Teach Yourself KMC

独習KMC vol.11

京大マイコンクラブ　著

ＫＭＣ創立４０周年によせて

初代会長　田中和彦

はじめに

初代ＫＭＣ会長の田中和彦と申します。１９７７年に創立したＫＭＣは今年、創立４０周年を迎えました。

６月１７日に、京都で記念イベントが開催され、創部以来の多数の卒業生、現役生が集まり、各世代がその時代を振り返るなど、大いに盛り上がりました。

ここでは創部当時の様子など紹介させて頂きます。

ＫＭＣの起源

私は１９７７年４月に京都大学工学部情報工学科に入学しました。小学校６年生の時にアマチュア無線の資格を取り、中学時代にはプログラム式電卓で２次方程式の解を求めるプログラムを組み、また高校時代には生物部に属しながらＴＴＬでデジタルマルチメーターを組み立ててレーザーと落体を組み合わせて重力加速度を測定し鈴木梅太郎賞を受賞しました。当時、ＮＨＫでは日立のＨＩＴＡＣ－１０を使ったＦＯＲＴＲＡＮ講座が放送されており、いつか自分のコンピューターを持ちたいという夢を持っていました。

ミニコンはとても買えませんので、Intel のビットスライスチップでＨＩＴＡＣ－１０のコンパチマシンを作ろうと思い立った私は、そのためにはシミュレーターを作ろう、そのために大型機を使わせて貰おうと、学科長の教授にお願いに行きました。

先生は、「君たちはいずれは、コンピューターを職業として扱うことになる」「別に焦る必要はないんじゃないか」「今しか、アマチュアでなければ出来ないことをしてはどうか」とおっしゃいました。

大型機を借用出来ないことにはガッカリしましたが、確かに先生の言われる通りだと思った私は、翌５月に「マイコンの会」を呼びかけて、毎週、木曜日に学内の教室に集まることを始めました。最初は、誰もマシンを持っておらず「マイコンを語る会」いうならば「ナイコンクラブ」でした。

会の名称は、私は、コンピューターは革命 (Revolution) 的ではなく、社会に応じて進化

(Evolution) するのだと思ったので “Computer Evolution Group”(CEG) としたのですが、仲間からはすこぶる評判が悪く、直ぐに、「京大マイコンクラブ」(KMC) となりました。

やがては、8bit CPU(当時は MPU) を使ったマイコンを各自が自作し、「フルキーボードが付いているんだぞ」と自慢したりするようになりましたが、前出の通り、当初は、最新の MPU のデータシートを持ち寄って「この MPU の命令がキレイだ」「いやこっちの方が動作が速い」などを議論していました。

市販機を購入する部員も現れ、機材を置ける部室をという話になりましたが、学内は盗難など物騒と聞いたので学外で探し、うどん屋さんの２階に畳が痛まぬようにベニア板を敷き詰めて部室が開設されました。

初めてのプロジェクト PLANET で広げた「ネットの輪」

部員各自が自作機や市販機を持ち寄ってワイワイやっていたのですが、研究室でイーサーネットをテーマに与えられていたこともあり、私は、これらをつないで何か出来ないかと考えました。実は私の自作マイコンは本体とキーボードを 20mA の電流ループでシリアル通信で接続していました。これはテレタイプに憧れてそのインタフェースをまねたからでした。この電流ループに複数のマイコンを接続したらと考え、定電流電源とフォトカプラを組み合わせて 4800bps での動作を確認しました。通信プロトコルは、衝突型とポーリング型の混在を可能としました。物理層は後にプラスチックファイバも追加されました。

データリンク層のプログラムはまず Z-80 用を作り、それを 8080 用、6502 用など各自が各マシン用にハンドアセンブルで移植しました。

応用プログラムは、２台のマイコン間でテニスを行う文字通りの「ネットテニス」や、オセロプログラムを自動で戦わせ鍛える自動対局、ロボット戦士を操縦して戦うシミュレーション、また、宇宙船を複数の機能を分散させて実現し宇宙空間を探検する「エリトザハース」など様々なものが、複数の部員の共同作業で開発され、雑誌「アスキー」で取り上げられたり、池袋で開催されたマイコンショーでデモンストレーションを行うなど、ＫＭＣのプレゼンス向上にも大いに役立ちました。(未確認情報ですが電子楽器のインタフェース MIDI のヒントになったとも聞いています)

「今は昔」を越えて

当時、私が自作したマイコンは、Z-80 2.5MHz、メインメモリ 32KB、記憶装置は、カセットテープに 300〜2400bps で記録した数 KB のファイルで、現在、私が使用している自作機、i7 3.1GHz、メインメモリ 32GB、記憶装置 24TB、自宅内総容量 100TB 以上、と比べると、現状は 10 の３乗〜９乗倍の容量、速度であり、驚くべき違いがあります。

現在、ポケットに入っているスマートホンは当時のマイコンどころかスパコン以上の性能があると言われています。

マイコン (パソコン) の意味も当時とは全く違っています。当時は、非常に特殊な道具でした

が、今は、一般的には使うこと自体に関心を持つことは少ないのではないでしょうか。

そのような激変した環境の中でマイコンクラブ、ＫＭＣが４０年も続いていることは正に驚きです。

ある部員は入部の際に「このクラブはマイコンを勉強するところじゃないよ」「マイコンで何かをする場所なんだ」と言われたそうです。

まさにこの精神が変わり者が集まったＫＭＣをかくも長く続けさせた理由だと思います。

最後に私ごとですが、私は今年、還暦を迎えます。長年勤めた会社も退職し、様々な衰えを感じる日々ですが、「今しか、アマチュアでなければ出来ないことを」の言葉を忘れず個人的に最新の技術を試行しその内容を個人ＨＰで公開しています。

＜武蔵野メディア研究所＞

http://www13.plala.or.jp/mml/

創立時の部員もまだまだ頑張ります。ＫＭＣの今後の発展にご注目下さい。

目次

KMC創立４０周年によせて i

KMC の最近の活動 1

新入生プロジェクト 1

その他勉強会 4

部員のコラム 6

WOLF RPG エディターのダウンロード機能で遊ぶ (polaris) 6

作ったゲームのテストプレイについて (hideya) 20

APOLLO 出展記 (tron) 24

格闘ゲームに学ぶ親切な入力設計とプレイヤーの悪用例 (asragi) 29

Slack bot でスロットを作ってみた (yu3) 38

あとがき 43

あとがき 43

新入生プロジェクト

DTM 練習会 2017

tron

ごきげんよう。担当の tron です。DTM 練習会は音楽を作ろうという人たちが集まるプロジェクトです。課題に沿って曲を作ってきて皆で聴く「整数回」と、音楽制作に関する講座をする「小数回」を交互に行っています。前期はコードやドラムなどの基本的なことを学び、課題で実践していました。後期はお題に合った曲を作る、という課題を主に出す予定です。出来た作品はネット上に公開するつもりなので、是非お聴きください。また、ネット上の音楽即売会「APOLLO」に出展したり、より難しい内容に取り組む「Advanced」を立ち上げるなど、新たな試みも行っています。今後もさらに研鑽を積み、様々な場で成果を発表していきますので、ご期待下さい！

お絵かきプログラミング (Piet) 勉強会 2017

kyp

こんにちは。お絵かきプログラミング (Piet) 勉強会担当の id:kyp です。隔週日曜日に開催されるこのプロジェクトは名前にもある通り、文字の代わりに色を解釈の単位とする Piet というプログラミング言語を用いて、毎回指定されるお題に即した様々なソースコード（拡張子は png です）を書いて（描いて）楽しむ集まりです。

KMC と Piet の関係は深く、この勉強会も 3 年目を迎え、沢山の熱心（で暇な）部員によるエディタや多くの芸術的作品も生み出されてきました。今年に入り、我々が考えていた Piet は作者が 2008 年に策定した追加の言語仕様を満たしていないことが判明したため、この勉強会も一時「Piet のようなもの勉強会」と改名されかけるなど大変な危機に遭いましたが、id:dama さんの尽力などにより真の Piet を書ける環境が整いました。

そして前回の勉強会では、ついに KMC 初となる Piet のソースコードが書かれました。今後はこの新しい、世界標準の Piet を更に発展させていきます。というわけで、Let’s piet!!!!!!

