明治大学

2007 年度

機関誌

エレクトロニクス研究部

明治大学
理科部連合会
所属

幹事長挨拶

本日は明治大学エレクトロニクス研究部の展示にご来場いただきまして、まことにありがとうございます。

現在、我々エレクトロニクス研究部は 1,2 年生合わせて 20 名の部員がおり、全員がソフトウェア制作を行うソフト班とロボット制作を行うハード班のいずれかに所属する形で研究制作を行っております。

今回の学術研究発表会における展示発表では、ソフト班からは各班員がプログラミングしたゲームの展示を、ハード班からは今年 10 月 7 日に行われたロボット相撲大会の出場機体の展示や班員の個人制作等を展示しております。展示物の詳細については、この機関誌の後頁においてそれぞれの発表内容を紹介しております。また、展示物それぞれに制作担当者が説明員としておりますので、お気軽にお声をおかけください。

2007 年 12 月 16 日

明治大学エレクトロニクス研究部 幹事長

菅沼 久浩

Meiji University – Electronics Lab.

ハード班

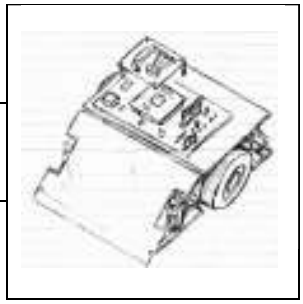
ソフト班

その他

ハード班

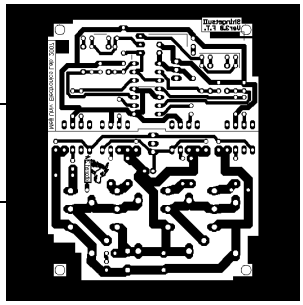
序

ハード班とは
相撲ロボットと全日本ロボット
相撲大会について

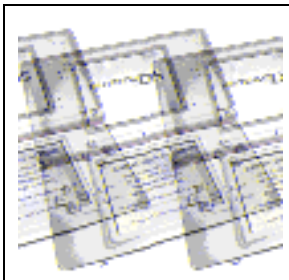


相撲ロボット制作
「ゴッドハンド」について

相撲ロボット制作
「新月Ⅱ」について



PWM 対応 3 信号
モータードライバーの制作



NintendoDS
オーバークロック化改造

明治大学エレクトロニクス研究部

ハード班について

われわれ、明治大学エレクトロニクス研究部ハード班では、主に相撲ロボットをはじめとする自立型ロボットの制作や電子工作を行っています。特に相撲ロボットにおいては、毎年行われる全日本ロボット相撲大会にむけ、日々改良を重ねています。また、ロボット相撲大会や学園祭(生明祭)に向けた制作以外にも、春休みを利用して制作活動を行い、その成果の一部を新入生勧誘時期に披露しています。

活動方針は「自作すること」を基本主体としており、回路はプリント基板の回路図・パターンを回路 CAD によって感光基板に手ずから焼き込んで制作しています。ボディの切断、穴あけなどのアルミ板やアクリル板の加工もほとんど自分たちで行っています。

なお、技術の向上には正しい知識とノウハウの伝授が必要と考え、われわれハード班では入ったばかりの新入生に対してハードゼミを行い、上級生が電子工作の基礎を指導しています。また、ゼミが一通り終わった後も、夏休み中の制作活動や学園祭に向けた準備の中で、ロジック IC や各センサーの使い方や回路の仕組みをはじめとした応用的なことを少しずつ教えていきます。

今年は、おもに相撲ロボットを 2 台制作し相撲ロボット大会に出場しました。

各機の詳しい説明は以降の、制作者による資料をご覧ください。

相撲ロボットと
全日本ロボット相撲大会について

相撲ロボットとは

名前どおり、相撲をするロボットです。ただし、二足歩行ではなく車輪駆動で、直径 154cm の円形土俵でロボット同士がぶつかりあう競技です。
相撲ロボットには大きく分けると 2 種類に分けられていて、ラジコン型と自立型の二つに分類されています。ラジコン型はリモコンでロボットの動きを制御するものですが、自立型は一度スタートしたらあとはプログラムに制御させるものになっています。なので、ほとんどの自立型のロボットには周りの状況を見るために赤外線センサーや超音波センサーなどのセンサー類がついています。これで周囲の情報を得て、戦います。
戦うロボットには、大抵が力の強いモーターで早く進み、敵に突撃していきますが、一部のロボットは馬力を強くして、敵の攻撃に耐えて、徐々に追い詰めていくタイプや、無差別に走り回り、偶然に任せるパワーのみのタイプなど、様々なものがあります。

全日本ロボット相撲大会とは

全日本ロボット相撲大会とは、富士ソフト株式会社と社団法人全国工業高等学校長協会が主催しているロボット大会のひとつです。他のロボット大会で有名なのは NHK ロボットコンテストなどがありますね。ロボット相撲大会もそれと同じで、個人、またはグループで作った相撲ロボットを戦わせて勝敗を競う場となります。出場者は高校生部門と社会人部門で大体 500 チームほどになります。
では、実際のロボット相撲大会のルールを見てみましょう。

～ルール～

・直径 154cm の鉄土俵で戦う。
・重量は 3kg まで。
・ロボットの大きさは縦 20cm×横 20cm までとする(開始時ので、変形可)。
・ロボットの高さは無制限。
・スタートスイッチを押してから 5 秒後で発進するようにするる。
・相手を場外へ押し出すと勝ち。2 本先取が勝ち進むことが出来る。
・出場選手は安全のために軍手、ゴーグルを着用する。
・暴走防止のための緊急停止回路を取り付ける。
・5 秒後に発進しないと戦意喪失とみなされ、1 本とられる。
・土俵を傷つけるような行為をしたものは失格。
・相手のロボットに対して破壊行為をしてはならない。

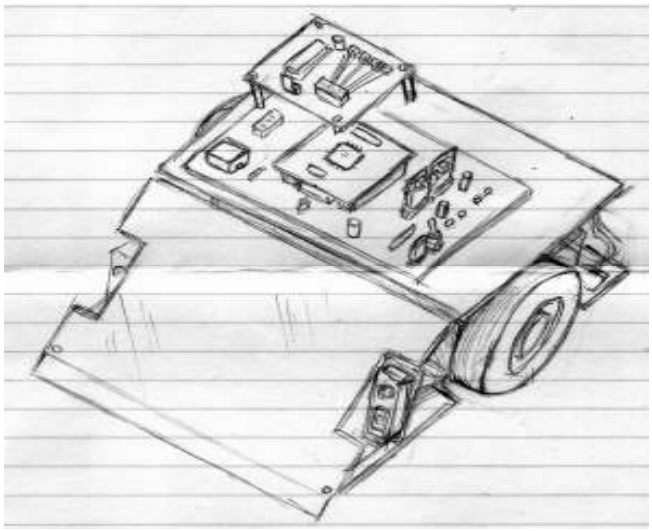
参考

全日本相撲ロボット大会(富士ソフト ABC) http://www.fsi.co.jp/sumo/

相撲ロボット制作
「ゴッドハンド」について

はじめに

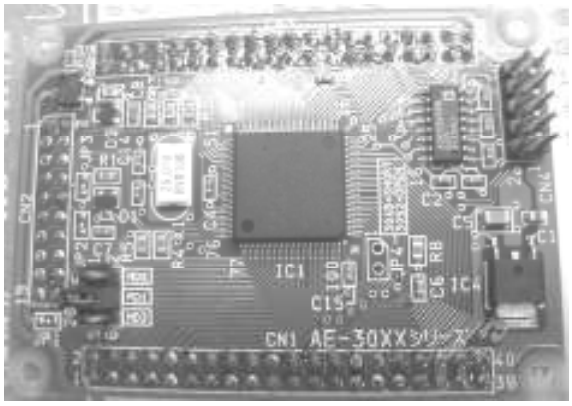
今年、ハード班 1 年は 3kg 級の相撲ロボットを制作しました。我々一年が制作した相撲ロボットは敵検知センサーをつけたスピードタイプで、名を「ゴッドハンド」と言います。挙動としては、敵を発見するまで彷徨い、検知後、突撃というタイプです。（図は手書きの完成予想図）



ゴッドハンドは頭脳となるマイコン(CPU)を積んだメインボード、神経たる各種センサー、モーター、車輪、バッテリー、緊急停止回路等を図のように組み立てます。パソコンによって、本体を動かすためのプログラムをマイコンに書き込みます。以下に各部品の役割を詳しく記します。

マイコン

マイコンとは電子機器の制御や演算に使うコンピュータのことです。パソコンから本体を動かすためのプログラムを書き込み、モーターへ流す電流をセンサーの入力に対する反応などで制御します。

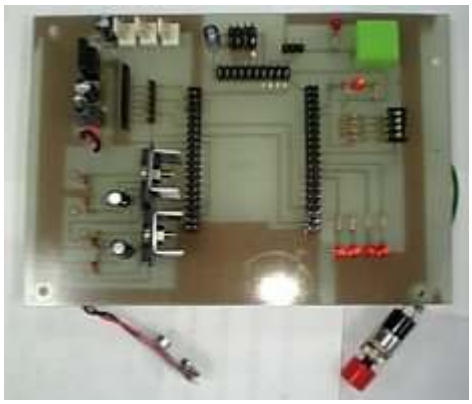


今回は H8/3052 いうマイコンを使っています。3052 は値段が安価、必要とするチャンネル数が適当、C 言語でプログラムを簡単に組むことが出来、小型で回路を作るのが容易など、電子工作をするのに向いています。右図はマイコンボードです。

メインボード

相撲ロボットの回路は主に各種センサーに関する部分、電源とモーターに関する部分の２つ分化されて配置されています。まず、電源部分は電源(下で説明している９V 乾電池)から来た電圧を３端子レギュレーターという部品によって５V に電圧を下げマイコンに電流を流し、マイコンから各部品に命令を出します。センサー部分はマイコンからの電流をバッファに通してセンサーがどのような情報を受け取った時にどういう電流を返すのかを制御します。

モーター部分は電流をモータードライブ IC というIC に流してモーターの前進、後退、反転などの動きを制御します。ちなみに、先輩方の作ったロボットはパワーを上げるため、一年が作ったロボットをはるかに上回る電流が流れています。そのため、IC では制御仕切れないため、先輩方のロボット(新月Ⅱ)に IC はついていません。次ページの図は、マイコンを未接続のものです。



メインボードはおそらく、相撲ロボットの中で一番複雑な回路なので、実際制作するときにも、どことどこをつなげるのか、などに苦労しました。また、デバック作業時にも「どこが違うのか」と、ミスした場所を探すのが大変でした。

モーター

今回、ゴッドハンド用のモーターとしてツカサ電工のギアドモーター、TG-06D を使用しています。以下にそのスペック表を記載します。

機種名	定格電圧 (V)	無負荷回転数 (rpm)	無負荷電流 (mA)	定格トルク		定格回転数 (rpm)	定格電流 (mA)	回転方向	質量 (g)	備考
				(mN-m)	(g-cm)					
TG-06D	12	5200	95	9.8	100	4250	550	両方向（CW）	120	バリスタ、コンデンサ

搭載するとき、ボディにうまくつなげるためのパーツを自作しなくてはなりませんでした。それ自体は去年のものを流用したのですが、少しゆがんでいたので、ヤスリでゴリゴリ……。心が折れそうでした。今回使用したものは、下の絵に描かれているギアーボックス付のものです。

バッテリー

ゴッドハンドに搭載したバッテリーは 7.2V のバッテリーです。下図のバッテリーを２パック使いました。これは 1.2V の電池を６つ直列にしたものです。電圧的には、先輩方が作ったロボットに搭載されているものに及びませんが、安価で手に入りやすいものです。

ボディ

重量制限があるため、軽いアルミをボディの材料につかいました。厚さ3mmのアルミ板に穴を開け、加工していきました。バンパーなどもアルミ製です。細かい部品(センサー類)などを車体につけるときの留め金には、厚さ 1mm のアルミ板を使いました。

また、これは義務ではないのですが、すべてのロボットは機体の底に強力な磁石をつけています。土俵が鉄板製なので磁力を利用し、機体に踏ん張る力を持たせられるのです。そうでもしないと、ロボット同士がぶつかったときに吹っ飛んでしまうからです。

ボディを制作するときも、微調整が必要でヤスリを使いました。厚みがあって、さらに細かい調整が必要だったので、何度も何度もゴリゴリ……。手が本当に疲れました。

緊急停止回路

相撲ロボットには安全のために遠隔操作可能な緊急停止回路を装着することが義務付けられています。これは、マイコンと直結させてテレビなどのリモコンから赤外線を受け取ってマイコンがモーターを遮断します。

その他特徴について

今回 1 年生が作ったものは 2 年生が作った通常の相撲ロボットと大きな違いが一つあります。

先輩方のロボットでは、モータードライバーが左右のモーターで 2 つに分かれているので、マザーボードからモータードライバーが独立していますが、我々のロボットはそれらを 1 つにまとめてあるので、モータードライバーに直接マザーボードがついています。これにより、回路全体がコンパクトにまとまっています。

反省点・今後の課題

今回、生明祭までに機体を完成させることが出来ませんでした。また、一年だけではまったく作業が進まず、上級生の方々に手伝ってもらいっぱなし、リードされっぱなし、頼りっぱなし、という状況が良くありませんでした。一年は、電気回路に対する知識をもっと増やすべきでした。

直前まで完成せず色々大変なスケジュールでしたが、次こそは早めに作業を始めるぞ、と思う今日この頃です。

ちなみに、回路はプリント基板を用いたのですが、それを加工するために使うエッチング液が飛び散って服が何着か大変なことに……次から慎重に作業をしようと思いました。

相撲ロボット制作
「新月Ⅱ」について

はじめに

新月Ⅱは 3kg 級相撲ロボットである。エレクトロニクス研究部にある全ての力を使い、妥協のない最強の相撲ロボットを目標に制作された。全日本ロボット相撲大会関東大会に出場し、1 回戦で敗れた。以下はその全体図である。

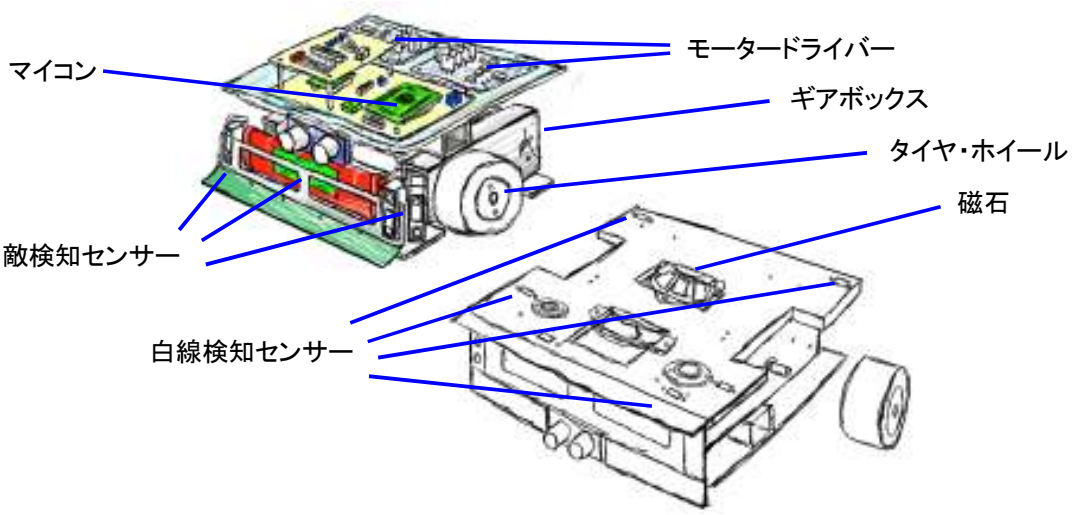


図 1 全体図

モーター

搭載するモーターは maxon motor の RE35 シリーズで公称電圧が 15V のものである。RE35 シリーズは直径 35mm、90W ブラシモーターである。モーターには専用の遊星ギアヘッド(18:1)が装着されている。

モーターの負荷がギアヘッドのみの状態で回転数を計測したところ表 1 のデータが得られた。タイヤの円周（52mm）から速度を計算し、24V で駆動することにした。1 秒間で土俵の端から端まで移動できる速さを目安としている。

表 1:モーターの電圧-回転数特性

電圧	12V	18V	24V	26V	30V
回転数	171rpm	511rpm	692rpm	746rpm	862rpm

ギアボックス

ギアヘッドを含めたモーターの全長は 10cm 以上あるため、2 つのモーターを 1 列に配置することはできない。モーターは車輪軸を中心に互い違いに配置し、歯数が等しいギアを 2 つ使用し、動力を伝達している。

