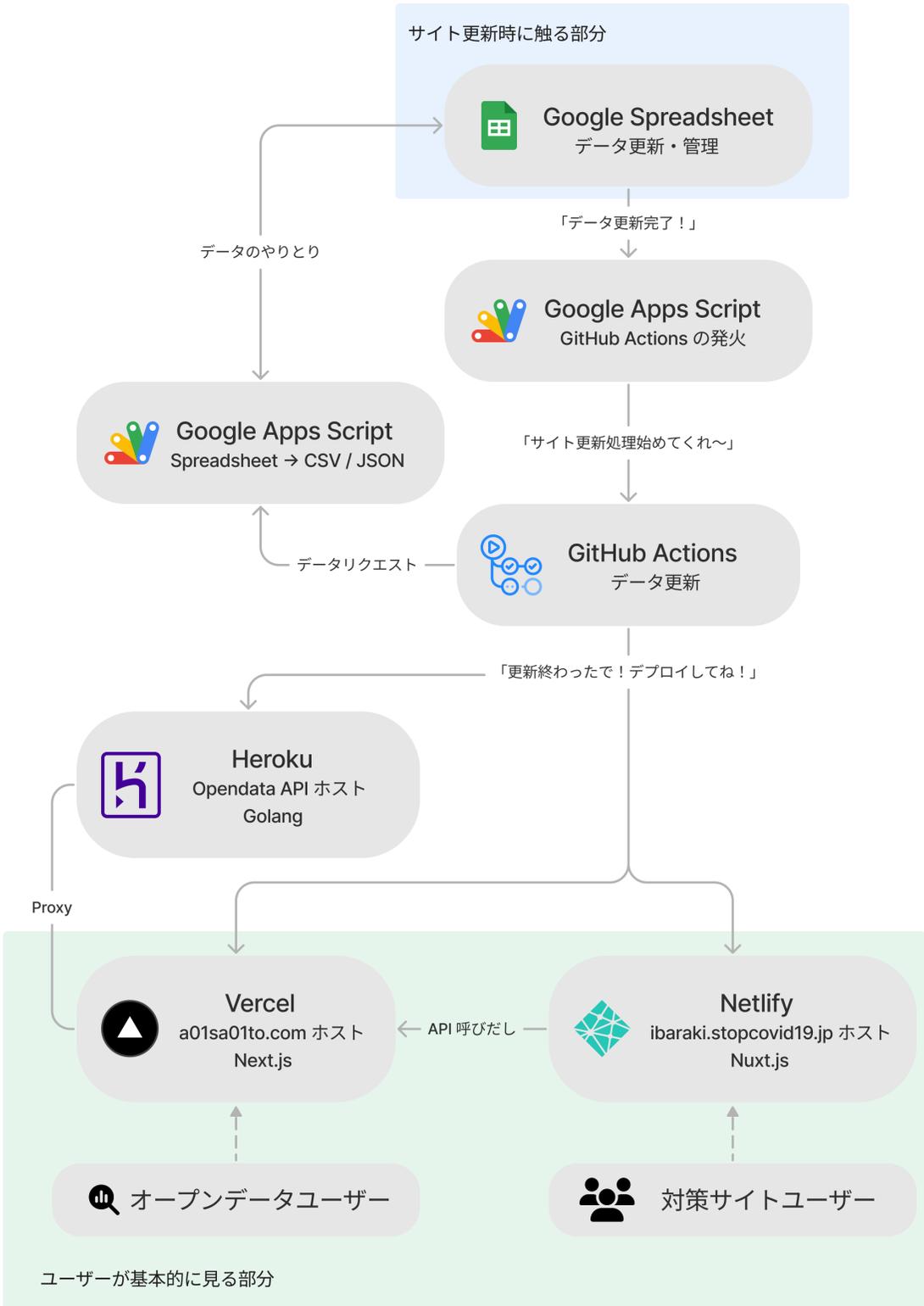
ちなみに、一番閲覧が多かったのは「市町村別感染者数」。県のサイトは PDF 公開だし一覧表示されてなかったの、表で見れる対策サイトは需要が高かったっぽい。また、GitHub 上でも、数週間で十数もの Star をいただきました。初めてこのくらいの大きな反応になったので、やりがいとうれしさを感じました。

2.4 対策サイトを支える技術

これまでの内容を見ればなんとなーくわかる気もしますが、**現在の**対策サイトを支える技術については以下のような感じです。

毎日の更新の流れとしては、まず私が(手動で)スプレッドシートを更新します。更新が終わったら、「更新」ボタンを押すと GAS から GitHub Actions の Dispatch イベントを発火させます。対策サイト、私のサイト、Opendata API の 3 つのリポジトリで発火された Actions では、それぞれスプレッドシート上のデータを GAS 経由で CSV/JSON 形式でダウンロードします。その後、データを処理してサイトをビルドし、各々デプロイします。

対策サイトのユーザーは対策サイトにアクセスすると、必要であれば Opendata API からデータを取得します。なお、Opendata API は Golang で構築されており、私のサイトから内部的にプロキシされています。プロキシでの通信時間を考慮しても、API を JavaScript (Node.js) で作るよりかなり高速です。JS 遅すぎ！！



▲ 図 2.6 : 現在の対策サイトを支える技術

第3章

パソコンが効かしてたら大学合格した話 + 大学生活所感

Ryoga.exe

3.1 はじめに

こんにちは、締め切りに大幅に遅刻した5期生の Ryoga.exe です。昨年度までこの古河中等に在籍していました。

今年度は久しぶりのけやき祭一般公開でかなり感動、嬉しみ。恐らく私も一般参加しているかもしれませんが、していないかもしれません。[1]

この部誌という文化は私が昨年勢いで始めたものですが、今年も続いておりとても嬉しいです。やはり自分の言葉での情報発信はとても良いものだと感じます。このような発信活動の場のひとつである本書を通して、読者のみなさまにも創作・技術に触れる楽しさ、好きなことに接する楽しさを感じてもらえると幸いです。

さて、私は今年の春に古河中等を卒業し、大学生になりました。筑波大学情報学群情報科学類(通称:coins)というところに入学していただき絶対賛学部1年生です。昨年私は所謂「受験生」だったわけですが、アドミッションセンター入試(AC入試)と呼ばれる少々特殊な入試方式で受験し、まともな「受験勉強」をせずに大学に合格してしまいました。一応後輩に私と同じ大学学類・同じ入学方式に気持ちのある方々がいるらしいのと、過去に古河中等から私以外で合格した人がいないので適当に書き残していきます。[2] [3]

[1] 全体集合 Ω の分割であり、情報量はない

[2] 今後、私のブログにも同じような文章が上がる可能性が高いです

[3] ステッピングモータとある曲を演奏してその様子を某動画投稿サイトに投稿したところ、かなり伸びたのでその記事にしようかと思ったが**なぜか**ストップがかかったためこんな記事になった

- 医学群
 - 医学類
 - 看護学類
 - 医療科学類
- 総合学域群
- 体育専門学群
- 芸術専門学群

文系理系から体育・芸術まで幅広い学問が学べる総合大学です。この中で私が受けた情報科学類は情報科学について専門的に学べる学類で、つくばの強いオタクが多数在籍しているところとしても有名です。

以下に筑波大学公式サイト (<https://www.tsukuba.ac.jp/education/organization-ug-list-0160/index.html>) からの引用を載せます。

教育目標

情報に関わる先端の科学と技術の基礎から応用までを習得し、同時に豊かな想像力とチャレンジ精神、高い社会的倫理観を培う。これらを基に、実世界の様々な課題の本質を理解し、その具体的な解決にリーダーシップを発揮して、秩序ある情報化社会の実現に貢献できる技術者・研究者を育成する。

構成分野について

情報科学類には、ソフトウェアサイエンス、情報システム、知能情報メディアの3つの主専攻があります。卒業研究では、ソフトウェア科学、計算機システム、知能情報、情報数理、情報電子などの分野から提供される多様なテーマを選択できます。

3.3 アドミッションセンター入試 (AC入試) とは？

総合型選抜のひとつで、筑波大学独自の入試です。AOではありません、ACです。出身高校からの推薦は不要ですが、書類選考と面接が課せられることが大きな特徴です。

AC入試では、「AC入試におけるアドミッションポリシー」を満たしている学生を選抜します。情報科学類におけるAC入試では、

1. 第1次選抜:「志願理由書(800字以内)」「自己推薦書(枚数自由)」を提出
2. 第2次選抜: 第1次選抜の合格者に対し、口述試験(面接)を行う。
3. 面接に合格した受験者が、AC入試の最終合格者となる。(大学入学共通テストの受験は必要ない) という感じの流れで行われます。

ちなみにAC入試は専願することを宣誓するものではないため併願校として受験することもできます。また、現高3生・既卒生・社会人・高卒認定試験の合格者など、広く受験資格を持つことも特徴です。

3.4 受験を決めるまで

なぜ筑波大学情報学群情報科学類を選択し、その上でAC入試を選んだかについて説明したいと思います。

推薦入試を決めるまで

古河中等1年次のころからプログラミングを行っており、情報系に気持ちがあったため中等入学当初から情報系学部への進学を考えていました。

4年次のころに、競技プログラミングに出会い、5年次になるにつれ情報科学という学問への興味ますます強くなりました。特にプログラミングでものを「作る」という興味から「研究する」という興味に変わっていきました。やはり研究するなら国公立大学だろう、と考え国公立大学への進学を視野に入れ始めました。当時は国公立大学ならどの大学でも同じようなものだろうと考えていたため、「適当な国公立大学で情報系の学部」という志望をしていました。最終的に志望校を決定したのは6年次の夏でした。[6]

5年次の冬頃、Twitter上でAC入試に合格した人を数人観測し、AC入試の存在を知りました。入試についていろいろと調べていくうちに面白そう！と感じ、決定まではいかなかったものの視野に入れていました。

6年次に上がるにつれ受験への焦燥感が高まるも、受験勉強よりもプログラミングやらの方が楽しかったのでずっとパソコンの前にいました。

私は勉強があまり得意な人間ではなく、特に所謂受験勉強と呼ばれるような機械的に知識を詰め込む作業は苦手でした。また、模試を受験すること自体に抵抗感があり苦手でしたし、学年全体にあった全体主義的な団体戦としての受験といった空気感が苦手でした。なにより、一般入試のシステムや、大学受験に対する空気感・学歴に対して云々いう人々[7]が嫌いで、かなりストレスを感じていました。そういうこともあり、「一般入試で合格するのは不可能ではないか」と感じていました。[8]

そこで当然目を付けたのが「推薦入試」です。

AC入試を選択するまで

情報科学を学べて推薦入試を選択できる国公立大学を探すといろいろ出てきました。調べていくうちに日本には主に学校型推薦と総合型選抜の2種類が存在するというを知りました。ここで、「学校でいろいろとやらかしていたり、未提出課題が溜まっていたり、何度か怒られが発生していたので学校からの推薦はもらえないのでは？それ以上に古河中等自体あまり好きではないので学校から推薦

[6] もちろん明確な志望校を決めていないなどと5年次での担任との面談では言えなかったのですが表向きには志望校を決めているふりをしていた

[7] 世間では学歴厨というらしい、クラスに数人いて始めはなんとも思わなかったがだんだんと苦しくなった

[8] というか嫌だった

をもらって受験したくない！」と思ったため、完全に総合型選抜にシフトしました。また、前述したとおり勉強ができない(特に国語が壊滅している)ため、できれば共通テストを受験しなかったと感じていたため、これらを前提に探しました。

調べていくうちに筑波大学の AC 入試を思い出し、かなり気持ちが強くなっていきました。大学でやりたいことができそうだなと感じたのもかなり大きかったです。また、筑波大学の情報科学類自体、授業や雰囲気が面白そうだなと感じ、行きたいなと思うようになりました。情報系のコミュニティが充実している点がとても好印象で、特に jsys や WORD などに入りたい気持ちが高まりました。その上、他学類の授業が取りやすいという点も理由としてありました。

以上のことから、[9] 筑波大学の AC 入試の受験を決めました。6 年次の 5, 6 月ごろでした。

ちなみに筑波大学の学校型推薦入試の方にも両方出願できるのですが、前述のことと AC 入試で入学すること自体に価値を感じていたためこちらの準備はせず、AC 入試のみに専念しました。

3.5 出願の準備

とりあえず、AC 入試で受験することなどを担任へ伝えました。またよくわからない書類に受験する旨を書いたりしました。学校には秘密にして受験したかったのですが、調査書などの提出が求められるなどがあり無理らしい。

エントリーは Web 上で行います、私は 8 月 30 日の夕方に行いました。

志願理由書

coins のアドミッションポリシーを参考にして書きました。志望した理由と今までにやってきたことや、これからやりたいことを絡めて書きました。学校で志望理由書を書く授業というかそんな感じのものがありましたが、そこで書いてみて返却された文章をもとに書き始めました。ただ、ここで返却されたものはなんとというかあまり的確な指摘・修正をしてくれなかったのが微妙でした。もちろん技術的なことを絡めて書いたわけですがそういったことに対するフィードバックはなんとも言えない感じでした。

同級生に知識情報・図書館学類 (klis) の AC 入試を考えている友人がいた [10] ため、2 人で添削しました。[11] また、母にフィードバックをもらい、なんとか良い感じの文章に仕上げていきました。

ちなみに指定された形式の用紙に書くのですが、パソコン使用可であったため頑張って原稿用紙の枠に合うように調節して書きました。[12] [13]

[9] 6 年次の 3 月ごろに情報科学類へ AC 入試で合格した人とお会いし、いろいろと話をする機会があり、そこで詳しく知れたのも大きかった

[10] なんと下の名前の読みが同じ

[11] なんだかんだで仲間がいることは精神的にも重要

[12] PDF の形式でしか配布されていないため、非常につらいものがある

[13] 友人はなんと MS ペイントで作っていた

自己推薦書

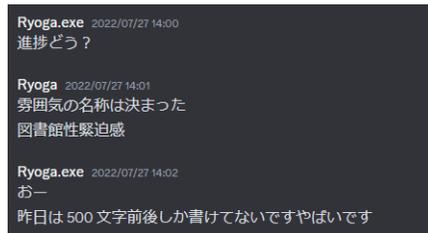
提出する書類の中でこれがメインです。ページ数もフォーマットも自由です。100枚以上書く猛者もいるらしいです。

私は以下のことを中心に書きました。これらを絡めて大学側から求められている**問題解決能力**を重視して書きました。

- 部活動での活動
- 開発したアプリやゲームについての話
- OSS への貢献活動
- 競技プログラミング・情報オリンピックへの参加
- 応用情報技術者試験の合格
- 情報科学の達人での学び・研究
- その他の活動 (文化祭実行委員やその他開発したもの・スキーなど)
- これから続けたいことや筑波大学でしたいこと

途中までは不安もあったのか思うように筆が進まずかなりまずかったです。

こちらに関しても前述した友人と添削し合ったりしました。なんなら推薦書に互いの名前を出しました。2人で進捗状況を確認したりできたのはかなり助かったです。添削・進捗状況の確認以外に Python での自然言語処理を教えたりしました。

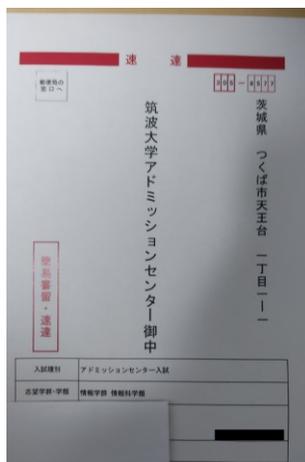


▲ 図 3.1 : 進捗状況を確認し合っている様子



▲ 図 3.2：主に放課後とかに PC 室に集まって作業したりしていた、途中なぜかなんらかの特微量ベクトルを共有している

それに加えて、昨年度に AC 入試で合格した人からの添削をしてもらいました。自己推薦書は内容が非常に専門的なことに傾くため、プログラマからの Lint はかなりありがたかったです。最終的に書きあがったのが 9 月 2 日の深夜でした。



▲ 図 3.3：実際に送った封筒

補足資料として作ったソフトウェアのソースコードや実行ファイル、動作している様子を収めた動画を USB メモリに入れて送りました。

3.6 一次選考結果発表

書類提出から一次選考結果発表まで1か月近くあるのでかなりメンタルがやられました。[14]