お絵描き練習プロジェクト 2017

nepnyan

こんにちは、kmc2 回生の id:nepnyan です。新歓プロジェクトの一つとして今年行ったお絵描き練習プロジェクトについて紹介していきたいと思います。初めに少しだけお絵描き練習プロジェクトの説明をすると、まあ名前の通りではあるのですが、絵を描いたことのある人も、描いたことのない人も一緒に集まって絵に関する知識なりを共有し、絵を上手く描けるようになろうというプロジェクトです。内容的には、主に人の描き方を取り扱い、ときどきパースや質感表現の勉強をしたりもしました。ベジェ曲線の定義についての説明もしましたね（お絵描きとは何の関係もありませんが…）。他には、とても絵が上手な外部の方からメイキングを見せて頂いたことがありました。細かい所にもとてもこだわっていて、とても勉強になりました。また夏休み明けからお絵描きプロジェクトの後期が始まります。後期では、新入生にも講師をしてもらう予定です。楽しみですね。

みんなでゲームを作る 2017

opesan

こんにちは。担当の opesan と申します。「みんなでゲームを作る」では新入生が中心となり、NF（京都大学文化祭）での頒布、展示や冬のコミックマーケットでの頒布を目的にゲームを作ろうと言うプロジェクトです。新入生には実際にディレクター、プログラマー、グラフィッカー、コンポーザーなどの役職についてもらってオリジナルのゲームを作り上げます。今年は fantasies と puzzle という名前の二つのゲームプロジェクトが進行中です。それぞれ文字通りファンタジーの世界観のあるアクションゲームと、パズルアクションゲームになっています。勿論新入生自身がゲーム案を持ち込んでおり、ゲームシステムや設定などを作っていっています。毎週火曜日に進捗報告会を行なっており、その場でゲーム制作が段々と進行していっているのを見て、担当者の私も完成にわくわくしております。また、上回生もそれまでのゲーム制作経験をいかして新入生のサポートを行なっています。皆和気藹々とした雰囲気で開発を行なっております。プロジェクトメンバー一同、鋭意制作中ですので期待してお待ちください！

競技プログラミング練習会 2017 Normal

suibaka

こんにちは．担当の suibaka です．本プロジェクトは，主に競技プログラミング初心者を対象として，基本的なアルゴリズムの解説，および問題演習をするプロジェクトです．

基本的には，30 分程度のアルゴリズム解説，その後 1 時間 30 分程度の問題演習をするとい

う流れでやっていました．ICPC の国内予選が近づいてきたときには，本番を想定して 2〜3 時間の問題演習をしたこともあります．

ちなみに，ICPC は世界規模の大学対抗競技プログラミングコンテストです．KMC から毎年たくさんの部員が参加しています．

後期からは，新入生が自分でトピックを選んで，それについて解説をしてもらう予定です．前期にはあまり触れなかった発展的内容を自分で調査してもらうことで，さらなる実力上昇を期待しています．

最後になりますが，今年もたくさんの新入生が参加してくれました．練習会以外でもコンテストに参加していたり，意欲的な人が多いです．早速 3 人チームを組んで，ICPC の国内予選に参加した新入生もいました．今後の活躍に目が離せません．

C#でゲームをつくろう 2017

ten

担当の KMC2 回生の ID:ten です。C#でゲームをつくろうは、プログラミング未経験者の人でもゲームを作れるようになるために勉強をするプロジェクトです。今年は Altseed+C#という環境でゲームを作ります。C#の文法のはなし（変数からクラスまで）や、Altseed というゲームエンジンの機能を、実際に簡単なゲームを作りながら教えています。また、オブジェクト指向などの話についても勉強をする予定をしています。このプロジェクトでは夏休み明けまでにゲームを作ってもらって発表会することを目標にしています。今年の新入生は積極的に参加していてすごいと思います。新入生も少しずつゲームの作り方が分かるようになってきているので、夏休み明けの発表会が楽しみです。

その他勉強会

ゼロから作る Deep Learning 読書会

utgwkk (a.k.a utagawa)

「ゼロから作る Deep Learning 読書会」の主催をしています、KMC3 回生の utgwkk[*1] です。

この読書会は、文字通り『ゼロから作る Deep Learning[*2]』という本をみんなで読んでいく会です。具体的には、担当者が 1 章を読んでスライドを作って発表し、それを聞く、という輪読形式をとっています。メンバーは、私のような機械学習にあまり馴染みのない人から、研究で機械学習を使うかもしれないという人や、ディープラーニングを使ったプログラムを書いたことのある人まで、いろいろな人が集まりました。

『ゼロから作る Deep Learning』は、外部のライブラリに頼らずに[*3]ディープラーニングをゼロから実装することによって、その原理を楽しく学んでいく、という本です。機械学習の重厚な理論を知る必要はないため、機械学習の知識がない人でもサクサク読み進めることができます[*4]。

4 月 22 日 (土) に読書会を始め、以降隔週土曜日に行ってきました。この原稿を執筆している時点[*5]では、6 章の「学習に関するテクニック」まで読み終えており、次回いよいよディープラーニングの肝である「畳み込みニューラルネットワーク (CNN)」の章に入ります。

[*1] KMC-ID: utgw

[*2] `https://www.oreilly.co.jp/books/9784873117584/`

[*3] numpy に思いきり依存していますが……。

[*4] 高校数学の知識は必要だと思います。

[*5] 2017/06/28

ボカロ調教練習会

amool

このプロジェクトの担当の amool です。DTM 練習会の子プロジェクトのつもりで立ち上げました。VOCALOID は人々の DTM への関心を高めた要因の 1 つです。…いえ、統計などは取っていないので完全に私の主観ではありますが。そんな VOCALOID ですが、KMC ではゲーム制作が盛んということもあってか、VOCALOID が使われることはあまりありません。そういうわけでこのプロジェクトでは VOCALOID をさわってみようというようなノリで始めました。とはいえ VOCALOID は有料なので、今回は UTAU という無料で使える歌声合成ソフトの使い方を勉強しました。しかしこの UTAU というソフト、気力とセンスさえあればかなり良い調声ができるのですが、これがなかなか難しい…。第一回の講座にはそこそこ？ 部員が来ていたのですが、後半にはどんどん過疎化してしまいました。実際、私も UTAU を使った作曲はしたことが無く、このプロジェクトはほとんど私が VOCALOID を部員たちに買わせようと布教するプロジェクトと化してしましました…。そんなこんなで半ば消滅してしまったプロジェクトになってしまいました…が、部室に IA という VOCALOID を導入するきっかけなれたので良かった…ですかね？ また、現在進行中のギャルゲーを作るプロジェクトでは、OP,ED として歌の制作の予定もあるので、これから VOCALOID を使った作曲も活発になればいいなといったところです。

WOLF RPG エディターのダウンロード機能で遊ぶ

polaris

こんにちは、KMC 1回生の polaris です。はじめまして。この記事ではゲーム制作ツール「WOLF RPG エディター」のダウンロード機能でやってみたことを紹介します。

WOLF RPG エディターとは？

WOLF RPG エディター[*1]は SmorkingWOLF 氏によって開発された無料のゲーム開発ツールです [1]。主に見下ろし型 2DRPG を制作することに特化していますが、ゲームの動作を定義する「イベント」を編集することによって他のジャンルのゲームも制作できます。

イベントとコマンド

ウディタ[*2]ではゲームやその中のキャラクターの動作定義を「イベント」という仕組みで行っています。例えば基本システム[*3]の戦闘を制御しているイベントはこのようになっています（図 1）。

[*1] `http://www.silversecond.com/WolfRPGEditor/` において配布されている。

[*2]「ウディタ」は WOLF RPG エディターの通称である。本記事でもこの呼称を使うときがある。

[*3]ウディタに最初から登録されている RPG 用のゲームシステムのこと。各種データベースとコモンイベントというゲーム内から共通して実行できるイベントの集合体になっている。

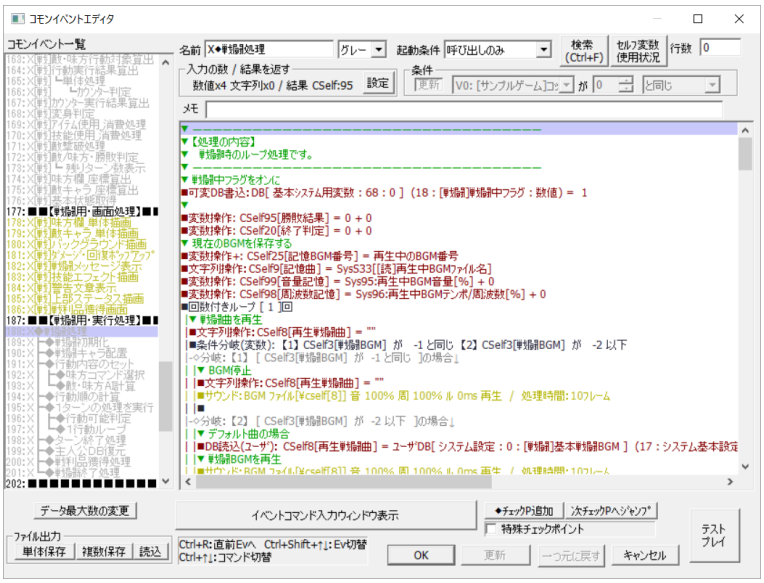

図 1: イベントをエディターで開いた様子

それらイベントを構成するものが「コマンド」で例えばメッセージを表示するものや他のイベントを呼び出すものがあります。

ダウンロードコマンド

それらコマンドの中であまり知られていない（と勝手に私が思っている）ものとして「ダウンロード」があります。これを使うと URL を指定してインターネット上からファイルをダウンロードしたり、サーバーに信号を送ったりできます。初期設定ではこのコマンドは非表示になっているのですが以下の設定をすると使えるようになります。

1. Editor のメインウィンドウの「オプション」→「エディターオプション」をクリック
2.「上級者向けコマンドを使用する」にチェックをつける
3.「設定完了」をクリックする

今回はこのコマンドでどのようなことができるかをテーマにいろいろ試してみようと思います。

とりあえず使ってみる

おことわり

扱うテーマの都合上本記事での実装はセキュリティや WebAPI の約束事などをすべて度外視しています。ここで紹介したものをそのまま実戦配備することはまず不可能[*4]ですのでご了承ください。またテストに使用した環境は WOLF RPG エディター Ver.2.21 と基本システム Ver.2.20、node.js Ver.6.11.0 で、サンプルゲーム上に実装していきます。

サイトをダウンロードしてみる

まずは手始めに Google 検索のウェブサイト `http://www.google.com/` をダウンロードしてみようと思います。サンプルマップ上に次のようにイベントを作成します（図 2、3）。