モーターとギアのハブ（イモネジでギアとシャフトを固定する部分）が干渉してしまうため、ハブ外径と基準円直径の比が大きなギアを選択した。使用したギアは以下の通りである。

表 2 使用したギア

メーカー/型番	協育歯車工業 S75S36B+0606
モジュール	0.75
材質	機械構造用炭素鋼
基準円直径	27mm

ギアボックスのフレームには 20mm × 40mm × 2mm のアルミ管を使用した。アルミ管を使用することで、部品数を減らし、加工のコストを下げることができる。

ホイール/タイヤ

3 年前に製作した初代新月では、スポンジタイヤに卓球用粘着ラバーを巻き付けたものを使用していた。[参考文献 2] グリップ力が非常に高かったが、ラバーの繋ぎ目が凹凸となり、スムーズな走行ができなかった。また、ホイールとシャフトの部の取り付け方法に問題もあった。昨年までの相撲ロボットは、直径 60mm 程度のホイール・タイヤにゴムテープを巻いて使用していた。

新月Ⅱでは、開発期間が限られているため、シータスク製の「相撲ロボット用車輪セット」を使用することにした。ホイールには標準アルミホイールと軽量アルミホイール、タイヤには飴ゴムタイヤとシリコンゴムタイヤがあるが、共に後者を用いた。

軽量ホイールタイヤを選んだのは、単に軽量化のためではない。モーターの端子と、反対側のギアボックスが干渉してしまうため、ホイールの内側を 2mm 削った。軽量タイヤは、ホイール内部が切り抜いてあるので、やすりのみで加工できる。

タイヤとホイールの接着は専用両面テープを使用し、1 回分のテープが付属している。一般アクリル系接着剤両面テープ（強力タイプ）で固定を試みたが、簡単に外れてしまう。市販品では、片面アクリル系、反対側がシリコン系の両面テープが存在する（専用両面テープもこのタイプ）が、入手困難である。両面がシリコン系接着剤の両面テープを入手したので、今後試す予定である。

前輪は小林製作所のベビーボールカスター（ボールキャスター）を使用している。

センサー

新月 II には、土俵の外周に引かれている白線を検知するセンサーが 6 個、対戦相手を探すための敵検知センサーが 5 個搭載されている。

白線検知センサー

白線検知センサーにはロームの赤外線センサーRPR-220 を使用した。赤外線 LED とフォトトランジスタが 7mm × 5mm × 7mm のパッケージに収められている。30kΩの可変抵抗を用いてベース電流を制御し、感度調整を行えるようになっている。出力は NOT 出力のバストランシーバー74HC640 によって論理値に変換される。

白線回避の終了判定をタイマーを用いて行うと、モーターの回転速度の影響を受けるため、バッテリーの状態の影響を受けてしまう。よって、センサーの情報を元に白線回避動作を行う必要がある。白線上での向きの情報を得るため、白線検知センサーを車体の回転軸を中心とした円周上に配置した。

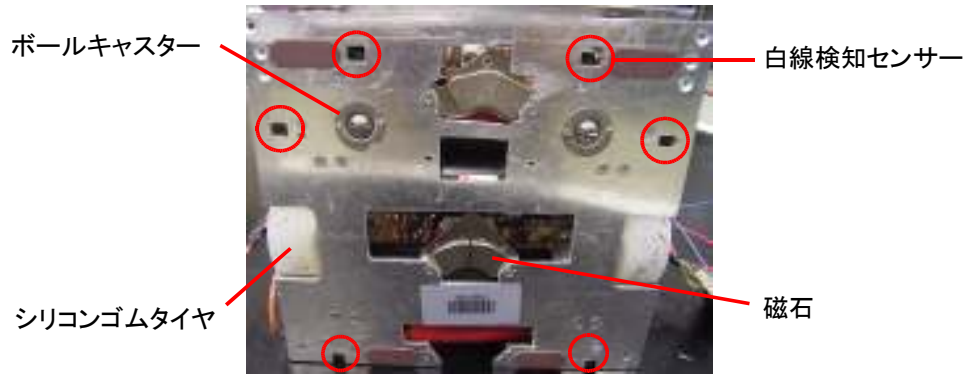


図 2 新月 II の底面

敵検知センサー

敵検知センサーは 2 種類のセンサーを使用している。その一つがシャープの PSD センサーGP2D12 である。このセンサーは赤外線を照射し、三角測量を行い、その測定された距離をアナログ出力する。出力値は距離と反比例した曲線を描く。センサーからの出力は、マイコンの A/D 変換で取り込むことができる。測定可能な距離は 約 10cm～80cm である。

GP2D12 は赤外線を使用するため、対象物が黒い場合は検出できない。アルミ板に黒いビニールテープを貼ったものでは 30cm 程度が限界であった。そこで、補助センサーとして超音波センサーを装備している。小型で使用しやすい浅草ギ研の「超音波距離センサーPING」を使用した。

図 3 PSD センサー

バッテリー

最近ではリチウムポリマー電池を採用する機体が増えているが、新月Ⅱではニカド電池を使用する。通常のラジコン・電動ガン用バッテリーパックでは重量がオーバーしてしまうため、EAGLE MODEL の 12V ミニバッテリーを 2 パック使用する。1 パック当たりの重量はわずか 200g である。

磁石

土俵は鉄板でできているため、弾き飛ばされないため、タイヤのグリップ力を上げるために磁石は必須である。磁石はハードディスクを分解し取り出したものを用いている。磁力を打ち消す金属板が付属しているため、モーターの真下にも配置することが可能で、取り付け穴も空いているので便利である。

新月Ⅱの制作に合わせ、直径 1500mm の鉄板を購入、塗装し、ロボット調整用の鉄土俵も制作した。

マイコン

メイン CPU にはルネサステクノロジ 16bit CPU、H8/3052 を搭載した秋月電子通商の AKI-H8/3052 と、超音波センサー専用として AKI-H8/3664、緊急停止回路に Microchip の PIC12F675 を使用した。H8/3052 は ITU（Integrated Timer Unit）の機能が充実しており、複数のモーターの制御に適している。

プログラム

開発環境

新月Ⅱのプログラムは C++で開発を行った。コンパイラは Renesas の H8 C++コンパイラを、統合開発環境は HEW（High-performance Embedded Workshop）を使用した。無償の評価版は使用期限を過ぎると、リンクサイズに 64KB の制限がかかるが、その他の機能は引き続き使用できる。

オブジェクト指向言語を使用することで、モーターやセンサーをオブジェクトとして扱うことにより、開発の効率化を図っている。

ロボットの制御

ロボットの制御は白線回避ターン、白線回避、敵探索の 3 つの状態を遷移することによって行う。例え白線回避動作中でも、対戦相手を発見した場合は探索を止め、敵の方向に向かう。ファジー推論、タスクスイッチなど、搭載可能な技術としては様々なものがあるが、新月Ⅱでは短い開発期間で最大の効果を得るために、シンプルなプログラムとなっている。1ms 間にロボットが移動できる距離は 2mm 程度であるため、この時間の遅れは許容範囲内である。

白線検知センサーのプログラム

白線検知センサーは、割り込みを使用して白線の回避を行っていた。しかし、感度を最大限落としても土俵表面の傷（表面の塗装が剥がれ、鉄板が露出した部分）に誤反応してしまう事があった。誤反応を回避するには、白い部分の太さを検出する必要がある。

そこで、割り込みを止め一定間隔でセンサーの状態を調べ、1ms 以上センサーが反応している場合に白線を検知したと見なすことにした。

デバッグ用モニター

新月Ⅱの開発では、デバッグのためにプログラムを RAM に配置するなどは行っていない。デバッグ用にシリアルポートで接続された PC にハードウェアの情報を送信し、リアルタイムにセンサーやモーターの情報を表示するデバッグモニターを作成した。デバッグモードではコマンドを入力することにより、モーターの速度を設定することも可能である。

標準 I/O ライブラリはプログラムサイズが大きいため、シリアルポートへ文字を出力する関数ライブラリは独自に実装した。この関数ライブラリには ESC 文字を使用した文字色の変更、カーソルの移動などの関数も含まれている。

図 4 デバッグモニター

最後に

ロボットを制作する上で回路や機構、加工の基礎知識は重要である。基本知識の有り無しで、1 日で終わる作業が 1 週間を要してしまうことも少なくない。また、メンバー全員が機構や回路の仕様について把握していることも大切である。その部品をどこで、どのように使用するのかを把握していれば防げる加工ミスが多い。限られた時間で制作をするため、作業効率は機体の完成度に大きな影響を与えるだろう。

新月Ⅱの制作を本格的に開始できたのは、関東大会を 1 ヶ月後に控えた 8 月の下旬であった。開発期間が非常に短く、大会は未完成の状態での出場となってしまった。生明祭（大学祭）の展示でも故障が相次いだ。来年は、春休みや夏休み前半で実験的な作業を行い、万全の体制で大会に臨むことを期待している。

相撲ロボット新月Ⅱ 制作スタッフ

4 年	武山 文信	2 年	平林 誠一郎
4 年	岩丸 虎喜	1 年	長谷川 博一
4 年	額 慎太郎	1 年	羽山 義弘
3 年	井坂 康行	1 年	渡辺 大貴

協力

M2	八木澤 圭	緊急停止回路の制作
4 年	後藤 慶之	モーター回転数の測定

参考文献

[1]PSD 調査報告書, 沼津高専 電子制御工学科
http://www2.denshi.numazu-ct.ac.jp/mirsdoc2/mirs0205/tech/num0014a/PSD.html

[2]相撲ロボット - 新月
http://www.isc.meiji.ac.jp/~eleken/hard/shingetsu/

[3]GP2D12 データシート
[4]RPR-220 データシート

[5]2007 年度 ハード講習テキスト, 明治大学オートメーション研究部
http://www.isc.meiji.ac.jp/~autome/data/2007hard01.pdf

[5]maxon 総合カタログ 04/05
[6]協育歯車工業カタログ KG803

PWM 対応
3 信号モータードライバーの制作

4 年 武山 文信
図 4 年 岩丸 虎喜

仕様

モーターの速度を制御するためには、PWM 制御が必要になる。しかし、今回使用するモーター（maxon motor RE35）は端子間インダクタンスが非常に小さいため、最適な制御を行うには数百 kHz での PWM が必要である。[参考文献 1] このような周波数で PWM 制御を行うことは FET やフォトカプラの性能により困難である。PWM は補助的な機能として使い（通常はデューティー比 100%として使用する）、効率は低下するが 40kHz で PWM 制御を行うことにした。

以上の点から、要求される仕様は次の通りである。
1. デューティー比 100%でも使用できる回路
2. PWM に対応
3. モーターがロックされた場合を考慮して、60V30A 程度の電流を流せる FET

回路の決定

デューティー比 100%で動作させることを考えると、H ブリッジ回路のハイサイドの FET に N チャンネル FET を使用することは難しい。ハイサイドのゲート電圧を生成するためにブートストラップ回路を使用する場合、定期的にモーターを OFF にする必要があるためである。[参考文献 1] 昇圧回路を使用する方法もあるが、回路規模が大きくなる欠点がある。

そこで新月Ⅱではハイサイドに P チャンネル FET を、ローサイドに N チャンネル FET を使用し、ローサイドで PWM を行う回路を採用した。[参考文献 2, 3]

FET の選定

製品としては非常に多くの FET が存在するが、簡単に入手できるものは限られている。3.の条件を満たす FET として、ルネサステクノロジの次の FET を採用した。TO-3P パッケージは十分な放熱能力があると判断し、放熱器は取り付けないことにした。

表 3:FET の性能

型番	D-S 電圧定格	D 電流定格	G-S 電圧定格	D-S 遮断電圧	入力容量	ターンオン遅延時間	上昇時間	ターンオフ遅延時間	下降時間
2SK2586	60V	60A	20V	1.0～2.0	3550pF	35ns	260ns	480ns	370ns
2SJ555	-60V	-60A	-20V	-1.0～-2.0	4100pF	32ns	270ns	570ns	360ns

D:ドレイン S:ソース G:ゲート

ゲートドライブ回路

フォトカプラ

モーター側電源と、マイコン側の電源をフォトカプラで独立させる。フォトカプラは東芝の TLP250 と TLP552 を用いた。

TLP250 はパワーMOSFET のゲートドライブ用フォトカプラで、入力容量が大きな FET を駆動するのに十分な電流（定格 ±1.5A）を流せる。 一方、TLP552 は NOT 回路出力の信号用フォトカプラで定格出力電流は 50mA である。ゲートを駆動するには電流が不足するが、大きめの抵抗を挿入しスイッチング時間を遅くすることにした。ハイサイドの FET のスイッチは方向変更時にしか行われないため、スイッチング時間が数μs 程度に延びても、十分な性能であると判断したからである。

TLP250 の出力電圧

TLP250 の出力電圧は、電源電圧が 30V の場合、標準で 27.8V(ハイレベル)、0.8V（ローレベル）となっている。ハイレベル時のモーター側 V_{CC} からの電位差は 2.2V で 2SJ555 のゲート遮断電圧（-1.0 ~ -2.0V）を超えているため、ゲートを遮断できない場合がある。後述するバージョン 2 の回路では、この問題を回避するため、2.0V のツェナーダイオードまたは 3 個直列の PN 接合ダイオード（1N954）を挿入した。バージョン 3 の回路では、ハイサイドのフォトカプラを TLP552 に変更した。

ゲート直列抵抗

フォトカプラと FET のゲートの間にはゲート直列抵抗が挿入されている。ハイサイド（TLP552→2SJ555）は 150Ωの抵抗が、ローサイド（TLP250→2SK2586）には 30Ωの抵抗が挿入されている。ゲート抵抗の抵抗値は FET の入力容量から求めることも出来るが[参考文献 4]、入力容量がデータシートそのままといった単純なものではない[参考文献 6]ため、フォトカプラの定格を超えない範囲の抵抗を使用した。ローサイドの抵抗が燃えるとの情報があり、1W の酸化金属抵抗を使用した。

ゲートドライブ電源

2SK2586 と 2SJ555 は 4V 駆動の FET である。ゲートを制御するには 0V（GND） ~ 4V 以上の電圧（ローサイド）と、24V（V_{CC}） ～ 20V 以下（ハイサイド）の電圧が必要である。また、ゲートソース電圧の定格が±20V であるため、V_{CC} と GND から直接電源を供給することはできない。

また、FET はゲートに電荷をチャージすることによって制御する。ゲートソース間電圧が大きいと、ゲートに流れる電流が大きくなってしまう。よって 4V（駆動に必要な最低限の電圧）に近い電圧で制御するべきである。ローサイドの電圧は三端子レギュレータ 7810 で 10V の電源を作り、ハイサイドは負三端子レギュレータ 7905 で(V_{CC}－5)V の電圧を生成している。ローサイドの電圧が 10V であるのは、少しでも TLP250 の推奨動作電圧（15V）に近づけるためである。

ロジック回路

PWM 信号は共有できるため、4 個の FET を 3 信号で制御できる。これにより H ブリッジ回路に入力禁止のパターンを入力することを防ぐことも出来る。ロジック IC には 3 ステートバッファーの 74AC125 を使用した。AC シリーズを使用する理由は、高速な動作が可能で、フォトカプラ内の LED を動作させるのに十分な電流を流せるからである。

デッドタイム生成回路

FET のゲートに電圧を加え、ドレイン-ソース間に電流が流れるまで、あるいは電流が遮断されるまでには時間がかかる。2SK2586 と 2SJ555 は、十分な電流が流れる場合、1μs 程度の時間が必要である。よって、モーターの回転方向を切り替える際に、全ての FET の状態を同時に切り替えると、中間状態の FET を通して貫通電流が流れてしまう。これを防ぐためには、全ての FET を OFF にしてから状態を切り替える。ソフトウェア側で実現する方法もあるが、ソフトウェアのバグで回路を破損してしまう可能性があるため、信号線にデッドタイム（全ての FET を OFF にする時間）生成回路を挿入した。