選考結果については家で確認しました。[15] 正直通るとは思っていなかったのでめちゃくちゃ声を出して喜んだ覚えがあります。[16]

それからは二次選考である面接の練習をしました。担任の先生や、coinsにACで合格した先輩などに練習をお願いしました。本当にありがとうございました。練習では自己推薦書の要約の練習をしたり、基本的な受け答えの練習、飛んできそうな質問について考えたりしました。

3.7 面接



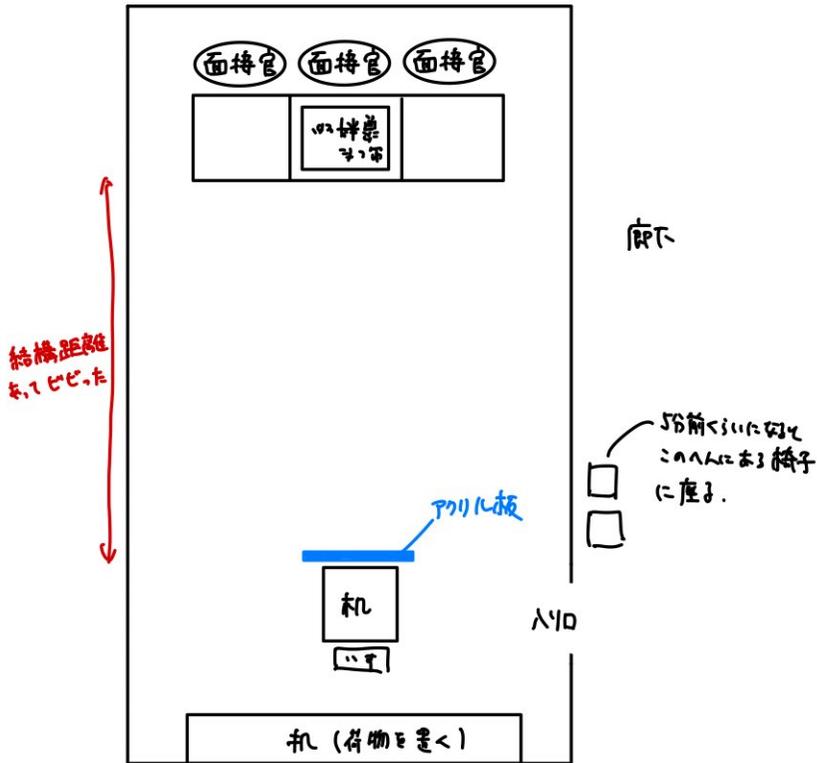
▲ 図3.4：有名な看板

前日はしっかりと早寝をしました。面接1時間前くらいに大学に到着しました。かなり着くのが早すぎたので待合室でかなり暇でした。

[14] ふらっと大洗に出かけて美味しい海鮮を食べたり、ゲームセンターに行き趣味のChunithmやオンゲキをするなどして気分をリフレッシュしました。

[15] 普通に平日なのであまりよくはないが、万が一精神が終わるので家で見るのが平和だと思いません…

[16] 心の準備ができてないまま間違えて適当なリンクをクリックしたらまさかの合格者一覧が出てきてでびっくりしたのもある



▲ 図 3.5 : 会場の様子

面接会場ではアクリル板がありました。次もあるのでしょうか…？

受験の雰囲気ですが、和やかだなと感じました。ほとんど雑談でした。面接官は教授2人とアドミッションセンターの職員の方1人の計3人でした。最初にアドミッションセンターの職員の方から緊張をほぐすような軽い雑談から始まり、教授からの質問に移って自己推薦書についてだんだんと掘り下げられていく…といった感じです。ボケるとウケを取れるのでとても平和で、どっと笑いを起こせたので楽しかったです。かなりの圧迫面接であったと言っていた人もいたので試験官や年度・学類によって変わるのかなと思います。[17]

なにより、楽しむのが一番だと思います。楽しく話せたら受かっていると思います。

[17] 教授と激しいレスパを繰り返して合格している人もいろいろ

受験後は、「あー落ちたなー」という気分で、ある意味開放的な気持ちになっていました。そのまま大学周辺の二郎系ラーメンを食べたり、内見をしったりしました。

3.8 最終合格者発表

最終合格者発表の一週間前の時期はいろいろと精神が不安定でした。なんとか乗り越えましょう。私の場合は Meta Hacker Cup 上位入賞者として Tシャツが届いたり、PCK で会津に行けることが完全に確定したり、PG BATTLE でスポンサー賞になったりしたイベントがあったのでなんとかなりました。

発表前日、一次の時の私のせいかもしれませんが「合格発表は家で確認しよう！」と教室の連絡事項に書かれていました。が、翌日普通に起きたら 1 限過ぎで、家で確認する car... という感じになったので家で確認しました。

嬉しかったです。

翌日に合格通知書が郵送され、合格したんだな…と実感しました。

3.9 合格してから入学まで

もちろん入試で終わりではないのでいろいろ活動については継続して行っていました。競技プログラミングや研究などを行っていました。特に前者ではパソコン甲子園 (PCK) 本選があり、いろいろと楽しかったです。余談ですが、PCK 本選参加者に筑波大学の科学オリンピック特別入試合格者がいてびっくりしました。

まあ学校の方もきちんと行っていました。一時期漠然とした不安からメンタルが完全に崩壊しました。まあまあ休んでしまいましたがなんとか卒業はできました。メンタルの大崩壊と劣等感に襲われ非常につらかったので受験を考えている方は合格後のメンタル管理もきちんと考えておきましょう…

3.10 大学生活所感

ここからは入学してからのことについて適当に書きます。

coins ラウンジが便利

ルールを守れば自由に使えるし、24 時間空いているのでとても便利です。空きコマに適当にふらつと寄って課題や作業ができたりします。飲食も可です。朝が来るまで作業をしても怒られません、私は深夜までいることが多いですが、友人と一緒に作業したりしてとてもいいです。

入学式の日に寄ってみたところ、自然と他の AC 合格者と一緒にになり、いろいろと話ができて楽しかったです。なんというか、高校時代にパソコカ系の話ができながくわずかだったのでこっちに来てから非常に楽しいです。

情報系コミュニティが多く楽しい

WORD や jsys、IPC、UNTIL. など大学内に情報系コミュニティが多く存在し非常に良いです。上級生にかなり有名な方がいたりするのでそういった方々と直接話ができるのはとても楽しいです。

学生による自治がしっかりとしている

感動しました。学生主体で動いている組織が多くあり、非常に良いです。どうしても古河中等生時代と比べてしまうのですが、本当の生徒(学生)主体ってこういうことなんだな、と感じています。

大学生活にはある程度の社会性が求められる

社会性をしっかりと付けましょう…

1 限前に起きるのが非常に困難 (個人差あり)

本当に起きられなくなります。私はやらかしましたが、単位云々には関係しないやつだったのでまだ助かりました。

3.11 おわりに

問題解決能力がある、高校生活で何かを成し遂げた、今辛くて受験に背を向けている方々などは筑波大学 AC 入試を選択肢の一つに入れてみてはいかがでしょうか？「今までやってきた活動を踏まえ、自分がアドミッションポリシーに沿った人物である」ということをアピールすることが重要です、何か社会的なステータスだったり、すごい賞を受賞していることが条件ではありません。

私のようなカス人間で良ければ喜んで相談に乗りますので興味があれば私の Twitter アカウントを特定して DM に突撃するか `contact [at] ryoga.dev` までお気軽にメールをください。自己推薦書やその他の事項について共有したり相談したりできます。

最後に、今回の受験でお世話になった方々に改めてこの場を借りて感謝したいと思います。ありがとうございました。

それでは。

第4章

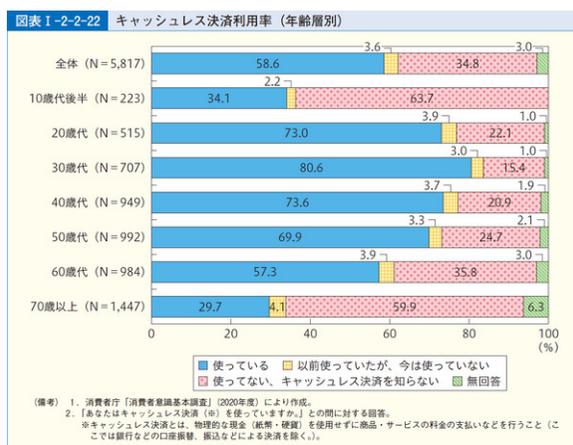
学生のキャッシュレス決済を考える

Chino Kafu

4.1 はじめに - 学生こそ決済手段を意識するべきである

ラビットハウスからこんにちは。香風智乃です。キャッシュレス決済という言葉が持つ囁かれるようになって幾許かの年月が経ち、最近では田舎の小さなスーパーのレジにさえ「PayPay 使えます」の文字が小躍りするようになりました。現金主義が蔓延していた日本も、世界的なキャッシュレス化の風潮や、パンデミックという現金決済に大きな負のインセンティブを与える出来事の影響により、徐々にキャッシュレス化が進んでいると理解してよいでしょう。

しかしながら、10代のキャッシュレス決済利用率は他の年代と比較して低い傾向にあります¹。恐らく「学生だからまだ要らない」とか「現金が減る感覚を知るべき」等と考えている人が多いのかと思います。



▲ 図 4.1：キャッシュレス決済利用率 (年齢層別)，令和 3 年版消費者白書より

しかし、果たして本当に学生にキャッシュレス決済は不要なのでしょうか？ 現金を取り扱うことによる負担は、現金決済者が想像している以上に大きいです。財布に紙幣と硬貨を入れて持ち運び、必要な金額を取り出してレジに出し、釣り銭を受け取りまた財布に収納するという手間が支払いの度に発生

します。その上現金の出納は自分で記録しない限り全て不可視です。それらの負担を「学生だから」といって無条件に負い続ける理由はどこにもないのです。その上キャッシュレス決済を利用すれば、いつ、どこで、いくらお金を使ったのかが全て自動的に記録されます。手元にあるお金を把握することも容易で、支出の計画も立てやすくなります。我々は、年齢の如何に関わらずキャッシュレス決済を利用すべきなのです。

4.2 キャッシュレス決済の種類

さて、一口にキャッシュレス決済といっても様々な手段があります。まずはその決済手段について理解する必要があります。

QRコード・バーコード

PayPay が代表的な例です。店舗が提示するコードを読み取るタイプ (Merchant-Presented Mode, **MPM**) と、客がコードを提示して店舗側が読み取るタイプ (Consumer-Presented Mode, **CPM**) の 2 種類があります。MPM は決済速度が早いものの、店舗側の導入コストが高くなります。CPM は店舗側の導入コストは低いものの、客側が自分でコードを読み取るため少々手間がかかります (金額も客側が入力するケースが多いです)。

支払い方法としては、事前に必要な金額を入金しておくプリペイド方式、利用時に銀行口座等から即時に引き落とされるリアルタイム方式、クレジットカードのように後から支払うポストペイ方式があります。大抵のコード決済はプリペイド方式とポストペイ方式を採用していますが、Bank Pay やゆうちよ Pay のようにリアルタイム方式を採用したコード決済も存在します。

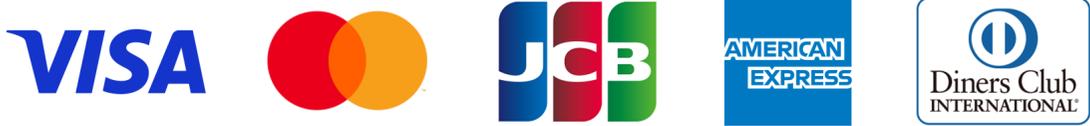
近年目覚ましい成長を見せている決済方法です。

電子マネー

このネーミングが混乱を招いているような気がしてなりません。日本国内では「電子マネー」という言葉が、主に FeliCa を利用したカード型キャッシュレス決済手段に対して利用されているように見受けられます。つまり、Suica を筆頭とする交通系 IC カードや楽天 Edy, iD, QUICPay といったものです。

Suica や楽天 Edy はプリペイド方式で、iD や QUICPay はクレジットカードを紐付ける²タイプになっています。10 代でも Suica を持っている人は相当数居るのではないのでしょうか。Suica は電車に乗るだけでなく、コンビニ等でチャージしたり支払いに使うこともできます。

クレジットカード



▲ 図 4.2：クレジットカードの5大国際ブランド

日本のキャッシュレス決済のほとんどを占める、ポストペイ方式の大御所です。Visa や Mastercard といった国際ブランドが介在することで、世界中のあらゆる加盟店で決済を行うことができます。

クレジットカードは基本的に高校生を除く 18 歳以上でないと作れません。例外として、**セゾンカード**と**オリコカード**は 18 歳以上であれば高校生でもクレジットカードの発行を認めています。また、**イオンカード**は卒業年度の 1/1 より発行を認めています。しかし、クレジットカードの利用にはリスクが伴うことを常に意識するようにしてください。特に支払いの延滞などは、その記録が CIC などの信用情報機関に記録され、新たなクレジットカードを発行したりローンを組んだりする際に影響を及ぼす可能性があります。

クレジットカードのタッチ決済



▲ 図 4.3：タッチ決済のロゴ

クレジットカードを非接触リーダにかざすことで支払う方法です。しばしば混同されるのですが、これは **iD** や **QUICPay** とは異なるものです。iD や QUICPay が利用する FeliCa と違い世界的に普及している NFC Type-A/B と呼ばれる技術を利用しているため、タッチ決済に対応している店舗では国を問わずに利用できます。まだ普及の過程にありますが、既に国内でもコンビニや大手チェーン店等はほとんどが対応しつつあります。カードをかざすだけなので、今までクレジットカードを利用する時に必要だったサインや暗証番号が不要になります³。さらに、Google Pay や Apple Pay のタッチ決済に対応しているクレジットカードであれば、スマホでクレジットカード決済が可能です。筆者はこの「Google Pay / Apple Pay のタッチ決済」に対応しているか否かが、これからクレジットカードを選ぶ時の非常に重要なポイントであると考えています。

ブランドデビットカード

ブランドデビットカードは、クレジットカードと同じように利用可能なリアルタイム方式のカードです。[ゆうちょデビット](#)や[SMBC デビット](#)のように銀行のキャッシュカードに付帯していることが多いです。

クレジットカードと同じように Visa や Mastercard, JCB のような国際ブランドのカード番号が発行されるのでクレジットカードしか使えない場所でも利用可能です⁴。

ブランドプリペイドカード

ブランドプリペイドカードは、クレジットカードと同じように利用可能なプリペイド方式のカードです。[Kyash](#) などが代表的な例です。

ブランドデビットカード同様、国際ブランドのカード番号が発行されます。

4.3 学生は何を使うべきなのか

ここまで主要なキャッシュレス決済手段についての簡単な説明をしてきました。では、学生はどの決済手段を使うべきなのでしょう。以下では、決済手段を絞り込むいくつかの基準を記します。

目的を見失わない (重要)

あまりにも沢山の「キャッシュレス決済」が存在するものですから、次第に、ユーザーとしてもどれを選べば良いのかが分からなくなってきます。特にコード決済事業者などは、それぞれがユーザーを集めようと躍起になって、あの手この手で様々なキャンペーンを打ち出しています。すると、「お得感」にめっぽう弱い我々は、あっちのアカウントを作って、こっちのカードも発行して……と無秩序に手を伸ばしてしまいがちです。しかし、我々は**キャッシュレス決済を利用する目的を見失ってはなりません**。その目的とは、

- 現金を取り扱う負担を軽減する
- お金の動きを把握しやすくする

に尽きます。キャンペーンや還元はあくまで副次的なもので、それらのために余計な複雑さを取り入れてしまっただけではキャッシュレス決済を使う意味がありません。現金の利用機会を減らせただけでも十分に儲けものだと考えるべきでしょう。

汎用性の高いものを選ぶ

どれほどよく出来た決済手段でも、1つの店舗でしか使えないのであれば意味がありません⁵。なるべく汎用性が高いものを選ぶことで、幅広いシーンで現金を扱うことなく生活を送ることができます。例えばコード決済なら J-Coin Pay より PayPay, カードの国際ブランドであれば American Express より Visa を選ぶといった具合です⁶。

年齢を考慮する

学生にとって一つの壁となるのが年齢です。例えばクレジットカードは18歳にならなければ作れませんし、デビットカードにも15歳以上という制限が課されている場合が多く見られます。当然ながら、自身の年齢で利用可能な決済方法を探す必要があります。