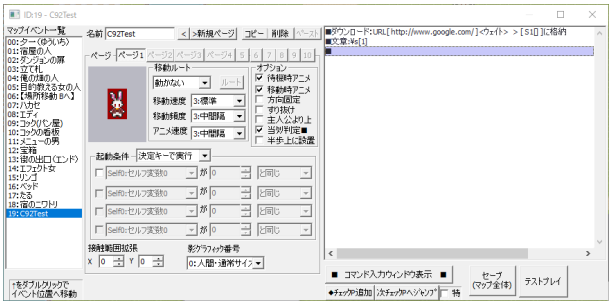

図 2: こんな感じ

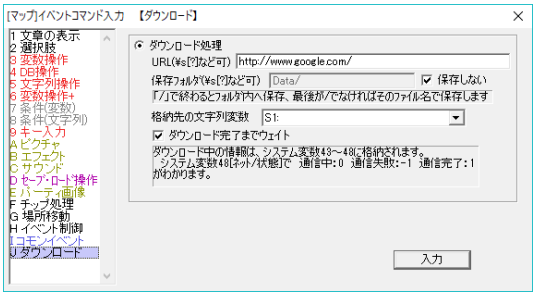

図 3: ダウンロードコマンドの設定

ここまで来たら一旦セーブしてテストプレイでイベントを実行してみます。実行結果は次のようになります（図 4）。

[*4]実用的なものにするためには例えば通信の仕組みを解読されないようにかんたんな暗号化を施したりする必要がある。使用する環境に合わせた対処が必要なこともある。

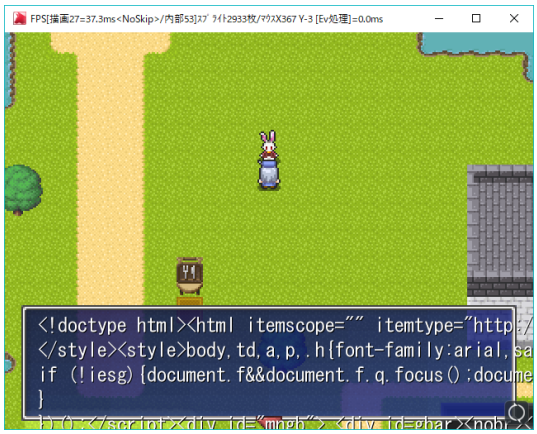

図 4: 実行結果。でもわかりますか？

実際にやってみて思ったのですが少々わかりにくいですね（笑）。メッセージに HTML タグが出てきているのでサイトのデータがそのままダウンロードされたとわかります[*5]。しかし文字列の抽出処理をする必要があることを考えるとこのデータでは処理することが少し難しそうです。

自作サーバーと通信

先程は Google の検索サイトにアクセスしてみました。でも自分の作ったものが表示できないとなんだか面白くないし著作権上のトラブルも発生するかもしれません。その上先程も述べたように HTML データではゲーム内での取り扱いが難しそうです。ということで node.js[*6]を使ってゲームと通信するサーバーを作ってみようと思います。そのソースコードを以下に載せます（コード 1)。

ソースコード 1: server.js：サーバーのメイン処理

```
1  // V- メインプロセス -V
2  console.log('Easy API Server');
3  console.log('2017 polaris@KMC');
4  console.log('/~/~/~/~/~/~/~/~/~/~/');
5
6  var http = require('http');
7  var url = require('url');
8  var qstring = require('querystring');
9
10 // httpServer を作る
```

[*5]実際に実行してみると文章の表示が途中で止まることがある。これはダウンロードしたデータ内にウディタで文字表示の制御に使われる特殊文字が含まれていたためと思われる。

[*6]JavaScript で書かれたプログラムを実行する環境。イベントに基づいた非同期動作ができるのでサーバーや Web アプリケーションの構築に使われる [2]。

```
11 console.log('Initializing http server...');
12 var server = http.createServer(function(request, response) {
13     var pathName = '';
14     console.log('request:' + request.url);
15     pathName = url.parse(request.url).pathname;
16     // どのAPIを呼び出すか？
17     // api. の後ろの名前で合わせる
18     if (pathName === '/api.hello') {
19         // API hello: プレーンテキスト形式で'Hello, world!'と返す。以下
               その処理
20         api_hello(response);
21     } else if (pathName === '/api.reflect'){
22         // API reflect: パラメータ d (api.reflect?d=(ココ！))にセットさ
               れた内容をプレーンテキスト形式でそのまま返す。以下その処理
23         var queryObject = qstring.parse(url.parse(request.url).query);
24         var input = queryObject.d;
25         api_reflect(response, input);
26     } else {
27         // 存在しないAPIが指定されたら404(Not Found)を返す
28         send404(response);
29     }
30 });
31 server.listen(80, 'localhost');
32 console.log('Server running at http://localhost:80/');
33 // A- メインプロセスここまで -A
34 // 以下関数群
35 // 404エラーを送る関数
36 function send404(response) {
37     response.writeHead(404, {'Content-Type': 'text/plain'});
38     response.write('Error 404(Not found)');
39     response.end();
40 }
41
42 // API実行時エラーをクライアントに知らせる関数
43 function sendAPIError(response, msg) {
44     response.writeHead(200, {'Context-Type': 'text-plain'});
45     response.write('API runtime error!: ' + msg);
46     response.end();
47 }
48
49 // API helloの実際の処理
50 function api_hello(response) {
51     response.writeHead(200, {'Context-Type': 'text/plain'});
52     response.write('Hello, world!');
53     response.end();
54 }
```

```
55
56 // API reflect
57 function api_reflect(response, input) {
58     if ( input !== undefined) {
59         response.writeHead(200, {'Context-Type': 'text/plain'});
60         if ( Array.isArray(input) ) {
61             response.write(input[0]);
62         } else {
63             response.write(input);
64         }
65         response.end();
66     } else {
67         sendAPIError(response, '(E-01)Input was not specified.');
68     }
69 }
```

結局のところ httpServer を作って API 名に応じた返答をプレーンテキスト形式でする、ということをしています。HTML ファイルでデータを送らずにテキストデータとしてデータを送受信しているところがポイントです。以下にサーバーに実装した API とその機能を記します。

api.hello　“Hello, world!” と応答します
api.reflect?d=string　パラメータ **d** に指定された内容をそのまま返します

それではこのサーバーと実際に通信してみます。先程のイベントのダウンロードコマンドを次のように変更して（図 5）**api.hello** を使ってみます。

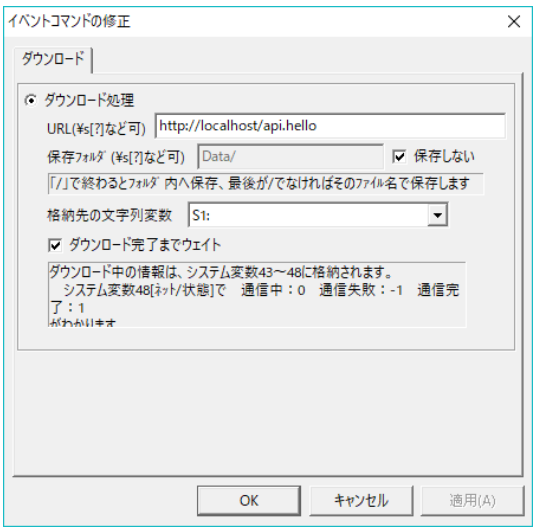

図 5: URL を “http://localhost/api.hello” に変更する（localhost は自分のマシンのこと）

変更を保存したらテストプレイを始める前にまずサーバーを起動します。**server.js** のあるフォルダでコマンドプロンプトを開いて

```
> node server.js
```

で起動できます。その後テストプレイでイベントを実行すると次のようになります（図 6）。

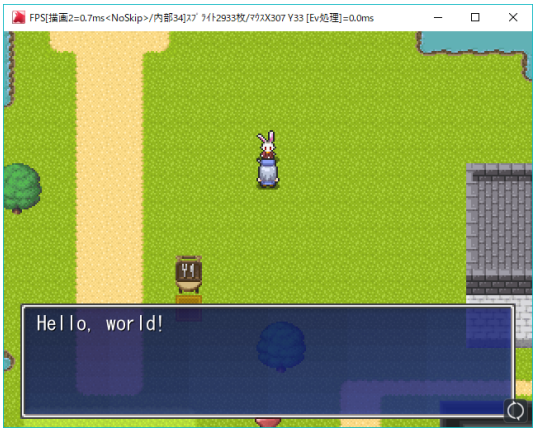

図 6: 実行結果

お次は入力です。**api.reflect** にゲームで入力した値を送ってもう一度それを受け取ってみます（無駄な処理とか言わない）。例のイベントを次のように編集します（図 7、8、9）。

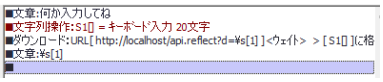

図 7: 変更後のイベント

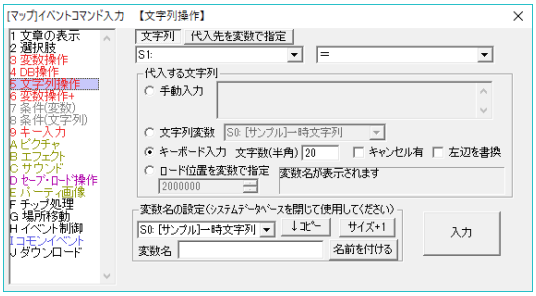

図 8: 入力を文字列変数 **S1** で受ける

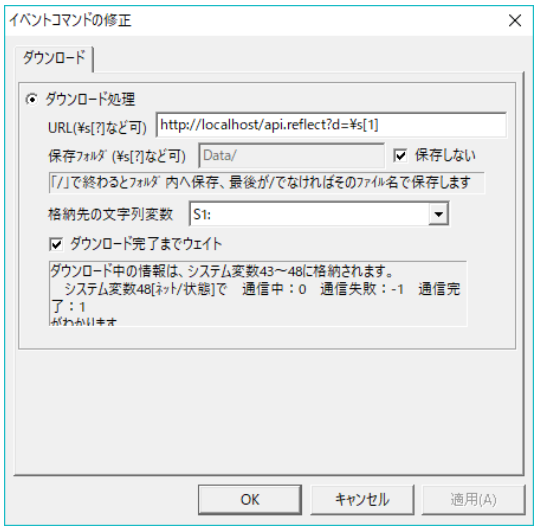

図 9: URL を図のように変更

それで「KMC サイコー！」[*7]と入力したときの実行結果が次なのですが、不思議なことになってしまいました（図 10）。

図 10: 「KMC ツサツイツコツーツ！」

というわけで原因を調べてみた結果、どうやら全角文字などは URL 用にパーセント記号と数字を組み合わせた形にエンコードしなければならないらしいということがわかりました。しかしどうやってやるのか？ 全角文字だけでも恐ろしい量あるじゃないか[*8]。と思ったのですが実はすでにコモンイベントとして実装された方がいらっしゃいました[*9]。それを用いてイベントを修正してみたものがこちらです（図 11）。

