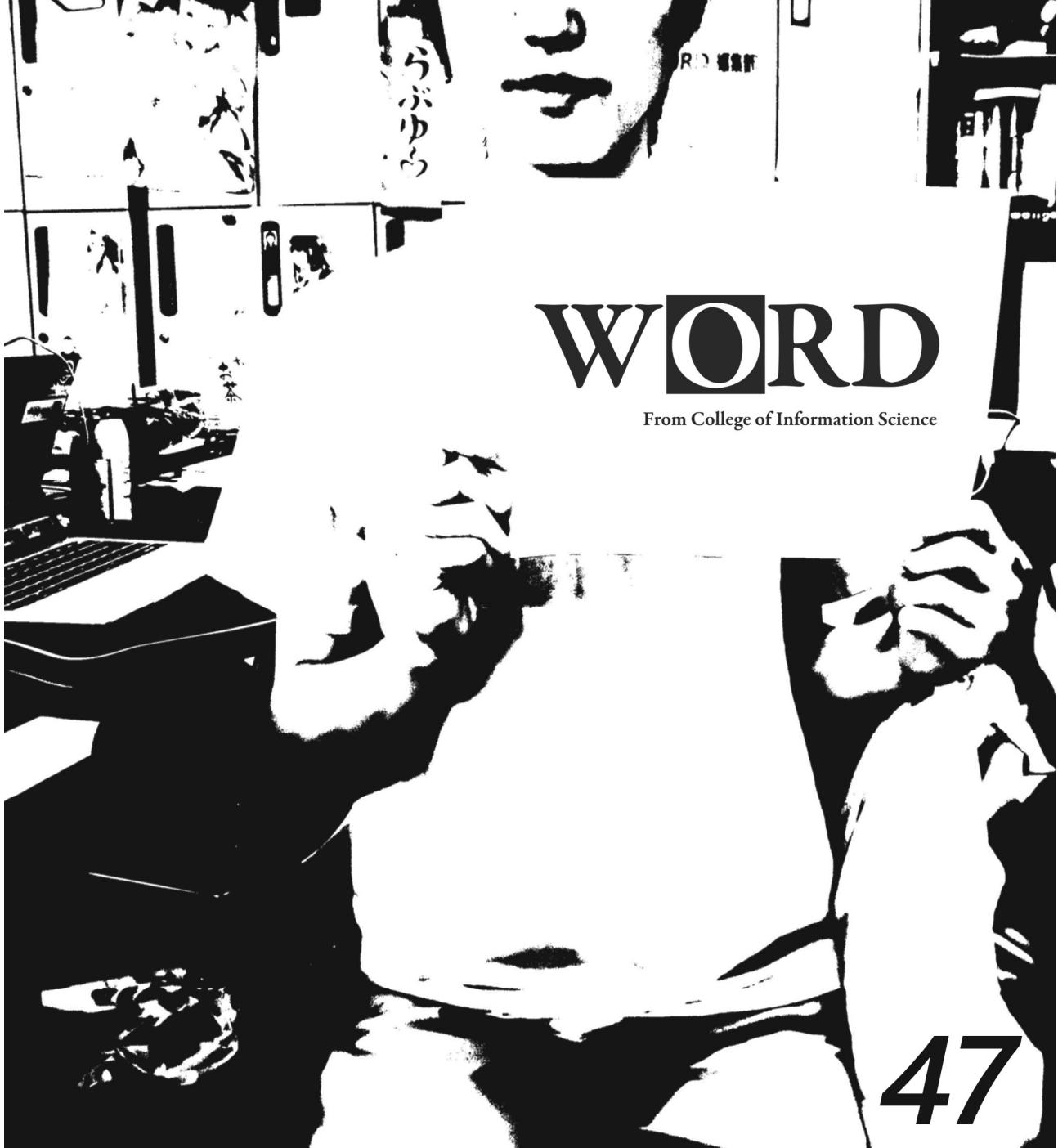


新元号



WORD

From College of Information Science

47

目次

情報系コンテストいろいろ	れっくす	003
Pairs に関する記録	ジェンガ 2003	009
UEFIで始めるOS開発	突撃隊	014
おけしょへうをしよう	はじょん	021
『ぶよぶよ e スポーツ』茨城県代表決勝戦レポート	らいりゅう号	025
編集後記		029

情報系コンテストいろいろ

文 編集部 れつくす

1 はじめに

こんにちは。いきなりですが皆さんは大学生活を楽しめていますか？ Twitter やサークル活動などで楽しい日々を送ってる人は多いでしょう。しかし、“情報系”の学生として生活を楽しんでいるかと聞かれると答えに困ると思います。

そこでこの記事では情報系の学生っぽい趣味の 1 つとして、コンピュータの技能について競うコンテストをいくつか紹介していきたいと思います。趣味で楽しく遊んでいたらコンピュータについて学べてしまうなんて一石二鳥ですね！

本記事では、競技プログラミング・Capture The Flag (CTF)・機械学習コンペティションの 3 種類のコンテストについて解説します。

2 競技プログラミング

競技プログラミングは本稿で紹介する中でも一番敷居の低いコンテストです。競技は単純で、問題文で指示された通りに動作するプログラムを書いて提出するだけ。大学のプログラミング課題と同じですね。細かい部分を文字で説明すると長くなるのでとりあえず問題を見てみましょう。

例題

あなたはアルファベット型のマグネットを使ってホワイトボード上に “WORD” という文字列を作ろうとしています。用意されたマグネットをすべて使ったとき、“WORD” は何個作れるでしょうか。マグネットは横一列に並べるものとします。

入力例

WIITOJYCFCOSDWCRWWOSMRDIVNFARDOB

出力例

3

競技プログラミングではこのような問題が数問出題され、参加者は問題を自動的に解くプログラムを提出します。すると、コンテストの主催者側でプログラムが正しく動作しているかをチェックし、正解ならば問題の難易度に応じた点数が与えられます。同点だった場合は解き終わるのが早かったほうが上位になるため、コードを書く速さも求められます。

ちなみに先ほど紹介した例題は、文字ごとで出でてくる回数をカウントして W, O, R, D のうち回数が最も少ないものを答えとして出力するプログラムを書けば解くことができます。

2.1 AtCoder

なんとなく競技プログラミングについて理解できましたか？ 次は実際にチャレンジしてみましょう。初めての方には AtCoder 社が開催している AtCoder Beginner Contest がおすすめです^{*1}。

The screenshot shows the AtCoder Beginner Contest 128 interface. The top bar displays the contest name and a Japanese flag. Below the bar, there are navigation links: トップ (Top), 質問 (Questions), 提出 (Submit), 提出一覧 (List of Submissions), 順位表 (Ranking Table), コードテスト (Code Test), and 解説 (Explanation). The main content area is titled "A - Apple Pie". It includes the following details:

- 実行時間制限: 2 sec / メモリ制限: 1024 MB
- 配点: 100 点
- 問題文**: 林檎が A 個、林檎の欠片が P 個あります。
林檎 1 個は、碎くことで林檎の欠片 3 個になります。また、林檎の欠片 2 個を鍋で煮込むことで、アップルパイが 1 個作れます。
今ある材料で作れるアップルパイの最大数を求めてください。
- 制約**: 入力は全て整数である。
 $0 \leq A, P \leq 100$

図 1 AtCoder Beginner Contest 128 の A 問題

月に 1~2 回の高頻度で開催されていて参加しやすい上、1 問は必ず超簡単な問題が出題されます。実力を表すレーティングの機能もあるので継続して参加することで成長も可視化することができます。また、「競技時間外でもとにかく問題を解きたいんだ！」という方は AtCoder の過去問や Aizu Online Judge^{*2} もおすすめです。さらに、学内には競技プログラミングの活動を行うサークル TPC もあって、年に 1 度行われている ICPC^{*3} にも参加しているようなので競技プログラミングが合っているようなら扉を叩いてみるとよいでしょう。

2.2 CodinGame

競技よりだと勝ち負けが発生してしまってモチベーションが続かない人もいると思います。CodinGame^{*4} というサイトでは、シューティングゲームの自機を操作して敵を自動で倒すといった、コードを書いてゲームをクリアする方式の問題が用意されています。他の人と競う競技タイプの問題もありますが、シンプルに問題を解くだけのパズル的な問題もたくさん用意されているので競わずマイペースでプログラミングしたい人におすすめです。もちろん慣れてきたらそのまま競技の方にも参加できるので自信がついたらチャレンジしてみま

^{*1} <https://atcoder.jp/>

^{*2} <http://judge.u-aizu.ac.jp/>

^{*3} 国際大学対抗プログラミングコンテスト <https://icpc.iisf.or.jp/>

^{*4} <https://www.codingame.com/>

しょう。サイトや問題の説明文などがすべて英語なので少し大変かもしれません、プログラミングをする上で英語のドキュメントを読むことは必須になるので、今のうちから慣れておきましょう。

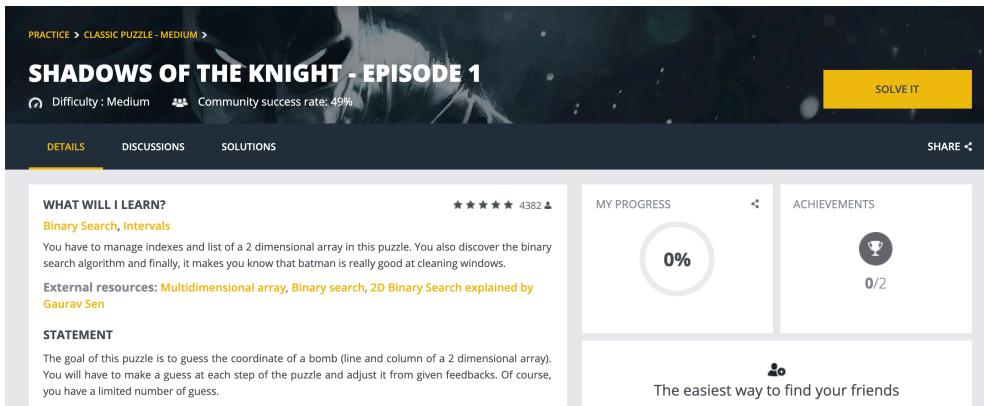


図2 CodinGame の問題概要画面

3 Capture The Flag (CTF)

コンピュータセキュリティに関する問題を解くコンテストです。筆者がハマっているコンテストなのでみんなも是非やろう。

入門がてら挑戦できる競技プログラミングとは異なり、こちらはコンピュータに関する深い知識が要求されるため、ある程度コンピュータの知識に自信がある人におすすめします。競技のフォーマット自体は至ってシンプルで、指示されたサーバに接続したり指定のファイルを調査したりして「フラグ」と呼ばれる文字列を入手し提出するだけです。問題ごとに点数が設定されているので最終的にはそれらの合計点で競います。フォーマットがシンプルな一方で問題の本質部分は相当に難しく、ジャンルとしては Web や暗号、バイナリ解析、エクスプロイト^{*5}など、多種多様な範囲から出題されます。一人で全ジャンル戦っていく超人も極稀にいますが、普通は複数人でチームを組んで分担しつつ問題を解きます。

紙面上で実際の問題を紹介するのは大変なので、C の簡単なサンプルコードとそれをコンパイルして逆アセンブル結果を載せておきます（図3）。なんとなく `scanf` を 2 回呼んで足し算（add 命令）して `printf` している流れが掴めると思います。実際にはこれが 100 倍くらい手強くなつたやつからパスワードを解析するといった感じになります。

なんと言っても CTF の醍醐味は実際に稼働しているサービスに対して攻撃を仕掛けられるという点でしょう。私たちが普段使用するような通常のサイトやサービスに対して攻撃をすると逮捕される可能性があるので攻撃してはいけませんが、CTF は管理者が一定のルール下^{*6}で攻撃を許可しているのでその心配がありません。そのため、Web アプリケーションに対して不正にログインして管理ユーザになりますたり、コード

^{*5} プログラムを乗っ取る攻撃を行うこと

^{*6} CTF でも攻撃していい範囲などを記述したルールが必ずあります

Listing 1 hoge.c

```
#include <stdio.h>

int main() {
    int a, b;
    scanf("%d", &a);
    scanf("%d", &b);
    printf("%d\n", a + b);
    return 0;
}
```

Listing 2 hoge の逆アセンブル結果 (main 関数)

```
00000000004005f6 <main>:
4005f6:    push    rbp
4005f7:    mov     rbp, rsp
4005fa:    sub     rsp, 0x10
4005fe:    mov     rax, QWORD PTR fs:0x28
400607:    mov     QWORD PTR [rbp-0x8], rax
40060b:    xor     eax, eax
40060d:    lea     rax, [rbp-0x10]
400611:    mov     rsi, rax
400614:    mov     edi, 0x4006f4
400619:    mov     eax, 0x0
40061e:    call    4004e0 <__isoc99_scanf@plt>
400623:    lea     rax, [rbp-0xc]
400627:    mov     rsi, rax
40062a:    mov     edi, 0x4006f4
40062f:    mov     eax, 0x0
400634:    call    4004e0 <__isoc99_scanf@plt>
400639:    mov     edx, DWORD PTR [rbp-0x10]
40063c:    mov     eax, DWORD PTR [rbp-0xc]
40063f:    add     eax, edx
400641:    mov     esi, eax
400643:    mov     edi, 0x4006f7
400648:    mov     eax, 0x0
40064d:    call    4004c0 <printf@plt>
400652:    mov     eax, 0x0
400657:    mov     rcx, QWORD PTR [rbp-0x8]
40065b:    xor     rcx, QWORD PTR fs:0x28
400664:    je     40066b <main+0x75>
400666:    call    4004b0 <__stack_chk_fail@plt>
40066b:    leave
40066c:    ret
```

