Lime

Limited Expression Reports

注：このイラストはフィクションであり実在の人物・団体・マスコットとは一切関係ございません。

素人が安くROMライタを作る

野村 賢

平成11年11月11日

1 はじめに

筆者はハードウェアに関しちゃ素人だ。だからお遊び電子工作しかできない。無知ゆえにいらん神経つかうことなくさくさくはんだづけが進む。で、完成すると結構それらしく動いてくれる。また、ICの出来の良さを証明してしまった。

2 なにはなくともROMライタ

電子工作するからにゃ、マイコンやらにゃ、ということでROMライタが必要になる。ROMライタがなけりゃプログラムはROMに手入力しなければならない。Z80をいじってた時のことだが、私はアドレスバスの下から7bit目を立てたところで嫌になった。ちなみに割り込みなんぞに注ぐエネルギーなどもちろんないのでプログラムは先頭番地からだらだら書かれてた。ROMライタは買うと高いので、自作してしまおうというのが今回のお話。

3 EEPROMを探せ

ROMライタを作る前にCPUとターゲットとするROMを決めねばならない。初心者におすすめのCPUは定番ではあるが、やはりZ80だ。資料の充実度と扱いやすさ、入手の容易さなど申し分ない。というわけでCPUは勝手に決めさせてもらった。次はROMだ。まず容量だが、Z80のアドレスバスは16bitで約64Kのアドレス空間をもつ。このうち何KをROM領域に割り当てるかを決めねばならない。(バンク切り替えは今回は置いておく) 例えば1/4を割り当てるなら16kのROMを購入すればよい。私の場合は8KのROMを使用した。これは今から述べるEEPROMが8Kのものしか入手できなかったからである。ちなみに8Kでも不具合はさほどない。で、そのROMの種類だが素人が扱うのに最も向いたROMはEEPROMであろう。例えばUV-EPROMなどのライタには2種類の電圧を作る必

要があるなど素人にとってはじゃまくさいのだ。残念ながらEEPROMはフラッシュROMによりほとんど淘汰されてしまったようで、入手は困難だ。フラッシュROMよりEEPROMが素人向けである理由はRAMとほぼピンコンパチであり内容の消えないRAMとでもおもって序盤を押しきれるからである。以下EEPROMを入手出来たものとして話をすすめる。

4 お値段は？

さて、ターゲットとなるROMが決定したところでROMライタに必要な部品を考えてみたい。言い忘れたがROMライタを作ることを決めたならターゲットとするROMは幾つか買いだめしておくといいだろう。市場にいつまでもあると思ってはいけない。

今回作るROMライタではPCにパラレルケーブルを介して接続する。そこでまずRS232C25ピン全結線ストレートケーブルが必要だ。つづいてROMライタ側にそのオスコネクタが必要になる。そして肝心なのがZIFソケットだがこれが高い普通にかうと2000-3000円はする。関西圏で手にはいる最も安いソケットに水色をした安物(1000-2000円)があるがお勧めできない。秋月で800円で売ってるソケットをぜひGETしたいところだ。ROMライタにかかる費用はこのソケット代だけが問題だ。残りは74HC193(4-bitカウンタ)70円x4とユニバーサル基盤(100-200円)きやすめに74LS14(30円)と500円足らずである。趣味の問題だがROMへの書き込み制御線にLEDをつけても値段はたいしてかわらない。RS232Cコネクタは(100円-700円)で手に入れられる。100円ジャンク基盤などからはずすと安く手にいれることができる。また、カウンタは4つをカスケード接続して16bitカウンタとして使うので8-bitカウンタ2つなど他の組み合わせでもかまわない。以下表1にまとめてみた。
わたしゃ当時いいソケットに恵まれず2000円ほどかかった。

5 製作

パラレルのデータバス8bitはROMのデータ線へそのままつなぐパラレルの制御信号で出力ができるのは4つなので、これをカウンタのカウントアップ、クリア、ROMのWE、OE、につなぐEEPROMのBUSY線は余った4つの入力につかえる制御線へつなぐ。
カウントアップはノイズ対策として7414(Schmitt Trigger Inverters)を通してみた。カウンタとROMのアドレス線の接続はしっかり確認して行うとよい。わたしゃA10とA11が入れ替わっててずっと後になってプログラムサイズ0x400以上のプログラムがうまく動かないという現象に悩まされた。(この法則がわかったのでアドレス線の結線ミスに気付けた)

表 1: ROM ライタ製作に必要な部品と値段

部品名	部品代 (円)	個数 (個)	注釈
RS232C-25pin コネクタ (オス)	100-700	1	特価で結構ある
ZIF ソケット	800-3000	1	秋月で 800 円
ユニバーサル基盤	100-200	1	秋月で 100 円
74HC193	70	4	
74LS14	30	1	
LED	25	1	高輝度青色 390 円
抵抗 (330Ω)	10	1	
リード線、はんだ等	100	-	適当
計	1445-4345	10	