[*7]「KMC サイコー！」の元ネタがわかった人はおそらくウディタユーザーである。

[*8]常用漢字だけでも 2136 字ある [3]。

[*9]「UTF-8 エンコード&デコード」という名前のコモンイベント。`http://silsec.sakura.ne.jp/WolfRPGEditor/CommonList/html/` で検索すると詳細が見られる。作者の Rtt 様に感謝！

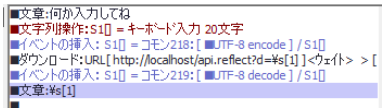

図 11: 要するに送受信する文字列はエンコード済みのものにしている

このコモンイベントの使い方は簡単で、エンコードデコードのどちらなのか指定して処理する文字列が格納された変数を指定するだけです。更に今回はサーバー側も変更を加えます。以下に変えたところを示します（コード 2)。

ソースコード 2: server.js：変更したところ

```
56 // API reflect
57 function api_reflect(response, input) {
58     if ( input !== undefined) {
59         response.writeHead(200, {'Context-Type': 'text/plain'});
60         if ( Array.isArray(input) ) {
61             response.write(encodeURIComponent(input[0]));     // ココ！
62         } else {
63             response.write(encodeURIComponent(input));        // ココ！
64         }
65         response.end();
66     } else {
67         sendAPIError(response, '(E-01)Input was not specified.');
68     }
69 }
```

どうやらサーバー側で URL を処理するときにエンコードされた文字が解釈されてもとに戻ってしまうらしく、送信する前にもう一度エンコードする必要があるようです。次に実行結果を示します（図 12)。いやぁ長かった（原因究明が)。

図 12: 実行結果。正常に全角文字が表示されている

さて、ここまででサーバーとの文字列の送受信の仕方と全角文字の扱い方を確認し実装しました。次はこれを使ってサンプルゲームの機能を増やしてみたいと思います。

応用してみる

それでは実際にダウンロードコマンドを応用した機能を基本システム上に構築してみましょう。オンライン＋ゲームといえば何でしょうか？ やはりオンライン対戦でしょう。ということでその実装を……?

> いやまて、そもそも基本システムの戦闘システムってどんな仕組みになってるんだ？ データベース内の一時ステータスを逐次送受信すれば良いんだろうか？ それでも状態異常の回復などは乱数で管理されているかもしれないじゃないか。もしそうだったらその管理はどこでやるんだ……？

こうして polaris は考えることをやめたのであった。っておい！ こんな中途半端なところで終わったらみんな怒りますよ！ もっとしっかりして k (ry
はい、というわけで別のものを考えてみました。

どんなものをを作るか？

いくつかのゲームから送られたメッセージをサーバーが保存して、それらのメッセージのうちランダムに選ばれたものをゲーム側で取得できるというようなシステムを考えてみました。もっと簡単に言えばこれまでにサーバーに送られた任意のメッセージを送り返すようなものです。どのような通信が行われるのか図にしてみました（図 13)。

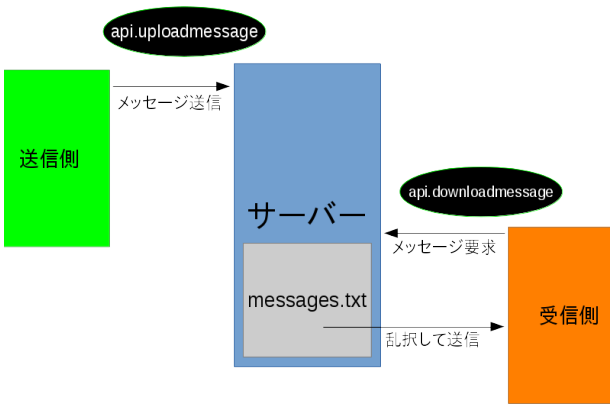

図 13: 上から順番に行われる

仕組み

必要な API を考えるためにもう少し上の機能を考察してみます。サーバーにメッセージを送るための API が必要そうです。これを **api.uploadmessage** としましょう。これにメッセージをパラメータとして渡すことにしましょう。ランダムなメッセージをサーバーから受信する API も必要です。これを **api.downloadmessage** としましょう。以下にこれらの API の仕様をまとめてみます。

api.uploadmessage?mes=message サーバーにパラメータ **mes** に指定されたメッセージを新たに記録します。メッセージは（サーバーのプログラムのあるディレクトリ）/userdata/messages.txt に保存されます

api.downloadmessage サーバーから無作為に選ばれたメッセージを受信します

それでは実際に実装してみます。以下に新しく作ったサーバーのソースコードの一部[*10]とウディタのイベントを示します（図 14）（コード 3、4）。

ソースコード 3: server2.js：22 行目付近

```
22      } else if (pathName === '/api.reflect'){
23          // API reflect: パラメータ d (api.reflect?d=(ココ！))にセットされた内容をプレーンテキスト形式でそのまま返す。以下その処理
24          var queryObject = qstring.parse(url.parse(request.url).query);
25          var input = queryObject.d;
26          api_reflect(response, input);
27      } else if (pathName === '/api.uploadmessage') {    // ココから
28          // API uploadmessage: パラメータ mes にセットされたメッセージを記録する。セミコロンは除去される
29          var queryObject = qstring.parse(url.parse(request.url).query);
30          var mes = queryObject.mes;
31          api_uploadmessage(response, mes);
32      } else if (pathName === '/api.downloadmessage') {
33          // API downloadmessage: 保存されているメッセージの中からランダムに一つ返す
34          api_downloadmessage(response);
35      } else {                                            // ココまで
36          // 存在しないAPIが指定されたら404(Not Found)を返す
37          send404(response);
38      }
```

[*10]server.js に機能を追加しただけのものなので全体のソースコードは載せないことにした。載せた部分の追加の他にもファイル操作用のパッケージを読み込む行を増やしている。

ソースコード 4: server2.js：81 行目から

```
81 // ココから追加したAPI群
82 // API uploadmessage
83 function api_uploadmessage(response, mes) {
84     // ここら辺の動作は API reflect と似ている
85     if ( mes !== undefined) {
86         if ( Array.isArray(mes) ) {
87             message = mes[0];
88         } else {
89             message = mes;
90         }
91         // セミコロンが入っていると困るので除去
92         message = message.replace(new RegExp(';', 'g'), '');
93         // ファイルに書き込む
94         fs.appendFile('./userdata/messages.txt', encodeURIComponent(
               message) + ';', (err) => {
95             if (err) {
96                 sendAPIError(response, '(E-03) Write failed: messages.
                       txt');
97                 return;
98             } else {
99                 response.writeHead(200, {'Context-Type': 'text/plain
                       '});
100                response.write('OK');
101                response.end();
102            }
103        });
104    } else {
105        sendAPIError(response, '(E-01)Input was not specified.');
106    }
107 }
108
109 // API downloadmessage
110 function api_downloadmessage(response) {
111     // セミコロン区切りのデータを読み込む
112     fs.readFile('./userdata/messages.txt', 'utf8', (err, data) => {
113         if (err) {
114             sendAPIError(response, '(E-02) Read failed: messages.txt');
115             return;
116         } else {
117             // セミコロンで文字列を分ける
118             var mesarray = data.split(';');
119             var nelements = mesarray.length;
120             // 末尾のセミコロンも処理されて、配列の最後に空白文字が入っ
                   てしまうので要素数から 1 減らしている
```

```
            var randomid = Math.floor(Math.random() * (nelements - 1));
            response.writeHead(200, {'Context-Type': 'text/plain'});
            // すべてのメッセージのうちから一つだけ送る
            response.write(mesarray[randomid]);
            response.end();
        }
    });
}
```

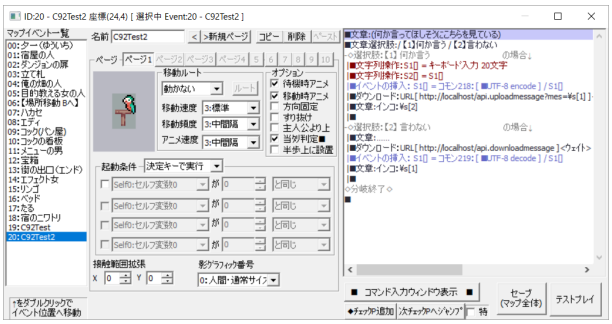

図 14: api.reflect の時のを応用した感じ

というわけで、できたウディタのゲームフォルダをコピーで２つにして[*11]それぞれから通信してみました。結果がこちらです（図 15、16）（コード 5）。

図 15: 左がアップロード側、右がダウンロード側。上部の端末に要求された API を表示している

[*11]同じ PC 上で動かすには片方のゲームタイトルを変更しておく必要がある。またウディタのバージョンが 2.20 より低くてもいけない。

図 16: 1 つめの送信なので片方から先ほど送られたメッセージが表示されている

ソースコード 5: messages.txt：パーセントエンコードされたメッセージが保存されている

```
5000%E5%85%86%E5%86%86%E6%AC%B2%E3%81%97%E3%81%84%EF%BC%81;
```