図3 ソースコードとコンパイルされた実行ファイルの比較

をカタカタッターンしてサーバを乗っ取ったりしたい人にはおすすめのコンテストです。もちろん面白がつて遊ぶだけではなく、CTF をプレイするうちに攻撃者視点でサービスを見るという力が身につくので、普段のプログラム開発のときにも「ここは攻撃者が真っ先に狙うから正しい実装は何か調べよう」といった感覚を持って開発することができます。

CTF の参加方法についてですが、CTFtime^{*7}というサイトから世界各地で開催される CTF を確認できるので興味のある人は毎週チェックしましょう。また、常設型と呼ばれる常時コンテストを開催しているタイプ

^{*7} <https://ctftime.org/>

のサイトもおすすめです。初心者向けおすすめなのは、picoCTF^{*8}とか、ksnctf^{*9}です。

4 機械学習コンペティション

機械学習熱の高まりとともに、いかにうまく機械学習のモデルを構築できるかを競うサイトが登場しました。また企業からの注目も相まって、企業が賞金を出してコンテストを開催するケースも増えています。ただ、あいにく筆者にはこの辺りの知識があまりないので、Kaggle というサービスの簡単な紹介だけします。

4.1 Kaggle

Kaggle^{*10}は機械学習コンペティションサイトの中で最大手のサイトです。目玉機能である賞金付きコンペティションの他に、データセットやカーネルと呼ばれる共同解析用の学習環境が提供されています。特にカーネルはコンペティションに参加することなく他の人が学習をどのように進めているかといった情報が記載されているため、読むだけでもかなり勉強になります。さらに、公開されているカーネルを自分のアカウント上にコピーしてコードを変更して結果を確認することもすべてブラウザ上からできるので、興味はあるけど手の出し方がわからない人にもおすすめだと思います。

Competition	Description	Prize	Teams
Two Sigma: Using News to Predict Stock Movements	Use news analytics to predict stock price performance	\$100,000	2,927 teams
Jigsaw Unintended Bias in Toxicity Classification	Detect toxicity across a diverse range of conversations	\$65,000	2,446 teams
LANL Earthquake Prediction	Can you predict upcoming laboratory earthquakes?	\$50,000	4,482 teams
Predicting Molecular Properties	Can you measure the magnetic interactions between a pair of atoms?	\$30,000	329 teams
Google Landmark Recognition 2019	Label famous (and not-so-famous) landmarks in images	\$25,000	281 teams

図 4 Kaggle で開催されている賞金付きコンペティション（執筆当時）

*8 <https://2018game.picoctf.com/>

*9 <https://ksnctf.sweetduet.info/>

*10 <https://www.kaggle.com/>

5 おわりに

情報科学類で提供されている講義を受けているだけでは講義で得られる分しか成長できません。かと言つて自分で勉強するのもなかなか続かないで、ゲーム感覚で楽しめるようなものを紹介しました。この他にも探せばたくさんあると思うので、何かを学びたいと思ったらゲーム形式のものがないか探してみるといいでしょう。皆さんも是非沼にハマって圧倒的成長をしましょう！

Pairs に関する記録

文 編集部 ジェンガ 2003 (@NTSC_J)

1 はじめに

And as I sat watching the intimate and highly personal video, stolen only hours earlier from one of my best friends, I realized that something important was missing from my life.

Trainspotting (1996)

こんにちは。ジェンガ 2003 です。つくばで独り暮らしをしていると、ふとした瞬間に孤独を感じことがあります。特に最近は気圧のせいか、そんな瞬間が増えました。

この記事では、孤独に駆られうつかり Pairs に入会してしまった筆者の体験を共有します。この情報が、本誌を手にとった方の導きとなれば幸いです。

2 Pairs とは

Pairs はいわゆるマッチングアプリの一つです。

さて、マッチングアプリとは何でしょうか。INFOSS 情報倫理は、次のように説明しています。

出会い系マッチングアプリは比較的最近始まったものです。インターネット上で異性同士を結びつけるアプリという点で、出会い系と同じような目的で使われることもあります。

出会い系マッチングアプリでは、入会時に免許証やパスポート等による年齢確認が行われます（18歳未満の児童は使用禁止）。また、実在の人物であることを証明するために Facebook のアカウント連携や電話番号での認証等を義務付けているところもあります。出会い系にはないユーザ認証の制限がかかりますが、知り合った人と実際に会ってトラブルに発展することがあります。

この記述は、筆者の経験と一致しています。ただ、知り合った人と実際に会ってトラブルに発展するかどうかはわかりません。筆者は知り合った人と実際に会ったことがないからです。

3 Pairs への登録

スマートフォンがあれば、Pairs への登録は簡単です。アプリをインストールすれば、あとは言われるままでするだけです。ただ、自分が実在し、18歳以上であることを証明するため、身分証の画像データをアップ

ロードする必要があります。

4 有料会員登録

Pairs は無料会員でもマッチングできます。しかし、あなたが男性なら、マッチした人にメッセージを送ったり、受け取ったメッセージを読んだりするのにお金がかかります^{*1}。

Pairs は月額制で、男性の基本料金は 3480 円/月です。また、さらにお金を積むと、男性はプレミアムオプション、女性はレディースオプションというものを利用できます。これがあるとメッセージが既読かどうかを見たり、プロフィールをフリーワード検索したりできるようになります。

筆者は今年の 4 月 1 日、3 ヶ月の契約とプレミアムオプションがセットで **6240 円** のパックを購入しました。読者の皆さんはこの額、どう思いますか？

5 プロフィール

自分のプライバシーにかかわる情報はしっかりと管理し、むやみに発信しないようにしましょう。

INFOSS 情報倫理

自分のプロフィールを設定します。プロフィールには、以下のような項目が用意されています。

- ニックネーム
- 写真
- 年齢（生年月日）
- つぶやき
- 自己紹介文
- Pairs プロフ
- 詳細プロフ

写真としては、顔や顔以外の写真を設定することができます。一応、1 人で写った無加工の正面からの顔写真が良いとされているようです。

つぶやきは、プロフィールがリスト表示されている時、写真の下に表示される文字列です。最大 24 文字で、1 日に 3 回までしか変更できません。

自己紹介文は、1000 文字まで書くことができます。長文を書く人もいれば、全く書かない人もいます。

Pairs プロフには、身長、体型、血液型、居住地、出身地（ともに都道府県単位）、職種、学歴、酒を飲む頻度、タバコを吸う頻度を入力します。この情報は他の利用者から見えるほか、おすすめの**お相手**^{*2}を推薦す