フォールバック手段を意識する

当たり前のことですが、一つの決済手段のみに依存することは高いリスクを負うことになります。キャッシュレス決済が使えない事態に備えてフォールバック手段を用意する必要があります。最低でも、現金を持ち運ぶことは必要です。基本的に現金はどのような場所でも通用します。キャッシュレス決済を利用するからといって現金を持ち歩かない理由はありません。筆者は、異なる種類のキャッシュレス決済手段を合計で2つ、加えて現金があれば良いと考えています。

4.4 結論

以上の基準を踏まえて、筆者が学生に勧めたいキャッシュレス決済手段を紹介します。

住信 SBI ネット銀行のデビットカード

まずは[住信 SBI ネット銀行のデビットカード](#)です。

私がこれを勧める第一の理由は、デビットカードとしては珍しく**年齢制限が無い**ことです⁷。一般にデビットカードは15歳以上でなければ作れないことが多いですが、住信 SBI ネット銀行に関しては、親権者の同意を得れば口座開設・デビットカードの発行が可能です。

また、タッチ決済 (Mastercard コンタクトレス) に対応している上、**Google Pay / Apple Pay のタッチ決済も利用可能です**。これによって、タッチ決済に対応している店舗ではスマホをかざすだけで支払いが完了します。

また、スマホアプリの出来が非常に良く扱いやすいこともメリットです。学生が自主的に管理する財布代わりに口座として、こうしたネットバンクは非常に適していると言えるでしょう。

PayPay

コード決済市場において圧倒的な地位を誇る PayPay は、やはりキャッシュレス生活を志向する上で欠かせない存在になってきます。他のコード決済が使えて PayPay が使えないという店舗は非常に少ないですし、PayPay のみに対応しておりクレジットカードが使えない店舗などもたくさんあります。もちろん PayPay には年齢制限も無いため自由に使うことができます。先述の銀行口座からチャージできるように設定すれば、なお有効に活用できると思われれます。

以上の2つが、筆者が学生に勧めたいと考えるキャッシュレス決済手段です。どちらを主とするかは個人の好みに依りますが⁸、この2つを利用していれば、キャッシュレス決済に対応している店舗であればほとんど現金を取り扱う必要はないはずです。学生の皆さんがキャッシュレス決済を有効活用し、手間を減らし、お金の動きを可視化し、快適な消費生活を送ることを切に願います。

4.5 あとがきのようなもの

昨年に引き続き「パソコン部の部誌」には似合わない技術色の薄い内容になってしまいましたすみません。皆さんの作業ブランチを覗いてみるとしっかりとパソコン部らしいことをしていて肩身が狭い……キャッシュレス決済も人々の生活を支えるデジタル技術の一環ということで、どうかご容赦いただければと思います。来年はAndroidアプリかWebアプリの開発に関する話を載せられれば良いなと思っています（一応構想があります）。

そして部誌の発行に携わった方々へ。2年連続で締め切りを大幅に超過してしまい本当に申し訳ありません。部長には大変なお手間をおかけしてしまいました。来年こそは締め切りまでにpushできるよう努力する所存です。

1. [第1部 第2章 第2節 \(5\)キャッシュレス決済に対する意識 | 消費者庁](#)
2. iDやQUICPay+でデビットカードを利用することもできますが、長くなるため割愛します。←
3. その手軽さゆえ、タッチ決済のみで決済ができる金額には上限があり、それを越えるとサインや暗証番号の入力が求められます。日本の上限は1万円です。ただし、Apple PayやGoogle Payは、CDCVMと呼ばれる仕組みによりスマホ側の本人確認(Face IDなど)によって上限額を越えた支払いが可能になります。←
4. 実際にはクレジットカードとブランドデビットカードとブランドプリペイドカードは区別されていて、本当にクレジットカードを利用していないと弾かれることも稀にあります。←
5. 特定の系列店のみで使えるハウスマネーと呼ばれるキャッシュレス決済も存在しますが、これはもちろん汎用性に欠けるものとなります。←
6. 特定の決済方法を貶める意図はありません。あくまで市場シェアの話をしています。←
7. [\[15歳未満のお子さまの口座\] キャッシュカードを... | よくあるご質問TOP | NEOBANK 住信SBIネット銀行](#)
8. 筆者の個人的意見ですが、PayPayを主とする方が便利かもしれません。デビットカードは明細が届くまでに数日かかる場合があり、その間は店舗名が確認できません。PayPayであれば店舗名が即時反映され、管理が容易になるかもしれません。←

第5章

ソフトウェアシンセサイザを作ってみよう

Nagaso.cpp

5.1 はじめに

こんにちは。6期生の Nagaso.cpp です。昨年に引き続き部誌の執筆に参加しました。今年ソフトウェアシンセサイザの自作に挑戦してみました。趣味の DTM でプラグインシンセサイザをいじっているうちに自作したくなったので、簡単なものを作ってみることにしました。

5.2 シンセサイザとは？

シンセサイザとは、電気が持つ波を変換にして音にし、それを組み合わせて様々な音色を出すことのできる電気楽器です。エレキギターやエレクトリックピアノも電気楽器ですが、これらは発音自体はアコースティック（「音を鳴らす」という行為自体には電気を要さない）なのに対して、シンセサイザは発音にも電気を使うという特徴があります。具体例として、DTM での作曲に用いられるプラグインシンセサイザ「Vital」を見てみましょう。



▲ 図 5.1 : Vital の画面

こちらが Vital の画面です。この画面を参考に、シンセサイザの主な機能を見ていきましょう。



▲ 図 5.2 : Vital : オシレーター

左上ではオシレータの設定をすることができます。オシレータは音の元となる波形を作りだすものです。波形には様々な種類があり、サイン波やのこぎり波、パルス波などがあります。この波形を更に細かく弄ったり、その他様々なパラメータを弄ったりすることができます。



▲ 図 5.3 : Vital : エンベローブ

右上ではエンベローブの設定をすることができます。エンベローブは、以下の6つのパラメータを変更することで、オシレータで設定した音の鳴り方を調整することができます。

- DELAY
音の入力があってから音が鳴り始めるまでの秒数を設定できます。
- ATTACK
音の入力があってから最大音量に達するまでの時間を設定できます。何も弄らなければ鳴った瞬間に最大の音量になります。
- HOLD
最大音量になってからその状態をキープする秒数を指定できます。
- SUSTAIN
音を鳴らし続けた時に最終的に到達する音量を設定できます。例えば、ピアノは鍵盤を押し続けると減衰して音は聞こえなくなるので、SUSTAIN は0になります。
- DECAY
SUSTAIN で設定した音量に達するまでの時間を設定できます。
- RELEASE
音の入力が終わってから完全に音が消えるまでの時間を設定できます。



▲ 図 5.5 : Vital : Filter

左下ではフィルターの設定をすることができます。フィルターは鳴っている音の高域あるいは低域の周波数の成分をカットして、音色を変化させることができます。

5.3 実装する機能

さて、ここからは本題に入っていきます。さすがに Vital のような完璧なものを作るのは今の私の技術力では無理なので、オシレータ・エンヴェロープの2つの機能を搭載したシンプルなシンセサイザを作ることにしました。

- オシレータ サイン波とのこぎり波の2つを実装し、切り替えられるようにします。また、Vital 同様波形を表示して視覚的に分かりやすくします。
- エンヴェロープ DELAY 以外の5つを実装します。

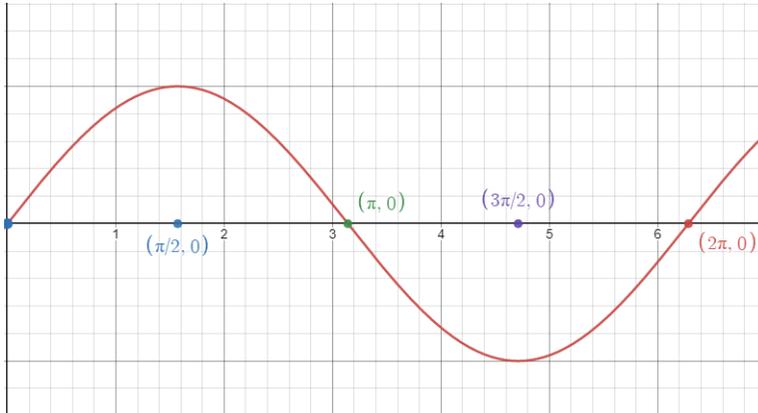
5.4 使用する技術

言語：C++

フレームワーク：OpenSiv3D v0.6.9

5.5 音の波形を生成する

シンセサイザは音の波形をこねくり回して音を鳴らす楽器ですから、まずは波形についていくつか知っておきましょう。以下は $y = \sin x$ のグラフです。



▲ 図 5.6 : $y = \sin x$

このグラフの周期は 2π ですが、この $0 \sim 2\pi$ までの形が一秒間に何回繰り返されるかを表す数を**周波数** (frequency) といいます。この値が大きければ大きいほど、高い音になります。また、グラフの中心から最大変位までの差 (今回であれば1ですね) を**振幅** (amplitude) といいます。この値が大きければ大きいほど、大きな音になります。

これらを踏まえたうえで、Siv3D 上で音声波形データを生成していきます。

音声波形データを生成する

Siv3D で音声波形データを生成するには、`Wave` クラスを使用します。`Wave` クラスは `WaveSample` という「`double` 型の (端的に言えば) そのタイミングにおける音の大きさを表す値」を配列として保持します。配列なので宣言時には要素数を指定する必要があります。これは波形の秒数とサンプリング周波数の積で算出することができます。よって、`Seconds` 秒の `Wave` は、サンプリング周波数を `44100Hz` (これは Siv3D 側に `Wave::DefaultSampleRate` として用意されています) として以下のように表すことができます。

```
const auto lengthOfSamples = Seconds * Wave::DefaultSampleRate;

Wave wave(lengthOfSamples);
```

この `wave` の各要素に対して、音量を指定していきます。for 文を回して `wave` の各要素にアクセスしています。

```
wave[i].left = wave[i].right = static_cast<float>(w);
```

`WaveSample` は左右両方の音量を持つ値なので、両方指定します。今回はパンを振る予定はないので、両方同じ値にします。この音量を `w` で表しています。実装の都合上、`w` は `double` 型にしているので、`float` 型に変換しています。そして、この `w` で表された音量は、波形のグラフで指定します。今回はサイン波を想定しています。

```
const double sec = 1.0f * i / Wave::DefaultSampleRate;
const double w = sin(Math::TwoPiF * frequency * sec) * amplitude;
```

`sec` は波形が再生されてからどれくらい時間が経ったのかを表しています。`Math::TwoPiF` は 2π を表していて、周期が 2π 、周波数が `frequency`、振幅が `amplitude` のサイン波の `sec` における座標を `w` に代入しています。

5.6 エンベロープを作る

エンベロープの部分を作っていきます。各パラメータ (Attack, Hold, Decay, Sustain, Release) は `struct` で以下のようにまとめて管理します。

```
struct AHDSRConfig {
    double AttackTime = 0.1;
    double HoldTime = 0.1;
    double DecayTime = 0.1;
    double SustainLevel = 0.6;
    double ReleaseTime = 0.2;
};
```

エンベロープ本体を、`AHDRSEnvelope` クラスとして実装します。以降はこの中身を書いていきます。

```
class AHDRSEnvelope
{
    // エンベロープの処理を書いていく
};
```

エンベロープでは、Attack、Hold、Decay、Sustain、Release の順に状態遷移して処理を行うようにします。まず、エンベロープの遷移状態を表す `AHDSRState` クラスを実装します。

```
public:
    class AHDSRState
    {
        Attack, Hold, Decay, Sustain, Release;
    };
```

さらに、エンベロープ内で使用する変数を定義します。

```
private:
    AHDSRState CurrentState = AHDSRState::Attack; // 現在のエンベロープの状態
    double ElapsedTime = 0.0; // その状態に遷移してからの経過時間
    double CurrentLevel = 0.0; // 鳴っている音の音量
    double ReleaseStart = 0.0; // AHDSRState が Release になったときの音量
```

ここからはエンベロープの各遷移状態における処理を書いていきます。これ以降は全て `public` です。`ADSRConfig` の各値と単位時間の値を表す `DeltaTime` を引数に持つ `EnvUpdate` 関数を定義します。`EnvUpdate` 関数は、波形に遷移状態に応じた処理をするようにします。これを `switch` 文を使って実装します。

```
void EnvUpdate (AHDSRConfig &ahdsr, double DeltaTime)
{
    switch (CurrentState)
    {
        // 遷移状態によって処理を分ける
    }
}
```

ここからは各遷移状態の処理を書いていきます。

`Attack` では、`CurrentLevel` を `AttackTime` 秒かけて 0.0 から 1.0 まで増幅させます。Siv3D には、ベクトル A からベクトル B への線形補完ができる関数 `Math::Lerp()` 関数が用意されているのでそれを使います。

```
case AHDSRState::Attack:
    if (ElapsedTime < ahdsr.AttackTime) // ahdsr.AttackTime 秒かけて増幅
    {
        CurrentLevel = Math::Lerp(0.0, 1.0, ElapsedTime / ahdsr.AttackTime);
        break;
    }
    // 処理がすべて終わったら各値を更新
    ElapsedTime = 0;
    CurrentState = AHDSRState::Hold;
    [[fallthrough]]; // そのまま Hold に移行する
```

`Hold` では、`CurrentLevel` を `HoldTime` 秒の間 1.0 に維持します。

```

case AHDSRState::Hold:
    if (ElapsedTime < ahdsr.HoldTime)
    {
        CurrentLevel = 1.0;
        break;
    }
    ElapsedTime = 0;
    CurrentState = AHDSRState::Decay;
    [[fallthrough]]; // そのまま Decay に移行する

```

Decayでは、`CurrentLevel` を `DecayTime` 秒かけて `SustainLevel` まで減衰させます。こちらも `Math::Lerp()` 関数を用いて線形補完します。

```

case AHDSRState::Decay:
    if (ElapsedTime < ahdsr.DecayTime)
    {
        CurrentLevel = Math::Lerp(1.0, ahdsr.SustainLevel,
                                   ElapsedTime / ahdsr.DecayTime);
        break;
    }
    ElapsedTime = 0;
    CurrentState = AHDSRState::Sustain;
    [[fallthrough]]; // そのまま Sustain に移行する

```

Sustainでは、ノートがオフになるまで `CurrentLevel` を `SustainLevel` に維持します。

```

case AHDSRState::Sustain:
    CurrentLevel = ahdsr.SustainLevel;
    break;

```

Releaseでは、ノートがオフになったときに `ReleaseTime` 秒かけて `CurrentLevel` を 0.0 まで減衰させます。ReleaseTime 秒経ったら常に 0.0 を返すようにします。こちらも `Math::Lerp()` を用いて線形補完します。

```

case AHDSRState::Release:
    if (ElapsedTime < ahdsr.ReleaseTime)
    {
        CurrentLevel = Math::Lerp(ReleaseStart, 0.0,
                                   ElapsedTime / ahdsr.ReleaseTime);
    }
    else CurrentLevel = 0.0;
    break;

```

最後に、処理が終わったら `DeltaTime` 秒だけ時間を進めるようにします。

```
ElapsedTime += DeltaTime;
```

これで `EnvUpdate()` 関数の実装が完了しました。

最後に、ノートがオフになったときに `CurrentState` を `AHDSRState::Release` にするための `NoteOff()` 関数を実装します。

```
void NoteOff()
{
    ElapsedTime = 0;
    ReleaseStart = CurrentLevel;
    CurrentState = AHDSRState::Release;
}
```