おわりに

いかがでしたでしょうか。もしこの記事が何かのアイディアの種になったならば幸いです。また、私は京都工芸繊維大学の学生なのですが、KMC のメンバーの方々は大学の枠を超えて暖かく迎えてくださいました。この場を借りてお礼をしようと思います。ありがとうございました。それではまた。

参考文献

[1] WOLF RPG エディター公式サイト, http://www.silversecond.com/WolfRPGEditor/

[2] node.js 公式サイト, https://nodejs.org/ja/

[3] 文化庁改定常用漢字表, 33 ページ, http://www.bunka.go.jp/seisaku/bunkashingikai/sokai/sokai_10/pdf/kaitei_kanji_toushin.pdf

[4] MDN JavaScript リファレンス, https://developer.mozilla.org/ja/docs/Web/JavaScript/Reference

＊ MDN JavaScript リファレンスはコーディングの際に大いに参考になったのでここで紹介した。

作ったゲームのテストプレイについて

hideya

あいさつ

こんにちわ、KMC8 回生の hideya です。訳あってまだ現役大学生です、うふふ。KMC に入部してからゲーム制作を始めたので、アマチュアゲーム製作歴も 8 年目です[*1]。

この記事では、私がゲーム制作を続ける中で培ってきた技術の 1 つである「ゲームのテストプレイを観察し、ゲームをより良くする」技術について書いてみます。

なお、この記事は KMC の 2017 年春合宿で行った講座のショート版記事です。元となった講座のスライドは SlideShare にアップロードされている[*2]ので、この記事を読んで面白いと感じれば（もしくは、文章よりもスライド形式で見たいと思うならば）そちらもどうぞ。私が過去に行ったテストプレイの実例などが載っています。

本記事におけるテストプレイの定義

本題に入る前に、先程何気なく使った「テストプレイ」という言葉を定義[*3]しておきます。本記事においてテストプレイとは「開発中のゲームを**開発者以外の者**にプレイさせる」行為を指すものとします。開発者自身によるプレイもテストプレイの範疇に含む事がありますが、本記事においてはそれらは取り扱わない[*4]ものとします。あしからず。

テストプレイの意義

それではまず、テストプレイを実施する事で得られるものを考えてみましょう。話のネタ、人生経験、外部への進捗アピール……様々なものが考えられますが、最も大事なのは「ゲーム内容への**客観的な**フィードバック[*5]」であると私は主張します。

[*1] その間に外部へ発表した作品数は 8 作品。シューティング系統のゲームが多め。

[*2] `https://www.slideshare.net/KMC_JP/hideya`

[*3] 2017 年 3 月の時点では、Google 検索した限りではこの言葉のちゃんとした定義が出てこないのである。

[*4] これらの行為の意義を軽視する意図は無いので、誤解なきよう。

[*5] 様々な意味のある言葉であるが、本記事においては概ね「行為の結果を確かめるための情報となる手応え、ならびにそれを行為者へ伝える行為」という意味。

ゲーム開発とはとどのつまり、開発者の脳内にあるゲーム構想を実装していき、その構想による楽しさ等をプレイヤーが体験できる状態にする、というアウトプット作業であると言えます。その為、アウトプットする主体である開発者は、自分の伝えたい事が本当に相手（この場合はゲームのプレイヤー）に伝わっているのか？　という事をある程度は[*6]気にしなければいけません。そのためにはやはり、下書きの状態でもよいので実際に誰か他人（アウトプットする側ではない立場の人）に試しに見せてみて、どう感じられたかを実際に確かめてみるのがとても有効となります。文章やスライド発表と同じですね。

テストプレイの実施によって修正しなければならない点が顕になるのは確かに苦しいことではありますが、ゲームのリリース後に判明するよりは遥かにマシです。作るゲームをより良いものにしたいのであれば、テストプレイは避けて通れない工程であると断言できます。

テストプレイで確認したい事

さて、では具体的にどのようなフィードバックが得られるのでしょうか？ それは開発の状況によって様々です。軽く列挙してみると、

- 想定した楽しさ[*7]が表現できているか
- 説明や機能が十分か
- 難易度やバランスは適切か

といったものが考えられます。どのフィードバックが必要なのかは開発の段階によって様々なので、そこは各部分の担当者やマネージャーが頑張って考えたり感じたりして下さい。

テストプレイを実施する前の準備

それでは実際にテストプレイを実施するにあたって、どのような準備が必要かを考えてみましょう。ゲームの開発はテストプレイができる段階まで十分に進んでいるものとします[*8]。

まずは前項で触れた通り、どのようなフィードバックを得たいかを考えます。新しく作った機能やステージ等に焦点を絞ると考えやすいでしょう。

テストプレイヤーの用意も必要です。そこら辺に居た協力してくれそうな人を捕まえるので大抵は十分ですが、得たいフィードバックによっては特定の人を狙い撃ちする必要があるかもしれません。シューティングゲーム初心者に向けたゲームを開発しているのであれば、シューティングゲーム初心者の視点からのフィードバックを得たいのは当然です。いずれにせよ、テストプレイヤーが（とりわけ、テストプレイさせるジャンルについて）どれほどゲームに精通しているか、腕前はどれくらいか、などは確認しておきましょう。あまりゲームを遊ばない人

[*6]「表現者がどの程度ワガママであるべきか」というのは深淵なる問いかけであるので、その方面は今は掘り下げない方針で。

[*7]例えばパズルゲームで「オブジェクトを纏めて大量に消す爽快感」を感じてもらえているか、といったもの。

[*8]簡単に書きましたが、実現するのはそう簡単ではない。

が言う「簡単だね」と、凄腕ゲーマーが言う「簡単だね」は意味合いが全然異なるのですから。

また、テストプレイヤーにどれだけ情報を与えるのか、についても考えるべきです。得たいフィードバックによっては、ある程度のゲームの説明をテストプレイヤーにしておいた方がよい場合もありますし、逆に説明しない方がよい情報が存在する場合もあります。分かりやすい極端な例を挙げると、「作成したチュートリアルが十分な操作説明をできているか」を確認したい場合には、テストプレイヤーにあらかじめ操作の仕方を説明してはいけません。当たり前ですね。「このステージのボスの強さは、ボス到達までの難易度と比較して適切かどうか」を確かめるのであれば、操作説明はあらかじめしておいて構わないでしょう。それに加えて、確かめたい内容を予めテストプレイヤーに伝えておけば、テストプレイヤーもそのバランスを意識してプレイを行い、その部分の感想をより明確にフィードバックしてくれる可能性が高まります。説明する内容によってテストプレイヤーの意識へ与えるバイアスについても考えておくことで、より有意義なフィードバックを引き出すことさえも可能なのです。

この辺りまで考えれば、テストプレイを実行する時でしょう。もっとも、この説明では記事にするために事前にしっかり考えていますが、実際には毎回ここまで厳密に考える必要は無く、「部室に暇そうな奴が居たからとりあえず触らせてみるか」程度の動機でホイホイ気楽にやって良いものでもあります。あれこれ考えるのにエネルギーを浪費するよりも、とりあえずデータを集めて開発を進める方が有意義な事は多々あります。そもそもテストプレイを開催する経験が無ければここまで考えるのは難しいので、経験を積んでから色々考え出すくらいで丁度良いとさえ私は考えています。あくまでも「考えようと思えばこれだけ考えることができる」という一例を示しただけ、という程度に受け取って下さい。

テストプレイの時きたれり

さぁ、実際にテストプレイをさせる時が来ました。気合を入れてテストプレイを観察しましょう。テストプレイ観察の際に主に見るのはゲーム画面ですが、それ以外の部分からも有意義な情報が得られることがあります。テストプレイヤーの表情やリアクションからはプレイヤーの感情を確認できますし、操作する手元の様子からは操作への適応度合いが分かります。可能な限りでよいので、様々な部分に気を配るようにしましょう。とはいえ、やはり沢山の場所を観察するのは大変であり現実的ではないので、プレイ後にテストプレイヤーに「あの場面は苦しかった？」「あの操作はしやすかった？」などと質問したり、或いはテストプレイヤーの方から感じたことを申告してもらう[*9]のが良いと思います。ただプレイを見て終わりではなく、そこから更なるフィードバックを能動的に引き出していきましょう。また、前節で考えた「今回のテストプレイで得たいフィードバック」について注意を向けるのは当然大事なのですが、だからといって視野狭窄にならず幅広い視野でプレイを観察する事も大事です。難しそうですが、難しいです。頑張りましょう。

[*9] これにはテストプレイヤー側に注意力や言語化能力が必要なので、アテにできるかどうかは場合に依るのではあるが。

余談に近い話ですが、実験データには外れ値が付き物です。あるテストプレイヤーのプレイ結果や意見が真理である保証はどこにも無いので、時にはそれらを無視するのが正しい事もあるという事実を忘れないでください。

こうしてテストプレイを実施し、その結果からフィードバックを得る事によって現時点での問題点や問題無い点[*10]を確認することで開発の進行度合いを正確に測ることができ、よりスムーズで実りのある開発が行えるようになります。そうです。テストプレイは開発の中間地点にすぎません。得られたフィードバックを活かすために必要な時間を考慮して、計画的なテストプレイの実施を心掛けましょう。

おわりに

ここまででテストプレイについての解説をしましたが、いかがでしょうか。きっと今頃、読者の皆さんもテストプレイを観察したくてウズウズしてきているのではないかと思います。他の様々なスキルと同様に、テストプレイの実施によってゲームを改善するスキルもまた、経験を積めば上達するものなのです。どんどんテストプレイを実施して、どんどん上達していきましょう。

それでは皆様、よきゲーム開発ライフを。BE ON YOUR GUARD.