^{*1} 女性は無料。

^{*2} Pairs 用語で、マッチングする対象はこのように呼ばれます。

るためにも使われているようです。

詳細プロフには、職種名（好きな文字列）、年収、結婚歴、子どもの有無、結婚に対する意思、子供が欲しいかどうか、家事・育児への積極性、出会うまでの希望^{*3}、初回デート費用をどう分担するか、性格・タイプ（リストから複数選択）^{*4}、社交性、同居人、休みの曜日、好きなこと・趣味（好きな文字列を3つまで）を設定することができます。

プロフィールのすべての項目を埋めることが推奨されています。

いずれの項目も、内容が公序良俗に反するものでないことを運営会社が確認するまで公開されません。とはいっても、つぶやきなどは送信した次の瞬間には承認されるので、承認手続きの一部は無人化されているんじゃないかなと思います。

6 コミュニティ

Pairs の特徴的な機能として、**コミュニティ**機能があります。これは自分の好きなものを表明する機能です。文字列でコミュニティを検索し、コミュニティに属している**お相手**を一覧表示することができます。コミュニティは趣味・ライフスタイル・価値観の3つのカテゴリに分かれていますが、趣味とライフスタイルは微妙に内容が重複しています。

コミュニティは簡単に作成できるため、あらゆるもののが存在します。本誌の読者が興味を持ちそうなものとしては、

- xkcd
- Ingress
- Steam
- group_inou
- 小林銅蟲
- 悪魔城ドラキュラ
- GitHub
- カレー大好き
- 筑波大学出身
- ラーメン二郎
- 激辛部
- 蒸留酒大好き

^{*3} マッチングしたらまず会いたいか、あるいはメッセージをある程度交換してからがいいのか、等

^{*4} 筆者の性格は優しい、素直、穏やか、親しみやすい、明るい、インドア、真面目、知的、誠実、楽観的、落ち着いている、謙虚、思いやりがある、好奇心旺盛、寛容、裏表がない、マイペース、気分屋です。



図1 コミュニティたち

- コンピュータ科学
- 花粉症に困ってます
- 仮想化
- Kotlin
- Haskell
- zsh 派
- StackOverflow

などのコミュニティが存在します。

特にニッチなコミュニティだと、属しているだけである程度話題が合いますから、合理的な仕組みだと思います。

ただ、同じ内容のコミュニティが複数存在したりしていて、いい加減なところもあります。

7 推薦

毎朝 10 時頃に、数人の気の合いそうなお相手が推薦されます。

しかし、筆者の感覚では全く気の合わなさそうな人しか推薦されません。実際、**いて座に相性ぴったりのお相手**とかほざいてくるので、この機能は全く信用できません。

8 マッチング

コミュニティ検索やフリーワード検索、推薦などで良いと思う**お相手**を見つけたら、**いいね**ボタンを押します。いいねされると通知が届くので、**お相手**がそれを見ていいねを返すと、マッチング成立ということになります。

いいねをするたびに 1 ポイント消費します。このポイントは毎月末に 30 ポイント追加されます。また、3 ポイントを消費して**メッセージ付きいいね**を送ることもできます。

マッチングが成立すると、メッセージを交換できるようになります。メッセージには料金はかかりません。メッセージも検閲されており、最初のメッセージで連絡先を送ったりすることは禁止されています。

いいねを返してもらえない場合は、**みてね**というボタンを押すことができます。みてねは 3~5 ポイント、メッセージ付きみてねは 6~8 ポイント消費します。

筆者は誰ともマッチしないのではないかと不安に思っていましたが、意外にも何人かとマッチングすることができました。

9 精神への影響

マッチングが成立したりメッセージが返ってきたりすると大変うれしい気持ちになります。これはツイッターでいいねやリツイートをされる感覚に似ています。

一方で、マッチングに失敗したりメッセージが返ってこなくなると大変悲しい気持ちになります。これは渾身のツイートが滑るよりもずっとつらい感覚で、長く尾を引きます。

私の精神が脆弱なだけかもしれません、Pairs は精神にかなりの悪影響があると思います。

10 おわりに

悲しくなってきた……

読者の方で、筆者より Pairs が上手な人がいたら、こっそりコツを教えてください。情報はツイッターか WORD のアンケート回収ボックスで募集しています。

UEFI ではじめる OS 開発

文 編集部 突撃隊

1 モダンな開発がしたかったんや

こんにちは！元気に OS 開発してますか？突撃隊です。

春休みが終わる直前になんとか緑本^{*1}を終えました。いやあ、素晴らしい本ですね、1日あたり 2~3 時間くらい持つていかれましたがその価値はある本でした。読者のみなさんもやってください。

さて緑本を終えて、なんかもう少しステップアップした OS 開発やりたいなーとなっていたので、緑本でやっていないことを考えてみました。そしたらあるじゃないですか、UEFI^{*2}です。緑本を読んだことのある人ならわかると思いますが、この本はがつたりレガシー BIOS^{*3}で開発していく、UEFI の U の字も本には出てきません^{*4}。それになんかレガシー BIOS ももう使われなくなるらしいし^{*5}、今後はフルスクラッチで OS 開発しようとしたら UEFI を避けては通れなくなります。デファクトスタンダードとなりつつある UEFI に今のうちに慣れて、ライバルのアツとコーナーで差をつけましょう。

2 UEFI アプリとしての OS ローダを書いてみる

2.1 目標

今回は UEFI アプリのお作法を学ぶ目的で、OS ローダとそれを利用した OS を書いてみます。プログラムを簡単にするために、OS ローダと OS 本体と一緒にコンパイルして単一の efi バイナリにしてしまいます。OS ローダから OS 本体のファイルを読み込んでスタートさせる方がかっこいいのですが、大変なので今回はやりません。OS 本体も OS ローダから渡される情報を元に、フレームバッファにデータを書き込んで画面に色を塗るだけの簡素なものにします。もっと色々やりたかったが時間が足らんかった^{*6}……。

以下は自分が作業したリポジトリの URL です。Minimal OS という意味で minOS と名付けました。word-article タグを参考してください^{*7}。

minOS: [<https://github.com/Totsugekitai/minOS>]

^{*1}『30 日でできる！OS 自作入門』という本のこと

^{*2} Unified Extensible Firmware Interface の略。ファームウェアの共通規格のこと

^{*3} Basic Input Output System の略。IBM のファームウェア規格

^{*4} 初版発行が 2006 年だからねしちゃうがないね

^{*5} https://uefi.org/sites/default/files/resources/Brian_Richardson_Intel_Final.pdf