これで `AHDSREnvelope` クラスの実装が完了しました。全体像は以下の通りです。

```
// AHDSR の初期値
struct AHDSRConfig {
    double AttackTime = 0.1;
    double HoldTime = 0.1;
    double DecayTime = 0.1;
    double SustainLevel = 0.6;
    double ReleaseTime = 0.2;
};

// AHDSR エンベロープ
class AHDSREnvelope {
public:
    // 遷移状態を表す State
    enum class AHDSRState {
        Attack, Hold, Decay, Sustain, Release
    };

    // ノートがオフになったときの処理
    void NoteOff()
    {
        ElapsedTime = 0;
        ReleaseStart = CurrentLevel;
        CurrentState = AHDSRState::Release;
    }

    // エンベロープの状態遷移
    void EnvUpdate (AHDSRConfig& ahdsr, double DeltaTime)
    {
        // 状態によって処理を分ける
        switch (CurrentState)
        {

            case AHDSRState::Attack:
```

```
    if (ElapsedTime < ahdsr.AttackTime)
    {
        CurrentLevel = Math::Lerp(0.0, 1.0,
                                   ElapsedTime / ahdsr.AttackTime);
        break;
    }
    ElapsedTime = 0;
    CurrentState = AHDSRState::Hold;
    [[fallthrough]]; // そのまま Hold に移行する

case AHDSRState::Hold:
    if (ElapsedTime < ahdsr.HoldTime)
    {
        CurrentLevel = 1.0;
        break;
    }
    ElapsedTime = 0;
    CurrentState = AHDSRState::Decay;
    [[fallthrough]]; // そのまま Decay に移行する

case AHDSRState::Decay:
    if (ElapsedTime < ahdsr.DecayTime)
    {
        CurrentLevel = Math::Lerp(1.0, ahdsr.SustainLevel,
                                   ElapsedTime / ahdsr.DecayTime);
        break;
    }
    ElapsedTime = 0;
    CurrentState = AHDSRState::Sustain;
    [[fallthrough]];

case AHDSRState::Sustain:
    CurrentLevel = ahdsr.SustainLevel;
    break;

case AHDSRState::Release:
    if (ElapsedTime < ahdsr.ReleaseTime)
    {
        CurrentLevel = Math::Lerp(ReleaseStart, 0.0,
                                   ElapsedTime / ahdsr.ReleaseTime);
    }
    else CurrentLevel = 0.0;
    break;
}

// DeltaTime 秒だけ時間を進める
ElapsedTime += DeltaTime;
}

private:
    AHDSRState CurrentState = AHDSRState::Attack; // 現在のエンベロープの状態
    double ElapsedTime = 0.0; // その状態に遷移してからの経過時間
    double CurrentLevel = 0.0; // 鳴っている音の音量
    double ReleaseStart = 0.0; // AHDSRState が Release になったときの音量
};
```

5.7 波形を生成する

生成した音声波形データとエンベロープのパラメータをもとにして実際に再生される波形を生成する `WaveGenerator` 関数を定義します。実装は以下の通りです。

```
Wave RenderWave(double seconds, double amplitude, double frequency, AHDSRConfig &ahdsr)
{
    uint32 Seconds = static_cast<uint32>(floor(seconds));

    const auto lengthOfSamples = Seconds *
                                   Wave::DefaultSampleRate;

    Wave wave(lengthOfSamples);

    AHDSREnvelope envelope;

    float DeltaTime = 1.0f / Wave::DefaultSampleRate;

    auto NoteOffTime = lengthOfSamples / 2;

    for (uint32 i = 0; i < lengthOfSamples; ++i)
    {
        if (i == NoteOffTime)
        {
            envelope.NoteOff();
        }
        const double sec = 1.0f * i / Wave::DefaultSampleRate;
        const double w = sin(Math::TwoPiF * frequency * sec) *
                          amplitude * envelope.GetCurrentLevel();
        wave[i].left = wave[i].right = static_cast<float>(w);
        envelope.EnvUpdate(ahdsr, DeltaTime);
    }

    return wave;
}
```

引数には、波形の秒数、振幅、周波数とエンベロープのパラメータを取り、音声波形データを返します。基本的な処理の流れは前の章で説明したものとほぼ同じです。秒数は GUI 作成の都合上 `double` 型で受け取りますが、波形の長さは整数値であるため床関数で小数点以下を切り捨てた上で `uint32` 型 (標準 C++ の `unsigned int` に該当) に変換しています。大元の音声波形データを生成した後、for 文を回して `WaveSample` 1つ1つに対して音源の増幅・減衰処理をしています。なお、エンベロープの Release 処理については、ノートがオフになるタイミングを自由に設定できるところまで出ていないので、内部側で波形の半分のところで固定しています。そのため、例えば波形の長さが3秒であれば、1.5秒のタイミングで Release 処理に入るようになっています。

5.8 波形を再生する

いよいよ `Main` 関数の処理に入っていきます。とは言っても特に何かするわけではなく、`WaveGenerator` 関数を呼び出して受け取った音声波形データを Siv3D の `Audio` クラスに渡して再生するだけです。あとは仮ですが GUI の処理も書いています。一行がかなり長いので、紙面の都合上見づらくなってしまいますがご容赦ください。

```
// GUIの描画用
const auto SliderHeight = 36;
const auto SliderWidth = 400;
const auto LabelWidth = 200;

// const Vec2 offset = { 50, 300 };

void Main()
{
    double amplitude = 0.2;
    double frequency = 440.0;
    double seconds = 3;

    AHDSRConfig config(0.1, 0.1, 0.1, 0.6, 0.2);

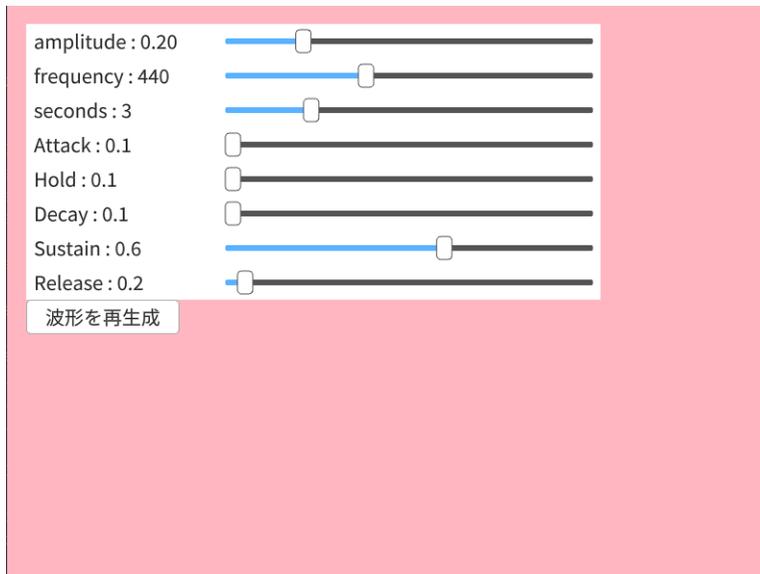
    Audio audio(WaveGenerator(seconds, amplitude,
                              frequency, config));

    while (System::Update())
    {
        Vec2 pos(20, 20 - SliderHeight);
        SimpleGUI::Slider(U"amplitude : {:.2f}"_fmt(amplitude), amplitude, 0.
0, 1.0, Vec2{ pos.x, pos.y += SliderHeight }, LabelWidth, SliderWidth);
        SimpleGUI::Slider(U"frequency : {:.0f}"_fmt(frequency), frequency, 10
0.0, 1000.0, Vec2{ pos.x, pos.y += SliderHeight }, LabelWidth, SliderWidth);
        SimpleGUI::Slider(U"seconds : {:.0f}"_fmt(seconds), seconds, 1, 10, V
ec2{pos.x, pos.y += SliderHeight}, LabelWidth, SliderWidth);
        SimpleGUI::Slider(U"Attack : {:.0}"_fmt(config.AttackTime), config.At
tackTime, 0.1, 3.0, Vec2{ pos.x, pos.y += SliderHeight }, LabelWidth, SliderW
idth);
        SimpleGUI::Slider(U"Hold : {:.0}"_fmt(config.HoldTime), config.HoldTi
me, 0.1, 3.0, Vec2{ pos.x, pos.y += SliderHeight }, LabelWidth, SliderWidth);
        SimpleGUI::Slider(U"Decay : {:.0}"_fmt(config.DecayTime), config.Deca
yTime, 0.1, 3.0, Vec2{ pos.x, pos.y += SliderHeight }, LabelWidth, SliderWid
th);
        SimpleGUI::Slider(U"Sustain : {:.0}"_fmt(config.SustainLevel), config
.SustainLevel, 0.0, 1.0, Vec2{ pos.x, pos.y += SliderHeight }, LabelWidth, Sl
iderWidth);
        SimpleGUI::Slider(U"Release : {:.0}"_fmt(config.ReleaseTime), config.
ReleaseTime, 0.1, 3.0, Vec2{ pos.x, pos.y += SliderHeight }, LabelWidth, Slid
erWidth);

        if (SimpleGUI::Button(U"波形を再生成",
                               Vec2{ pos.x, pos.y += SliderHeight }))
        {
```

```
        audio = Audio(WaveGenerator(seconds, amplitude,  
                        frequency, config));  
        audio.play();  
    }  
}
```

また、(仮ですが) GUI は以下のようにになっています。画像の各値を自由に変更できるようにしています。



▲ 図 5.7：シンセサイザのGUI

5.9 おわりに

オシレータ1つとエンベロープというごくごくシンプルなものにはなりますが、なんとかシンセサイザを作ることができました。音響工学の知識が皆無であるため、最初は開発の方針すらも立ちませんでした。色々調べてなんとか作り上げることができ嬉しく思います。ゆくゆくは実際の DAW に VST プラグインとして使えるようにして、私の作曲に生かせるようなものが作れたらと思います。また、大学で研究したいと思っている音響工学と計算機科学においても、何か生かせればと思います。最後になりますが、拙い文章であったにもかかわらずここまで読んで下さりありがとうございました。去年に引き続き、部誌の執筆に参加できたことを嬉しく思います。来年は(浪大しなれば)大学生になりますが、部誌が存続するのであれば是非参加したいと思います。またどこかでお会いできることを楽しみにしています。ありがとうございました。

参考文献

ソフトウェアシンセサイザを作る その1：サイン波でMIDIを再生する
(https://qiita.com/agehama_/items/7da430491400e9a2b6a7)

Special Thanks

- Siv3D
- Reputeless (Siv3D 開発者)
- agehama (参考文献執筆者)
- この記事を読んでくださった方々

第6章

ニコニコ動画風コメント掲示板2はこうやって作られた

張替健太/hrkn63_hnm

6.1 はじめに

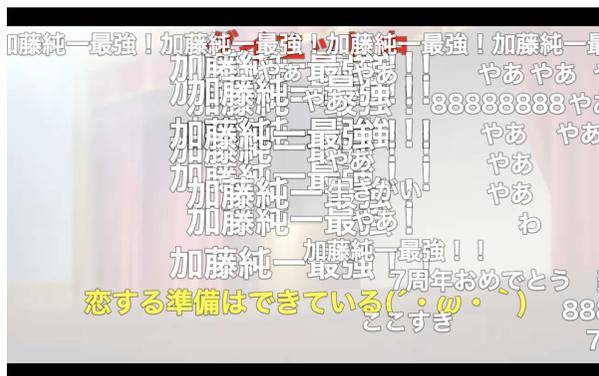
みなさんこんにちは。6期生の張替健太です。趣味で生徒会をやったり、SNSを作ったりしています。僕の記事では、おそらく文化祭当日に展示されているであろう「ニコニコ動画風コメント掲示板2」について紹介します。

ソースコードはこちらです。<https://github.com/hrkn63-hnm/nikoniko-board>

6.2 ニコニコ動画風コメント掲示板2とは

まずは「ニコニコ動画風コメント掲示板2」の「ニコニコ動画風」について。

ニコニコ動画(<https://www.nicovideo.jp/>)とは、YouTubeと同じ動画投稿プラットフォームの一つです。誰かがあげた動画を他の誰かが閲覧して楽しむことができます。ニコニコ動画の醍醐味といえ、この弹幕コメントです。



▲ 図 6.1：弹幕コメントのいい例

【実況】アマガミやるお (^ω^) part1/ うんこちゃん
(<https://www.nicovideo.jp/watch/sm27054304>)より

このニコニコ動画のような熱いコメントが横から流れる掲示板。それが「ニコニコ動画風コメント掲示板2」です。



▲ 図 6.2 : 実際のプロダクト

画像は開発中のものです。実際のものとは少し違うかも。

この「ニコニコ動画風コメント掲示板」は去年の文化祭でも掲示しました。そのパワーアップバージョンが今回のものになります。

6.3 ニコニコ動画風コメント掲示板2を支える技術

`Next.js` というフレームワークを用いています。データベースは `Firestore` を使っていて、`GitHub` へ push すると `Vercel` にそのまま deploy できるようになっています。

`Next.js 13` では `yarn create-next-app` を実行すると `Tailwind.css` を使うかどうか聞いてくるようになり、誘惑に負けてそのまま採用しました。

6.4 技術的な話

個人的トピックを厳選しました。

1. 型定義について

`TypeScript` を使用しているので型定義を行うのは当たり前ですが、`tsconfig.json` ファイルの型定義設定をいつも甘くしているせいで痛い目に遭いました。

今回、コメントが何文字か、コメントのフィールドに含まれるIDがどんな数で割り切れるのか、で分岐させてCSSを変更させることで色や流れる高さを変えるような仕様を目指しました。

各コメントデータのフィールドに含まれるIDを調べるのは少し面倒で `forEach` などをこねくり回して使う必要があるので、「ドキュメントIDをそのまま参照して割り切れるかどうか調べればいいや！」をしました。

結果、`Firestore` のドキュメントIDは `string` 型でないといけないのにも関わらず、`number` 型をぶち込んでいることに気が付かず格闘しました。

ベストプラクティスではないことはわかっていたのですが、ドキュメントIDは `string` 型にして、分岐させるときは `Number()` を使って `number` に戻して処理を行いました。

2.この世の数字の分類について

先ほどの話にもある通り、「どんな数で割り切れるか」「割った余りがいくつか」でコメントを分類しているため、「2で割り切れるかどうか」という処理しか行わなければ、ドキュメントID「10980」と「34506」は同じCSSが適応され、二つのコメントが被り、見づらくなってしまいます。

僕は脳死で「3で割り切れるかどうか」「5で割り切れるかどうか」というように素数で処理を重ねましたが、

これは絶対に良くない!!

というのも、これは明らかに偏りの出る分類方法だからです。

1~100の中で2の倍数は50個。

2の倍数でなく、3の倍数なのは $33-16=17$ 個。

2の倍数の方が明らかに多く、ムラが出来てしまいます。16→8→4→2の順で調べた方がいいのか、`else if` を使うのが良くないのか。はたまた、うまく分類なんてできないのか...