6 書き込みプログラムを作る

読み込みファイルの形式についてだが、べつに俺フォーマットでも良いがインテルヘキサ形式などにしておくと既存の Z80 アセンブラがそのまま吐いてくれて便利だ。インテルヘキサ形式では一行ごとに先頭アドレスが埋め込まれており、行ごとに不連続なアドレスのデータを持っている。インテルヘキサ形式のファイルを扱うときは行ごとの開始アドレスをキーにソートしておくとカウントアップしかできない今回のライタでも高速に書き込みができる。
簡単なプログラムなのですぐにつくれると思うが、注意点としては usleep() などでタイミングをとるのではなく ROM の BUSY 線を ACK として同期をとるのが良い。プログラムの書き方で書き込み速度はかなり改善できるので挑戦してほしい。経験的な話だが読み込みは wait 無しでループをまわしても問題は無いようだ。読み込みはベリファイに使える。ベリファイは書き込み毎に行うのではなく、後から一括して行うのが良い。というのは何らかの原因でカウントアップしなかったり、2 カウントアップした場合の検出が出来ないからである。またしても経験的な話だが書き込みの失敗はカウントアップの失敗が多いようだ。

7 おわり&いいわけ

肝心の回路図はありません。tgif で書くのは面倒だからです。どうしても見たい人は私のノートを見に来てくださいまし。
でも回路図見てしまうとオリジナルなとこ無くなって面白くないぞ！しかしいつのまにやら OB になってしまった。月日がたつのは早いのぉ。今回書いた ROM ライタは春先に作ったもので、これを使ってもっと面白いことしてるんだけども原

稿にする元気がないのですじゃ。

参考文献

[1] トランジスタ技術 SPECIAL-No49 徹底解説 Z80 マイコンの全て CQ 出版者

[2] トランジスタ技術 SPECIAL-No29 マイコン独習 Z80 完全マニュアル CQ 出版者

[3] トランジスタ技術 SPECIAL-No36 基礎からの電子回路設計ノート CQ 出版者

[4] トランジスタ技術 SPECIAL-No38 Z80 システム設計完全マニュアル CQ 出版者

[5] 98 最新 74 シリーズ IC 規格表 CQ 出版者

[6] Application Notes for Character Mode LCDs Debsitron

[7] Z80 逆アセンブラ hoja ver 2.01 UNIX 版 README

[8] Small C version C3.0R1.1 (with ZCC) README

[9] BootStrap PC のハードウェアを理解する pp.50-62

雑記帳 (Athlete になりませんか?)

服部 保

平成 11 年 11 月 18 日

1 はじめに

皆さんこんにちは。電情 4 回の服部です。今回は、最近起こったことについて、考察を加えてみたいと思います。一応、

- AMD、新 CPU Athlon(アスロン) を発表
- intel、新チップセット i820 の 9 月発表に失敗
- intel、新 PentiumIII(開発コード Coppermine カッパーマイン) を発表

という、最近雑誌を賑わせているネタについてみてみたいと思います。

なお、お断りとして、私は AMD シンパなので、随所に intel シンパな方が不快感を催すであろう記述があると思われます。その不快感・ムカツキ等については当方では一切関知しません。intel シンパである自分を呪って下さい。

2 AMD、新 CPU Athlon を発表

以前から K7 というコードネームで開発が進められていた AMD の新 CPU、Athlon が、この 8 月 17 日から日本国内で正式発売の運びになりました。

PentiumII などと形状は同じだが互換性のないスロットを採用したこと (理由は後述) や、AMD しかチップセットのサプライヤがないこと、対応マザーボードの開発状況が不透明だったことなどから、波瀾含みのスタートとなりましたが、それにしてはわりあいと順調な滑り出しでした。発売開始直後に、当時唯一の対応マザーボードだった MS-6167(台湾 MSI 社) の一部ロットに電源関係のトラブルがあり、交換騒ぎにもなりましたが、その後は新規格の出始めにありがちなトラブル以外、致命的なものは出ていないようです。

実際、私は、1 週間後には大学院入学試験を控えているにもかかわらず、正式発売日の朝の開店時刻前に大阪・日本橋のパーツショップに行き、Athlon500MHz とマザーボードをセットで ¥62,000 ほどで購入しました。別段、相性問題等もなく、ちゃんと稼動しています (本稿もその Athlon マシンで執筆しています)。FreeBSD 3.1-Release、OS/2 Warp v.4、Windows98 は正常に動作します。

どれほどのメーカが対応マザーボードを出すか、というのも当初心配されていましたが、本稿執筆時点では、MSI、FIC、GIGABYTE、Asustek、BIOSTAR、FreeWay(パーツショップ TWOTOP のブランド。実際は Asustek の OEM) の 6 社から発売され、今後 EPoX などのマザーボードメーカからリリースが予定されていて、これも一安心といったところです。

さて、Athlonを取り巻く状況の一部を見たわけなんですが、「じゃあ、Athlonってどんなな CPU なわけ？」と思われる方もおられるでしょうから、ちょっと触っておきましょう。