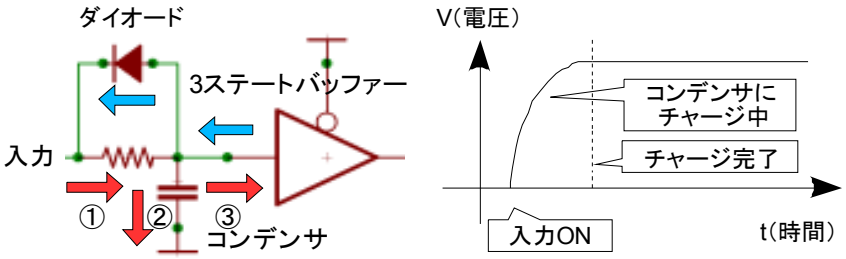


図 1:デッドタイム生成回路

デッドタイム生成回路は図 1 のようなダイオード、コンデンサ、抵抗で構成された回路である。図 1 の回路では、入力が 0V から 5V に変化する場合に電流は抵抗を通してコンデンサに流れ、出力が十分な電圧になる時間を遅延させる。一方、入力が 5V から 0V になる場合はコンデンサからダイオードを通して電流が流れ、すぐに出力側の電圧は 0V に落ちる。遅延時間は次の式で求められる。[参考文献 5]

$$t_{dt}\ (\text{遅延時間}) = 0.8 \times C \times R$$

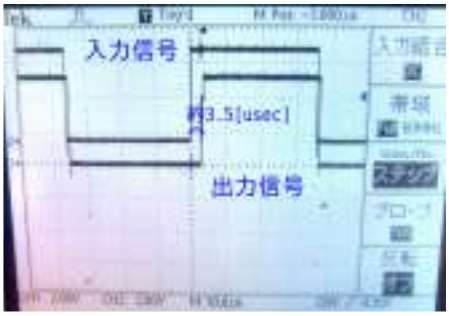


図 2:デッドタイム生成回路の入力信号と出力信号

図 2 は V_{CC}=5V、R=24kΩ、C=100pF、マイコンポート保護抵抗 5.1kΩの条件でデッドタイム生成回路の入力と出力をオシロスコープで測定したものである。この回路のダイオードは 1N954 であるが、Ver3 の回路では、順方向電圧が低く高速に動作するショットキーダイオード 1SS120 を使用した。

制作過程

試作バージョン

試作バージョンは最低限の部品のみが載った回路である。部品の配置もかなり適当で、回路が非常に大きい。TLP250 の出力電圧対策も行われていない。

バージョン２

バージョン 2 は全日本ロボット相撲大会関東予選の際に使用した回路である。デッドタイム生成回路が搭載されたが、設計ミスでハイサイドの FET の信号線にしか挿入していなかったため効果は無かった。何度か FET または配線が焼けるトラブルが発生した。原因究明のため、大会直前にツェナーダイオードを３個直列の 1N954（図３）に変更している。

図 3:3 個の 1N954

図 4:焼き切れたパターン

図 5:皮膜が溶けてしまった配線

バージョン３

バージョン３の回路では、全ての FET に対しデッドタイム生成回路を挿入した。これにより、部品数が増えたが基板レイアウトを見直すことにより、バージョン 2 の回路より一回り小さくなっている。ハイサイドの FET 用フォトカプラを TLP552 に変更し、ゲート制御電圧の問題も解決した。バージョン 3 の回路からは、基板構成が各モーター用基板（2 枚）とゲート電源供給回路となっている。TLP552 と 2SJ555 がそれぞれ 1 回ずつ故障したため、ゲート抵抗を大きくし、デッドタイムを増やした。

図 6:Ver3.0 の回路

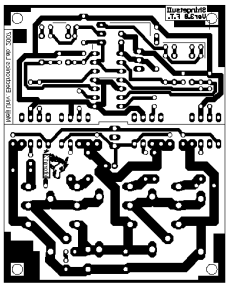


図 7:プリントパターン

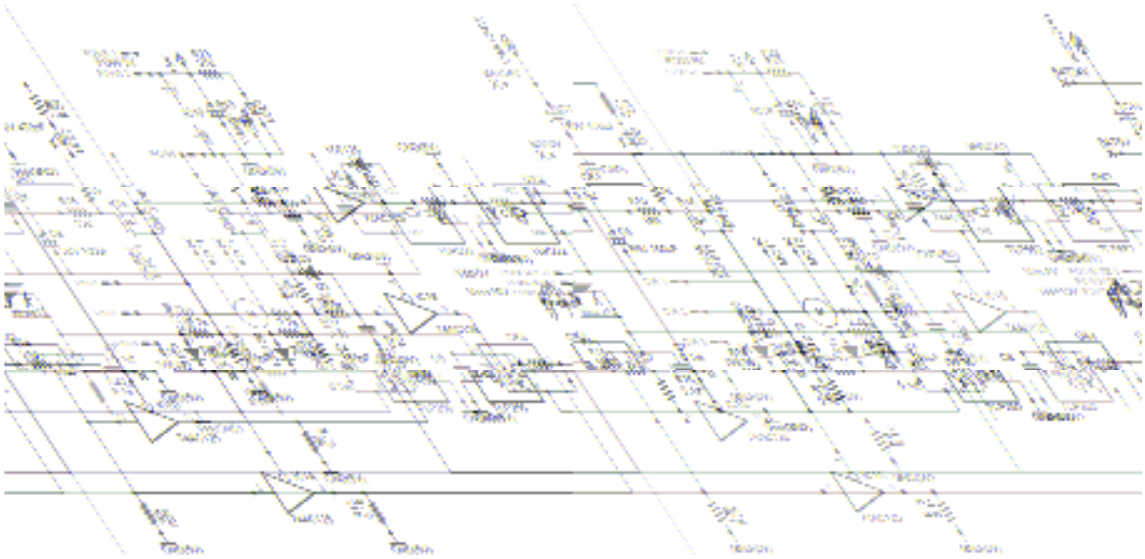


図 8:モータードライバーVer3.1 の回路図

反省点

モータードライブ回路の反省点としては、回路がオーバースペックになってしまったことである。その結果、回路の規模が大きくなり、ロボット本体への搭載が難しくなった。また、FET の入力容量が大きく、スイッチング特性が良くないという問題もある。

安定性についても未知である。時々FET とフォトカプラに発生する故障の原因は未だ解明できていない。引き続きテストを行い原因を究明する必要がある。

参考文献

[1]NOTE - 回路設計編 - フルブリッジ - モータドライバ, m.kimpara
http://www.ceres.dti.ne.jp/~kkimpara/robocon/note/circuit-design/fullbridge.html

[2]FET によるモータードライバーの作成, 宮田耕自
http://www.geocities.jp/mimiin/tips/fet/fet.html

[3]目指せ究極のオリジナルサーボ, バーニング宮田
http://www.geocities.jp/mimiin/TC.pdf

[4]Robocon2003　回路設計資料～パワーMOSFET 編～, 舞鶴高専 創造技術研究会
http://machidapc.maizuru-ct.ac.jp/sken/siryou/sekkei2003/mosfet/mosfet.htm

[5]デッドタイム回路の CR 定数, ドルクスラボ
http://www.geocities.co.jp/Technopolis/5348/00-10dt.html

[6]パワーMOS FET 活用の基礎と実際, 稲葉 保, CQ 出版

[7]H.achida@MNTC H-Bridge motor Driver by Power MOS-FET, 町田秀和
http://machidapc.maizuru-ct.ac.jp/machida/elec/motor/motorbody.htm

[8]TLP250 データシート

[9]TLP552 データシート

[10]2SK2586 データシート

[11]2SJ555 データシート

NintendoDS オーバークロック化改造 プレイ速度を 1.4・1.7 倍速に！

はじめに

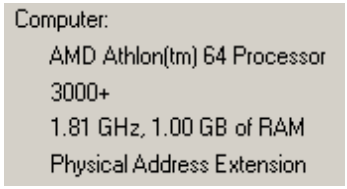
どうもゲームが遅い・・・そう感じたことはないだろうか？ニンテンドーDS をもっと高速で動作させることができれば快適になるはずだ。そこで、オーバークロックにより、高速化を図った。

オーバークロックとは何か？

何も知らずにオーバークロックを行うことは大変危険。まずはパソコンのオーバークロックのメカニズムを知ろう。

PC でのオーバークロック

パソコンは進化が早い製品である。しかし、次から次へと最新のハイエンド製品に買い換えられる人はまずいないと思う。少しでも性能を上げたいと思うのが人の心理だ。そこで登場したのがオーバークロックという手法である。手軽に処理速度を上昇できるために、根強い人気がある。



画面 1:PC のクロック表示。
1.81GHzが動作周波数となる。

オーバークロックとは、パソコンの内部周波数を変更する改造である。PC 内部は一定のクロックと呼ばれる信号に基づいて動作している。クロックを上げれば処理速度も上昇するというわけだ。

初期の改造方法は基板上の水晶振動子を交換するというものだった。しかし、水晶振動子ではその特性上、細かい周波数の調整ができない。調整するには、たくさんの水晶振動子が必要である。また、半田付けを行わなければならないという面倒な作業があったため、時代とともに消えていった。ここ最近では、ソフト側からオーバークロックする手法が用いられている。細かく調整できる上に、温度に応じてクロックを変更する製品もある。これなら誰でも簡単に性能を向上できるというわけだ。

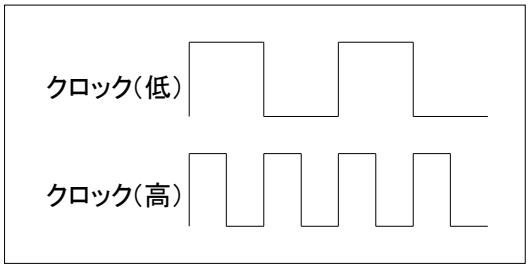


図 6:動作クロックの例。ひとつの山が 1 クロックに相当する。

しかし、オーバークロックには欠点がある。高クロックになればなるほど、素子は大電流を消費する。そうなると、激しい発熱を生じる。うまく冷却できなければ、素子が焼け焦げてしまうことがあるので注意したい。また、配線長の問題や、素子の限界などもあり、大きく性能が変化しない場合もある。

ニンテンドーDS でのオーバークロック

パソコンとは構造が異なり、処理速度に加えて、ゲームのプレイ速度も内部周波数で決定される。**そのため、ゲームの高速化をすることができる。**さらに、シンプルで低スペックなため、発熱の心配がない。水晶を交換するだけで、簡単にスペックアップできるのが特徴だ。

水晶振動子とは？

水晶に圧力を加えると電流が発生する。これを圧電現象という。反対に電圧を加えると一定の周期で発振する。このことを逆圧電現象と呼ぶ。この現象を利用して一定周期のクロック周波数を作っている。用途によってさまざまなサイズのものがある。

写真 1:水晶振動子の例

改造済みニンテンドーDSL の紹介

今回はニンテンドーDSLite をオーバークロックした。誰でもできるように本体の水晶振動子を取り外さずに、並列に接続する手法をとった。さらに、水晶を２つ搭載することにより、1.7・1.4 倍速を自由に切り替えられるようになった。

1.7 倍速ではすべてのソフトに対応できないため、1.4 倍速モードも搭載している。一部カードのチップが１．７倍速に対応しないため、このような策をとっている。

写真 3:速度を切り替えるスイッチはペン収納部の横にある。これで3つの速度を操ることが可能だ。

写真 2:実際に改造された NDS。
外見はスイッチ部分以外に変化はない。
写真は DSL だが、DS でも同様の改造は可能。

写真 4:内部には水晶振動子 2 個を取り付けてある。両面テープなどで固定してある。

実際の使用感レポート・レビュー

1.7 倍速ともなると、大きな差を感じることができた。画面の動きに合わせて音楽も高速化されているためだと思われる。実際にポケモンなどの育成系ゲームをプレイしたところ、戦闘などで高速化を体感することができた。

また、シムシティでテストしたところ、重いものの、日付が速く進むことを確認した。ロード・セーブも多少高速化されているので、大変快適だ。一部、1.7 倍速では起動できないカードが存在するが、1.4 倍速でカバーしているため、問題は感じられない。ただ、スイッチ部分の見た目が変化してしまうのは避けられないし、半田付けが必要という点では敷居が高く感じられるかもしれない。手間がかかるのは事実だが、相応の魅力を持つ改造であることは間違いないだろう。（喜虎）

表 4:速度によるメリット・デメリット

	メリット	デメリット
1.7倍速	最高速度を目指せる。	起動しないカードがある。
1.4倍速	ほぼすべてのソフトを起動できる。	
通常の速度		

ゲーム中にもスピードを切り替えられるのが特徴。高速化時のロード・セーブ・wi-fi・通信対戦・スリープも問題なく行える。時計は高速化されず、通常通りに進む。

詳しい改造方法はこちら
nds.jpn.org ゲームボーイを調べてみる。
http://nds.jpn.org/

ソフト班

明治大学エレクトロニクス研究部

ソフト班について

ソフト班では、プログラム言語を用いてゲームなどのソフトウェアを制作しています。使用するプログラム言語に指定はなく、自分の好みで自由に選ぶことができます。

ソフト班もハード班同様に、5 月から 7 月にかけて新入生を対象にソフトゼミを行っていますが、ソフトゼミは A と B に分かれています

ソフトゼミ A では C 言語という基本的なプログラム言語を用いて、プログラミングの基礎を学びますが、ソフトゼミ B では A で学んだ知識を用いて、実際のゲーム制作に必要な画像の表示やテクニックを学びます。

ゼミが終わり次第、ソフト班員は実際にゲームを作り始めます。また、定期的にソフト班会議を開き、進行状況を見せ合って意見交換などをしています。

11 月初頭には部内発表会を行います。サークルメンバーや OB に自分の作ったものをプレゼンテーションして様々な意見を貰い、その意見を元に更に良いものに仕上げていきます。

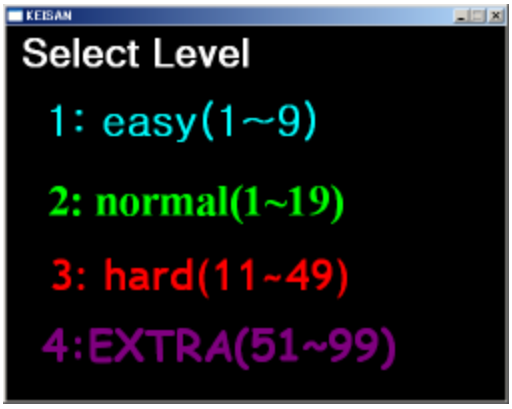
今年も様々なゲームが揃いました。新しいジャンルに挑戦したものや特殊な入力機器を使ったもの、携帯ゲーム機で動くものまであります。是非、ソフト班が頑張って制作した作品をご覧ください。

THE かけ算

ひたすらかけ算をして、60 秒で何問解けたかを競い合うゲームです。4 段階のゲームレベルがあります。

操作方法

レベルの選択（1～4）・答えの入力・・・テンキー
答えの確定・・・エンターキー
答えの訂正・・・できません
その他の操作・・・ゲーム画面に従う



開発環境

C 言語 ＋ SDL
（ゲーム等のマルチメディア関係のソフトウェアを開発するためのライブラリ）

努力した所

ただ計算を繰り返すだけでは面白味があまりないので、正解数が上がっていくごとに背景の画像を変化させたり、コンボ数を入れてみたりしました。また、今までプログラミングの経験がなかったため、わからないところがたくさんあり、様々な本を読んで少しずつ作りました。

反省点

初めは答えの訂正の機能やランキング機能を実装しようと考えていましたが、まだプログラミングに慣れていないため実装することができず、最低限の機能をつけただけになってしまいました。次に作るソフトではより発展的な機能をつけたいです。

おわりに

　似たようなソフトがたくさんありそうですが、自分なりには予想以上のものができて満足しています。少しでも楽しんでいただければ幸いです。

画像提供

無料素材・WEB デザイン　フリー素材屋 Hoshino
（http://www.s-hoshino.com/）

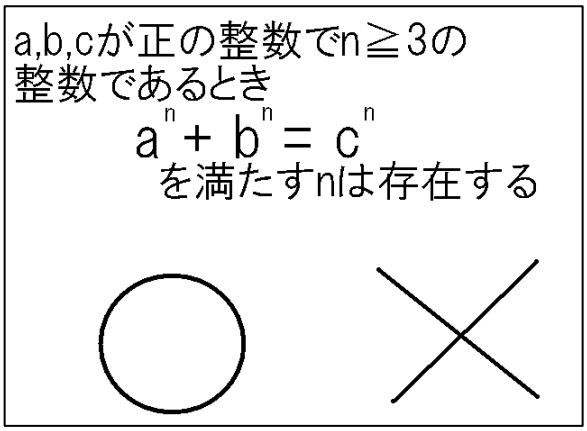
○×クイズやらないか？

画面に出てくる文章が正しいなら○、間違っているなら×を選び、正解数を競う。作者の趣味全開の理系の問題が出題される

操作方法

十字キーの左右で○×を選んでください。左が○で右が×です。

その他の操作は随時画面に表示されるようにしています。



開発環境

SDL・・・ソフトゼミで教わったのでそのまま流れで使用するに至った。

（SDL とは Simple Directmedia Layer というライブラリで、画像の表示などゲームを作るのに比較的便利である。）

挑戦してみたかったこと

タイピングゲーム

当初はタイピングゲームを作ろうとしたがいざ構想を練り始めるとどのようにプログラムを組めばいいかネットで調べても全く理解できなかったので断念し、現在の二択クイズに落ち着いた。