プログラミングが上手な人はこういう時うまく対処できる人なんでしょう。多分。

3.コメントを横に流すCSS

実は、`ChatGPT` に聞いてCSSをコピーしました。多少の改善は行なったものの、凄すぎます。

ニコニコ動画のようにテキストを横に流すCSSを教えてください

ニコニコ動画のようにテキストを横に流すには、CSSの`marquee`プロパティを使用することができます。

以下は、`marquee`プロパティを使用してテキストを横に流すCSSの例です。

```
css
.marquee {
  white-space: nowrap; /* テキストを折り返さないようにする */
  overflow: hidden; /* 領域外の部分を隠す */
  box-sizing: border-box; /* padding&borderを要素の内側に含める */
  animation: marquee 20s linear infinite; /* アニメーションを指定 */
}

@keyframes marquee {
  0% {
    transform: translateX(100%); /* 最初はテキストを領域外に移動させる */
  }
  100% {
    transform: translateX(-100%); /* 最後はテキストを領域外に移動させる */
  }
}
```

このCSSを適用するためには、HTMLで`<div>`要素などの適当な要素を作成し、そのクラスに`marquee`を指定します。

```
html
<div class="marquee">ここに横に流したいテキストを入力</div>
```

このようにすることで、テキストが横に流れるようになります。ただし、`marquee`プロパティは、一部のブラウザではサポートされていないため、動作しないことがあります。また、Webアクセシビリティの観点からも、使用は推奨されていません。代替手段を検討することをお勧めします。

tailwindcssを使っているせいかうまくいきません

すみません、`marquee`プロパティは、現代的なブラウザで動作しない場合があるため、使用は推奨されていません。代わりに、JavaScriptやCSSアニメーションを使用することをお勧めします。

以下は、tailwindcssを使用してテキストを横に流すための例です。

```
html
<div class="overflow-hidden whitespace-nowrap">
  <div class="inline-block animate-marquee">
    <!-- 横に流したいテキストを入力 -->
    <span class="text-red-500 font-bold">テキストを横に流す</span>
  </div>
</div>
```

```
css
@keyframes marquee {
  0% {
    transform: translateX(0%);
  }
  100% {
    transform: translateX(-100%);
  }
}

.animate-marquee {
  animation: marquee 10s linear infinite;
  display: inline-block;
  white-space: nowrap;
}
```

`whitespace-nowrap`クラスを使用することで、テキストが折り返されず一行に表示されるようになります。`overflow-hidden`クラスを使用することで、領域外の部分を隠すことができます。

また、`animate-marquee`クラスを作成し、`animation`プロパティを使用して、テキストを横に流すアニメーションを指定します。`display: inline-block`を指定することで、テキストが横に流れるようになります。

上記の例は、``要素に対して適用していますが、適当な要素に対しても同様に適用できます。また、`animation`プロパティの値を変更することで、アニメーションの速度を変更することができます。

▲ 図 6.3 : ChatGPTの功績

6.5 おわりに

自分で好きなことをやっているうちに、知識なんて自然とつくもんです。

この掲示板を数時間で作ることができたのも、ホスティングの仕方、データのやり取りの仕方、JSXの条件分岐の仕方、`useState`の使い所など昔はできなかったことひとつひとつが勝手に身につけていたからです。

この記事を読んで興味を持った人、Webに興味がある人なんかは、初めの方に記載したソースコードをコピーして同じように作ってみてね。

読んでくださり、ありがとうございました。

第7章

茨城県教育情報ネットワークの脆弱性を報告した話

eno1220

本記事は、茨城県教育情報ネットワークが認知していない未修正の脆弱性を公開することを意図したものではありません。また、茨城県教育情報ネットワークを構成する各 web サイトなどで脆弱性を発見した場合は、報告をするようにしてください。

7.1 はじめに

こんにちは。2022-23 年度パソコン部副部長の eno1220 です。普段は競プロや web 開発を行っています。さて、この記事では 2022 年 12 月に私と卒業生である Ryoga.exe が発見した、茨城県教育情報ネットワークの脆弱性について紹介を行います。

7.2 茨城県教育情報ネットワークについて

自分で説明するのは面倒なので、公式の説明 [1] を引用します。

茨城県教育情報ネットワークは、統一したセキュリティ対策のもとで県立学校とその他の教育機関を光ファイバー回線で結び、より高度な「教育の情報化」の実現を目指して構築されたネットワークシステムです。教職員向けポータルサイトは、県内の小学校・中学校・義務教育学校・高等学校・中等教育学校・特別支援学校とすべての教職員約 26,000 人の情報共有基盤として、学習支援や校務の効率化など様々な活用が進んでいます。

調べたところによれば、平成 17 年ごろから（つまり現在 6 年次生である私が生まれた頃から）運用を開始しているようで、全国的に見ても早い段階から「教育の情報化」の実現を目指していたことが推察されます。

[1] <https://kyoiku.pref.ibaraki.jp/education-policy/page-10899>

教育情報ネットワークは2022年8月にリニューアルが実施され、それまで運用されていたシステムと比べてさまざまな機能 [2] が追加されました。また、このリニューアルでポータルサイトが刷新され、フロントエンドが `Vue.js` で記述されるなどモダンな技術で構成されるようになりました。

ポータルサイトには、私たち生徒や教員がアクセスすることができ、資料の共有 [3] や教材の閲覧を行うことができるほか、教員に関しては公文書の共有、お知らせの発信などを行うことができます。

7.3 脆弱性の発見までの経過

- 2022年12月6日深夜に、eno1220 がポータルサイトの設計やコードを見るためにブラウザの Developer tool で遊んでいた
- この中で、セッションストレージ [4] にログイン時に API サーバから fetch してきたユーザ情報 [5] が保存されていることに気づいた
- 翌日12月7日の部活中に前日に気づいたことをもとに Ryoga.exe とポータルサイトを調べていたところ、脆弱性を発見した

7.4 脆弱性の概要

セッションストレージに保存されているユーザ情報を書き換えることで、ポータルサイトの特定ページにおいてシステム管理者モードへと権限を不正に昇格することができる。これにより、本来閲覧することのできないコンテンツが閲覧できる状態となった。

7.5 脆弱性の詳細

ポータルサイトは前述の通り、フロントエンドは `Vue.js` で記述されており、API サーバとやり取りをする形で動作しています。ポータルサイトにログインするには、茨城県教育委員会から各生徒や教職員に提供されている Google アカウントが必要であり、これを用いて SAML 認証を行なっているように見えます。 [6] (筆者は認証・認可に詳しくないため詳細は省略する。)

Developer tool の「ネットワーク」タブから通信の様子を監視していると、ログイン後に API サーバを叩いてユーザ情報を取得し、セッションストレージに保存していることがわかりました。セッションストレージに保存されている情報には、ユーザ名やユーザ ID、組織（学校）の ID などが含まれていました。

[2] 公開されている範囲で言えば、「教材データベース」が一般の方でも閲覧できるようになりました。

[3] 主に異なる学校間や教育委員会から県内全体に発信されるような資料の共有を行なっている。

[4] ブラウザのデータを保存しておく領域の一つ

[5] ユーザ名、個人 ID、学校 ID、ポータルサイトの設定など

[6] Google Cloud Platform の提供する機能を用いてうまく構築していると考えられる。

さて、調べを進めていくと、この中に `accountDivision` という項目があることに気づきました。値が `b1` になっていることと合わせて考えてみると、これはユーザの分類を表しているであろうことが推測されます。この値を適当に書き換えていたところ、画面上に「モード切替」ボタンが出現し、「システム管理者モード」へとモードが切り替わることを発見しました。

ポータルシステムのトップページには「お知らせ」を表示する欄があり、生徒が閲覧した場合においては何も表示されないようになっています。ところが、「モード切替」ボタンを押し、「システム管理者モード」へとモードを切り替えると、「お知らせ」を表示する欄に投稿された「お知らせ」のリストが表示されてしまいました。

なお、上記以外で「システム管理者モード」へとモードが切り替わることで本来とは異なる挙動をするページはありませんでした。(上記以外の脆弱性は発見できませんでした。)

7.6 その後の経過

本来の挙動とは異なる表示がなされたと判断し、すぐに管理者に報告を行いました。

- 2022/12/7 17:30 脆弱性を発見
- 2022/12/8 09:30 脆弱性の詳細を報告
- 2022/12/8 15:30 脆弱性の再現の確認と開発元への通知完了の報告を受ける
- 2022/12/26 15:00 脆弱性修正の完了の報告を受ける

修正完了の報告を受けるまでに期間が少し空いていますが、どうやら報告を受けたかなり前には修正が完了していたようです。(12月20日ごろに学校を經由して連絡をいただいた際に、学校側で不手際があったらしい)

7.7 おわりに

脆弱性の報告をして、なにか責められるのではないかと...?とも思っていました。そのようなことはなく、迅速に対応いただきました。対応いただいた管理者の方々、開発者の方々に感謝いたします。

ところで、これを機にCTFとかやってみるか~と思ったまま半年くらいなにもやっていません...大学に行ったらやってみたいですね。

第8章

技術と人間 + pygameについて

Yowashita

8.1 はじめに

97.3%、皆さんはこの数字を見て何を思い浮かべるでしょうか。このパーセントを見てこれだと即決できる人はいないでしょう。しかし、実はこれは皆さんの身近なものにかかわる割合なのです。もう少し考えてみましょうか。

…もう何か思いつきましたか？では、正解発表をします。答えを言ってしまうと実はこれは2023年の高校生のスマホ所持率なのです。

このデータに従えば、仮にひとクラスに100人いたらそのうちスマホを持っているのはなんと97人という事になります。どうでしょうか、人によってはこれでも少ないと思うかもしれません。全員スマホを持っているのが当たり前だと思う人も多いでしょう。しかし、スマホは誕生してからたったの29年しか経っていません。それなのにいまではこれほど私達の日常に浸透しているのです。驚くべきことだと思いませんか？まあ、とはいってもそう言ってる私はスマホを持っていないんですけどね。上記の例に従えば、私は100人の内の3人となります、すっごくレアな存在なんですよ。だからといって、別に家が貧乏からという訳ではありませんよ、パソコンは持っていますしね。では何故スマホを持っていないかと言うと私はそれを必要だとは思わないからです。その理由について次の章で述べたいと思います。

8.2 スマホについて

確かにスマホは便利です。どこにいてもだれとでも連絡を取れますし、分からないことがあったら一瞬で調べることが出来ます。暇な時には手軽にゲームをすることも、さらには勉強にも使えます。正直、この現代でスマホを持っていない高校生なんて頭おかしいんじゃないかとも自分でも思いますし。実際、困ったことはたくさんありました。例えば外に行っても道が分からないとか、外で友達と連絡が取れないとか。一番困ったのは至る所で携帯電話番号を要求されることです。LINEでもChatGPTでも、Googleアカウントの作成にも、様々なサービスを受けるためには私の持っていない電話番号が必要なのです。私はそれを持っていないばかりにGoogleアカウントを作るときに大変な思いをしました。アカウントを作るにはGoogleから送られる認証番号を入力する必要があるのですが、それは通常、携帯電話番号にSMSという形で送られます。しかし、私にはそれがない。困っているとなんと流石は天下

のGoogle様、家の電話番号を代わりに入力し、固定電話で知らせてくれる機能があったんです。早速それを使い、電話がかかってくるのを待ちました。数秒後、受話器が鳴り、急いでとると、機械音声の無感情のトーンで「1...3...9...」という風に認証番号が淡々と言われました。携帯番号があったら入力するだけで済んだのにこんなめんどくさい遠回りを強いられたのです。このある意味貴重経験によっては私にスマホの必要を強く実感しました。スマホがあれば、こんな経験をする事もなかったのに、当時はそう思ったものです。

しかし、これほど様々な不便を経験してもやはり結局私はスマホは持たない方がいいと判断しました。何故か？ ちょっと極端かもしれませんが、スマホは私にとって有害でしかない様々な経験を通して感じたからです。皆さんは経験したことありませんか？ スマホがなければって。周りの人から良くそんな話を聞きます。スマホがあったから勉強できなかった、スマホを見ていて事故にあった...、このような話を上げれば枚挙にいとまがありません。ほとんどの人間はその本性が非常に弱いものです。近くに誘惑があればすぐそれに飛びついてしまう。完璧に自己を律せる人間なんてほぼいません。スマホはそんな弱い人間を蝕む...行ってしまえば悪魔の道具です。スマホの悪影響について調べると次々出てきます。鬱、依存症、睡眠時間の減少、放心...、有名な話ですが、かのスティーブジョブスは自分の息子にスマホを与えなかったそうです。その悪影響をだれよりも知っているから。このように、ひとたびスマホの悪影響について調べたら、誰もがスマホを嫌うでしょう、しかしいったい何人がそれを手放せるでしょうか。これを読んでいるあなたもきっと、いや絶対捨てられないでしょう。だったら最初から持たなければいい...なんてさすがに言いすぎですけど、まあそんな感じで私はスマホを持たないのです。もう少し理由を上げるとスマホが高いからとか、思考力がそがれるからとか、スマホが高いからとか、スマホが高いからとか色々な理由があるんですけどね。しかし、勘違いしないでほしいのは私はスマホを完全に嫌悪しているわけではないのです。今までめっちゃくちゃ悪口を言ってきましたけどスマホが憎いわけではないのです。悪いのはスマホと言う優秀な技術ではなく、それを使う人なのです。事故を完璧に律することが出来る人はいないと言いましたが、その努力をすることは誰でもできます。結局のところ大事なものはスマホなどの技術との付き合い方なのです。

8.3 技術との付き合い方

今までの長い長い文章は前振りでここからが本題です。技術とは使う人によって毒とも薬ともなります。例えばナイフを強盗と外科医に渡したとしましょう。同じナイフでも明らかにその使い方は両者で違くなるでしょう。散々書き下ろしたスマホも同じです。人によって使い方は全く違います。一日中スマホを眺めて時間を無駄にしたと嘆く人もいれば、スマホに時間制限をかけるなどしてその影響を最小にとどめようとする人もいます。しかしながら、私が思うに実際にスマホの影響を少なくしようとしている人はそれほど多くないと感じます。不思議なことだと思いませんか？ 悪影響を認識しながらその対策をする人があまりいないなんて。私の無知のせいかもしれませんが、そのような物スマホぐらいだと思います。ではなぜ一体そうなっているのでしょうか？ 様々な理由が考えられるでしょうけれども、私はスマホ・インターネットに対するルール作りがその技術の驚異的な発展速度に追いつけていなかったからだと思います。最初に書きましたが、スマホは誕生してから29年しか経っていません。それなのに今はスマホが必要不可欠な時代になっているのです。私が小学生の頃はスマホやインターネットに対する教育なんて全くありませんでした。中学生になってから少しずつスマホ

やインターネットに対する教育が進んできたという印象です。つまり、この世代または前の世代はスマホに対して無知のまま生きてきたのです。だからスマホの悪影響について知っても手放せない、それほど深刻なものと捉えられないから、況や知っていたとしてももはや生活の一部となっているスマホを今更切り離すのは難しい。それが今の現状です。だからといってそのままにすればいいという訳ではありません。明らかな悪影響を前に何もしないのは愚か者のすることです。この現状に対して必要なのはスマホに対する理解です。無知なら知ればいい、ただそれだけです。「スマホは絶対手放さない」、「スマホは命よりも大事」、なんて浅いことを言ってる暇があるならその薄い板についての理解を深めてほしいです。そうすることで、多少なりとも自分の現状を理解でき、自分にとって何が大切か分かるはずですから。

今の情報通信社会では技術の発展速度は歴史上類を見ないほどです。だからこそ学校などの組織は技術に対応できない。だったら自分で調べるしかないのです。最近話題になっているAIも同じです。AIの代表格となったChatGPTは人間と遜色ないレベルの知識を持ち、一億人以上の人が利用しています。宿題の問題の解き方をChatGPTに聞けば完璧な答えがすぐに返ってきますし、なんなら人間顔負けのロールプレイもできます。その高性能な機能を悪事に使う人も多く、AI賛成派と反対派で世界の世論は二分されており、多くの人はAI時代という新しい時代に少なくとも何かしらの恐怖を抱えています。ではその恐怖はどこから生じるのでしょうか？・・・それは人々の無知から生じるのです。恐怖はたいてい無知から生じます。暗闇を怖がるのもそこに何がいるか見えない、分からない、知らないから。AIを怖がるのもその実態が何で、どんな影響があって、私達にどんな変化をもたらすのか誰も分からないからです。人々はそんな漠然としたイメージに怖がるのです。だから、理解することが必要なのです。技術と付き合うためには正しい理解、知識が必要なのです。漠然としたイメージの霧を晴らし、その正体を見極めることこそ大切なのです。折角あなたのポケットの中に世界の叡智へと導いてくれる道具があるのなら、それを使って霧を晴らすべきです。それこそ有意義な使い方だと思いませんか？

8.4 最後に

これまでの話でスマホとAIを例に出しましたが、驚異的な進化を遂げる技術は他にもあります。例えばロボット、画像生成、IoT、クラウド、5Gコネクティビティ、ユビキタスネットワーク……数えればきりがありません。この時代を生きているのは他人ではなく紛れもなくあなたです。現代の、多くの情報が錯綜するこの世の中はまさに、情報が氾濫しているようなものであり、そのすべてが何のセーフティーもなく私達に降りかかってきます。だからこそ、その情報の波に流されないようにするために私たち一人一人がそれらを取捨するという主体的な行動が必要です。そのために正しい知識と理解が必要なのです。そうすればどんな技術がやってきても上手く付き合うことができますし、恐怖に陥ることもないでしょう。長々と書いて申し訳ありませんが、次の言葉で締めとしたいと思います。