^{*6} 記事の執筆は 2019/05/20 に行われた

^{*7} 2019/05/26 現在、develop ブランチではローダから minOS 本体のファイルを読み込んで起動することができます

2.2 環境構築

2.2.1 UEFI の SDK

UEFI アプリを開発する SDK は主に以下の 2 つが存在します。

- EDK2
- gnu-efi

EDK2 は Intel が作った SDK で、UEFI アプリ以外に、UEFI フームウェアや UEFI ライブリも全てこれまで作れちゃう優れものです。その反面扱いが少しややこしいです。gnu-efi は UEFI アプリを作ることを目的にして作られた SDK で、扱いは簡単ですが命名規則や ABI が UEFI の規格書と違っていたり、API がところどころ実装されていなかったりと不完全です。

ドキュメントの検索のしやすさという観点から、今回は EDK2 で開発します。

以下のコマンドで EDK2 のリポジトリを clone し、環境変数を設定したり、作業用ディレクトリや必要なファイルを作ったりします。

```

1 $ git clone https://github.com/tianocore/edk2.git
2 $ cd edk2
3 $ source edksetup.sh
4 $ mkdir -p LoaderPkg/Applications/MinLoader && cd ./LoaderPkg
5 $ touch LoaderPkg.dec LoaderPkg.dsc Applications/MinLoader/MinLoader.inf

```

設定ファイルである、`LoaderPkg.dec`、`LoaderPkg.dsc`、`MinLoader.inf` と `Conf/target.txt` の設定は、『EDK II で UEFI アプリケーションを作る^{*8}』を読んで参考にしてください。

以降は `edk2/LoaderPkg/Application/MinLoader/` で開発します。

2.2.2 エミュレータ

`qemu-system-x86_64` を使いますが、`OVMF.fd` という UEFI フームウェアのファイルが必要です。ここ^{*9}からダウンロードしてきましょう。また、`OVMF.fd` から `bios.bin` というファイルも作ります。

```

1 $ cp /path/to/OVMF.fd bios.bin

```

2.3 コード解説

これで準備が整いました。早速ローダを書きましょう。UEFI の仕様書とにらめっこしながら書いたローダがこちらです。

^{*8} <https://osdev-jp.readthedocs.io/ja/latest/2017/create-uefi-app-with-edk2.html>

^{*9} <https://sourceforge.net/projects/edk2/files/OVMF/>

```
1 // MinLoader.h
2 #include <Uefi.h>
3 #include <Library/UefiLib.h>
4 #include <Library/BaseMemoryLib.h>
5 #include <Library/UefiBootServicesTableLib.h>
6 #include <Protocol/GraphicsOutput.h>
7 #include <X64/ProcessorBind.h>
8 #include <Guid/FileInfo.h>
9 #include <Uefi/UefiMultiPhase.h>
10 #include <Uefi/UefiSpec.h>
11
12 #include "kernel.h"
13
14 #define MEM_DESC_SIZE      4800
15
16 unsigned char mem_desc[MEM_DESC_SIZE];
```

まずはヘッダファイルからです。たくさんインクルードしています。ここでインクルードしているファイル群の実体は edk2/MdePkg/Include 内にあります。

次に本体の [MinLoader.c](#) を見てみましょう。

```
1 // MinLoader.c
2 #include "MinLoader.h"
3
4 EFI_STATUS
5 EFIAPI
6 Uefi_main(EFI_HANDLE image, EFI_SYSTEM_TABLE *st)
7 {
8     struct bootinfo_t bootinfo;
9     EFI_STATUS status;
10    EFI_GRAPHICS_OUTPUT_PROTOCOL *gop = NULL; // 画面描画に必要
11
12    // start up GraphicsOutputProtocol
13    EFI_GUID gop_guid = EFI_GRAPHICS_OUTPUT_PROTOCOL_GUID;
```

```

14     status = gBS->LocateProtocol(&gop_guid, NULL, (void**)&gop);
15
16     // カーネルに渡す引数を設定
17     bootinfo.vinfo.fb = (unsigned long long)gop->Mode->FrameBufferBase;
18     bootinfo.vinfo.fb_size = (unsigned long long)gop->Mode->
19         FrameBufferSize;
20
21     // ExitBootService()の処理を開始
22     // メモリマップを取得せなあかんらしいので取得
23     UINTN mmapsize = MEM_DESC_SIZE;
24     // GetMemoryMap()で返ってくる値を格納する変数たち
25     UINTN mapkey, descriptorsize;
26     UINT32 descriptorversion;
27     do {
28         // メモリマップ取得
29         gBS->GetMemoryMap(&mmapsize,
30                            (EFI_MEMORY_DESCRIPTOR *)mem_desc, &mapkey,
31                            &descriptorsize, &descriptorversion);
32         status = gBS->ExitBootServices(image, mapkey);
33     } while (EFI_ERROR(status));
34
35     // カーネルのプログラムに情報を渡す
36     start_kernel(&bootinfo);
37
38     return EFI_SUCCESS;
}

```

それでは解説をしていきます。

まずUefi_main関数ですが、第1引数にEFI_HANDLE型のimage変数を、第2引数に(EFI_SYSTEM_TABLE *)型のst変数を受け取り、戻り値の型はEFI_STATUS EFI APIに設定しています。今回は第1引数だけ終盤に使います。

関数内を見ていくましょう。

いきなり(struct bootinfo_t)型の変数bootinfoを宣言していますが、これはminOS本体に渡す情報詰める変数です。構造体定義はminOS本体の方のkernel.hでしています。

その後に (EFI_GRAPHICS_OUTPUT_PROTOCOL *) 型の変数 `gop` を宣言しています。

`EFI_GRAPHICS_OUTPUT_PROTOCOL` 型は、画面描画に関する情報や関数を提供する構造体です。

次に 13~14 行目を見ます。ここでの目標は `gop` に値を詰め込むことです。まずは `EFI_GUID` 型の変数 `gop_guid` を宣言して、`EFI_GRAPHICS_OUTPUT_PROTOCOL_GUID` を代入しています。`EFI_GRAPHICS_OUTPUT_PROTOCOL_GUID` は定数で、`GraphicsOutput.h` 内で定義されています。突然 `gBS` という変数が出てきますが、これは `EFI_BOOT_SERVICES` 型で、`UefiBootServicesTableLib.h` により宣言されているので `MinLoader.c` 内で宣言しなくとも使えます。これを用いて `EFI_BOOT_SERVICES` `.LocateProtocol()` という関数を呼び出しています。 `LocateProtocol()` に `&gop_guid` と `NULL` と `(void**) &gop` を渡すと、`gop` に値が詰められます。以降は変数 `gop` のメンバにアクセスすることで、画面描画に関する情報が手に入ります。