難易度

クイズをやるにあたって難易度を程よく高いものにしたつもりだが、成功していないようだ。

多人数プレイ

クイズを一人で黙々と解いていくだけでは面白く無さそうだったので二人で対戦できるようにしたかったが、アイデアと技術が足りず、現段階では実装できず。

工夫したところ

問題など文字の表示はペイントで作った文字の画像を画面に貼り付けるという方法をとった。そのため問題を 1 問追加するのにも手間が掛かってしまうのがこの方法の欠点である。問題をランダムに出題するために rand() をはじめ使ったのだがなぜか決まった順番で問題が出題されてしまったので、経過した時間を元にランダムな値を得てそれに対応した問題を表示するという形に変え、そしてうまくいった。

得点をテキストファイルに書き出すようにした。現段階では特に何にも使えないが後になにかに使えたらと考えている。また処理を可能な限り関数として main 文の外に出しできるだけ可読性を高めた。初心者の私にとって困難を極めたがある程度までできた。

反省点

自分では問題文などの文字の表示が一番の問題ではないかと思われる。現段階はペイントで作った問題文の画像を貼り付ける方式、SDL には直接画面に文字を表示することができる仕組みがあるのだがどういじっても表示されない。この問題が解決できなかったことが悔やまれる。

次にゲームがあまりに質素であること現段階でＢＧＭなし、効果音なし、エフェクトなし、面白さなしと、ないない尽くしである。クイズの問題数も少なめなのでこれから総数で６０問以上になるようにしたい。

今後の課題

何度もしつこいが文字の表示を何とかしたい。

クイズを二択だけでなく四択や、連想、そしてできるなら当初の目的であるタイピングで回答できるようにする。

二人でもできるように改造し、早押しなどで対戦できるようにしたい。

得点の算出の改善。現在一問正解で１０点となっているが、これを回答に掛かった時間を元に加減算し、よりゲームらしく体を整える。

ただでさえ地味なクイズゲームな上にエフェクトが全く無いため、非常に寂しい。よって少しでもエフェクトをつける。

得点をランキング形式で表示したい。

クイズを通して体験者に「科学って面白い」と思わせ、少しでも興味を持ってもらいたい。そのためにクイズの内容、形式や解説など充実させたい。

感想

　大学に入って初めてプログラムに触れ、実際に作品を制作し、プログラムの難しさ、奥深さ、楽しさを体感できた。アイデアはあってもそれを実現する技術が無いことにもどかしさを感じたり、何度繰り返してもエラーを吐かれて絶望したり、同じ一年のゲームをみてあまりに自分とレベルが高くて愕然としたり。つらいことはたくさんあるが思惑どおりにプログラムが動いたときはやっぱりうれしい。これからも勉強し、知識をため、技術を身につけ、思い通りにプログラムを動かせるようにしたい。

戦略シミュレーション（笑）

このゲームは、自軍のコマを操作して敵軍を全滅させるか、フィールド上の領地を敵よりも先に占領する事を目的とします。おそらく『戦略シミュレーション』に分類されるでしょう。某ファンタジックシミュレーション RPG に触発されて製作しました。

はじめてのゲーム制作だったので、まずはとにかくゲームとしての形にすることを目指して作りました。「とりあえず動くこと」「ある程度のプログラミングの技術を身に着けること」が目標でした。

開発環境

VS.net 2005、素材は Photoshop Elements 2.0 にて自作しました。

基本ルール・操作方法

前述した通り、このゲームの最終的な目的は以下の２つです。この条件のどちらかを敵より先に満たせば勝ち。逆に敵に先を越されたら負けです。

1.敵軍を攻撃により全滅させる
2.フィールド上の陣地（５箇所）を占領する。

ゲームがスタートすると、まず左上のコマ（１Ｐとする）を動かせます。画面右下の矢印のボタン（①）を押すと、このコマが１マスずつ移動します。このコマが陣地のあるマスに辿り着き、１０秒経過すると、そのマスの陣地を占領します。これを全ての陣地で行い、全て占領すれば勝ちになります。一度敵に占領された陣地も、同様に奪い取ることも可能です（もちろん逆に敵に奪われることもあります）

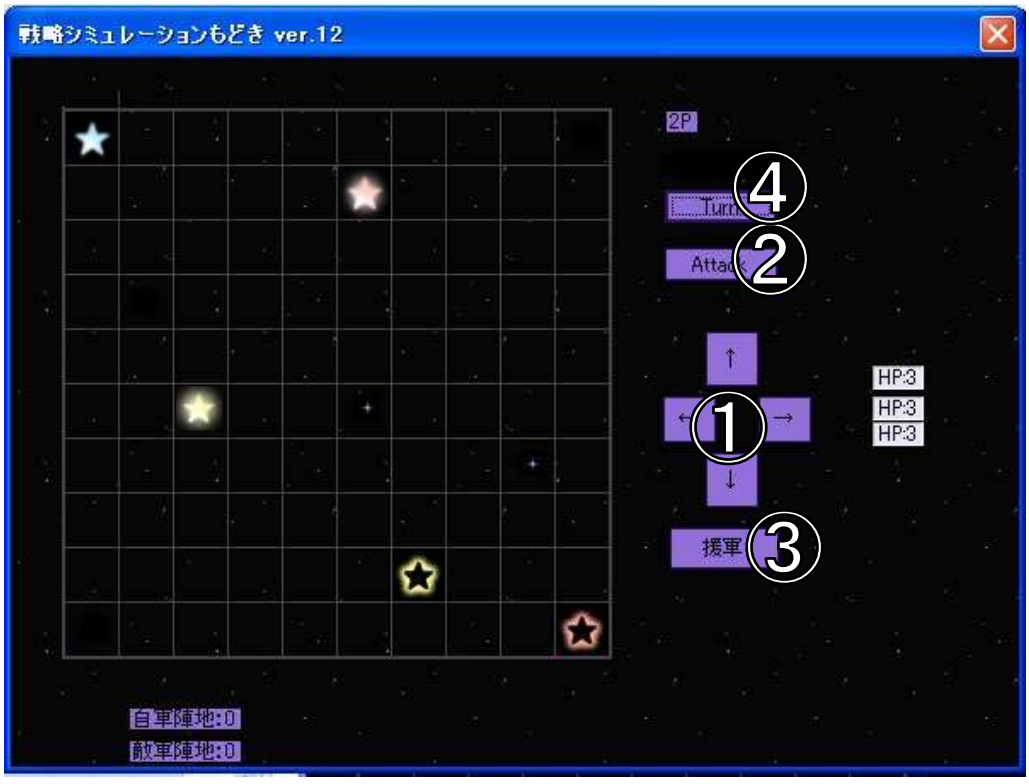
自機コマに隣接したマスに敵がいれば、Attack ボタン（②）で攻撃します。

②ボタンを押すと、敵がいる方向に対応した①ボタンの色が変わります。その状態で①ボタンを押すと、敵に攻撃します。これを繰り返し、敵の HP が0になれば敵を倒せます。（おそらく倒す前にこっちが先に死にます。しかも、倒してもすぐ別の敵が湧いてきます。敵が強すぎる気がするのは仕様です）

左上のマスが空いていれば援軍ボタン（③）で自軍が増えます。最大５機まで増やせます。

また、移動や攻撃の時には「行動値」という値を消費します。最初は行動値が３あり、移動で１、攻撃で２消費され、値が0以下になったら行動出来なくなります。この場合、Turn ボタン（④）を押すと行動値が回復し、再び操作可能になります。さらに自軍が２機以上ある場合、操作するのが別のコマになります。

敵は勝手に移動・攻撃・増殖・占領します。増えるとスピードが速くなるので（仕様です）、早めに決着をつけた方がいいかな・・・と思います。



座標の管理

座標の管理（どこに何のコマがあるか、敵はどちらの方向にいるか）にて二次元配列を使いました。

zahyou[10][10]という配列を用意し、例えば１Ｐが（3,4）にいたらzahyou[3][4] = 1; を代入する。右に移動するときは zahyou[4][4] == 0 （０：マスが空になっている）かを調べてから、１を代入して・・・と処理しています。おそらく初歩の初歩だと思いますが、ややこしくて私にとっての第一の難関でした。

敵の動き

１番苦労した所です。持てる知識をフル活用して試行錯誤を繰り返した結果、ソースを見ても何をやっているのかよく分からないという非常事態が発生しました。

敵の行動（移動・攻撃）は全てタイマーを使っています。最初は１秒に１回の行動ですが、そのまま敵が増えると動きが鈍くなるので、敵の数に反比例してタイマーの時間が短くなるようにしました。

また、どの敵が動くか、どの方向に動くか等は完全にランダムです。そのため、壁に向かって進み続けたり、全く見当外れの方向に突き進んでしまうことがあります。手を加えて、もう少し頭のいい行動が出来るように改良する事が今後の課題です。

今後の課題

結局、最後まで「クラス」が理解できませんでした。約７ヶ月間かけて勉強した末、「とりあえず大切なものだよ」って事は分かりました。次の製作までにはなんとか習得したいです。お・・・オブジェクト指向・・・？

また、操作もクリックではなく、キーボードで出来るようにすれば、もっと快適に遊べたのではないかと思いました。

細かいところを挙げるとキリがないのですが、とにかく全体的に勉強不足でした。精進します。

ぼくのげんかいあくしょん

このゲームは、UFO をほかの敵を踏みつけたり、回避しながらステージの奥にあるゴールポイントをめざすアクションゲームです。難易度は単純だけど結構きつめです。

操作説明

メニュー/ゲームオーバー/クリア画面
[ENTER キー]ゲーム続行
[Esc キー]ゲーム強制終了
(ゲームオーバー画面のときのみ[SPACE キー]でも可能)

プレイ画面
[矢印キー]左右の移動
[SPACE キー]ジャンプ
[Esc キー]ゲーム強制終了
(逃げ出したかったらどうぞ)

ゲーム説明

基本マ●オと同じようなゲームです。敵は何回か踏めば倒せます。踏みつける際に[SPACE キー]を押しっぱなしにするとジャンプの反動が大きくなります。これを利用しないとクリアできない場所があるので覚えておいてください。
とがったブロックにさわると死にます。敵に横からぶつかっても、敵に上からさわられても死にます。

プログラムの簡単な説明

ステージは、同一フォルダ内にある txt ファイルから読み込む形をとってあります。

txt ファイルの中身は、数字と！とアルファベットの小文字・大文字だけで作られておりそれぞれ、[0]は何もないところ。[1]は通常ブロック。[2]は画面上では目視できないけど一部の敵キャラクターの動きの制御に関わったブロック。[！]は自分キャラクター初期位置の座標。[アルファベットの小文字各種]はそれぞれ雑魚 a から雑魚 f までの対応した敵のマップ上での初期位置の座標。[G]はゴールポイントの座標。となっています。

これらの読み込んだ座標を 48 ドット×48 ドットでくぎられた配列に数値をよみこんでステージを作ってます。

敵の動きは、ただ決まられたところを決められたようにしか動かない単純なものです。一応全部の敵の動きを書いておきます。

1・・・崖または[2]ブロックがあるところまでひたすら直進する敵
2・・・一定の場所を地面からの一定の距離に達するか、天井にぶつかって下に向かって移動を開始して、地面についたら再び上に向かって移動する敵。
3・・・一定の場所をジグザク飛びはね続ける敵。
4・・・空中を[2]ブロックがあるところまでひたすら直進する敵。
5・・・空中などで[2]ブロックにあたることによって向きを変えて上下する敵。
6・・・壁などにぶつかるまでひたすら横に跳ね続けながら進む敵。

本当は他にも自分と相手の距離によって動きを変化するやつや、弾を撃ってくる敵も作ろうと思いましたけど、それらはめんどくさかったため作ってなくて、全員タックルのみです。ただ、自分より敵のほうが動きを早く設定してあるのでタックルだけでもなかなか強いはずです。ちなみに敵は殺しても殺しても、初期位置の座標が画面外に出ると自動的に復活するようにできてます。

開発環境

Borland C++ compiler(コンパイラー)
SDL(ライブラリー)
PNGediter(png ファイルへの変換)

反省点

最初、ステージをすべて bmp ファイルに直線やら塗りつぶしとかですべて描いたりしてました。かなり時間がかかったのに座標とかでの制御もしてないので壁への判定は必然的に全部壁の座標で制御され、左右への壁の当たり判定だけでも if 文が１００行を余裕で超してました。またそのときにリアルでのバイト先で客さんの対応に腹立って、憂さ晴らしになぜかプログラムを全部消してしまって、最初の時のプログラムときはまるまる２週間以上無駄にしました･･･。

次に 1 から作ったときに、「また永遠と if 文書くのがめんどくせぇ」と思って考え方を変えて配列を使うようにしました。ただ、まだファイルの読み込みを知らなかったので、壁の上下左右の判定のためのステージの配列への値入れに for 文やらで、ここもまた 100 行近くになってました。

そのあと、先輩からアドバイスをもらって、ファイルからステージの配列への値を入れるようにして、ステージを書くのも背景とブロックにだけ書いてそれを読み込んだとおりに設置するって型。つまり今のスタイルになりました。ただここまでできるのに 2 ヶ月以上を費やしていたのでもっと早く気づくべきだった･･･、とかなりへこみました。

今後の課題

このゲームのジャンプは重力が実装されてないので常に速度が一定になっているためなかなか動きが不自然になっているのが自分でも残念。敵の動きも弾を打ってくるやつとか、もっと複雑な動きをするやつを作れば良かったとすこし後悔。。。なので、今度もう一回アクションを作ってみたくなったので、そのときはちゃんと重力なり、敵が自分の位置に合わせた動きとかするようにしたりと、すこし複雑にしていきたいと思ってます。

シューティング（？）ゲーム

明治大学エレクトロニクス研究部に入り、プログラミングを教わってから始めて作ったゲームです。実際に教わった言語はC言語でしたが、今回はHSP（Hot Soup Processor）で制作しました。

アイディアがなかなか纏まらず、“アクションにしようか？シューティングにしようか？”ぐらいの所から始まりました。結局シューティングにしようと思い至った際、どうせならまぞげーにしようと決めて作り始めてしまった為、今回のような作品になってしまいました。

開発環境

Hot Soup Processor ver3.1

ゲーム内容

・上から敵機が出てきて特攻してくるのを避けつつ撃退していくシューティングゲーム･･･のつもりです。
・敵は特攻してくるだけです。弾は撃ってきません。
・敵が速かったり奇妙な動きを見せるのは仕様です、がんばって避けましょう。
・当たり判定は相当シビアです、でっかいです。
・自機が常に動いていないと得点が減点されます。
・100機墜とすと一機増えます（最大5機）。
・敵に当たったり、得点が-400点になると一機減ります。
・全機墜ちてしまうとゲーム終了です。
・一応、5千点に到達すればクリアになります（というよりあまりにもクリア出来ない為、初期設定の1万点から下げました）。

操作方法

Esc キー（START ボタン）：終了
↑↓→←（方向キー）：方向移動
スペースキー（ボタン 4）：自弾発射（←押しっぱなしで出続けます）
* () 内はパット使用時

制作にあたって

敵の動きに変化をつけたかったのでランダムで横に動くように設定しました。

また、マゾゲーにするために常に移動していないと得点が減点されるようにしてあります。それに際して、「方向キーが押されていない時に減点する」とすると逆方向の方向キーが同時に押されても減点されなくなってしまうので、その時にも減点されるようにプログラムを工夫しました。

反省点など・・・

まあ色々思い至る節はありますが･･･。まず大きなものとしては、本当はボムのようなものも使えるようにしたりボス的なものも用意したかった等、色々やりたいことはあったのですが、途中でハード班の制作に重点をおいた為此方の制作に時間を割けなくなってしまい、夏を過ぎたあたりからは殆どプログラムを弄れなかったので、細かい調整ぐらいしか出来なかった事です。それに伴い BGM も入れることが出来ませんでした。