「Knowledge is power (知は力なり)」 - フランシスコ・ベーコン

8.5 おまけ:pygameについて

締めにするとか言ったけど、それは前章までの話という事で、ここからはさっきまでの話とは全く関係のないおまけの話です。ちょっとパソコン部として前の話だけではあれかなと思うのでこれをかくことにしました。すごく適当ですけど。タイトルのpygameというのはPythonというプログラミング言語の中のライブラリです。ライブラリと言うのは「ある目的のために機能をまとめたパッケージの総称」です。まあそれはどうでもいいのでこのpygameについて説明したいと思います。これはPython環境下でゲーム開発を簡単にすることを目的に作られたライブラリです。実際にコードを書いてみます。

```
# 事前にパソコンに導入したpygameを読み込む
import pygame

# 初期化する
pygame.init()

# スクリーンの大きさを決める
screen_width = 600
screen_height = 600

# リフレッシュレートの設定
FPS = 60
clock = pygame.time.Clock()

# 色
black = (0,0,0)

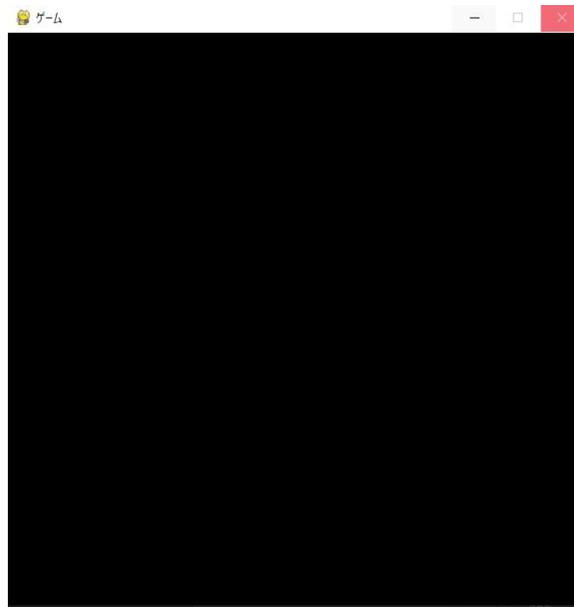
# ウィンドウの設定
screen = pygame.display.set_mode((screen_width,screen_height))
pygame.display.set_caption("ゲーム") # タイトルを決める

# メインループ
run = True #ゲームが動いている時
while run:
    # 背景
    screen.fill(black)

    for event in pygame.event.get():
        if event.type == pygame.QUIT: # ×マークを押したらゲームが閉じるようにする
            run = False
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_ESCAPE: # Escキーを押したらゲームが閉じるよう
にする
                run = False

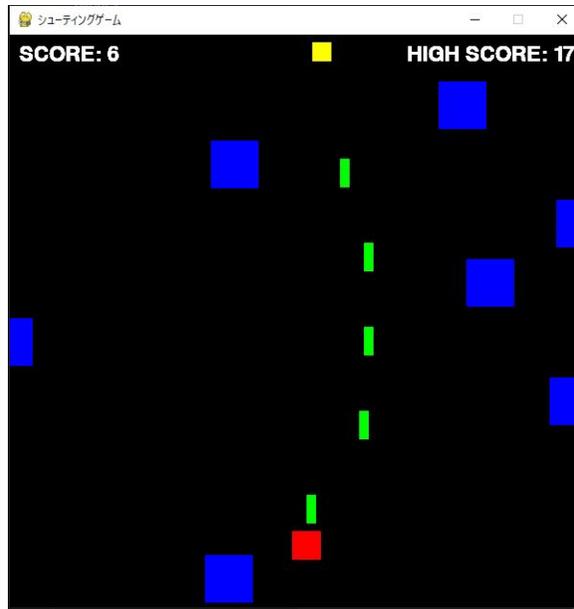
pygame.quit()
```

このコードを実行したら



▲ 図 8.1 : 黒枠

という黒枠が出てきます。はい、このコードはこれだけです。でも、これだけではなんにもできませんからここに様々な要素を付け加えます。例えば操作できる自機を追加したり、敵を追加したり、弾を追加したり、スコア機能を追加したり。そうすることで1つのゲームを作ることが出来ます。この要素の追加という点においてpygameは非常に便利なのです。キー入力の結果とから、スコアとか、最初からある程度の型が作られているのでゲーム開発がやりやすくなっています。その機能を使って私も文化祭用にゲームを作りました。そのコードは400行近くあり長すぎるので書けませんが、代わりに画像を乗せたいと思います。



▲ 図 8.2 : 完成品

このように、自機と、そこから出る弾と、敵とスコアを追加して簡単なゲームを作りました。時間がたつごとに敵の速度が上がるようにも設定しています。そして作ったゲームはそのままでいいのですが、pyinstallerという別のライブラリを使うとexe化することが出来ます。exe化とは「開発言語に依存することなくプログラムを実行できるようにしたもの」のことで、これを使うことによってPython環境のないパソコンでもゲームを動かすことが出来るのです。実際私もノートパソコンで作ったゲームをexe化して学校のパソコンに入れました。

概要だけは説明したのでおまけはこんなもんにしたいと思います。pygameに興味を持ったという人は調べてみて、触ってみるのが一番でしょう。習うより慣れよ、という事でこれで終わりにしたいと思います。ここまで読んでくださった人はありがとうございました。

第9章

AIと一緒にゲームを作ろう!

tarokko

9.1 はじめに

こんにちは。6期生の tarokko です。先日、幕張メッセで行われたイベントのゲーム大会に行く機会があったのでニンテンドー3DSを持って行ったのですが、一日で12人のプレイヤーとすれ違いました。まだすれ違い通信をすることがあるのかと感動した反面、3DSというハードの時代の終わりを感じ、名残惜しく感じました。

本章では他の変態たち(誉め言葉)の記事を読んでよくわからないと思った人にも読みやすいように心がけ、プログラミングに関する知識がない人向けに注釈を入れた記事を書いていますので、ぜひ最後までお読みいただくと幸いです。

9.2 AIについて

昨今のAIの発達は凄まじく、ニュースでも取り上げられることが増えてきました。具体的には対話型AIの ChatGPT や BingAI、画像生成AIの Midjourney や Stable Diffusion などが話題に挙げられることが多いと思います。それ以外にも作曲AIや小説AIなどが存在し、ジャンルは多岐にわたっています。本章では、対話型AIの ChatGPT が出力したソースコード [1] でゲームを作っていきます。

9.3 ゲーム制作におけるルール

ゲームを作っていく前にいくつかルールを定めていきたいと思います。

1. 無料版の ChatGPT(GPT-3.5) にコードを出力させる
2. 使用言語は JavaScript
3. 動作環境は JSFiddle
4. 基本的にソースコードはAIが出力した文章をそのまま使用する
5. ソースコードに含まれる数値は例外として人の手による変更を許可する

これらのルールを定めた理由は以下の通りです。

[1] コンピュータに命令を与える文字列のこと

1. 無料版であれば誰でも同じ環境が使用できるため、真似がしやすいから。また、インターネットに上がっている似たような記事では、GPT-4を使っている記事がいくつか見受けられたため、それらとの差別化のため。
2. 環境構築 [2] の必要がないから。
3. ブラウザで実行コンピュータプログラムをコンピュータ上で動作させることできるサービスの中でも、使いやすく感じたから。
4. AIでゲーム制作をするという趣旨のもと、人の手による介入を避け、AIが出力したソースコードだけでゲームを作っていくため。
5. 主に座標の値をいじることにより見栄えを良くするため。

ルールについては以上ですが、以降の内容を読んでいくにあたって1点だけ頭に入れておいてほしいことがあります。それは、私はJavascriptについての知識は全く持っていないということです。それどころかプログラミングの経験は約5年前にC言語という言語に触れただけであり、そのC言語の記憶すらとても怪しいという状況です。これらを踏まえたうえで以降を読むことで、プログラミングの知識がなくても簡単なゲームが作れてしまう時代に入りつつあるということを実感していただけたらと思います。

9.4 ゲーム制作

ここからゲーム制作に手を付けていきます。制作に失敗したゲームについても書いてあるため、制作に成功したゲームだけを読みたい方は **3 スネークゲーム**まで読み飛ばしてください。

ドクターマリオ

始めに目を付けたゲームは **ドクターマリオ** です。このゲームについては以下の通りです。

このソフトは、1990年に発売されたファミリーコンピュータ用のアクションパズルゲームです。ドクターとなったマリオの研究室のビンに発生した3種類のウィルスを、次々と投げ入れるカプセルを使って退治していきます。ウィルスは同じ色のカプセルを縦か横に4個以上並べると消えます。カプセルを回転させたり移動させたりしながら積み上げていき、ビンの中のウィルスをすべて退治するとステージクリアです。

任天堂ホームページから引用

ウィルスとカプセルをそれぞれ3色の記号で表すことができるため、このゲームを作れると思い、試してみました。できたコードとその実行結果がこちらです。

[2] あるシステムやソフトウェアを実行するために必要な設定を行うこと

```
<!DOCTYPE html>
<html>
<head>
  <title>Doctor Mario</title>
  <style>
    #gameboard {
      position: relative;
      width: 240px;
      height: 480px;
      border: 2px solid black;
    }
    .virus {
      position: absolute;
      width: 24px;
      height: 24px;
      background-color: #00FF00;
      border: 2px solid black;
      border-radius: 50%;
      top: 0;
      left: 0;
      animation: virusMovement 0.8s ease-in-out infinite alternate;
    }
    .pill {
      position: absolute;
      width: 24px;
      height: 48px;
      background-color: #FFFF00;
      border: 2px solid black;
      border-radius: 8px;
    }
    .score {
      position: absolute;
      top: 490px;
      left: 0;
    }
    .level {
      position: absolute;
      top: 490px;
      right: 0;
    }
    @keyframes virusMovement {
      from {
        transform: translate(0, 0);
      }
      to {
        transform: translate(0, 24px);
      }
    }
  </style>
</head>
<body>
  <div id="gameboard"></div>
  <div class="score" id="score"></div>
  <div class="level" id="level"></div>
  <script>
    (function () {
```

```
const VIRUS_SIZE = 24;
const PILL_SIZE = 24;
const ROW_SIZE = 24;
const CREATE_VIRUS_INTERVAL = 1000;
const FALLING_INTERVAL = 500;

let gameboard = document.getElementById("gameboard");
let scoreElement = document.getElementById("score");
let levelElement = document.getElementById("level");

let score = 0;
let level = 1;
let virusCount = 0;
let pillFalling = false;
let pillInterval;

let createVirusInterval;

function createVirus() {
  let row = document.createElement("div");
  row.classList.add("row");
  gameboard.appendChild(row);

  let virusPosition = Math.floor(Math.random() * 10);
  for (let i = 0; i < 10; i++) {
    let virus = document.createElement("div");
    if (i === virusPosition) {
      virus.classList.add("virus");
    }
    row.appendChild(virus);
  }

  virusCount++;

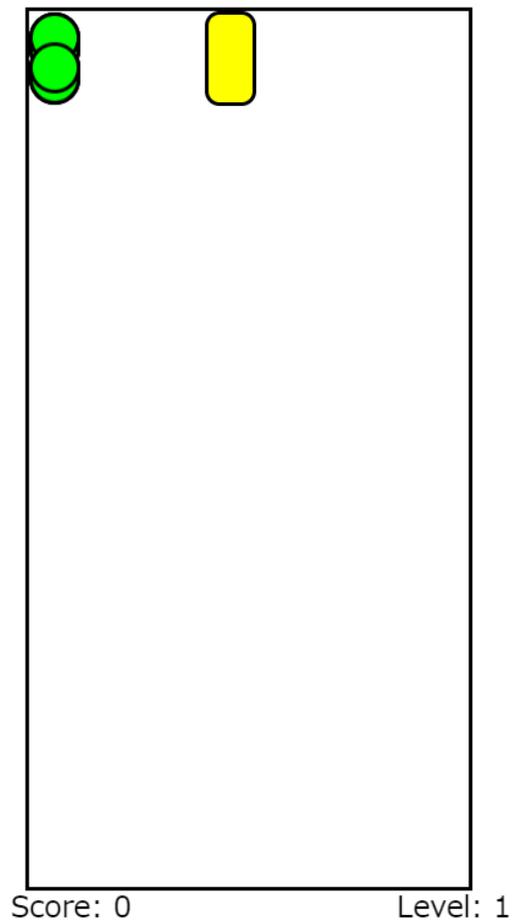
  if (virusCount === 20) {
    clearInterval(createVirusInterval);
    pillFalling = true;
    pillInterval = setInterval(movePillDown, FALLING_INTERVAL);
  }
}

function createPill() {
  let pill = document.createElement("div");
  pill.classList.add("pill");
  pill.style.top = "0px";
  pill.style.left = "96px";
  gameboard.appendChild(pill);
  pillFalling = true;
}

function movePill(direction) {
  let pill = document.querySelector(".pill");
  let left = parseInt(pill.style.left);
  if (left + direction * PILL_SIZE < 0 || left + direction * PILL_SIZE
> 216) {
    return;
  }
}
```



```
    }  
    switch (event.keyCode) {  
      case 37:  
        movePill(-1);  
        break;  
      case 39:  
        movePill(1);  
        break;  
    }  
  });  
  
  startGame();  
})();  
</script>  
</body>  
</html>
```



▲ 図 9.1 : ドクターマリオ失敗作

左上で絶えず生成される緑色のウイルス？達がぷかぷか浮いて、横に出現するカプセルが落ちるのをただひたすらに見守るゲームが完成しました。ちなみにカプセルが一番下まで到達すると Score が -100 されます。個人的にはこれほこれの意味不明すぎて面白いのでアリ。

このあとも試行錯誤を重ねてみたのですが、残念ながらこれが一番まともなコードでした。作ることができなかった原因としては初期状態で発生するウイルスのコードを組むことが難しいからだと考えました。

ぷよぷよ

ドクターマリオは諦め、次に**ぷよぷよ**を作ってみようと思いました。このゲームについては以下の通りです。

同じ色の"ぷよ"を4つつなげて消していくシンプルなルールで、行く手をはばむゆかいなキャラクターたちと対戦していくアクションパズルゲームです。

セガ公式サイトから引用

ウイルスという障壁がなくなったのでどうにかなるだろうと思い、試してみました。できたコードとその実行結果がこちらです。

```
<!DOCTYPE html>
<html>
<head>
  <title>Puyo Puyo</title>
  <style>
    #field {
      width: 220px;
      height: 440px;
      background-color: #ccc;
      border: 2px solid #666;
    }
    .block {
      width: 20px;
      height: 20px;
      border: 1px solid #666;
      box-sizing: border-box;
      position: absolute;
    }
    .red {
      background-color: red;
    }
    .green {
      background-color: green;
    }
    .blue {
      background-color: blue;
    }
    .yellow {
```

```
        background-color: yellow;
    }
</style>
</head>
<body>
  <div id="field"></div>
  <script>
    const colors = ["red", "green", "blue", "yellow"];
    let currentPuyo = null;
    const field = [];
    for (let i = 0; i < 22; i++) {
      const row = [];
      for (let j = 0; j < 12; j++) {
        row.push(null);
      }
      field.push(row);
    }
    const fieldElement = document.getElementById("field");

    function render() {
      for (let i = 0; i < 22; i++) {
        for (let j = 0; j < 12; j++) {
          const block = field[i][j];
          if (block !== null) {
            block.style.top = i * 20 + "px";
            block.style.left = j * 20 + "px";
          }
        }
      }
    }

    function createPuyo() {
      const puyo1 = document.createElement("div");
      const puyo2 = document.createElement("div");
      const color1 = colors[Math.floor(Math.random() * colors.length)];
      const color2 = colors[Math.floor(Math.random() * colors.length)];
      puyo1.classList.add("block", color1);
      puyo2.classList.add("block", color2);
      puyo2.style.top = "20px";
      currentPuyo = [puyo1, puyo2];
      fieldElement.appendChild(puyo1);
      fieldElement.appendChild(puyo2);
      puyo1.style.left = "80px";
      puyo2.style.left = "100px";
    }

    function canMoveDown() {
      const [puyo1, puyo2] = currentPuyo;
      const [x1, y1] = getBlockCoords(puyo1);
      const [x2, y2] = getBlockCoords(puyo2);
      if (x2 >= 21 || field[x2 + 1][y2] !== null) {
        return false;
      }
      if (x1 >= 21 || field[x1 + 1][y1] !== null) {
        return false;
      }
    }
  </script>
</body>
</html>
```

```
    return true;
  }

  function moveDown() {
    if (!canMoveDown()) {
      fixCurrentPuyo();
      createPuyo();
      return;
    }
    const [puyo1, puyo2] = currentPuyo;
    const [x1, y1] = getBlockCoords(puyo1);
    const [x2, y2] = getBlockCoords(puyo2);
    field[x1][y1] = null;
    field[x2][y2] = null;
    field[x1 + 1][y1] = puyo1;
    field[x2 + 1][y2] = puyo2;
    render();
    setTimeout(moveDown, 500);
  }

  function getBlockCoords(block) {
    const top = parseInt(block.style.top);
    const left = parseInt(block.style.left);
    const x = Math.round(top / 20);
    const y = Math.round(left / 20);
    return [x, y];
  }

  function fixCurrentPuyo() {
    const [puyo1, puyo2] = currentPuyo;
    const [x1, y1] = getBlockCoords(puyo1);
    const [x2, y2] = getBlockCoords(puyo2);
    field[x1][y1] = puyo1;
    field[x2][y2] = puyo2;
    currentPuyo = null;
  }

  createPuyo();
  setTimeout(moveDown, 500);
</script>
</body>
</html>
```