17~18 行目では、カーネルへ渡す情報を `bootinfo` に詰めています。今回はフレームバッファのアドレスとサイズが欲しいので変数 `gop` から手に入れています。

必要な情報は手に入れたので、UEFI 環境から脱出します。UEFI 環境から脱出するには、`EFI_BOOT_SERVICES.ExitBootServices()` を呼び出します。22~30 行目はそのための準備です。

メモリマップを取得する必要があるので `EFI_BOOT_SERVICES.GetMemoryMap()` を呼び出します。22~25 行目は呼び出しに必要な変数を宣言しています。メモリマップのサイズである `mmapsize` だけは値を指定しておきます。`MEM_DESC_SIZE` はカーネル本体が入るサイズを入れてください。`GetMemoryMap()` が呼び出されると、`mapkey` に情報が詰められるので、それを `ExitBootServices()` の第 2 引数に入れてやります。第 1 引数は `Uefi_main` 関数が受け取った変数 `image` を入れてやります。`ExitBootServices()` が成功すると、UEFI の機能がほとんど使えなくなり、UEFI 環境から脱出します。

カーネルのプログラムに情報を渡し、ローダプログラムは役目を終えます。

最後にカーネルのコードを見てみましょう。

```
1 // kernel.h
2 struct video_info_t {
3     unsigned long long fb;
4     unsigned long long fb_size;
5 };
6
7 struct bootinfo_t {
8     struct video_info_t vinfo;
9 };
10
11 void start_kernel(struct bootinfo_t *binfo);
```

```

1 // kernel.c
2 #include "kernel.h"
3
4 void start_kernel(struct bootinfo_t *binfo)
5 {
6     struct video_info_t vinfo = binfo->vinfo;
7     unsigned long long *fbptr = (unsigned long long *)vinfo.fb;
8     unsigned long long fbsize = vinfo.fb_size;
9
10    unsigned long long i;
11    for (i = 0; i < fbsize; i++) {
12        fbptr[i] = 0x00FF000000FF0000;
13    }
14
15    while (1) {}
16
17    return;
18}

```

これは簡単ですね。ローダから渡された情報を元に、フレームバッファのアドレスにデータを書き込んでいます。

2.4 ビルドと実行

ここまで `edk2/LoaderPkg/Applications/MinLoader/` 内には、`MinLoader.h`、`MinLoader.c`、`kernel.h`、`kernel.c` の 4 つのファイルがあるはずです。この状態で `edk2/` 内でシェルに `build` というコマンドを打ち込むと efi アプリがビルドされます。ビルドに成功すると、`edk2/Build/` に `Hoge0kgX64/` のようなディレクトリが生成されます。ディレクトリをたどっていき `X64/` というディレクトリに入ると、`MinLoader.efi` が生成されているはずです。自分の環境では `edk2/Build/LoaderPkgX64/NOOPT_GCC5/X64/` 内に生成されていました。

それではエミュレータで実行しましょう。カレントディレクトリを `NOOPT_GCC5` として、以下のコマンドで実行します。

```

1 $ qemu-system-x86_64 --bios OVMF.fd -pflash bios.bin -hda fat:rw:hda-
  contents

```



図1 成功時の様子(印刷の都合上灰色ですが実際は赤いです)

これで `qemu` が起動します。起動すると UEFI Shell が起動するので、プロンプトが出たら以下のコマンドを打ち込みます。

```
1 Shell> fs0:  
2 FS0:\> MinLoader.efi
```

すると画面が赤く染まります。おめでとう！

3 さあみんなも

やってみよう！

4 資料集

- UEFI Specification [<https://uefi.org/specifications>]
- osdev-jp wiki [<https://osdev-jp.readthedocs.io/ja/latest/>]
- 大神 祐真 (Ohgami Yuma) 氏の資料 [<http://yuma.ohgami.jp/>]

おけしょ～うをしよう

文 編集部 はじめ

1 はじめに

この記事は「処世術」として化粧をしている人にベースメイクを上手くやってのけるコツを伝える目的で書かれました。数ある「処世術」の中でもベースメイクは特に困難だと私は思っています。上手くやらないと「浮き」ますよね。「浮く」の具体的な説明は後述しますが、簡単に言うと「化粧をしたらかえって肌が汚く見える」現象です。今回これを中心に原因と対策を紹介します。

肌全体の化粧はその人の肌質や季節で適したもののが変わるために、「雑誌やインターネットを頑張って調べたけれど結局何がいいのかよく分からなかった」という経験がある人もいるのではないでしょうか。そんな状況下で自分に合った化粧方法や化粧品を模索するのはきっと大変なはずです。ですからこの記事では、化粧方法を劇的に変えたり化粧品を買い足したりするのではなく、現状に少し工夫を加えてよい結果を得ることを目指します。

2 化粧工程おさらい

他の処世術と同じで化粧には明確な正解がありません。難しいですよね。何パターンか紹介しますので、あてはまらない場合はどれか近いものを参考にしてください。

- (a) スキンケア→BB クリーム→フェースパウダー
- (b) スキンケア→下地→フェースパウダー
- (c) スキンケア→下地→パウダーファンデーション→フェースパウダー
- (d) スキンケア→下地→リキッドファンデーション→フェースパウダー

ついでに用語もチェックしておきましょう。今回のはじめんメソッドでは粉か液体かが最大のポイントです。

- (A) スキンケア——肌の調子を整える「液体」
- (B) 下地——肌の表面を滑らかにする「液体」
- (C) パウダーファンデーション——肌のあらを隠すための「粉」
- (D) リキッドファンデーション——肌のあらを隠すための「液体」
- (E) BB クリーム——B と D の機能を複合した「液体」
- (F) フェースパウダー——顔のテカリを抑え、既に塗られているものを落ちにくくする「粉」

3 化粧が「浮く」とは

ベースメイクを劇的に難しくしているのがこの「浮き」現象です。先ほど「化粧をしたらかえって肌が汚く見える」現象と説明しました。もっと具体的には

- ・化粧がよれてまだら模様になっている（パターン i）
- ・粉っぽくなっている（パターン ii）

2 パターンを指します。前者はカレーが微妙に残っている洗っている途中のカレー鍋、後者は干ばつの地面のイメージです。ちなみに表面が平らで色と質感が均一なのが理想状態です。浮くタイミングは化粧をした直後だったり時間が経つてからだったりと様々ですが、この記事では特に区別しません。

4 環境構築

まず環境を整えましょう。プログラミングより環境構築は簡単です。条件は2つだけで、「明るい場所で」「近づいて」です。明るさは学校の教室くらい、鏡との距離は10cmくらいをおすすめします。つまり、浮いているかそうでないかを確認できる環境です。肌の質感は均一か、カサついているかあるいはテカっているか、現状をチェックすることが改善に繋がります。私はテープLEDを鏡のフチに貼りました。