また、マゾゲーにしようとして作っていた為、クリアがどう考えても無理なゲームになってしまったことです。初期設定の関係で敵が速くなり過ぎていたり、ゲーム開始時に動いていないとすぐに墜ちてしまったり...etc...（実際にゲームをやってみた人から“訳が分からん”と言われ続けてしまった・・・）多少改良はしたものの、まだ難しいようです・・・提出の間際になって敵の速さがおかしい原因が分かったのですが、改良する時間が無かったため放置してあります。

今後の課題

フルスクリーン化： ゲーム自体がまだ小さいので本や HSP のヘルプを見ながらやろうと思っています。

敵の速度、動きの調整：反省点でも書いたように今の状態だと敵が速過ぎるので普通の速さに戻す予定です。座標の初期設定に問題があるようなので、敵の動きの部分だけ初期設定の枠から外せば直ると 考えています。

敵の弾の発射：現在敵が特攻してくるだけなので、変化をつけるために必要な要素だと思いました。自弾の発射と同じ要領で設定すれば出来ると考えています。

ボムの使用：途中まで出来ているので、後は当たり判定をつければ使用可能になります。ただ、まだ単純なものなので改良しようと思っています。

難易度の変化：敵の速度を調節して難易度を変える予定です。

等々他にもありそうですが・・・とりあえず思いつくのはこんな感じになっています。

参考

HSP3.0 プログラミング・テキスト ちょっとずつ作っていこう (HSP 3.0)
http://www005.upp.so-net.ne.jp/h-masuda/hsp30/index.html

狂気のシューティング

明治大学エレクトロニクス研究部（以下エレ研）に今年入部し前期と後期の半分を使い最初の個人制作物を作りました。その内容、制作過程、来年の抱負を今回の機関誌としてまとめました。

コンセプト

ゲームを１つ作るにあたり自分が一番好きなシューティングを作る考えがありましたが、エレ研内でも同様にシューティングゲームを作る人が前期の制作方針発表の時点で多かったため自分は他に無い自機が弾を出さない避ける専門のゲームを作ることにしました。

自機が弾を出さないと制作初期から決めていたときに敵機の弾に拘る事も同時に考えました。夏休み序盤でシューティングゲームとしての基本はできましたが貴重な夏休みのペース配分を間違え後期は締め切りに追われる状態になりました。（詳しくは反省点で）

挑戦したかったこと

・最近のシューティングゲームのような非直線的な弾
・ほどよい難易度

操作方法

十字キー；
対応する方向に移動

開発環境

DX ライブラリ
Microsoft Visual Studio 2005

弾の軌道

敵の弾の軌道は相手と自分の現座標を保存しその２つの座標から１つの軌道を作り、さらにそこからラジアン単位で傾きをずらし、もう１つの軌道を作るという工程を基本として計 4 種類の弾の軌道を作ることができた。

反省点

弾の軌道に関しては満足できる程度まで作ることが出来ましたが難易度の調整が難しく単純に弾を増やす、速くするだけでは調整が難しく、そのさじ加減に苦労しました（納得できる難易度まで調整できませんでしたが･･･）

また、すべての弾を同じ画像で使っているため軌道の違いが見ただけではわかりにくいもので、その違いをわかりやすくするため交互に２つの画像を弾に使うことを目指しましたが、理解できず挫折。

このような繰り返しによるロスのため締め切りまでに完全にゲームを完成させることが今年の最大の反省点です。

来年の抱負

期限を守る

エレ研内で利用した SDL を DX ライブラリの代わりの用いることで上記のロスをなくしたりエレ研内でも相談しやすかったかもしれない。

DX ライブラリも独学のやっつけであったためもっと多くのことに利用したい。

LOVELESS　DEVIL

文章を読み、進むサウンドノベルゲームです。選択肢によってエンディングや、ストーリーが変わります。内容は、ファンタジー系ノベルです。主人公を中心とし、数人のキャラと物語りを進めていきます。

操作方法

ENTER、もしくは左クリック　　文章を進める
右クリック　メニュー
（メニュー画面では、左クリックで操作します）

開発環境

吉里吉里　KAG３
BLUE　iiEditor110
Photoshop　CS
吉里吉里／KAG 用開発環境

やりたかったこと

前回より CG にカを入れたかったです。枚数とか･･･CG 集の機能もつけてみたかったので、つけてみました。あとは、メニュー画面とか･･装飾系。メニュー画面では、文字の速さの設定とか、最初に戻れる機能とかをつけたかったです。

それと、OP ムービー。これだけはやってみたかったので、入れました。
それと、ボイス。入れてみたいです。

反省点

全体的にまだストーリーが書き終わってないです･･。それと CG 集の機能もまだ未完です。キャラ絵も枚数的に全然足りてないので、もっと増やさないといけないと思ってます。

それとスキップ機能をつけていなかったので、OP ムービーでスキップできずに困る事態が多々あったり･･･。あとは、ボイスがやっぱり無理でした。

3×3

3×3 は NintendoDS(NDS)向けに作られたゲームです。実機やエミュレーターなどで動作します。

開発環境

DevkitPro + PAlib

実行環境

M3 DS Simply + NDS Lite
R4 + NDS Lite
などで実際に動作することを確認しています。

ゲームのルール

主に操作はタッチペンで行います。

タイトル画面
どこでもいいので、下画面をタッチすれば、ゲームがスタートします。

ゲーム画面
上画面には下画面と同じような 3×3 のマスに 1～3 までの数が書いてあります。下画面のマスをタッチして数字を変えていって上画面と下画面のマスをすべて同じにしてください。同じにすると得点が入り、次の問題が上画面に表示されます。これを 60 秒間の間どれだけ出来るかを競います。
下画面に九つ表示されているマスをタッチすれば数字が 1→2→3→1…の順で変わっていきます。

リザルト(結果)画面
どこでもいいので、下画面をタッチすれば、タイトル画面に戻ります。

「3×3」でやりたかったこと

家庭用ゲーム機で動くゲームを作る

今回このゲームを作った一番の動機がこれです。パソコンでゲームを作るのはそれほど難しくありません。開発環境も色々ありますし、ゲーム開発ライブラリも色々ありますが、家庭用のゲーム機の開発環境は一般に公開されていません。しかし、~~暇人が~~有志がオープンソースの開発環境を作り上げました。このことを知り、

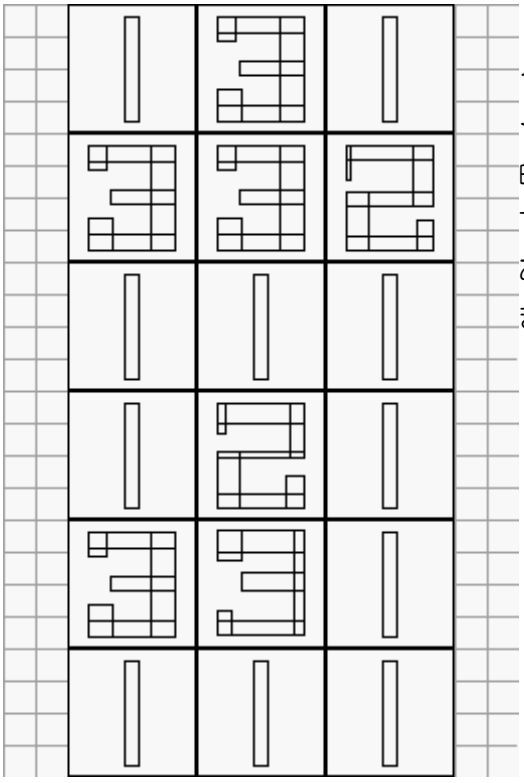
「じゃあ試しに作ってみよう！」

と思って実際に開発環境と実行環境を整えて開発してみました。

特殊な入力方法を使う

キーボード入力やゲーム用のコントローラーはもう飽きた。

~~SP なんて機種としては何の魅力も感じない。~~



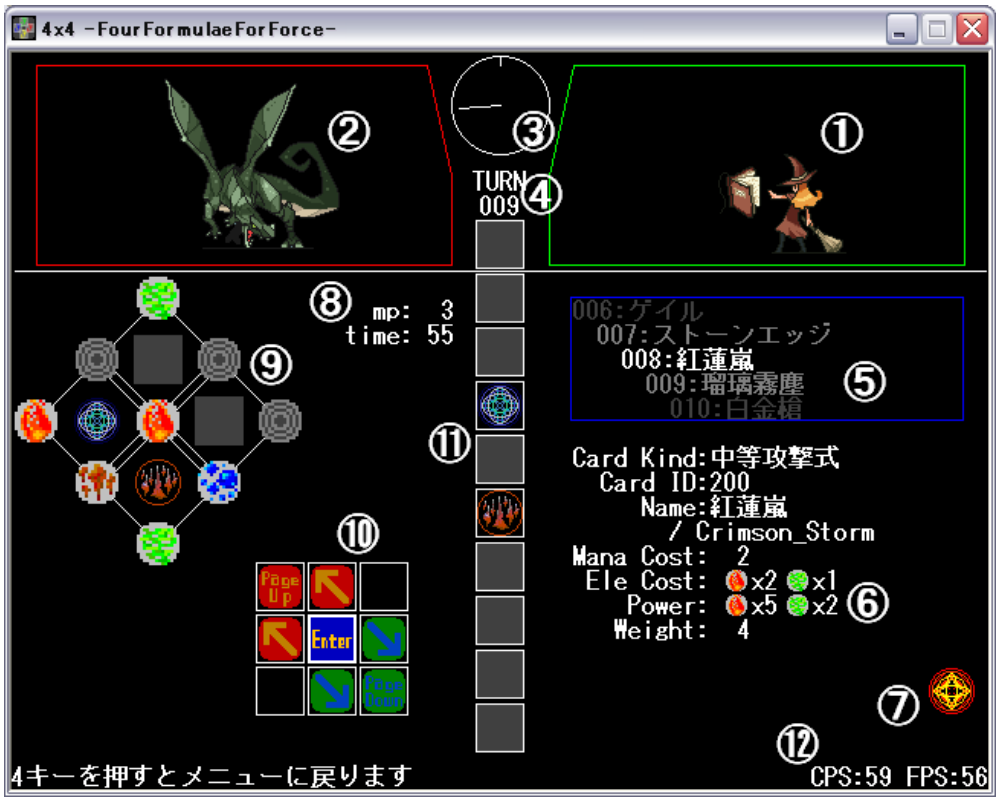
新たな入力機器を試してみようという動機もありました。タッチパネルはゲームの入力機器としてはなかなか魅力的で色々アイディアが思いついたのですが、今回実現できなかったことが多々あったので、そこは自分の力不足を感じました。

4x4 -FourFormulaeForForce-

「4x4」は、トレーディングカードゲーム的な何かを目指したゲームです。
色々やりたいことを求めすぎて大風呂敷を広げてしまい、結局今年だけでは完成させることは出来ませんでした。

というわけで、今回の展示では現在出来ている一部だけを公開という形に。現在プログラムソースが所謂スパゲティプログラムだったり、実現したいことのビジョンが見えなくなったりしているので、近々企画の洗い直しをするつもりです。
来年の今頃には、きっと完成版が…! ~~出来たら良いなぁ~~

操作方法 (操作並びに画像は開発中のものであり、以降変わる可能性があります)



#基本操作

	パッド	キーボード
決定	1	Z
取消	2	X
中断	3	C
選択	十字ボタン	カーソルキー
強制終了		ESC

#戦闘画面

①	自キャラ画像。
②	敵キャラ画像。
③	ターンタイマ。 この針が一周すると、1 ターン経つ。
④	ターンカウンタ。現在のターン数を表示。
⑤	ページ表示。
⑥	式詳細表示。
⑦	式アイコン表示。
⑧	mp,time 表示。
⑨	エレメントと式のフィールド。
⑩	コマンド選択。
⑪	式ストック。
⑫	CPS と FPS の表示。

式:この世界で魔法を使う為に使われるもの。転じて、魔法そのものを指す。

⑩のコマンド選択では、取りたいコマンド方向の選択ボタンを押したまま決定ボタンを押すことで選択します。

また、選択対象となっているものは、青い四角で囲まれます(画面では、ページが選択対象)。選択対象は、以下のように変化します。(決定で矢印方向へ、取消で左へ)
自キャラは、1 ページに 1 つずつ式が書かれた魔導書で戦闘します。従って、ページを選ぶことは式を選ぶことと同義です。

ルール

1)基本

1 ターン(3 秒)経つと、mp が 1 増え、⑪の式ストックが 1 つ上昇します。

式ストックには、宣言された式が順次溜められ、一番上まで到達すると発動します。(現在のバージョンではメッセージが出るだけですが;)

2)エレメントの宣言

エレメントの宣言は、1mp を消費します。mp が無いときは宣言できません。エレメントフィールドは全部で 9 つあり、何処に宣言するかは自由です。また、既にエレメントがある場所への宣言も自由です(この場合、既にあったエレメントは上書きされ、新しく宣言されたエレメントに置き換わります)。宣言されたエレメントは、永続的にフィールドに残ります。

3)式の宣言

式の宣言には、式詳細に書かれている Mana Cost 分の mp を消費します。式フィールドは全部で 4 つあり、何処に宣言するかは自由です。しかし、エレメントとは違い、**既に式のあるフィールドに上書きして宣言することは出来ません。**

式は宣言されると、その Weight 番目の式ストックに溜められます。このとき、この式は式ストックに割り込むように溜められ、それより下(Weight 番目を含む)にストックされていた式は下に押しやられます。これにより、**Weight が重い式の発動は軽い式の宣言によって邪魔をすることができます。**

4)式の発動

1)でも書いたとおり、式ストックの一番上まで到達した式は自動的に発動します。発動に成功したか否かにかかわらず、発動した式はフィールドから削除されます。

このとき注意しなければならないのは、**式が発動する時点**で、その式が宣言されたフィールドの周り 4 つのエレメントフィールドに、式詳細の Ele Cost に書かれたエレメントが無ければならないことです。

例えば、式の Ele Cost：火 x2,風 x1 の場合、

成功

失敗
(火が足りない)

成功
(水があっても関係ない)

となります。発動する時点でエレメントがあればよいので、式の宣言後にエレメントを宣言してもなんら問題ありません。また、エレメントフィールドの中央、左上、右上、左下、右下のエレメントは、複数の式で共有しますので、宣言には注意が必要です(特に**中央は 4 つの式全てで共有します**)。

式の発動に成功した場合、相手には Power に書かれている属性分のダメージが与えられます。(現在のバージョンではメッセージが出る d(ry

「4x4」のコンセプト

#「4x4」のコンセプト其の一　～ 運要素の排除 ～

カードゲームでは、普通各自ターンにのみ行動できる、というのが常ですが、それではどうしても先攻と後攻に分かれてしまいます。そして、先攻と後攻でどちらかが有利になってしまうことが非常に多い。これを排除したかった。純粋に、実力だけでの勝負をしたかった。これが、下の「リアルタイム性」にも繋がっていきます。

また、運要素の排除の為、一般的なカードゲームではほぼ有ると思われる「手札」の概念がありません。全ての式は、いつでも使うことが出来ます。しかしながら、1 ターンがたった 3 秒で過ぎ去るので、ページをめくっている時間は非常に無駄であります。ページの配列を決めることから、既に勝負は始まっています(これも使い古された表現だな…)。(現在の仕様では)式は何回使ってもなくならないため、(このままの仕様で行くなら)複数のページに同じ式を置いておくのも一つの作戦になるかもしれません。

#「4x4」のコンセプト其の二　～ リアルタイム性 ～

これは、『バ〇ン･カイトス』シリーズ辺りに影響されてますね。先攻と後攻の有利不利を解消するには、先攻と後攻と言う考え方を排除すればよい、という非常に単純な理由から、リアルタイム性を敷こうと考えました。

スタックとキューを組み合わせたような現在の式のストックフィールドは、「ウェイト」という言葉から連想しました。つまり、「ウェイト(Weight)が重い」ものは下に配置される為「ウェイト(Wait)が長くなる」のです。(実は、これは FF3 の武器･防具のデータの一つ「重量」から来ていたりします。)