▲ 図 9.2：ぷよぷよ失敗作

パッと見それっぽくはなりましたが、ランダムな色のぷよが生成されるだけのプログラムになってしまいました。何度実行してもぷよの色が変わる以外の変化はなく、このあと試行錯誤してもぷよが動くことはなかったです。

残念ながらこれも失敗に終わってしまいました。

スネークゲーム

落ち物パズルは難しいと思い、作るゲームのジャンルを大幅に変えてみました。Google で**ヘビゲーム**や**スネークゲーム**などと検索すると遊ぶことができるゲームを目標に作っていきます。

このゲームではプレイヤーがヘビを操作し、フィールドに存在するフルーツをたくさん食べることが目的のゲームです。フィールドの壁やヘビ自身の体にヘビの頭が衝突するとゲームオーバーとなります。また、フルーツを食べることでヘビの体が長くなり、だんだん難易度が上がっていくものになっています。

これまでの2つのゲームでは「Javascriptで動く〇〇のコードを出力して」から始まり、何度も出力をさせた結果のコードを載せていたのですが、今回はなんと最初の出力でこのコードとその実行結果が出力されました。

```
<!DOCTYPE html>
<html>
<head>
  <title>Snake Game</title>
```

```
<style>
  canvas {
    border: 1px solid black;
  }
</style>
</head>
<body>
  <canvas id="gameCanvas" width="400" height="400"></canvas>
  <script>
    // ゲーム画面の設定
    var canvas = document.getElementById("gameCanvas");
    var ctx = canvas.getContext("2d");
    var blockSize = 10;
    var width = canvas.width / blockSize;
    var height = canvas.height / blockSize;

    // スネークの初期位置
    var snake = [];
    snake[0] = {x: 5, y: 5};

    // フルーツの初期位置
    var fruit = {x: Math.floor(Math.random() * width), y: Math.floor(Math.random() * height)};

    // スコアとゲームオーバーの設定
    var score = 0;
    var gameOver = false;

    // ゲームループ
    function gameLoop() {
      if (gameOver) {
        return;
      }

      setTimeout(function() {
        requestAnimationFrame(gameLoop);
        draw();
      }, 100);
    }

    // 描画
    function draw() {
      // スネークの移動
      var head = {x: snake[0].x, y: snake[0].y};
      if (direction === "right") {
        head.x++;
      } else if (direction === "left") {
        head.x--;
      } else if (direction === "up") {
        head.y--;
      } else if (direction === "down") {
        head.y++;
      }
      snake.unshift(head);

      // フルーツを取得した場合
```

```
    if (head.x === fruit.x && head.y === fruit.y) {
        score++;
        fruit = {x: Math.floor(Math.random() * width), y: Math.floor(Math.random() * height)};
    } else {
        snake.pop();
    }
}

// ゲームオーバーの判定
if (head.x < 0 || head.x >= width || head.y < 0 || head.y >= height) {
    gameOver = true;
}
for (var i = 1; i < snake.length; i++) {
    if (snake[i].x === head.x && snake[i].y === head.y) {
        gameOver = true;
    }
}

// 背景を描画
ctx.fillStyle = "#ccc";
ctx.fillRect(0, 0, canvas.width, canvas.height);

// スネークを描画
ctx.fillStyle = "green";
for (var i = 0; i < snake.length; i++) {
    ctx.fillRect(snake[i].x * blockSize, snake[i].y * blockSize, blockSize, blockSize);
}

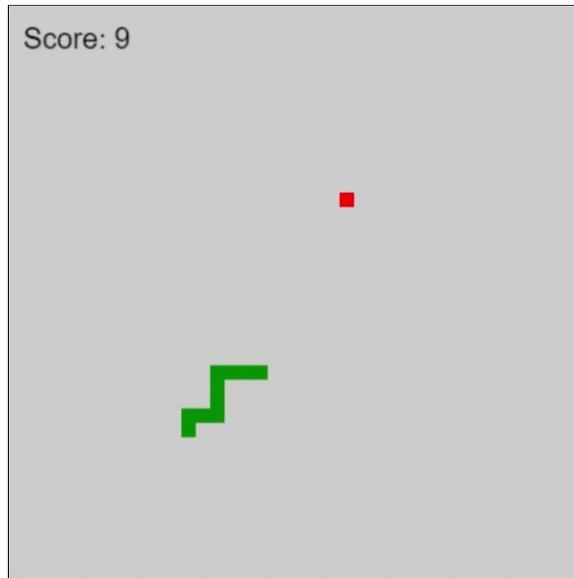
// フルーツを描画
ctx.fillStyle = "red";
ctx.fillRect(fruit.x * blockSize, fruit.y * blockSize, blockSize, blockSize);

// スコアを表示
ctx.fillStyle = "black";
ctx.font = "20px Arial";
ctx.fillText("Score: " + score, 10, 30);
}

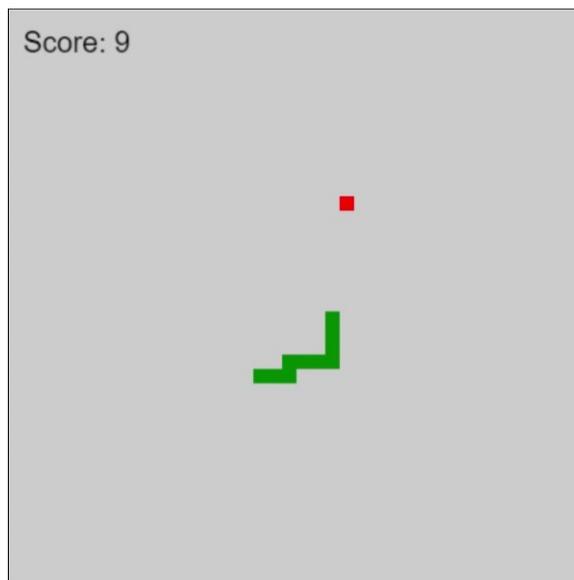
// キーボード入力
var direction = "right";
document.addEventListener("keydown", function(event) {
    if (event.keyCode === 37 && direction !== "right") {
        direction = "left";
    } else if (event.keyCode === 38 && direction !== "down") {
        direction = "up";
    } else if (event.keyCode === 39 && direction !== "left") {
        direction = "right";
    } else if (event.keyCode === 40 && direction !== "up") {
        direction = "down";
    }
});

// ゲームを開始
```

```
gameLoop();  
</script>  
</body>  
</html>
```



▲ 図 9.3 : スネークゲーム1



▲ 図 9.4 : スネークゲーム2

ちゃんと動く!!!すげえ!!!

画像しか載せることができないというのがとても歯がゆいですがこのゲームの最低限のルールをおさえたコードが一瞬で出てきました。そしてスネークゲームを作れというだけの命令だったにもかかわらず、出力されたコードにこれらの機能が実装されていたのには驚愕でした。

- ヘビがフルーツを食べるとScoreが増える点
- ヘビの体や壁に当たるとゲームが進行不能=ゲームオーバーになる点
- コメントでそのコードが何を表しているか説明してくれている点

この後も改良を重ね、この記事の提出時点では、ゲームオーバーになったときにリトライするためのボタンの実装を完了させました。文化祭当日ギリギリまで改良を重ねていきますので、ぜひ遊んでいただけると幸いです。

9.5 おわりに

本章ではAIでゲームを作ってもらうことをテーマに、制作過程を紹介してきました。しかし、AIでゲームが作れるからといってプログラミングの勉強をしなくてもよいかというと、実はそうではありません。AIが生成するコードにもミスが含まれているため、それを修正できるほどの理解はしておくべきだと思います。

また、AIで生成したゲームを公開するときにも注意が必要です。あくまでもゲームを作ったのは自分ではなくAIです。ソースコードやゲームを公開する際にはそれを念頭に置いて、節度を守った利用を心がけるようにしましょう。

9.6 参考文献

- Introducing ChatGPT (<https://openai.com/blog/chatgpt>)
- jsfiddle (<https://jsfiddle.net/>)
- GPT-4を使ってぶよぶよ作ってみたときの感想 (<https://zenn.dev/corocn/articles/b4c473ed779935>)
- GPT-4だけで作ったpackman… (<https://twitter.com/shotyas/status/1636261249297297409?s=20>)
- GPT-4に「ブラウザで動くテトリス作って」って言ったら一瞬でできた🤖… (https://twitter.com/djrjo_vr/status/1636230792799215616?s=20)
- ドクターマリオ | Wii U | 任天堂 (<https://www.nintendo.co.jp/titles/20010000003946>)
- SEGA AGES ぶよぶよ (<https://archives.sega.jp/segaages/puyo/>)

第10章

「文字の自動生成の仕組み」について

Kikuchi Hiromu

10.1 はじめに

今回、文字の自動生成について書き綴っていきます。まず、文字の自動生成とは何でしょうか。これは、簡潔に言うとAI（人工知能）がフォントを作ることと言えるでしょう。

フォントとは従来、人の手によって一つ一つデザインされていました。そのため、日本語のフォントを作るのに膨大な時間を要していました。英語圏のフォントと比べるとわかりやすいと思いますが、日本語には漢字、カタカナ、ひらがなと文字の種類がとても多いです。フォントを作成するにあたって、人手と時間がかかるとなると、フォントの種類は決して豊富といえるほどなく、利用するのにもお金がかかります。

このような背景があり、フォント作成をAIに任せることができるのであれば任せたいという考えから、文字の自動生成が研究され始めました。そして、私はそこに興味を持ち、仕組みを調べました。あわよくば、私自身でプログラムを書いてみたいと思っていましたが、仕組みを理解することは容易ではありませんでした。そのため、誰にでもわかってもらえるような説明をしていきたいと思います。

AIというと「万能」という印象があるかもしれませんが、実際には何もできません。少し言い過ぎたかもしれませんが、それでも、最初は何もできません。何度も同じことを練習することで、ある特定のことができるようになっていくのです。つまり、フォントを書くためには、文字をどう認識するのか、文字をどう装飾してフォントにするのか、というようにいくつかの段階を踏んで、最終的な目的のAIが出来上がります。このことを頭の片隅に入れながら読んでいただくとわかりやすくなると思います。

10.2 文字をどう識別するのか

「文字」といってもたくさん種類があります。今は何もわからないため、まずは「数字」に絞って教えることにしましょう。数字であれば、0から9までの10文字しかありません。

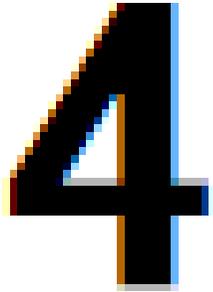
これから、私はパソコンを通してAIに数字の識別を学習させていきます。

まず初めに、ある1つの数字をAIに渡してみしましょう。それは、手書きで書かれた「4」であるとしましょう。すると、AIは認識、思考、識別の過程を踏んで、「9」と答えました。

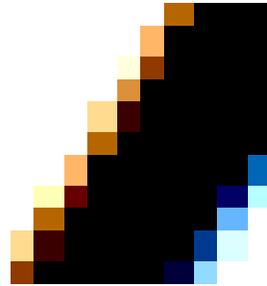
誰でも最初は間違えるものですよ。これを正しく答えさせるようにするにはどうすればいいでしょうか。一つ一つ順を追って教えるため、「認識」、「思考」、「識別」の3つに分けて教えましょう。

「認識」

認識とは言いつつ、AIにはコンピュータを通して教えているので、データの形で判断してもらうため、数字を渡すしかありません。また、AIは人間のように目で見えることはできません。ではどうするのかというと、受け取った数字のデータを、縦横同じ数のピクセルで分割し、それぞれのピクセルに記録されている、白から黒のグレースケールでデータを数値化します。いきなり言われてもよくわからないと思うので、図を用いて補足します。



▲ 図 10.1 : 4



▲ 図 10.2 : 4拡大

文字を拡大してみると、グレースケールのドットで文字が構成されていることがわかります。つまり、文字を細かい正方形で切ると1ピクセルになり、そのピクセルには、白を0、黒を1とすると、0から1の濃さで表すことのできる色がついています。

このことから、AIが数字を認識するとは、ピクセルの順番とその色の値（0から1）記録することです。数字も文字も同じ方法で認識できます。

「思考」

AIが文字を認識すると、これは何の数字であるか考えます。もし、人もAIと同じように数字を細かく分けた状態であるピクセルで与えられたら、なんの数がかかれているのか気づけないでしょう。数字を認識するには、ある程度ピクセルよりも少し大きなまとまりで数字を認識しているはずで、つまり、認識したデータを組み合わせて小さな塊をいくつか作ることで、思考します。具体的にどう考えているのかを目に見える形で表すことは難しいです。そうはいつでもAIが考えることはすべて数式になっています。

「識別」

識別とは、AIが思考の末たどり着いた結果といえるでしょう。頭の中で考えたことをほかの人に伝えるように考えをまとめ、わかりやすい形に直して結果を表します。識別では考えた答えを整える程度のことしかしません。

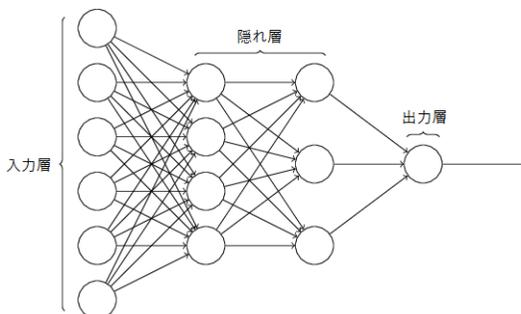
このようにAIが学習するには、認識、思考、識別を何度も繰り返すことで、コツをつかんでいきます。正確には、正しく答えられるように主に思考の方法を少しずつ変化させ、正しく答えられるようにします。そして、AIは漢字も同様に学習することで文字を識別することができますようになります。

10.3 学習プロセスの数式化

「認識」、「思考」、「識別」は、あくまでも人間の頭の中で起きていることです。コンピュータは基本物事を2進数で把握し、常に計算しています。そのため、先ほどのAIの学習プロセスである「認識」、「思考」、「識別」をニューラルネットワークと呼ばれる数式で表さなければいけません。少し説明を付け加えると、ニューラルネットワークとは、人間の思考プロセスを数式化したもので、AI学習の基本的な学習方法の一つです。

ニューラルネットワーク

ニューラルネットワークは、「入力層、隠れ層、出力層」の三つからできています。それぞれの層が、「認識、思考、識別」に対応するようになっています。つまり、入力層では文字を識別することであり、隠れ層では思考し、出力層で文字を識別した結果を表します。このことを数式に表したものが、ニューラルネットワークです。後々重要な基本的な考えになるので、一つずつ説明していきます。入力層が認識、隠れ層が思考、出力層が識別に対応するようになっており、説明も対応させて書くので、見比べながら読むとより理解が深まるでしょう。



▲ 図 10.3 : ニューラルネットワーク

入力層

ここでは、認識で説明したことを行います。AIが手書き文字を受け取ります。この画像を数式的に表すことが目標です。そのため、画像を28×28の大きさに区切り、一枚の画像を784ピクセルに分けます。このピクセル一つ一つには、0から1の（白から黒に数字を対応させている）濃さで表すことができます。そのため、一枚の手書き文字は、0から1の数値を784個並べたデータとして扱えます。

この入力層には784個のデータが入るように784個の数値が入るニュートロン（丸い容器のようなもの）があります。そのニュートロンに一つ一つ数値を入れることで、手書き文字の認識ができます。

隠れ層

入力層で認識したデータをあらゆる方法で思考します。入力層は一つでしたが、隠れ層は何層にもなる場合があります。3層以上になるとニューラルネットワークがからディープラーニングと言われるように、層が多いほどより複雑な処理、分析が可能になります。