5 原因と対策

化粧が浮く原因と、なぜそれが浮きに繋がるか、どうやって対策するか、を説明します。カッコ内は対応するパターンを示します。出かける前に化粧をする段階で実践してみてください。

5.1 肌のコンディションに原因があるケース

- ・乾燥している（ii）

肌の乾燥した部分に粉などが引っかかり、乾燥が目立つことになります。スキンケアで使う液体の量を増やし、顔を触ったときにカサつきを感じない位にしてみましょう。

- ・汗をかいている（i）

汗の上に化粧をすると、汗が流れたときに化粧も流れ、まだら模様になります。部屋の温度を下げて完全に汗がひいた状態で化粧を始めましょう。

5.2 下地・BBクリーム・リキッドファンデーション等、「液体」を付ける工程に原因があるケース

- ・付けすぎ（i・ii）

乾燥しやすい部分は時間が経つと粉っぽさが目立ち、逆に皮脂が出やすい部分は皮脂の上で化粧が動いてまだら模様になります。少量ずつ、薄く均一に付けることを心がけてください。それが難しい場合はテクスチャ^{*1}が固い可能性があります。次の項を参照してください。

- ・テクスチャが固い（i・ii）

^{*1} 肌に付けたときの質感

顔に均一に付けられなかつたり、付けすぎてしまつたりする可能性が高くなります。私の場合、伸ばすときに肌に引っかかる感じがあれば固いと判断しています。付けようとしている液体に化粧水や乳液をませたり、スキンケアで使う液体を増やして肌をよりしっとりした状態にしたりと、滑らかに伸びせる状態に調整するのがおすすめです。

5.3 パウダーファンデーションやフェースパウダー等、「粉」を付ける工程に原因があるケース

- 付けすぎ (i・ii)

余分な粉のせいで、粉っぽさやよれが目立ってしまいます。付けた後に、粉の粒が見えない位がおすすめです。特にiiにあてはまる部分や口元や頬などの乾燥しやすい部分に付ける粉は少量にするのが無難です。付けすぎた時は何も付いていないパフやブラシなどで軽く擦って落としましょう。

- 顔に付いている液体の量が多い (i)

余分な液体の上で化粧が動き、まだら模様になります。粉を付ける前にティッシュや手の平で顔全体を押さえ、「ベタベタ」でなく「しっとり」している位に調節します。

5.4 まとめ

ここまでつらつら事項を並べましたが、つまりは

- 濡れた顔に粉を付けない
- とにかく薄く付ける

だけやればなんとかなるということです。

6 修正

どんなに頑張っても浮くときは浮きます。気づいた時に直しましょう。簡単な装備で外出先でもできる方法だけを紹介します。

- iに気づいた場合

ティッシュや指先などでまだらになっている部分の化粧を落とします。乳液（なければワセリンやリップクリーム）を少量付けて落とすと肌を傷めません。何も持っていない時は指先に水をつけてチョチョっとやれば大丈夫です。フェースパウダー等がある場合は最後に付け直して完成です。

- iiに気づいた場合

ワセリン（なければリップクリームなど）を少量付け、カサつきが見えないようにします。何も持っていない時はとりあえず水でも付けておきましょう。

7 私の場合

最後に、私が4月から5月にかけてのベースメイクを軽～くご紹介して終わりにします。前述した工程で言うと最初のものに近いです。

- 化粧水→ミルククリーム^{*2}→BB クリーム→ハイライト→シェーディング→フェースパウダー

この手法は夏の気温には耐えられないので(汗をかくとテカテカになる)、これから季節よい子は真似しないように。

8 おわりに

本当はそもそも「処世」せざるを得ないという世の中が辛いんですよね～。高校卒業までは化粧をすると非難されるのに、それ以降はできて当然と見なされるのは不条理だと思います。

^{*2} 3CE 製の顔をとにかく白くするクリーム

『ぷよぷよ e スポーツ』 茨城県代表決定戦レポート

文 編集部 らいりゅう号

1 はじめに

『ぷよぷよ』は 1991 年に誕生して以来、老若男女問わず人気を博しているアクションパズルゲームである。昨年には競技性を重視した最新作『ぷよぷよ e スポーツ』が発売され、近年の e スポーツブームと共に更なる人気の高まりが期待されている。茨城国体の文化プログラムにて行われる全国都道府県対抗 e スポーツ選手権の種目の 1 つに『ぷよぷよ』が選ばれたことも記憶に新しい。

この記事では、2019 年 5 月 18 日に行われた全国都道府県対抗 e スポーツ選手権『ぷよぷよ e スポーツ』茨城県代表決定戦の様子をお届けする。



2 全国都道府県対抗 e スポーツ選手権とは

2019 年の国民体育大会は『いきいき茨城ゆめ国体』として茨城県で開催される。その茨城国体の文化プログラム事業の一環として、全国都道府県対抗 e スポーツ選手権(以下“e スポーツ選手権”)が開催されることになった。正式な種目という形ではないものの、国体で e スポーツの大会が開かれるることはメディアでも大きく取り上げられた。

e スポーツ選手権の競技種目は次の 3 つ。『グランツーリスモ SPORT』、『ウイニングイレブン 11』、そして『ぶよぶよ e スポーツ』である。10 月の本大会へ出場する選手を選出するために、この春から各都道府県でそれぞれの種目ごとに代表決定戦が開かれているのだ。

代表選手は各都道府県 1 名ずつなのだが、開催地ということで茨城県の代表選手は特別に 2 名選出される。4 月 6 日にはその 2 名の内の 1 名を決める『ぶよぶよ e スポーツ』特別先行代表決定戦がイオンモールつくばにて行われ、プロプレイヤー^{*1}のぎいろ選手が勝利を収めた。5 月 18 日に同じくイオンモールつくばで行われた代表決定戦は、残る 1 つの代表選手枠をかけた戦いとなる。

代表決定戦の流れについて説明しておこう。代表決定戦は予選と代表決定トーナメントの 2 ステップからなる。

予選

『チャレンジモード 公式チャレンジ』(以下『公式チャレンジ』)のスコアアタックを 2 回行い、それぞれのスコアの高い方をその選手の獲得スコアとする。獲得スコアの高さで順位を決定し、上位 16 名が代表決定トーナメントへ勝ち進む。『公式チャレンジ』は本大会のために実装されたモードで、おじやまぶよが出現しない環境で 2 分間のうちにどれだけスコアを獲得できるかを競う。

代表決定トーナメント

予選上位 16 名によるトーナメントである。対戦は『ぶよぶよ通』ルール^{*2}を用いて 1 対 1 で行い、1 試合 2 本先取・2 セット先取で勝敗を決める。優勝者は茨城県代表として本大会に出場する権利を得る。