#「4x4」のコンセプト其の三

～配置による妙とゲームリソースの管理～

これは、某「カードヒー〇ー」の影響が大きいです。某ゲームの場合、モンスターの配置が非常に重要でした。従って、モンスターの位置を移動できるカードや特技の存在は、ゲームの流れを一気に変えてしまうほどの力を持っていました。また、ゲームリソースの管理を『ストーン』によって行っています。毎ターン『ストーン』を 3 個配られ、それを使うことで、「モンスターの召喚」、「モンスターのレベルアップ」「モンスターの特技の使役」「マジックカードの使役」「マスターの特技の使役」を行っています。また、「マスターの HP」も『ストーン』によって表現され、「マスターがダメージを受ける」「モンスターが倒される」ことでも『ストーン』を得ること出来ます。これは、格闘ゲームの必殺技の概念に近いかもしれません。即ち、「ダメージを受けることでパワー（ストーン）が溜まり、強力な（ストーンを多く消費する）技を発動出来る」ということです。

「4x4」では、毎ターン MP を 1 つ回復し、MP を消費することによって、「エレメントの配置」、「式の発動」を行います。また、エレメントの配置は式の発動条件を満たさなければなりません。これにより、「上手くエレメントを配置すること」は「ゲームリソースの無駄遣いを無くす」と言う意味で非常に重要なものとなります。

現在では、中心のエレメントが 4 つの式全てに使われる為、非常にバランスの

悪いものとなっていますが、これについては何れ解消します。また、今のところダメージ概念を作っていないので、どうなるかはわかりませんが、これも何れ何らかのダメージボーナスを付けたいと思います。

開発環境

Microsoft Visual C++ 2005 を使用。
開発初期段階では、BCC Developer を使用していました。
DX ライブラリ
　http://homepage2.nifty.com/natupaji/DxLib/

クレジット

キャラクタ画像：【R ド】
　http://www.geocities.co.jp/Milano-Cat/3319/
　この場を借りてお礼申し上げます。どうもありがとうございました。

「4x4」のこれから。

「はじめに」でも書きましたが、まず企画の洗い直しをして、ソースの整備をしたいと思います。

また、重要なのが「チュートリアル」の整備ですね。とにかく、私は、操作方法が分かりにくい、とっつきにくいゲームを作ってしまう傾向があります。某「カード〇ーロー」では、「取説いらずの親切設計」と説明書に書いてあるくらい(！)のチュートリアルがありましたから、それを目指しつつ、敵の AI やらカードの種類やらインタフェイスやらを整え、完成できるように頑張りたいと思います。

てとりす

「てとりす」はご存知(?)のとおり某有名落ち物ゲームです。
普通のものと違い、1から3ピースのブロックを使ったものも登場します。

開発環境

BCC Developer
Score Magazine (音楽制作)
PCMPlayer (音楽再生)

操作方法

タイトル画面

BGMを選択する(0-4キー)とゲームが始まります(0はBGMなしですが…)。
BGMの種類（ちなみに、すべて初代GB版の後期バージョンで使用されているコロブチカ を元にしています。）

1. **リコーダー**
2. **アコースティックピアノ**
3. **ハーモニカ**
4. **スチールドラム**

ゲーム画面

左右カーソル	･･･	移動
下カーソル	･･･	速く落下
スペース	･･･	回転
T	･･･	ゲーム終了

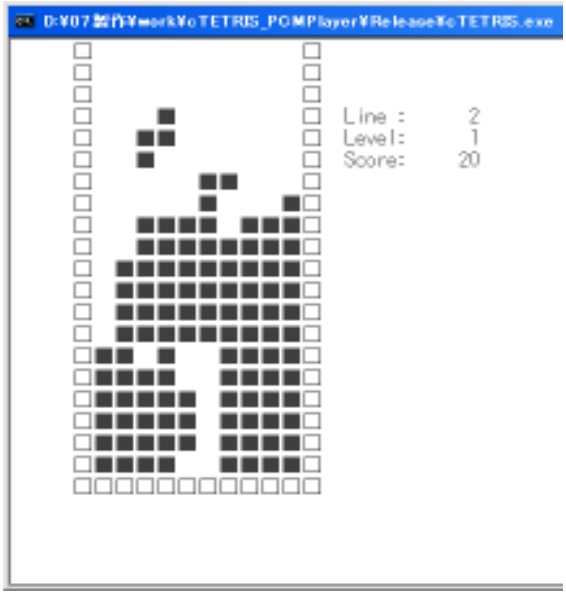
終了後

S	･･･	BGM選択に戻る
Y	･･･	終了

ゲームのルール

開始すると、上からブロックが落ちてきます。移動、回転等してブロックを積んでいき、ブロックが横 1 列に並ぶとその段が消えます。もちろん、2 列、3 列、4 列並べばその段数だけ消えます。

ブロックが上まで積みあがり、新しいブロックが出現することができなくなるとゲームオーバーです。T を押して意図的に終了するか、ゲームオーバーになるまで続きます。
(適当なラインでクリアするモード? 設定していません。)
(時々妙なブロックが落ちてきますが臆せずに。)



回転アルゴリズム

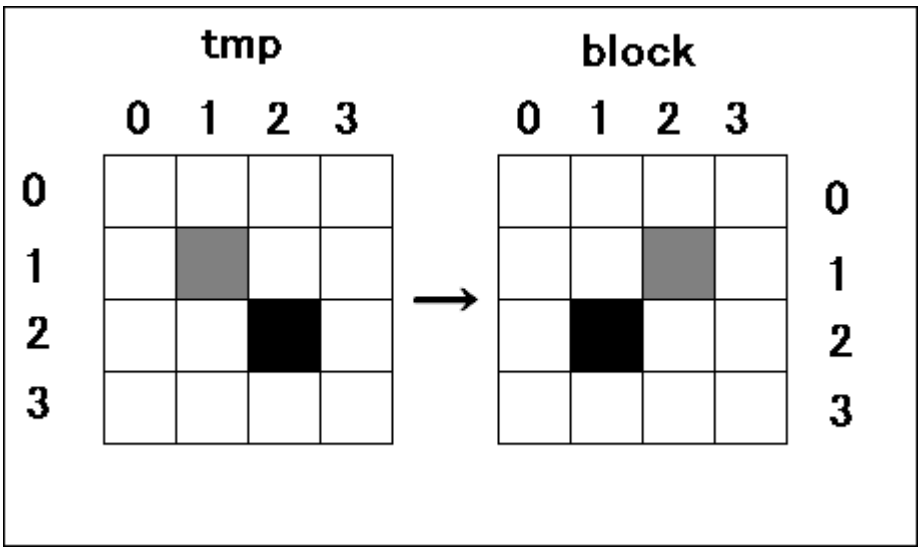
ブロックは 4x4 の配列で宣言してあります。
回転前に一度、今動かしているブロックのすべてのマスの情報を同じ 4x4 の 2 次元配列 tmp に保存し、

```
for(i=0;i<4;i++)
        for(j=0;j<4;j++)
                block[i][j] = tmp[3-j][i];
```

とすることによって回転しています。例えば以下のようになります。

block[1][1]=tmp[2][1]
block[1][2]=tmp[1][1]　(図 1 に灰色で示す。)
block[2][1]=tmp[2][2]　(図 1 に黒色で示す。)
block[3][2]=tmp[1][3]

図 1.　回転アルゴリズム （注意：上の例と左右逆になっています。）



やりたかったこと

1. 次に出るブロック表示　→　いろいろやってみたが上手く行かず挫折 orz…
2. 1 マスのブロックが着地したら爆発　→　いろいろ考えてやめた
（できる限り GB 版に近づけるために）

Orbit

Orbit は万有引力の法則を用いて、惑星の周回する衛星や彗星の動きを再現し、ついでにゲームにしてしまおうという考えから生まれたゲームです。前年の反省から、誰にでもプレイしていただくべく、極力キーボードの使用を控える、ゲームの目的を単純にするといった外面的なこと、ソースコードのメンテナンス性を上げる等の内面的なことの両方にいくつかの目標を設定し制作しました。

開発環境

開発環境はマイクロソフト社の VisualC++ 2005ExpressEdition を用いました。これは無償で提供されています。有償の上位バージョンと比べて多くの機能が制限されているそうですが、小規模な開発ならあまり気にする必要もなさそうです。

また、今回の制作では、DirectX に関するライブラリを一つ使いました。機能は限定されていますが、導入、実装が非常に分かり易いので、C 言語の勉強を一通り終えてゲーム制作をしてみようという方におすすめです。

DX ライブラリ（http://homepage2.nifty.com/natupaji/DxLib/）

ゲーム内容について

Orbit は万有引力の法則を上手く利用し、自機（ボール）をゴールに導くことが目的のゲームです。それだけではどうかと思ったので、何人かで競い合えるようにスコアを算出するようにしています。ステージ内で移動した距離＝スコアとなるため、なるべく長い距離を飛行させてからゴールに入れることができれば、高得点を得られるでしょう。また、ステージが進むにつれて、質量や大きさの違う惑星を複数配置し難易度を上げるといったこともしています。

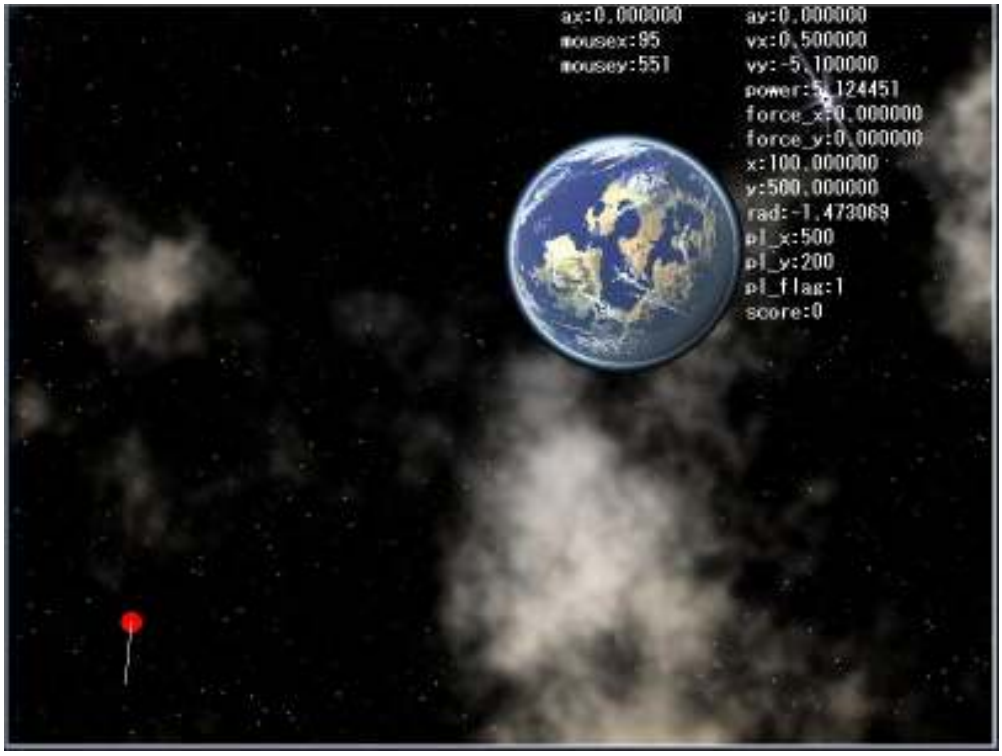
制作するにあたって

制作するにあたって、考えなければならない問題がありました。広大な宇宙空間をどうやって PC モニター内に落としこむかです。色々と述べていく前に、万有引力の法則を忘れたもしくはご存知ない方もいらっしゃると思いますので、ここで一度示しておきます。

$$F = G\frac{Mm}{r^2}$$

F が二物体が受ける力（作用反作用の法則で互いに逆向き）、M と m は二つの物体のそれぞれの質量、r は二物体間の距離、G は万有引力定数です。

とは言っても難しいですよね。今回は**物体が受ける力は質量に比例する**ということがわかっていれば全く問題ありません。



さて本題です。今回は太陽系の惑星の画像を使用したので、データもそれに合わせるつもりでした。しかし、考えていたよりも惑星同士の質量の差が大きかったのです。地球を 1 とすると、水星はその 20 分の 1 倍、木星は 330 倍にもなってしまいました。つまり、上の式に寄れば水星は地球の 20 分の 1 倍の大きさの力でしかボールを引っ張ることができないけれども、木星は 330 倍の力でボールを引っ張ることができると言い換えることが出来ます。

また、惑星のスケールも水星、地球、木星では大きく異なります。地球が野球のボール程度の大きさならば、木星は人間の子供くらいの大きさです。これ程スケール違っていては正直言ってゲームになりませんから、上手くゲーム風に、それでも非現実的にならないようにアレンジが必要でした。

このアレンジ作業はゲーム制作において非常に難しい部分です。あまりに地球との比に値を近づけてしまえば、どの惑星が表れてもボールは似たような動きにしかしませんし、あまり遠ざけると今度はものすごいスピードでボールが画面外へ飛んでしまったり・・・(笑)。

この後、ボールだけでなく、惑星側も動かそうとするとさらに大変なことになりますが、それは三体問題、多体問題というキーワードで検索していただけるとお分かりになるかと思います。

実装部分について

万有引力の法則をプログラムとして実装するのは、あまり難しくありませんでした。分母がゼロになる（二物体の中心間の距離がゼロになる）とどうなってしまうのか、という懸念がありますが、その前にボールと惑星が接触したら処理を止めるということにしてしまえば、これは解消されます。

実際に物体の速度の計算に用いるのは加速度なので、万有引力の法則で算出した力を運動方程式に代入してこれを求めることになります。物体の速度は、加速度の積分で求められるので近似的には毎フレームごとに速度の変数に加速度を足していくことで、ボールの速度を算出しています。

次にステージ部分の実装には、外部ファイルからの読み込みには対応していないので、関数のテンプレートを用意しています。ボールの初期位置を設定する関数、ゴールの位置を設定する関数、ステージに登場させる惑星を設定する関数の 3 つを書き換え、書き足しをすることで、ステージの追加を実現します。設定する項目が非常にすくないので、外部ファイル読み込みへの対応もしやすくしています。

今後の課題

今後の課題です。今回の発表会までに実装できなかった部分が主になります。

スコアランキングの実装:

せっかくスコアをつけたのに、これが無くては・・・。昨年は実装したのですが、適当なプログラム だったため、バグがいくつか発見されてしまいました。クイックソートやインサーションのアルゴリズムを勉強して、実装したいです。

ポインタの有効活用を:

せっかく C++を使っているのだから、ポインタを上手く使えるようになりたいです。Orbit 以前に 作っていたものは、ポインタがらみのトラブルでほっぽり出してしまうという悲惨なことに・・・。

ステージ定義の外部ファイル化:

これもやってみたいことの一つです。Orbit はステージを作る時に設定しなければならない情報 が少ないので、もし実現すれば、無尽蔵にステージが増えるに違いありません。

StackOverFlow 問題

実は Orbit の前に制作していた作品があったのですが、StackOverFlow な問題が長い間解決できず、開発を投げてしまいました。Orbit の開発を始めてからしばらくあって、StackOverFlow な問題の原因と解決法を教えていただいたので、Orbit の内容とは直接関係ありませんが、今後のため（誰のため？）にも機関誌に掲載したいと思います。

Stack(スタック)とは

まず Stack とは何かご存知でない方もいらっしゃると思いますので、これについて説明したいと思います。

Stack とはデータを管理する方法のひとつで、データの追加と取り出しができます。そのデータは、順に下から積み上げられ、取り出すときは上から取り出します。イメージとしては積み木を積むときと丁寧に崩すときというのが近いと思います。

C 言語では、これを意識しなくても、この考え方が導入されています。

Stack の実際

例えば、関数(kansu1)の中で関数(kansu2)を呼び出した場合、Stack は次のようになります。

```
-----------
(1)kansu2
(2)kansu1
-----------
```

次に、kansu2 が実行を終えて kansu1 の処理に戻ったときには、一番上にある kansu2 が取り出され、Stack は次のようになります。

```
-----------
(1)kansu1
-----------
```

kansu1 の実行を終えれば、kansu1 も Stack から取り出されます。このようにして、関数を実行するたびに Stack に積み上げられ、一番後に入れたものが一番最初に取り出されていきます。

StackOverFlow

StackOverFlow とは、Stack に積める関数の量が限界を超えてしまったときに発生するエラーです。前の項では特に触れませんでしたが、関数には関数のサイズがあります。これは、関数内で宣言されている変数の量などで決まります。したがって多くの変数を宣言すれば、関数は大きくなり Stack はより満たされやすくなります。VisualC++においては Stack のサイズは 1MByte と決まっています。この容量を超えるような関数を呼び出したり、前の項で挙げたような〃関数内で関数を呼び出す”ことを非常に多く繰り返すと、結果として StackOverFlow が発生してしまいます。これが発生しやすい状況としてはクラス型変数を大量に宣言したときなどがあります。