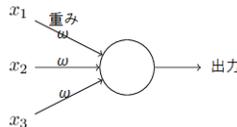
出力層

出力層で最終結果を表します。隠れ層で分析したデータをもとに、AIが導きだした答えをわかりやすく表現します。

精度向上

これまで一つの手書き文字を受け取るとAIがどのように答えを導き出すのかを説明してきました。これからは、どのようにしたらより正しく答えられるようになるのかを説明します。つまり、AIの正確さを上げるためにすることがいくつかあるのです。

まず初めに、「重み」をデータの受け渡しの時につけます。具体的に言うと、たいてい文字の場合ノートに書かれたマス目の真ん中に文字をかきます。そのため、四隅には空白がでがちです。つまり、マス目の真ん中にあるピクセルと四隅にあるピクセルを比べると前者のほうがより重要で、正解にたどり着くための鍵となりやすくなっています。この状態が重いと表現しましょう。つまり、「重み」をつけるということは、その重みをつけたピクセルを強調することができます。人間が無意識に重要な情報だけを考慮するように、AIにも重要な情報を「重み」をつけることによって、より考えやすくします。



▲ 図 10.4：重み

この重みをデータの受け渡しの時につけるといのは、次の層にいくときそれぞれのニューロンが持っている数値に重みをかけるということです。ニューラルネットワークの図でニューロン同士が線でつながっていますが、それが重みを表しています。重みを別の言葉で表現するならば、重要度、伝わる速さとなるでしょう。この表現を理解できるのであれば、重みを十分理解したといっても過言ではないでしょう。

この重みは自由につけることができますが、最終的な結果を大きく左右します。これは重みを持つとても重要な性質です。なぜならば、重みを変数として扱うことで、AIの精度を上げられるからです。しかし、ニューラルネットワークの図を見るとわかるように、線が重みであるので、たくさんの異なる重みがあります。これを一つ一つ変えて精度を上げるには、AIといっても膨大な時間がかかります。ではどうするのでしょうか。その前に説明すべきことがあるので、また後で説明します。

損失関数

急に損失関数と言われましてもという感じですよ。実はコスト関数とも言われることもあります。といっても説明にはなりません。損失関数とは、予想値と実際の値の差を測定する関数と定義しています。つまり、AIが考えた答えが正しい場合、その差は0になります。この差を0に近づけることが、AIの精度を上げることになります。損失関数はいくつか種類があるため、状況に応じて使い分けません。

ここで重みが登場します。この損失関数の値を0に近づけることを目的に、重みの値を変化させます。そしてこの時、勾配降下法というアルゴリズムを用いて、損失関数が0に近づく重みを決めます。

これでAIが文字を識別できるようにする学習方法とその精度向上の仕方を説明しました。これでAIはようやく文字を認識できるようになりました。次は、文字を書くことを教えます。ただ文字を書くのではなく、装飾的にフォントを作るイメージです。

10.4 文字をどう装飾してフォントにするのか

まず初めに、目的を確認しましょう。最終的な目的は、AIがフォントを自動で生成できるようにすることでした。より砕けた言い方をすると、とある書体の文字をいくつか受け取り、それをもとに似た書体の文字（フォント）を作成することです。ここには、受け取らなかった文字も、その受け取った文字の書体にしたフォントを作成することを含んでいます。今まですべてをデザインしてフォントを作成していたことに代わって、いくつかの文字のフォントのデザイン案でほぼすべてのフォント完成させられることにするということです。

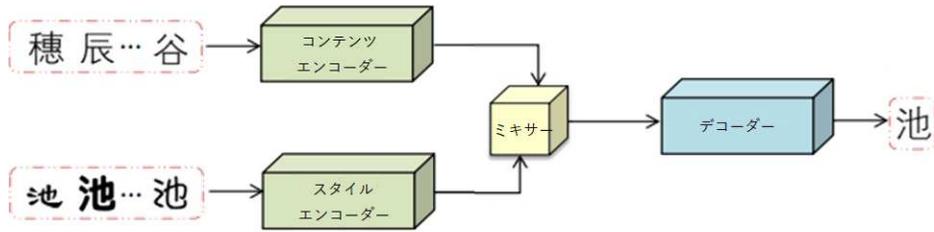
ではどのような学習をすればよいのでしょうか。今回は、その学習を行ってAIを学習させた「zi2zi」というプロジェクトで行われていることを説明していきます。

「zi2zi」というプロジェクトでは、実際に文字の書体変換をスムーズに行うことができます。つまり、自動でフォントの作成をすることができているのです。ただ、文字の書体を変換するだけでなく、文字の太さもスムーズに変換できています。これは、条件付き敵対生成ネットワーク (*Conditional Generative Adversaria Network*) というネットワーク構造になっているからです。今後CGANと呼んでいきます。

CGANを知らないからと言って、頭を抱えることはありません。文字の識別を学習させるときに、ニューラルネットワークを用いました。それと似たAIを学習させるネットワーク構造であるのがCGANです。CGANとは、与えられた条件のもとに、その条件に応じた画像を生成することができます。条件とは、手書き文字だと思ってください。そして、生成される画像がフォントです。CGANによってフォントを作成することができるようになるので、文字をCGANについて説明していきます。

CGAN

CGANは軽く説明したように、与えられた条件（手書き文字、もしくはフォントのデザイン案）をもとに条件に適した新しい画像（フォント）を生成することができます。CGANには、Generator（生成器）とDiscriminator（識別機）の2つの構造からなります。



▲ 図 10.5 : CGAN

Generator (ジェネレーター)

ジェネレーターは、ニューラルネットワークとほぼ同じ機能を持ちます。つまり、ジェネレーターでは、手書き文字を受け取り、それに応じた文字を生成することができます。CGANにはディスクリミネーターもありますが、実はジェネレーターだけで文字を生成できてしまうのです。

それはさておき、ジェネレーターについてももう少し詳しく見ていきましょう。ジェネレーターをさらに分けると、エンコーダー(Encoder)とデコーダー(Decoder)に分けられます。

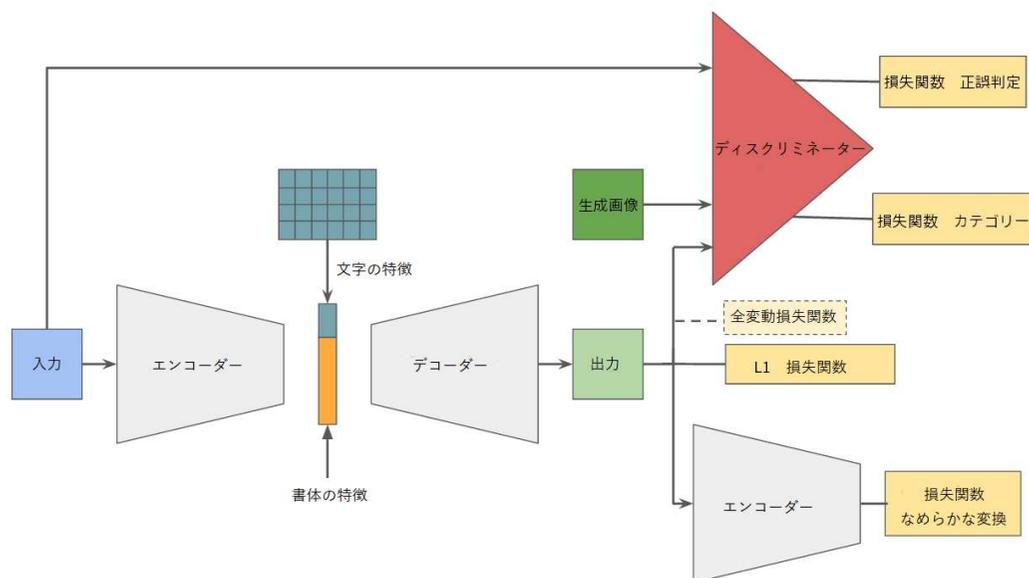
エンコーダーは、手書き文字を受け取り、そこから特徴を抽出することをします。文字を認識するのに必要な、ニューラルネットワークでいう入力層の役割を担っています。それに加えて、手書き文字から特徴を抽出しますが、これは何でしょうか。フォントを生成するためには、文字を書くことと、文字をどのように装飾するかを学ばなければいけません。フォントを作成する際に、文字の形を知り、どう装飾するか(書体の違い)を知らないとフォントを作れないからです。そのため、特徴を抽出ということは手書き文字から、「文字の形」とその文字の「書体」(どう装飾的に書かれているか)のコツをつかむというような感覚ですが、別々に特徴を抽出します。同じ字でも書体が違えば、フォントとしては異なります。そのため、入力層と同等の役割を担うエンコーダーは、字形の特徴を抽出する「コンテンツエンコーダー」と書体の特徴をつかむ「スタイルエンコーダー」の2つがあります。それぞれのエンコーダーは、手書き文字を特徴ベクトルに変換して、次にデータを受け渡します。特徴ベクトルとは、手書き文字を数値に変換した膨大なデータと思ってください。ちなみに「ベクトル」というと、向きと大きさを持った値というイメージが強いですが、数学的な「ベクトル」は、行列と同義であるでしょう。つまり、データの羅列で多次元にわたる数値を持っています。

デコーダーは、文字画像を生成します。ニューラルネットワークでいうところの出力層です。デコーダーには、エンコーダーで抽出した2つの特徴を混ぜてから、画像を生成しています。その2つの特徴を混ぜるところを、ミキサーと呼びます。エンコーダー、ミキサー、デコーダーの順に並んでいます。この並びとそれぞれの役割を考えると、入力層、隠れ層、出力層を連想してしまいますね。

Discriminator (ディスクリミネーター)

ディスクリミネーターは、ジェネレーターで生成した画像が本物に似ているかどうかを判断する役割を担っています。ジェネレーターで生成されたフォント画像は、何もしないとフォントではなくただの画像程度で、文字として日用使いすることは難しいでしょう。ディスクリミネーターはジェネレ

ーターで生成した画像が、どれくらい受け取ったデータ（本物）と似ているか比べます。そして、違うところを指摘します。この過程は、ニューラルネットワークにあった損失関数と似ています。しかし、実際にはディスクリミネーターは違いを指摘しません。生成された画像と受け取った画像を見せられ、どちらが本物かを見分けるのです。このとき、エンコーダーは自分で生成した画像をデコーダーに選んでもらえるように（デコーダーをだますと表現することもできますが…）努めています。このことから、CGANはエンコーダーとディスクリミネーターが切磋琢磨しあうことで、生成されるフォント画像の精度が向上していくことを可能としてくれます。



▲ 図 10.6 : ディスクリミネーター

10.5 文字の自動生成の結果

今まで説明したことがプログラムに組み込まれています。コンピュータでさえも膨大な時間がかかるため、できるだけ効率化した計算等を行っています。しかし、その効率化した計算式が難しいため、説明は省かせていただきます。このことに興味がない人は、AIが自動生成した文字は実際のところどれだけ正確なのかをご覧になって下さい。この画像は「zi2zi」プロジェクトの結果報告の一部です。気になる方に限らず、本研究をぜひともご覧になってください。

久	久	肠	肠	柞	柞	裘	裘	肋	肋
亮	亮	樞	樞	又	又	哲	哲	苜	苜
阙	阙	洧	洧	材	材	簿	簿	装	装
序	序	魁	魁	坤	坤	瞬	瞬	晋	晋
骺	骺	熟	熟	又	又	吹	吹	澜	澜

▲ 図 10.7：生成画像

左がもともとの字（グラントゥールズ）、右が推測された字（生成画像）です。

第	第	第	第	第	第	第	第	第
壺	壺	壺	壺	壺	壺	壺	壺	壺
話	話	話	話	話	話	話	話	話
使	使	使	使	使	使	使	使	使
徒	徒	徒	徒	徒	徒	徒	徒	徒
襲	襲	襲	襲	襲	襲	襲	襲	襲

▲ 図 10.8：フォントファミリー

10.6 あとがき

私が数年にかけて、理解しようと取り組んできたことをまとめました。しかし、数式やプログラムのようなより専門的なことはまだ説明できる程十分に理解できていないため、重要な部分以外は説明で触れませんでした。また、その難解さを上回る興味がありました。ですから、理解することが難しかったり、逆にもっと知りたいと思ったりと様々な感想を持たれたと思います。そのような場合にも以下のサイトをご覧になるとより考えが深まると思います。ぜひ一度は訪れてみてください。

手書き文字認識 - https://nnadl-ja.github.io/nnadl_site_ja/chap1.html

「rewrite」のプロジェクト - <https://arxiv.org/pdf/1711.06454.pdf>

「zi2zi」プロジェクト - <https://github.com/kaonashi-tyc/zi2zi>

ニューラルネットワークとは何ですか - <https://youtu.be/aircAruvnKk>

執筆者紹介/コメント

Nagaso.cpp

古河中等6期生。パソコン部2022-2023年度部長。文化祭実行委員IT管理部部署長としてオープニングムービーを始めとする動画関連や楽曲制作等の創作分野を担当。趣味はDTMとソフトウェア開発。

コメント: お金が無いので投げ銭してください。

eno1220

古河中等第6期生。パソコン部2022-23年度副部長。文化祭実行委員会IT管理部副部署長を務めており、公式Webサイトや電子チケット、内部で使用しているシステムの開発や運営を行っている。趣味はWeb開発や競技プログラミング、OS自作など。情報科学の達人4期生。

コメント: 文化祭準備がとても忙しいです。誰かたすけて～

張替健太/hrkn63_hnm

どこにでもいる古河中等6期生。今日でちょうど生後216ヶ月を迎える大きな赤ちゃん。校内SNS「Tone」を作っていた生徒会長。情報科学の達人の動画が見終わらない。

コメント: 働きたくないので「自分で会社作って社長になることにした」と妹に宣言した。「すぐ潰れそう」と言われた。

Yowashita

古河中等第6期生。パソコン部古参兵の一人。ただ昔からいるというだけで特に何かを成したわけではない。html、css、ruby、pythonなど様々なものに手を出してきたが、そのどれも中途半端に終わった。

コメント: 光陰矢の如し

tarokko

古河中等6期生。中等近くのデイリーヤマザキの売り上げに微力ながら貢献している。

コメント: #アラブ首長国連邦で人気のSB69をよろしくお願ひします

Kikuchi Hiromu

古河中等第6期生。気まぐれパソコン部員。Pythonを用いたAIの勉強、HTML・CSSなどでWEBサイトの作り方の勉強をしている。

コメント: 自分の知らないことにも興味を持って生きてみよう

a01sa01to

古河中等第3期生。現在は埼玉大学工学部情報工学科3年。埼玉大学プログラミングサークルMaximum 代表。応用情報技術者。今回のけやき祭には実行委員会IT管理部のアドバイザーという立ち位置で携わっている。競技プログラミングをはじめ、Web開発や低レイヤー分野など、幅広くプログラミングを行っている。JOIやICPCといった競技プログラミングコンテストのほか、Web Speed Hackathon や UTE-1 などのハッカソンにも参加している。

コメント: 2023年8月26日に、プログラミングコンテスト「Maximum-Cup 2023」を行う予定です！ぜひ参加してね！！

香風智乃

古河中等4期生。部誌に非技術的な話を挟み込む常習犯。最近は Arch Linux 盆栽の過程でタイル型WMを試したりしています。Hyprland は良いですよ。

コメント: ラビットハウスでは stera terminal を導入してクレジットカード決済, 電子マネー決済, コード決済に一気に対応しました。最近では同人誌即売会に Square を持ち込むサークルも増えてきたそうで, あらゆるシーンでキャッシュレス化が進展していることをひしひしと感じます。

Ryoga.exe

古河中等第5期生。現在は筑波大学情報学群情報科学類1年。応用情報技術者。今回のけやき祭には実行委員会IT管理部のアドバイザーという立ち位置で携わっている。Web開発や競技プログラミング、CTF、ゲーム開発、アプリケーション開発などを行っているよ。最近はdotfilesの沼に落ちている。情報科学の達人3期生。WORD編集部。jsys23。

コメント: 最近業務やらでAstroを触っているが非常に良すぎる…それと大学が楽しすぎてやばい、パソコンオタクはみんな来よう！

KSS PC BOOK 2023

2023年6月3日 初版発行

著者 KSS PC Club

印刷所 ウィルダイレクト

© 2023 KSS PC Club

KSS PC BOOK 2023