3 茨城県代表決定戦当日の様子

2019 年 5 月 18 日 11 時、買い物客で賑わうイオンモールつくば内のイーストコートにて県代表決定戦の火蓋が切られた。会場の中心には特設ステージが設営されており、その手前に予選とトーナメントで使用される PS4 とディスプレイが備えられた机が並んでいた。さらにその隣では試遊体験コーナーがあり、体験者にはオリジナルデザインの紙製のサンバイザーが配布されていた。

11 時から 14 時 30 分まで予選が行われた。予選は事前登録の他に当日参加枠が 20 人分ほど用意されていた。どうやら 20 人というのは目安で厳格な上限というわけではないらしく、台さえ空いていれば何人でも参

^{*1} 本稿では日本 e スポーツ連合の公認を受けたプレイヤーのことを指す。

^{*2} 最もオーソドックスな対戦ルールの 1 つ。互いにおじやまぶよを送り合い、自分のフィールドにぶよが積めなくなったら負け。



加できそうであった。参加賞を貰えるということもありカジュアルに当日参加する人も多かったようだ。人通りが多く目立つ位置に会場があったので、参加はせずとも足を止めて観戦する人も多く見られた。

予選の様子について解説していこう。予選で採用されている『公式チャレンジ』ルールは2分間にどれだけスコアを稼げるかを競うというものである。対戦相手はおらずおじやまぶよも降ってくることのない1人プレイ専用のルールだ。連鎖中も時間は止まらない、時間切れになつても連鎖中であれば連鎖が終わるまではゲームが進行するなどといった特徴がある。2分という時間をどのように配分するか、細かく連鎖を稼ぐかそれとも一発勝負で大きな連鎖を狙うかという戦略が必要になってくるのだ。選手ごとにどのような戦略をとるのかがこのルールの見どころである。

予選参加者のプレイを見てみると、多くの選手が階段積みをしているようだった。階段積みとは、ぶよぶよの積み方の中でも最も有名なものの1つである。シンプルかつ安定感があり初心者にも組みやすい反面、折り返し³部分を作るのは難しいのが特徴だ。なるべく小さいスキで折り返し部分を作る戦略が取られることが多い対戦ルールではあまり見られない積み方でもある。『公式チャレンジ』ルールだと、比較的じっくりとぶよを積む時間が確保できるので安定感のある階段積みを選ぶ選手が多かったようだ。また、当日参加者の多くも階段積みをする傾向にあった。

筆者も当日参加枠で予選に挑戦した。受付を済ませた後、すぐにスタッフに誘導され座席へと移動。コントローラとヘッドホンをセットしていざプレイを開始。普段はGTR⁴を使っているが、周りに合わせて階段積みで攻めてみることにする。ヘッドホンのおかげで周りの様子が気にならず、まるで家にいるときのような感覚でプレイすることができた。そしてまるで家にいるときのような雑魚つぶりを發揮し、対戦相手がないルールなのにも拘わらずゲームオーバーとなってしまった。参加賞のオリジナルポーチを手にすごすと退散したのであった。

14時30分に予選はつつがなく終了した。最終的に予選参加者の数は50人となったようだ。

³ 連鎖を上へ伸ばすための型を指すぶよぶよの用語。

⁴ 折り返し部分を安定して作る積み方の1つを指すぶよぶよの用語。

14時45分には、代表決定トーナメントに出場する16名の選手の発表が行われた。予選スコアの1位から順に選手の名前が呼ばれ、会場は大きな拍手に包まれる。同時に、代表決定トーナメント出場スコアのボーダーは427,290点だったことが発表された。少なくとも13連鎖は必要といったところだろうか。茨城のぶよぶよプレイヤーは非常に高レベルなようだ^{*5}。

15時に代表決定トーナメントが開始された。トーナメント第1回戦と第2回戦は、ステージと手前の対戦台のそれぞれで同時に4組の試合が行われる。司会進行のお姉さんのやたらドスの利いた「ぶよぶよバトル～」という掛け声に「スタート！」と観客全員で合わせるのが大会の流儀らしい。いざ対戦が始まつてみると、同時に4組の対戦が進行するせいか観客の注意が分散し、会場はいまひとつ盛り上がりに欠けるのだった。とは言え、そのほうが選手にとっては静かで好条件だったのかもしれない。それぞれの対戦台で白熱する勝負が繰り広げられていく。

あれよあれよと言う間にトーナメントは進行し、気づくと準決勝戦になっていた。ここからの対戦は1戦ごとにステージで執り行われる。対戦前には出場する選手へのインタビューがあり、選手たちはそれぞれに大会にかける思いを語った。数回の観戦を経て、観客側もルールを理解し始めたのか段々と会場は静かな熱気に包まれていく。さつきまでことあるごとに筆者のつま先を踏んできていたどこぞの男子小学生も、今はステージから目が離せないようだった。

準決勝戦と3位決定戦が終わり、ついに決勝戦が始まる。決勝戦へと駒を進めたのはTekku選手とふじもん選手。Tekku選手は今年の4月13日にプロ認定を受けたばかりの優勝候補である。対するふじもん選手は4月の先行代表決定戦には参加しておらず、実力が未知数のダークホースである。

対戦開始前はプロのTekku選手が優位に思われたが、いざ対戦が始まるとふじもん選手の優勢が続く。その勢いのままストレートで2セットを先取したふじもん選手がTekku選手を下したのであった。対戦後のインタビューでふじもん選手は「茨城県代表として1つでも多く勝ち進みたい」と語り、会場の全員から大きな拍手が送られた。

このようにして、eスポーツ選手権『ぶよぶよ e スポーツ』茨城県代表決定戦はその幕を下ろしたのであつた。ここまで触れなかつたが、WORD所属のれっくす選手も本大会に出場しており、なんと4位という結果を残した。おめでとう！

4 おわりに

駆け足気味になってしまったが茨城県代表決定戦の様子をレポートした。筆者がちゃんと記録を取つていなかつたトーナメントの対戦の内容について詳しく知りたい人はぶよぶよの公式ブログ^{*6}を読むとよいだろう。

この記事が執筆された5月下旬時点では都道府県代表決定戦は絶賛開催中であり、また10月にはここ茨城で全国大会が開催される。ぶよぶよやeスポーツに興味があるなら選手たちの勇姿を見に行ってみてはいかがだろうか。

^{*5} 参考までにお隣の栃木県のボーダーは24,185点であった。

^{*6} <https://ameblo.jp/puyo-sega/entry-12462221032.html>

情報科学類誌

WORD

From College of Information Science

新元号

発行者 情報科学類長

編集長 巨畠和樹

制作・編集 筑波大学情報学群
情報科学類 WORD 編集部
(第三エリア C 棟 212 号室)

2019 年 6 月 26 日 初版第 1 刷発行

(256 部)