[*10]欠点を確認するフィードバックも大事だが、良い点を確認するフィードバックも同様に大事である。既に十分良く出来てる部分を必要以上に調整して時間を無駄にする事を防げ、何よりも製作に自信が持てて今後のモチベーションを保つのに大いに役立つ。

APOLLO 出展記

tron

ごきげんよう。KMC2 回生の tron です。KMC は 2017 年 6 月 17 日から同 19 日まで開催されたネット上の音楽即売会 APOLLO A-06 に出展しました。これはその記録です。

APOLLO とは

APOLLO は、pixiv 株式会社の運営するネットショップサービス Booth 上で開催される、同人音楽作品の即売会イベントです。KMC が参加した A-06 で (文字通り)6 回目の開催となります。詳しいことはここでは割愛しますので、さらに知りたい方は公式サイト[*1]をご覧ください。

ことの始まり

5 月 2 日

部員の opesan が APOLLO の情報を持ってきました。「夏コミ前に APOLLO でアルバム作るのもいいかも」とのこと。それを聞いて私がやる気になり、翌 3 日に参加を仮決定しました。

進む準備

5 月 15 日

出展が例会で承認され、参加が正式決定。またこの頃、APOLLO 用に作ろうとしている音楽集とは別に、「KMC 40th Anniversary Compilation Album」(以下、40thACA) を APOLLO で販売する話が持ち上がります。KMC は 2017 年に 40 周年を迎えるということで、そのためのテーマソングを作る企画がありました。それが少し形を変えて、皆で 40 周年記念のアルバムを作ろう、ということになったのが 40thACA です。諸事情で CD での配布は難しいかな……？ということで APOLLO で一緒に配布しよう！　ということになりました。

[*1] https://booth.pm/apollo/

5 月 22 日

音楽集のタイトルが「KMC Music Collection vol.1 隣り合わせの♪と青春」(以下、KMCvol1)に決定。元々 KMC 内で募集していたのですが締切直前になっても案が出ず、私が考えるしかない、と思って「KMC Music Collection」[*2]という名前を考えた所、いつの間にか誰か[*3]が「隣り合わせの♪と青春」という名前を出していたので、合体させてこの名前になりました。

6 月

6 月 1 日

40thACA のアートワークが完成。サイバー感溢れるクールな 1 枚です。

図 1: 作:maztani

[*2]再帰的頭字語。頭字語いいですよね。
[*3]未だ正体不明。

6 月 3 日

KMCvol1 の、6 月 4 日には 40thACA の曲提出締切がやってきました。正直厳しい日程でしたが、多くの曲が集まりました。

6 月 11 日

KMCvol1 のアートワークが完成。タイトルを反映した楽しげなイラストです。

図 2: 作:kurimotz

また、40thACA の曲順も決定。曰く、「アコースティック系→ピコピコ→ EDM みたいな流れのつもりです」とのこと。

6 月 12 日

KMCvol1 の曲順が決定。こちらは曲の繋がりがしっくりくるかどうか、を基準に決めました。

ラストスパート

6 月 13 日

APOLLO への登録を開始。値段の決め手がなくてちょっと苦労（300 円にしました）。

6 月 14 日

重大な問題が発覚。40thACA を APOLLO で頒布することが伝わっていない人が何人もいたのです。Slack でも会話にさらっと出ただけでちゃんと伝えることを忘れていました。具体的に言うと、部内の Wiki に書くのを完全に忘れていた。それでいて伝えた気になっていたのです。実は 40thACA と KMCvol1 の両方に同じ曲を出した人もいたのですが、そうした人から「そうと知っていたら被らせなかった」「被りを解消したい」といった声が上がり、曲を整理することに。結局、当初は両方共 15 曲あったのが 11 曲になりました（ってことは、8 曲も被っていたということ……）。

6 月 16 日

被りの件もなんとかなり、登録も無事完了。あとは本番を待つのみ。

本番

6 月 17 日

APOLLO が開幕。また、この日は KMC40 周年を祝うイベント「KMC 40th Anniversary Conference」が開催されました。40thACA に本来入るはずだった曲を会場で BGM として流し、来場者から好評を得ました。また、この会場で APOLLO での頒布の宣伝をしたり、終了後に行われた懇親会の会場で LT[*4]と称して CM をしたりしました。

6 月 19 日

APOLLO が閉幕。売上はまあ、そんなすごくはありませんでしたが、初めてにしてはいいのではないか、という感じでした[*5]。

[*4] Lightning Talk の略。5 分程度の短いスピーチ。
[*5] どのくらいが普通なのかわからないのでなんとも言えないのですが……

まとめ

APOLLO はもちろん、音楽だけでの頒布自体初めての試みということもあり、いろいろと上手く行かなかったところもありましたが、結果的には参加してよかったと思います。次回も参加を考えていますので、その時はぜひ 1 度聴いていただければな、と思います。

格闘ゲームに学ぶ親切な入力設計とプレイヤーの悪用例

asragi

自己紹介

- asRagi（らぎ）
- 京都大学工学部物理工学科所属
- DTM、ゲーム製作など
- 部誌 vol.10 表紙イラスト担当

まえがき

2D 格闘ゲームというリアルタイム性・アクション性を備えたゲームでは、プレイヤーが快適に操作できるよう入力周りに数多くの工夫が凝らされている。高速で展開するゲームに対しプレイヤーは素早く入力を行う必要があり、そうした場合に、正確性を欠くことは容易に想像ができる。以下では、これらの数多くの工夫のうちの一部の設計例を紹介する。
また、ゲームを遊ぶプレイヤーは大きな熱意と好奇心でもってゲームの「研究」を行うことがある。特に格闘ゲームをプレイする「格ゲーマー」はその傾向が強く、開発者側から用意されている選択肢を利用し尽くすだけに留まらず、共通システムの隙やバグまでもを利用して日々勝利に近づくための研究を続けている。前述した開発者側が行った実装例と共に、こうした熱心なプレイヤーによって応用、もとい悪用された例を紹介していく。
技術系のサークルのため学びを重視した体裁となっているが、「へーそんなテクニックあるんだふーん」くらいの気楽さで読み進めてもらうこともおススメする。

同時押しの設計

格闘ゲームは複数のボタンの同時押しによってキャラクターに異なる動作をさせるというシステムを搭載していることが非常に多い。例えば波動拳を “↓↘→＋P（P:パンチボタン）” で入力すると通常の波動拳が発動する一方で、“↓↘→＋PP（P ボタン二つ同時押し）” と入力する

図 1: 強パンチと弱パンチの入力例

と強化された波動拳が撃ち出される、などが挙げられる。
ここで問題となるのは、「高い精度で同時押しの入力を行うことは難しい」ということである。特に近年の多くのゲームは 60fps で処理が行われているが、複数のボタンを同時に押すタイミングのズレを 60 分の 1 秒未満に抑えることは案外難しく、慣れたプレイヤーでも中々安定しない。
図 1 のように、強パンチボタンと弱パンチボタンが 1F ズレて入力されたとする。1F[*1]は非常に小さい時間であるため、プレイヤーが同時押しするつもりで入力しない限りはこのような入力はまず受け取ることはなく、これはプレイヤーが同時押しのつもりで入力したものとして解釈すべきである。このとき、同時押しを素朴に「同一のフレームに複数のボタンが押されたとき」で検出していると、この入力は同時押しとして検出されない。

故に、近年の多くの格闘ゲームでは同時押しの入力に関して次のような実装が行われている。異なるボタンが 1F のズレで入力された場合には、図 2 のように後に押されたボタンを「その前のフレームに押されたボタンとの同時押し」として受け取り、「必要」であれば前にボタンを押されたタイミングで開始してしまった動作（この例では強パンチボタンの攻撃）をキャンセルし、同時押しの動作に切り替える。ここで言う「必要」とは、その同時押しに対応する動作が存在する、ということである（同時押しで特殊な動作をしないボタンの組み合わせの場合、このような処理を行う必要が無い）。

後に押されたボタンを同時押しとし、前に押されたボタンの動作をキャンセルすることで、同時押しの入力猶予を広げている。

同時押しの設計を利用したテクニック

同時押しの 1F のズレを許容する設計によって生まれた代表的なテクニックを紹介する。

[*1] 1F=1 フレーム。60 分の 1 秒に相当。

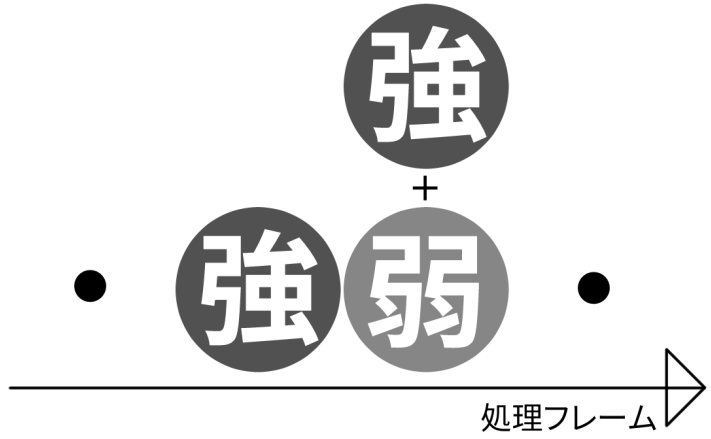

図 2: キャンセルの例

ずらし目押し（通称:辻式）

格闘ゲームでは、コンボ（タイミングよくボタンを押すことで攻撃から連続で相手に攻撃を当てること）を成功させるなどのために、自身の動作の硬直が解除されてから最速でボタンを押す必要がある場面は多い。この硬直が解除されてからの最速入力を安定して実現するためのテクニックが辻式と呼ばれるテクニックである。