現実にある物に例えると、天井の低い部屋（Stack）で積み木（関数）をどんどん積み上げていくと、最後には天井に達して積めなくなってしまうような状況が StackOverFlow と同じと言えます。

解決法

原因によって解決法も変わってきます。ここでは、いくつかの原因に対する解決法を掲載しておきます。

１．クラス型変数が原因になっているとき。

クラス型変数は、C++の非常に便利な機能の一つです。しかし、多くの機能を一つのクラスに盛り込もうとすると、クラス型変数の容量は増加していまいます。すなわち、クラス型変数を定義した関数は大きくなってしまうのと同じです。これが問題かと思われそうですが、クラス型変数が大きくなってしまうことには余り問題はありません（多分・・・）。問題になってしまうのは、変数宣言の仕方をよろしくないからです。

例：あまり良くない（？）クラス宣言の仕方

```
class A{
        double a;
        double b;
～～～～～～～～～～
};
int main(){
class A[1000];
～～～～～～～
}
```

クラス型変数が大きい場合、このような宣言の仕方は StackOverFlow を誘発します。次にその解決法を示します。

解決法：

```
class *A[1000];
for(int I = 0; I < 1000 ; i++){
        A[I] = new class();
}
```

どちらも同じ A[1000]という変数が作られる点では同じですが、main 関数のような、最後に Stack から取り出される関数などでは、変数を Stack に置かずに、ヒープというメモリ上の領域に置く方が都合が良いのです。それはなぜでしょうか。Stack の容量は 1MByte と決まっているというのは前々項で書きました。一方でヒープ領域は PC に搭載されているメモリと同じだけ使うことができるからです（実際には半分が良いところでしょうけど）。最近の PC は数 GB にも及ぶメモリを積むことも多くなってきましたから、ヒープ領域は Stack の容量に比べて遥かに大きいので、こちらを有効に活用すべきでしょう。

2.関数を大量に呼び出している時

例えば、ある関数（kansu1）の中でさらに関数(kansu2)を呼び出すという操作を考えてみます。Stack は最後に入れたものが一番最初に取り出されますから、1 回目では、

```
-----------
(1)kansu2
(2)kansu1
-----------
```

のようになり、さらに kansu2 の中には kansu3 を呼び出すというプログラムが書かれていたとすると、

```
-----------
(1)kansu3
(2)kansu2
(3)kansu1
-----------
```

当然このようになります。

kansu3 が実行し終わるまでは、kansu1 も kansu2 も実行を終えられないので、Stack に残留し続けることになります。ここから先、kansu(n)の中には kansu(n+1)を呼び出すというプログラムが書かれていたとすると、

```
-----------
(1)kansu(n+1)
(2)kansu(n)
～～～～～
(n)kansu2
(n+1)kansu1
-----------
```

このように n+1 個の関数が Stack に残留することになります。これが関数を呼び出し過ぎている場合にあたります。このように無限に関数を呼び出してしまうプログラムを意図的に書くことは実際には無いでしょう。しかし、ミスや条件の指定ミスでそのような状況に陥ることは少なからずあると思います。この場合に起こる StackOverFlow の解決には、処方箋はありません。この例のように、関数内でもう一度自身を呼び出すというのはテクニックの一つとしてありますし、非常に便利な面もあります。ですから、プログラマ自身がこのようなコードを減らそうとしなければ根本から無くすのは難しいと思います。

まとめ

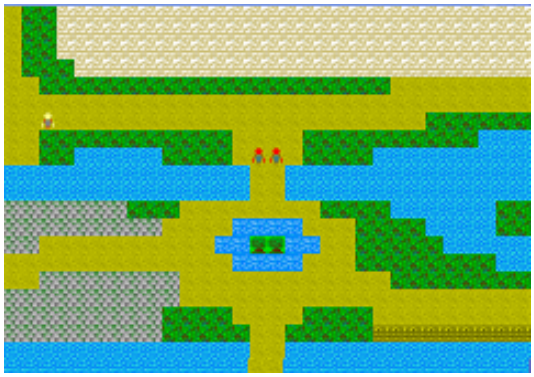
StackOverFlow は Stack の容量を超えるような巨大な関数を実行したり、非常に多くの関数を Stack に溜めてしまうと発生する。解決法としては、new 演算子を用いてヒープ領域に変数を置くことや、関数から関数を呼び出し続けると言った動作のプログラムを書くことを避けると言ったことが挙げられる。

ただいま修行中 II

シミュレーションゲームという名のノベル、シミュレーション、アクションの要素が含まれているハイブリット、しかし基礎的なソースでのみで作られたゲームです。

ルール・操作方法

以下のいくつかの「モード」があります。



・ロールモード

スキップはできません。
自動的に次に進みます。

・タイトル画面

↑ ↓で移動。Enter で決定。

・ノベルモード

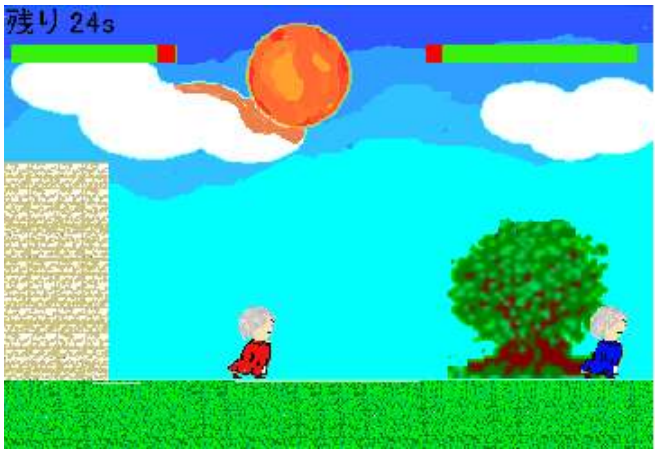
Enter で進む。

・シミュレーションモード

Enter で「移動」コマンドが出現。もう一度 Enter で移動範囲が表示されます。そのとき↑、↓、←、→の 4 つで進む方向を決めます。間違えたら Backspace を押すと元に戻ります。決めたら Enter で移動します。移動後ターンが経過し敵も動きます。

ターン経過後また Enter を押すと「移動」コマンドが出て、移動できます。

敵とプレイヤーが重なると戦闘開始です。



バトルモード

←、→で移動、↑で跳びます。長く↑を押すと高く飛べます。敵をかわしながら攻撃して画面上の敵の HP を 0 にすると勝てます。プレイヤーの HP が 0 になったり、画面左上の制限時間が 0 になると最初に戻ります

攻撃は Tab で火の玉発射、Ctrl を押しっぱなしにすると火の玉がプレイヤーの周りを旋回します。マップ上にいる敵を全て倒すとエンディングロールが流れます。覚えなくても画面の下の方を見ていればヘルプメッセージが出ています。

やりたかったこと

市販もされている某シミュレーションゲームに似たものを作ろうとして、ここまでしかできませんでした。本当はストーリーも書いていたんですが、それも中途半端だった上にストーリーにあわせて絵を書く必要があることに気づいて挫折しました。

やり残したこと

これでストーリーが作れればよかったと思うのですが・・長くなるので。絵が相変わらずなのは勘弁してください。著作権のこと考えると作ったほうが、 と思いまして。

一応サブプログラムの土台はできたので、これを肉付け(応用)できればまだまだよくなると思うのですが・・すみません。敵が特攻野郎なのは・・仕様です。

開発環境

Hot Soup Processor 3.0

ラバーリングアクションゲーム つくってみた

ゴムロープを投げてゴールを目指すゲーム。ゲームセンターＣＸ第５シーズンで課長が挑戦した「海腹川背」に魅了され、勢いだけで“似たようなもの”を作ろうと決心。主人公は単なる棒人間。せめて海腹川背さんのように名前を付けてあげようと思ったが、

考えよう！→・・・→ （ ﾟдﾟ)ﾎﾟｶｰﾝ →・・・→マンドクセ

の繰り返しにはまったのであきらめました。使用言語はHSP3.1。

ルール・操作方法

敵キャラや障害物を避けながらゴールとなる看板を目指してステージをクリアしていきます。踏み潰すのはもちろん、ロッ○バスターなどで敵を倒すことは出来ません。

・ロープ未使用時

←　：　左に歩行
→　：　右に歩行
○ボタン・・・決定ボタン
×ボタン・・・ジャンプ
□ボタン・・・長押しでロープを出します

・ロープ使用時

←　：　左方向に力を加える
→　：　右方向に力を加える
↑　：　ロープを伸ばす
↓　：　ロープを縮める
□ボタン　：　離すとロープを放します

※コントローラーのボタン数によって変わるので、KEYSETTING で変更してください
※ボタンの位置はPS コントローラーに対応

プログラムについて

・ゴムロープの動作

このゲームのメインとなる処理。作る前はＨＳＰで処理しきれるのか？と思ったが、簡単な物理法則で処理させたため計算量が少なくすみ実現できた･･･。バネ係数や主人公の重さ、重力加速度など数値を適当に設定したためか、ある条件の下でバネがとてつもなく早く伸び縮みしてしまう場合があるので、ある速度でロープが伸びきったときにロープを切るという処理を加えた。決して係数の調整やプログラムを書き直すのが面倒だったからではない。

・ブロックにロープがひっかかったときの処理

プログラム全体でこの処理が一番時間をかけたと思う。ロープがブロックの角に引っかかる際、ロープの長さ、ロープにかかる力などを計算しなおす必要があった。計算内容は秘密。たぶん間違ってるから(笑)。ギリギリ違和感のない動作をしてくれたかと思えば、階段状のブロックの角に２点以上引っかかる場合に問題が発生。ロープがブロックから離れる際の処理が原因でいろいろと試行錯誤し、これもなんとか違和感のない動作を実現できたと思う。

・敵キャラの動作

敵キャラはHSPDXFIXのスプライト機能を使用。これにより当たり判定や、キャラクタの管理（キャラのアニメーション、キャラの新規登録、削除)、移動の制御が簡単に行える。特にキャラのアニメーションを容易に実現できるのはアクションゲーム製作にとってかなり役に立つものだった。ただ敵はブロック上を左右に歩くだけで単調な動きとなってしまったので、今後作り直す機会があれば敵の動作も豊富にしていきたい。

やり残したこと・・・

ステージ分岐

海腹川背の特徴のひとつ。ステージのゴールは２つあり、分岐していく。これにより最終ステージに早くつけたり、遠回りしてしまったりする。面白そうだったので、実装しようとしたが・・・、分岐できるほどのステージを一人で作るのが大変だったので実現できず。

移動する台

これを作ればもっと充実したステージがつくれたのだが、ロープの処理がさらに複雑になるので断念。

敵キャラ

踏み潰したり、ロッ○バ○ターで殺したりするのは教育上よくないので、本家のように敵はあくまで“捕獲”することによってステージ上から削除したかったが、ゲームの製作熱がなくなってしまったため敵キャラについては放置。敵に当たるとティウンティウンします。

ロープが出る位置

おなかから出ます(笑。キャラはプログラムができてから描いたため、プログラム組んでいるときは一応中心からでるようにしていました。ロープの出る位置を変数で作成していなかったため直すのが面倒・・・。ロープを投げるアニメーションもつくらないと・・・

バグ

ロープがブロックの角にひっかかり、その後逆方向に進み、ロープがブロックの角から離れる瞬間に、変なところにロープの表示が・・・。

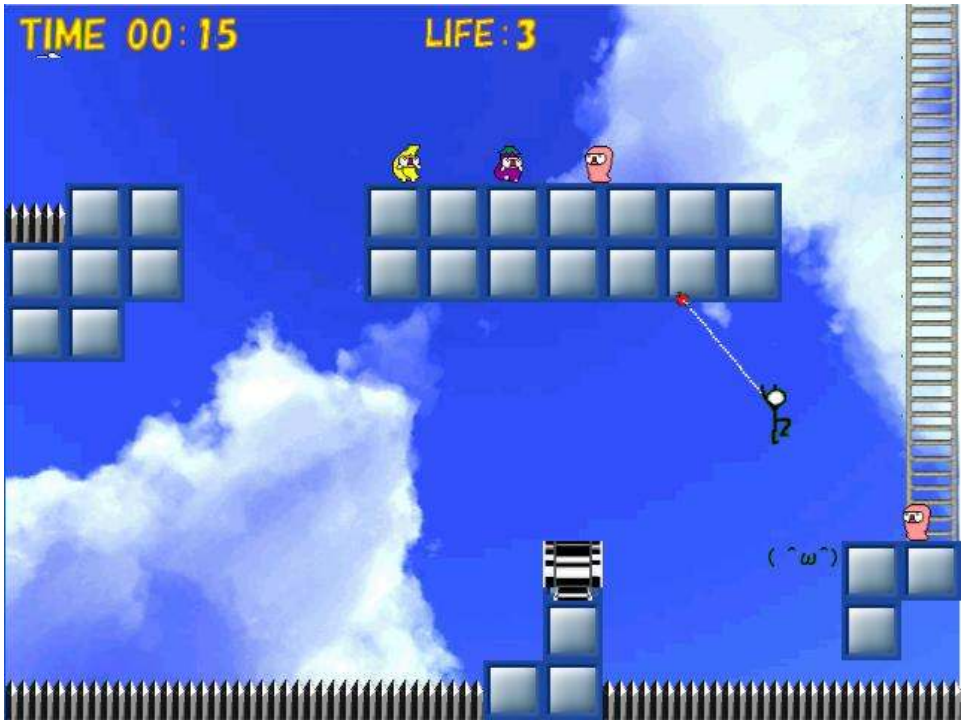
予想外な動作

落下中にロープを投げると、落下速度がリセットされる模様。これによりより遠くへ飛ぶことができる。直そうと思ったが、これはこれでありかなと思ったので放置。でもこの技を使うとゲームが簡単になってしまうんだ・・・

Windows VISTA への対応

(´・ω・`)･･･。

ゲームプレイ画面



敵はそれぞれ、

“そんなバナナギコ”(左)

“ボケナスギコ”(中央)

“たらこギコ”(右)

画像は　http://matsucon.net/　から拝借しました。ブーンの顔は大砲の弾の代わり。トゲにあたるとティウンティウンします。

音楽は今年もマッチメイカァズさんにお世話になりました。

海腹川背について

「海腹川背」は、1994 年 12 月に株式会社 TNN からゲーム参入第一弾の作品として発売されたスーパーファミコン用アクションゲーム。続編はエクシングから「海腹川背・旬」「海腹川背・旬～セカンドエディション～　マル安シリーズ⑴」が発売されている。こちらはプレイステーション用ソフトである。

主人公は「海腹川背」という名の１９歳の女の子。ゴムロープの先端に付いたルアーを障害物に引っ掛けてゴールとなるドアを目指す。ゴールは決して１ステージ１つあるのではなく、ゴールとなるドアが二箇所以上置かれているステージもあり、ステージが分岐する。中には相当なテクニックをつまないとクリアが非常に困難なステージが存在する。

伸縮性のゴムロープを使用することで従来のアクションゲームでは不可能だった様々な動きができ、幅広いアクションが楽しめる作品となっている。またニコニコ動画や youtube などで最速クリア動画があげられており、タイムアタックという遊び方が見受けられるようになった。

ちなみに株式会社TTNは「海腹川背」のあとに、「史上最強リーグセリエ A エーススストライカー」というサッカーゲームを出したが、その後ゲーム業界から撤退している。

俺の限界シリーズ final

俺の限界シリーズ final ～俺の限界 pop'n music～(以下『俺限 final』)は、ニコニコ動画で BEATMANIAⅡDX コントローラで「フタエノキワミ」を演奏していたのを見て感動して作ってみたものです。手元に pop'n music のアーケードスタイルコントローラがあったので、このコントローラでも同じような動作をさせることができるだろうと思い、作り始めました。

開発環境

メインスクリプト :Hot Soup Processor v3.0
SE,BGM :Sound Engine Free
DPP ケーブル :第三科学研究所製　PS 用 DPP ケーブル 2P

ゲーム内容について

俺限 final には譜面が存在しません。それは作者がド下手だから譜面が無いとか、制作に充てる時間が足りないなどという理由ではなく、初心者にも受け入れてもらえるように、という考えからなのです。コナミ社製大型メダルゲーム機「ARABIAN CRYSTAL」では 2 ボタンのポップンのミニゲームがありますが、それすら不自由なユーザーさんが多いのが事実。譜面なんか必要なくて、直感で叩いて遊んでいただく事の方が有意義だと思ったためです。本当です。本当ですってば。