前述の例のように、強パンチの 1F 後に弱パンチが入力されたとする。仕様により、この入力は「強パンチの 1F 後に弱パンチ + 強パンチの同時押しが為された」と解釈される。

このとき、「弱パンチ + 強パンチ」の入力に特殊な行動が割り当てられていない場合、同時押し時のボタンの優先度の実装によって強パンチあるいは弱パンチが行われる。

つまり「強パンチの 1F 後に強パンチ（強パンチと弱パンチの同時押し）が入力された」という状態になり、2F 間に渡り連続して強パンチの入力が行われたことになるである。

2 連続で入力された強パンチのどちらかが期待される入力のタイミングであれば良いため、これにより入力の成功猶予が 1F 延長される。

ストリートファイター 4 などをはじめとする多くのプレイヤーがマスターしているテクニックであり、この技術を使いこなし高難度のコンボを実戦で成功させるプレイは多くの観客を沸かせるものである。

投げ抜け仕込み攻撃（ボタンの入力優先度の応用）

投げ抜け仕込み攻撃は同時入力時のズレに対する設計によって生まれたテクニックではないのだが、後に紹介するテクニックでも重要な「投げ」という行動についての説明も兼ねてここで紹介しておく。

格闘ゲームの攻防において「投げ」という行動は攻撃に対してガードを固める相手を崩す主な

手段であり、非常に重要である。格闘ゲームにおける「投げ」は次に列挙する仕様を採用している場合が多い。

- 「投げ」はガードすることができない。
- 「投げ」は「投げ」の入力によって相殺することができる（主な回避手段）。
- 空中にいる（と判定される）状態の相手を投げることはできない。
- 「投げ」を行うためには立ち状態である必要がある（しゃがみ状態では出せない）。

投げ抜け仕込み攻撃は以上の仕様を利用したテクニックである。
ストリートファイターでは「投げ」の入力は「弱パンチ＋弱キック」が割り当てられている。また一方で、ボタン同時入力時の優先度は「弱パンチ＜弱キック＜中パンチ＜中キック＜強パンチ＜強キック＜投げ＜コマンド必殺技など」として実装されている（たとえば、弱キックと中キックを同時に押すと中キックが出る）。
ここで、しゃがみ状態のまま「弱パンチ＋弱キック＋中キック」と同時押ししたとする。この時、しゃがんでいるので投げは出ず、上記のボタン入力優先度より中キック攻撃が繰り出される。
しかし、もしこの入力を行う直前に相手から「投げ」によって掴まれていた場合はどうなるか。「投げ」の入力は「弱パンチ＋弱キック」であり、「投げ」に対しては「投げ」の入力で相殺することができることを考える。プレイヤーの入力はしゃがみ状態で「弱パンチ＋弱キック＋中キック」と同時押しであり、「弱パンチ＋弱キック」が含まれているため、相手の投げに対して投げ相殺となり、「投げ」の仕掛けを回避することができるのである。
つまりこの入力によって、相手に「投げ」で掴まれていた場合は投げ相殺による回避、そうでない場合は中キック攻撃が出る、というように自動的にベターな行動をとることができるのである。
このテクニックが投げ抜け仕込み攻撃と呼ばれるものである。
中キックのような攻撃ボタンの代わりに、必殺技やジャンプなど他の行動で置き換えることもでき、幅広く使われているテクニックである。

移動投げ

一部の技は、最初にある程度踏み込んでから蹴りを出す、のように攻撃モーションの前半部分でキャラクターの位置が移動する。
勘の良い方ならこの時点で何か嫌な予感のようなものを感じ取られるかもしれない、あるいは既におおよその察しがついているかもしれないが、解説させていただく。
あるキャラクターの強キックが仮に前述のように前進してから攻撃を行うタイプの技であるとする。
このとき、「強キックの 1F 後に弱パンチと弱キックの同時押し」と入力すると、強キックの最初の 1F だけ行った後に強キックをキャンセルして投げが出るのである。
たったの 1F であるが、強キックによりキャラクターが前進移動してから投げるため、相手が

「投げ」を警戒しない間合いから仕掛けることができ、非常に強力である。
この強キックに相当する技によってはキャラクター１人分ほども前進する場合もあり、キャラクターの個性として評価されることもあるほどである。

ジャンプ移行フレーム

格闘ゲームの醍醐味の１つに、コマンド入力によって必殺技を繰り出すことが挙げられる。コマンドの例として、波動拳を繰り出す “↓↘→＋P” や、昇龍拳を繰り出す “→↓↘＋P” などが挙げられる。
ところで、多くの格闘ゲームではレバーを上に入力することによってキャラクターはジャンプするが、入力した途端にキャラクターが浮く（＝空中状態になる）わけではない。レバーを上に倒してから数フレームほどの無防備な「ジャンプ移行フレーム」を経て、空中状態へ移行する。これは入力した直後に空中状態へ移行してしまうと、相手の攻撃に対してガードでなくレバーを上に倒し続けて空中状態で攻撃を受けた方が被害が少ないという場面が多くなるためであると考えられる（システムに依るが、空中で攻撃を受けると地上で受けた場合と比べて被害が小さくなる場合が多い）。
このジャンプ移行フレームには次のような特徴がある。

- ジャンプ移行中は地上立ち状態として判定される。
- ジャンプ移行中は「投げ」攻撃を受けない。
- ジャンプ移行中に必殺技の入力があった場合、ジャンプ移行をキャンセルして必殺技を発動する。

ジャンプ移行中に「投げ」を受けないのは、プレイヤーが「投げ」を回避するためにジャンプ入力を行ったにも関わらず投げられてしまうという事態の発生を避けるためであろう。
ジャンプ移行中に必殺技の入力を受け付ける実装は特に注目すべきである。このシステムの存在によって、例えば波動拳を “↓↘→↗＋P” のように誤ってレバーを斜め上まで動かしてしまった場合にもジャンプ移行中フレームがキャンセルされることで波動拳が正しく発動される。
同時押しの実装と共通して言えることだが、プレイヤーのこうした入力の若干のズレを許容する、遊びをもたせた設計がよりよいゲーム体験を生み出すためには必要不可欠である。

ジャンプ移行フレームを利用したテクニック

上記のジャンプ移行フレームの仕様を利用したテクニックを一部紹介する。

投げ無敵付与

波動拳のようなコマンド必殺技を “↓↘→＋P” ではなく、敢えて “↓↘→↗＋P” と入力し、一瞬のジャンプ移行フレームを経てから必殺技を繰り出すことで必殺技発動前に「投げ」に対して無敵（攻撃を受けない状態）になるテクニックである。通常のキャラクターはジャンプ移行フ

レームが 2〜5 フレーム程度であり実用性にやや欠けるが、一部の重量級キャラクターなどはジャンプ移行フレームが 8 フレーム程度ある場合があり、十分な威力を発揮する。

ジャンプキャンセルキャンセル必殺技

多くの格闘ゲームの一部の攻撃技は、攻撃が相手に当たったタイミングで必殺技コマンドを入力することで、攻撃技の後隙をキャンセルし必殺技を発動することができる。この仕様によってコンボを狙うのが近年の格闘ゲームの主流である。同様に、攻撃が相手に当たったタイミングでジャンプの入力を行うことで、攻撃技の後隙をキャンセルしジャンプに移行することができる技も存在する。これを「ジャンプキャンセル」と呼び、相手を打ち上げるような攻撃からジャンプキャンセルを経て空中で追撃をする、または後方にジャンプをして後隙を誤魔化す、などといった戦略を取ることができる。

これらの技の中には、「必殺技キャンセルができないが、ジャンプキャンセルはできる」といった性能の攻撃技が存在する。本来なら必殺技キャンセルができないのでコンボに利用することはできないのだが、「攻撃を当てる → 攻撃が当たった瞬間にジャンプ入力（ジャンプキャンセル）する → ジャンプ移行フレーム中に必殺技コマンドを入力」とすることによって、これら攻撃技から必殺技を繰り出し、コンボにすることができるのである。

仕様の裏を突いたテクニックにように思われるが（おそらく元々は実際そうだったのだが)、近年の格闘ゲームにはこのジャンプキャンセルキャンセル必殺技のテクニックの使用を前提として調整されているキャラクターも存在する。前述の移動投げや後述の無敵付与にも言えることだが、格闘ゲームは元々バグであったものをプレイヤーがテクニックとして利用し、それを開発者側が正式なテクニックとして後継作に採用する例が非常に多いという特徴がある（キャンセル必殺技も元々はバグであった)。

バクステ無敵付与

ほぼスレイヤー（ギルティギアシリーズのキャラクター）限定のテクニックであるが、おまけ程度に紹介する。

多くの格闘ゲームのキャラクターはレバーを後ろに素早く 2 回倒すことでバックステップ動作を行い、相手から距離を取る。このバックステップ動作には相手のあらゆる行動に対し無敵になる時間が与えられていることが多く、相手の攻めから脱出する強力な手段の 1 つであり、このバックステップを読んでその後隙を狙うといった攻防も繰り広げられる。

スレイヤーは他のキャラクターと異なり、**バックステップ動作をジャンプでキャンセルすることができる**という特徴を持つ。

これを利用して、「バックステップをジャンプでキャンセルする → ジャンプ移行フレームを必殺技でキャンセルする」とすることで、バックステップから必殺技を出すことができるのである。バックステップ動作中は無敵であるため、必殺技が繰り出される直前まで無敵状態で相手の攻撃を退けながら必殺技を繰り出すことができる強力なテクニックであるが、更に恐ろしいことには、**バックステップ動作をジャンプでキャンセルしても無敵時間だけはキャンセル後も**

そのまま残るという**バグ**が存在していたため、スレイヤーはあらゆる必殺技を無敵状態で繰り出すことができてしまっていたのである。
しかしこれも例に漏れず、後続作では無敵時間を調整するなどを経てスレイヤーにのみ引き続き正式に搭載されている。
余談だが、筆者は 2017 年 7 月時点で未だにこのテクニックに対する明確な攻略法にたどり着けておらず、スレイヤーと対戦する度に泣かされていることを記しておく（情報お待ちしております)。

あとがき

今回ここで取り上げたテクニックはほんの一部にすぎず、格ゲーマーは日々開発者の実装と闘い、さらに新たなテクニックを生み出し続けている。
これを読む皆様が開発者側として読んでいたかゲームプレイヤー側として読んでいたかはわからないが、開発者の皆様にプレイヤーの入力を素朴に受け取るだけではなく、常にプレイヤーの意思決定に基づいた処理を実現できるように工夫する必要性やそうした工夫が（良くも悪くも）プレイヤーに与える可能性について少しでも考えて頂けたなら、またプレイヤーの皆様には実装のスキマをついて新たなテクニックを考え生み出す楽しさを感じて頂けたなら幸いである。

おまけ：asRagi 式ハンバーグのレシピ

材料

ハンバーグを 7 つまとめて焼いて冷凍、食べるたびにハンバーグソースだけ作るという戦略で運用しているレシピのためハンバーグは 7 つ分、ソースは 1 人前の分量で記載する。

■ハンバーグ（ハンバーグ 7 つ分）

- 牛豚合いびき肉 500g（冷蔵庫に入れておく）
- 生卵 2〜4 個
- 牛乳 50ml
- たまねぎ 中 3 つあるいは大 2 つ
- 米（炊いたもの） お茶碗 1 杯分
- 氷 2〜4 個
- 牛脂 2〜3 個
- 塩 7g

■醤油ハンバーグソース（ハンバーグ 1 つ分）

- 醤油 大さじ 3
- みりん 大さじ 1

- おろし生姜 小さじ 0.5 未満
- 有塩バター 10g

・調理手順（ハンバーグ）

■みじん切りにしたたまねぎを加熱する　たまねぎは全てみじん切りにする。加熱はフライパンでやると大変なので全体に熱が通る程度にレンジでチンだけしている（フライパンでやってもよい）。

■十分大きいボウルに生卵・牛乳・加熱したたまねぎ・米・氷・刻んだ牛脂を入れてよく混ぜ合わせる　ハンバーグのタネを作る際には肉の温度を焼くまで上げないことが重要である。氷を入れることでたまねぎと米の温度を下げつつ水分を補う。
牛脂は刻んでから入れる。氷も砕いた方がいいが、砕かなくても問題は無い。
米はパン粉のかわりである。米の食感が気になる場合は米をつぶしてから入れると良い（かもしれない）。

■ボウルに挽き肉と塩を入れて粘り気が出るまでよく混ぜ合わせる　体温で肉の温度が上がらないようヘラなどで混ぜ合わせるとよい。

■ハンバーグのタネを手のひらサイズに丸めて油を多めに入れた冷たいフライパンに並べる
上記の分量でだいたい 7 つのハンバーグを作ることができるが、全てフライパンに並べるのは無理なので、2 回に分けて焼く。
油はサラダ油でもオリーブオイルでもよいが、多めに使うのでサラダ油がいいかもしれない。

■弱火でじっくり加熱する　表面を強火で焼いてから弱火で中まで加熱するやり方は肉汁を閉じ込めることを重視した焼き方であるが、asRagi 式では最初から弱火でじっくり焼くことで、肉のくさみを外に逃がすことを重視している。
肉汁のようなもの（灰汁）が多く出てきたらキッチンペーパーでふき取り、適宜フライパンに油を足す。
中まで火が通ってさえすればよいので、焼き加減は各自の判断に任せたい。

■焼きあがったハンバーグをお皿に取り出して完成！　必要ならラップに包んで粗熱を取ってから冷凍庫に入れる。

・調理手順（醤油ハンバーグソース）

■フライパンの油をキッチンペーパーで軽くふき取る　前述のとおり、肉のくさみを取るための弱火なので、肉汁は利用しない。

■醤油・みりん・おろし生姜をフライパンに入れ、弱火で軽く沸騰するまで混ぜ合わせる　焦げに注意すること。

■バターを溶かしてもう一度軽く沸騰させたらソース完成！　ハンバーグにソースをかけたら写真を撮ってツイッターで自慢するとよい。

Slack bot でスロットを作ってみた

yu3

はじめに

こんにちは。KMC6 回生の yu3 です。インターネット上では主に yu3mars と名乗っています。KMC に入部してからは、特に作りたいものが思いつかない時期を乗り越え、主に内輪での活動をしていました。最近になって外部へのアウトプットをして他人の反応を観測する遊びを始めました。

この記事では、部内 Slack でスロットができるようにする bot を作ってみた話をします。

bot 作成のきっかけ

KMC ではチャットツールとして Slack が使われています。また、部員による bot 作成が流行しています。具体的な bot の例は前回の部誌で記事になっているので、興味がある方はそちらも読んでみてください。

そんな環境でチャットをしていたある日、私はスロットをする bot に夢中になる部員を発見したのです……。

図 1: スロットに夢中になる部員

プロトタイプ作成

このスロットを改良すれば、部員に遊んでもらえて承認欲求が満たせるのでは？ と思った私は、早速 bot の作成に取り掛かりました。今回は、slack-ruby-client というライブラリを用いて作成しました。

図 2: スロット

オリジナルとの差別化

やはりスロットといえば 10 連、ということで、スロット 10 連を回せるようにしました。

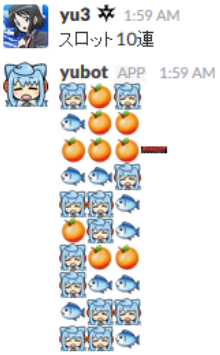

図 3: スロット 10 連

KMC 部員はどうせ脆弱性を突きに来るので、大きすぎる数は回せないようにしておきます。

図 4: スロット 100 連

どうせなら当たりが大量に出ると盛り上がりそう、というわけで接待モードもついでに実装します。

図 5: 接待モード

当然のように 10 連も実装します。

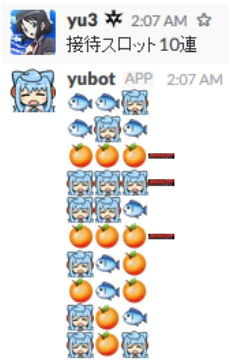

図 6: 当たりがたくさん出て楽しい

何故か部員はあまり接待されたがりません。単純に当たりが出ればいいというわけではないようです。

データベースを使ってみよう

スロットといえば、そう、課金です。スロットを回すたびにコインを消費するモードを実装しました。

図 7: 課金して回すスロットは楽しい

最初は当たり時のコイン増加枚数を多めに設定していたのですが、課金したがる部員が現れたため、急遽引き下げました。

おわりに

サクッと作った bot でも、思った以上に遊んでもらえます。ユーザーの反応が得られるのは、思った以上に嬉しいものです。

特に作りたいものがない……と思っている読者の方は、本記事を参考により良いスロットを作ってみてはいかがでしょうか。

あとがき

■jf712　ここにはあとがきを書きます。

■wass80　CSS+HTML 組版にしたかったですが時間がなく断念。LaTeX に戻りました。

■hakurin　挿絵担当、hakurin です。気がついたらもう老害おじさんですね

■yu3　部誌デビューしました。ようやく「これ私が書いたんです」って言えるものができて嬉しいです。

■polaris　~~やべぇ何を書けばいいんだ~~近況など：

- まだ大学生活に慣れない
- この夏に PC を新調する（したい）
- ハードウェアの勉強もやりたい

最後までお読み下さりありがとうございました（え、これを最初に見たって？）。

■hideya　先日セガサターンのパッドを改造して PC に接続できるようにしました (テキトーな USB パッドの中身を移植しただけ)。やっぱりサターンパッドの方向キーは最高ですね。ちなみに我が家はプレステ派閥でした。

■tron　前回は記事[*1]に合わせて 575 を詠んだので、まともに[*2]あとがきを書くのは初めてです。えーと、短い記事ではありますが、読んでいただければ嬉しいです。ﾂｷﾞ ﾊﾓｯﾄｽｺﾞｲﾉﾓｯﾃｸﾙｶﾗｵﾀﾉｼﾐﾆ……

[*1]「575　詠んだら皆　良い気分」という、部員の詠んだ 575 をまとめた記事です。
[*2]「文章である」ぐらいの意味です

独習 KMC vol.11

2017 年 7 月吉日　初版発行

著作・発行　京大マイコンクラブ
表紙デザイン　koh
挿絵　hakurin
印刷　コーシン出版

メールアドレス　info@kmc.gr.jp
Web　https://www.kmc.gr.jp/

落丁・乱丁の際は在庫がある限りお取り替えいたします。上記のメールアドレスまでご連絡ください。

独習KMC vol.11

■利用分類
部誌

■製品分類
KMC

■レベル
初級 中級 上級

Kyoto University
Micro Computer Club

https://www.kmc.gr.jp