ソースのほとんどはフェードアウト処理などばかりで、そこまで難しいプログラムではありません。唯一苦労したのはコントローラと PC をつなぐ DPP ケーブル及びドライバの入手でした。現在使っているのは第三科学研究所の"PS 用 DPP ケーブル 2P"です。このケーブルはプリンタポートを使っているためほとんど遅延がなく、ハードな使い方でも非常に使いやすいものになっています。ただし、コントローラの反応限界(スイッチ的な意味で)があるため、アーケードのように叩くのには向いていません。正直言って。つーかヤフオクで 3 万も出すくらいなら新品買ったほうが何かと良かったんじゃないのかと当時の俺に言ってやりたい。

操作方法

選曲画面 :青ボタンで左右、緑ボタンで SE 選択と BGM 選択の切り替え
プレイ画面 :BGM に合わせて叩くもよし、フリーダムに演奏するもよし
飽きたら START ボタンと SELECT ボタン同時押しで戻ります

今後の課題

1.BGM,SE のラインナップ
　一般の方にも受け入れられるラインナップにする。
　また、新しく声を録音して収録数を増やしたい。
2.卒業する

誰でもできる(かもしれない)NDS ソフト開発

この文章は、略すと NDS となるゲーム機の開発環境の構築について書いてあります。この文章で NDS の自作ソフトが増えることを願います。

コンパイラなどのインストール

コンパイラのインストールは専用のインストーラを使うことによって簡単に行うことができます。

http://www.devkitpro.org/category/devkitpro/ の Downloads から devkitPro Windows Installer をダウンロードしてください。ダウンロードしたものを実行すればインストールが始まり、コンパイラやライブラリ、エミュレータなどがインストールされます。使い方はページに収まりきれないのでここでは省略しますが、コンパイラは gcc なので gcc が使える人ならばそれほど難しくないと思います。

情報資料の少なさに絶望した！

NDS 開発はまだまだマイナーです。ただでさえ情報も少なく、日本語のものとなると数えるほどしかありません。某巨大掲示板のプログラム板に専用スレッドがありますので、分からないことがあればそこに質問してみるのもいいかもしれません。**間違っても京都にある N 社に問い合わせないように！**

ライブラリを使ってみる。

NDS 開発は難しいです。プログラム言語は C/C++ですが、専用の知識などが求められる為、はっきり言って簡単に開発できるものではありません。そこで、ある程度簡単に開発する為のライブラリがありますのでここで軽く紹介しておきます。

SDL DS port

とっても有名なマルチメディアライブラリ SDL の NDS 移植版です。SDL の知識がほぼそのまま使えます。ただ一部の機能がまだ未実装だそうです。

PAlib

NDS 専用ライブラリ。NDS で出来ることはほぼ網羅しています。ただ仕様が独特で初心者にとっては使いやすいが、中級者以上には細かいところが気になるという仕様になっています。また最新のコンパイラの仕様変更に対応していない為、現在のところ最新のコンパイラではこのライブラリは使えません。(対応する予定はあるそうです。)

最後に

簡単にですが NDS の開発環境の構築ついて書きましたが、これだけではわからないことが多いと思います。詳しく知りたい方はインターネットなどで情報を探してみてください。

z_

ざ_

ざk_

ざkk_

ざっき_

雑記_

雑記。_

エレ研・御用達飲食店紹介

我々エレクトロニクス研究部(以下エレ研)は、神奈川県の生田にある明治大学生田キャンパスに本拠を構えています。

この生田というのが中々の陸の孤島で、一般に名の知れたお店はそんなに多くありません。ですが、近隣に何も無いというワケでもないので、そんな「隠れた名店」チックなお店をご紹介していきたいと思います。

大学周辺

まずここで特筆すべきは我が生田キャンパスの正門を出てすぐ目の前にある喫茶店「タック」です。ここは立地とも相まって、完全に大学生をターゲットとしていて、安い・うまい・多いと 3 拍子揃った、若者の胃袋とお財布の両方を満足させてくれる喫茶店です。ステーキ皿に 20cm を超える高さまで山のように盛られたスパゲッティは圧巻の一言。

ちなみに、エレ研の優しい先輩は『昼休みが終わった頃』にやってきて、後輩に大盛りスパゲッティを奢ってくれます。ありがたく食べ切りましょう。残したら…ｱｯｰ

次に語るべきは「OK ストア」でしょう。チェーン店ではあるものの、欲しい物は何でも揃うかと思えるほどの豊富な品揃えと、買い手であるこちら側がどうやって利益を得ているのか疑問に思うほどの低価格っぷりは、多くの働いたら負けだと思っているエレ研部員の強い味方です。

「はなまる食堂」というお店もあります(チェーン店という話ですが筆者は他店舗を知りません…)エレ研部員はここでのんびりとビールを嗜むこともありますが、日頃の鬱憤を晴らす場としての、通称「はなまる会議」が行われることもあります。先輩を前にしては言えないようなこと…後輩に対する愚痴…等々。筆者自身ももしかしたら標的にされているかと思うと…ｶﾞｸｶﾞｸﾌﾞﾙﾌﾞﾙ…

生田周辺（駅的な意味で）

隣駅・向ヶ丘遊園には、かなりの量の信者を抱える「蓮爾(読み：ハスミ)」なる二郎系列のラーメン屋があります。

はっきり言って味は人を選びますが…選ばれてしまえば幸せになれます。店頭に『うちのラーメンは量が多いので初めての方には小ラーメンをおすすめします』といった感じの張り紙がしてありますしね。

まぁエレ研部員には狂信者が多いので、中には『大ラーメン』を完食する猛者もいれば毎週通う猛者もいますけれども。

さらにその隣駅・登戸には、チェーン店ですがエレ研が大きな飲み会を行った後によく行く「カラオケの鉄人」が。22 時過ぎまで飲んだ後に、0 時から 5 時までのフリータイムで徹夜カラオケをするんですが酔っているせいで明らかに皆のテンションがおかしいことが多いです。

飲み会にはいなかった社会人の先輩がいつの間にか混じってたり、この前もテンションに任せて『曲名に x の文字が入っている曲』縛りが行われたり…色々楽しいことがあります。

新宿(店名省略…筆者が忘れたとも言う)

新宿のラーメン屋の『北極ラーメン』は毎年恒例のメニューです。どこが北極なのか全くわからない程の激辛ラーメンで、もしや北極で食えば防寒着がいらないとでも言いたいのか！と思うほどなのですが…これがウマイんですよ。唇が痛くなること請け合いですが本当にウマイ！これも『心優しい』先輩が『何も知らない』後輩に奢ってくれます。

一通り書いてみた感想

1 ページ目はノリノリだったんですが…途中でネタが尽きたので 2 ページ目が全然書けず、レイアウトで誤魔化してしまったのが心残りと言えば心残りです。まぁあれですね、エレ研部員は大食いが多いですね。

というか安くて量の多い店ならどこでもいいんじゃねぇのかこいつら…

生田病

出典:『エレペディア(Elepedia)』

生田病(いくたびょう)は、生田農工大学[1]の、特にエレクトロニクス研究部の男子学生に多く見られる症状の総称のこと。正確には病気ではなく症候群に近いものであるため「生田症候群」とでも呼称すべきだが、一般的にはこの名称で広く知られている。他の理系大学でも似たような症候群が存在することが知られているが、名称を統一しようという動きは今のところ無い[要出典]。

症状

生田病の症状は多岐に渡り、全てをここに書き記すことは出来ない。よって、ここでは症状の一例を挙げる。感染者は以下の全て症状をあらわすとは限らないが、普通複数の症状があらわれる。人によっては、デメリットだけではなく、メリットとなる症状があらわれることもある。

・絵に描かれた人を「二次元」、現実世界の人を「三次元」と呼ぶ

・オタクではないひとを「一般人」と呼ぶ

・どのような女性も可愛く見える

・ネットスラングが日常会話に混じる

・「一般人」とカラオケに行くことが困難になる

・働く気が起きなくなる

・散財が激しくなる

・「蓮爾[2]」と呼ばれる食べ物への執着

・タイピング速度が速くなる

・ゲーム、アニメ、漫画等の知識量が膨大になる

・2/14 や 12/24・25 等の日になると気分がひどく落ち込む

また、感染者は「一般人」との接触を避け感染者同士で行動したがる傾向が見られる為、症状が進行しやすくなるのも特徴の一つである。

感染源

生田病の感染源は、未だにはっきりとは解明されていない。しかしながら、感染者には次のような共通部分があり、それが感染を引き起こすのではないかと考えられている。

・女性の極端に少ない環境

・限定された空間に多くのメディア(特に、パソコン、漫画雑誌、ゲーム機等)が存在する

・レポート提出や授業、或いは生田坂[3]による疲弊

しかしながら、「感染したから、周囲の環境がそのように変化するのではないか」との意見もあり、この議論は平行線を辿っている[要出典]。

また、感染者には、一般入試やセンター試験利用入試ではなく、AO 入試や(特に付属校からの)推薦入試で大学へ入学を決めたものが比較的多いことが分かっているが、これが生田病にどのように関連するのかも解明されていない。

予防手段

現在有効な予防手段は存在しない。生田病の感染者は、異口同音に「いつの間にかなっていた」「気付かぬうちになっていた」と唱える上、感染源がはっきりしないからである。

「ゲーム、アニメ、漫画等との接触を控える」ことが予防に繋がる、という説があるが、「それができるなら生田病は存在しないだろう」というのが大方の意見である。

治療法

生田病に対する治療法は確立されていない。これは、感染者が生田病であることを自覚しているにもかかわらず、治す意思が見られることはほとんど無いためである。

よって、これからも治療法が確立されることは恐らくないであろう。

脚注

1. 明治大学生田キャンパスの皮肉込めた言い方。

2. 蓮爾　登戸店のラーメンのこと。初めて見る人は大抵その量に圧倒される。脂分の多いスープに詰め込まれた麺は割り箸ほどに太く、その上に柔らかく煮込まれた豚肉、更にその上に茹でもやしが山のように盛られる。

3. 生田キャンパスが高台にある為、生田駅から来る学生が登らなければならない大学の入口から建物までの急な坂の通称。

関連項目

・バレンタインデー中止のお知らせ

・2 次元と 3 次元の狭間

この項目「生田病」は、書きかけの項目です。加筆･訂正などをして下さる協力者を求めています。

アキシブ系って何ですか？

どうも、某エレ研民に「**属性は音楽**」と言われたでじたるじゃんき～というものです。この文章はアキシブ系という音楽ジャンルについて調べてみたものです。アキシブ系という言葉を wikipedia で偶然発見し、主なアーティストの項を見て

「なんで俺このアーティスト全部聞いたことあるのに、
このジャンル名知らないんだ？」
ということで調べてみました。

アキバ系+シブヤ系=アキシブ系？

Wikipedia によるとアキシブ系とは渋谷系を源流とした音楽性やファッション性を持つ音楽が、アニメソングやアイドル歌謡として積極的に用いられることでアキバ系への浸透がすすむ現象を、J-POP のひとつのジャンルとして再定義した造語ということらしい。

これだけ読んでもよくわかりません。

渋谷系の音楽に秋葉系の歌詞を乗っけたということなのだろうか？渋谷系のアーティストがアニソン歌ったらアキシブ系じゃないのか？色々疑問点が出てきましたが、とりあえずアキシブ系のアーティストを調べて聞いていくことにしましょう。

そもそも渋谷系って？

ちょっと話を変えて、渋谷系って何？と思う人が大多数だと思うので書いておきます。Wikipedia によるとそれまでの流行りであったイカ天バンドなどの流れとは一線を画し、1980 年代のニューウェーブや 1960 年代～1970 年代のポップスなどの洋楽を中心に、幅広いジャンルの音楽を素地とした都市型志向の音楽だそうです。しかし、これだけ読んでもよくわかりませんね w

簡単にいうならお洒落なポップスですが、それだけの音楽というわけでもないです。どんな音楽か知りたい人は、フリッパーズ ギター(小沢健二,小山田圭吾(現 Cornelius))のカメラトークをおすすめしておきます。10 年以上前のアルバムですが、すばらしいアルバムです。

そもそも秋葉系って？

今思ったら、秋葉系の音楽ってなんだろ？アニソン？電波？アニソンだと、普通のアーティストまで秋葉系扱いになってしまうし、電波ソングと秋葉系は微妙に違う。

色々考えたんですが、とりあえずここではサブカルチャー(オタク文化)を意識した音楽としておきます。どんな音楽かといわれると困ってしまいますが、知りたい人は、MOSAIC.WAV の Future-Fiction：AKIBA-POP!! をおすすめしておきます。いろんな意味で秋葉系音楽というのを前面に押し出してるアルバムです。

@T○WER の違和感

タ○ーレ○ードのネット通販のサイトである@T○WER でアキシブ系の CD を集めたリストが見つかったので実際に見てみた。

出てきたのは何故か p○p'n music のサウンドトラックばかり…

他にも塊○などのサントラが出てきました。違うだろ、何かが違うだろ…と思いつつ他のアーティストもあったので気になったのを独断と偏見で以下に挙げてみました。

Perfume

最近とっても有名になったアイドルグループ Perfume。音楽に関しては capsule の田中ヤスタカがプロデュースしててクラブ系統のサウンドでかなりかっこいい。アイドルソングなんてどうせ駄目…という人でも大丈夫です。
そういえばニコニ○動画でア○マスの MAD によく使われてました。アイドル繋がり(?)で相性が良かったんでしょうかね？
~~黒歴史扱いらしいが…昔はモモーイともやっていたそうな…~~

YMCK

8bit ポップユニット YMCK。言い方を変えるとファミ〇ンポップユニット。音と歌詞、それとジャケットやブックレットなどがかなりファ〇コンを意識した作りとなっています。

2nd アルバムにはゲストで高橋名人まで参戦。16 連射します！歌います！現在のところアルバムは二枚。どちらもなかなかの良作です。次のアルバムでメジャーデビューするそうです。a〇ex から…それをどう思うかはあなた次第ということで(ry

orangenoise shortcut

渋谷系直系のような爽やかなポップスユニットの orangenoise shortcut。2nd 以降のアルバムは季節感を意識したアルバムとなっているのも特徴だと思います。

どのアルバムもすばらしく、個人的に応援していたのですが、残念なことに 2008 年の 1 月で活動を休止するそうです。ところでこれのどこに秋葉系の要素があるんでしょうか？どの曲も渋谷系音楽そのものです。

結局アキシブ系って？

独断と偏見で三組を選びましたが、他にもアキシブ系と定義されているアーティストはいっぱいいます。今回は紹介していませんが、アニメのキャラソンが渋谷系の音楽だったりするのもアキシブ系と定義されるようです。つまり、一部例外を除いて、秋葉系の要素と渋谷系の要素が入っていればアキシブ系と定義していいみたいです。

次回、日本語ラップとガン〇ムでまたお会いしましょう～(嘘)

編集後記

古今東西、どんな組織に関わらず同好の士集まれば機関誌なるもの成り立つわけで、自分が所属するエレクトロニクス研究部、通称：エレ研においても活動報告をかねた機関誌が脈々と続いています。

今回は編集にあたり、例年のものよりも特に“見られる“ことを意識した作りを目指してみました。読み手側に立ってとって飽きないよう、読みやすいよう心がけたつもりです。多くの試行錯誤を含み、まだまだ至らない部分もあるとは思いますが来年以降の課題として次に続けていきたいです。

多くのコンテンツがある中で、全ての人が内容に満足を覚えることは無いと思います。人にとって“面白さ”や“読みやすさ”は違いますし、載っている“情報の価値”は人によって様々です。中には非常につまらない、読みにくいと感じる方もいらっしゃるかもしれません。しかしそれでも、何か一つでも良いので、この機関誌を手にとって良かったと思える記事があれば幸いです。

最後にこの場を借りて、編集に当たって作業の大幅な遅延にも関わらず、較正や製本作業を手伝ってくださった先輩方、ならびに多くのエレ研民に感謝の言葉を述べたいと思います。

ここまで長々とおつきあいいただき本当にありがとうございました。

エレクトロニクス研究部 機関誌 2007 年度版

発　行　平成 19 年 12 月 15 日 初版

著　者　矢田裕基/吉川祐輔/北見謙典/平林誠一郎 ほか
編　者　関智章
発行者　明治大学エレクトロニクス研究部
　　代表者 菅沼久浩
http://www.isc.meiji.ac.jp/~eleken/

Printed in Ikuta