Athlonは、

- 200MHz の CPU バスクロック (バスプロトコルは DEC ALPHA CPU 用 EV6 互換)
- 複数の x86 命令デコーダ
- 128KB のデュアルポート・スプリット L1 キャッシュ
- 3 つの独立した整数パイプライン
- 3 つのアドレス演算パイプライン
- 完全にパイプライン化されたスーパースケーラ out-of-order スリーウェイ浮動小数点エンジン

という技術的特徴を持っています。

命令デコーダや演算パイプラインなどを複数持つことは、最近の PC 向け CPU ではごく当たり前になっているのでよいでしょう。128KB の L1 キャッシュは、PentiumII などに比べて 4 倍大きな容量になっています。命令/データに各 64KB が割り当てられており、メインメモリへのアクセスを減らすのに大きな効果を果たしています。

次にあげられるのは、200MHz の CPU バスクロックです。intel 製 CPU では、バスクロックは 100MHz ですが、その倍の周波数になります。とはいっても、本当に 200MHz のクロックなのではなく、100MHz のクロックの立ち上がり/立ち下がりの両エッジを用いることで、200MHz 相当のクロックとしています。これは、DDR(Double Data Rate) 技術とも言われるもので、ごく単純な仕掛けで高クロックが出来ます。しかも、400MHz までのスケーラビリティを持たせており、来年中にはバスクロック 266MHz や 300MHz の製品も出るのではないかといわれています。

また、バスプロトコル (CPU ―チップセット間のデータ伝送規約) には、DEC ALPHA CPU 用に開発された EV6 を使用しています。DEC ALPHA は、世界ではじめて動作周波数 1GHz を達成するなど、かなり高速な CPU です。そういった CPU 向けに開発されたバスプロトコルを用いることで、更なる高速化にも対応したいとの考えでしょう (実は、他にも理由があるが、後述)。

しかし、Athlon の目玉は、浮動小数点エンジンです。完全にパイプライン化された浮動小数点エンジンは、x86 系プロセッサでは Athlon ではじめて実装されています。パイプラインピッチ 15 段の 3 本の浮動小数点エンジンは、単精度浮動小数点演算で 2.4Gflops、倍精度浮動小数点演算でも 1Gflops 以上 (600MHz 動作時) を叩き出します。

いままでは、「AMD の CPU は整数演算は intel より速いけど浮動小数点演算が遅いから、ビジネスアプリにはいいけど 3D には向かない」と言われていました (特に K6-2 では酷評されることもままあった) が、この Athlon では PentiumII(III) を遥かに凌駕する浮動小数点演算性能を獲得し、3D 関係においても intel 製 CPU に対してアドバンテージを持つに至っています。

と、文章で書くよりも数値を示した方が速いでしょうが、AMD の WEB サイトにベンチマーク結果集があります[1]ので、そちらをご参照頂くことにします。また、雑誌にベンチマーク結果がよく載っているので、それをご参照下さっても結構です。ただし、日経 BP 社系の雑誌のベンチマークは、比較の際あまり使い物にならないことを指摘しておきます (測定環境が明確でなかったりするので)。

[1] (URL: http://www.amd.com/japan/products/cpg/athlon/benchmarks/benchmarks.html)

ただし、Athlonにも欠点がないわけではありません。それは、上に書いたように高性能なCPUにするため、トランジスタ数が2200万にも達しており、ダイサイズ・消費電力(発熱量)ともに非常に大きくなっていることです。ダイサイズ自体は、ユーザ側からは大して問題になりませんが、消費電力の問題はかなり大きな問題になります。というのも、Athlon500MHzでも30Wを超えるという消費電力は、PentiumIIIの倍以上にもなります。この点は、改善の余地があると思いますが、ごく最近のロットの700MHzなどでは、かなり消費電力の削減がなされ、600MHzなみの消費電力に押さえられているようです。また、AMDは、現在0.25uアルミニウム配線プロセスでAthlonを生産していますが、現在、0.18uアルミニウム配線プロセスへの移行を進めており、来年前半には0.18u銅配線プロセスのラインも稼動し始める予定になっており、消費電力の点についてはずいぶん小さなものになるようです。

話は変わりますが、AMDはintel互換CPU業界ではリーダ的存在です。他の互換CPUメーカには、Cyrix、IDT、RISEなどがありますが、いずれも知名度が低く、特に前2社は台湾VIA Technologiesに買収されています。これに対してAMDは、昨年にK6-2搭載の低価格PCが発売されるなどして、知名度は一気に上がりました。とはいっても、実は以前から、高速なintel互換CPUメーカとして一部ではその名を知られていました(ほかにも、プログラマブルロジック製品などでも有名だった)。実際、intelの8086よりもずっと高速な8086を作ったり、動作クロック80MHzの80386(intelのものは32MHz程度まで)を作ったりしていたのです。

なぜ、こんな話を書くかというと、この互換CPUというのが原因で、AMDはCPU事業においては、常にintelとの訴訟問題で揉めていました。この訴訟問題は、80386以降から次第に激しくなり、K5やK6が出る頃にはかなり激しいものとなっていました。もちろん、K6-2やK6-IIIについても例外ではありませんでした。この訴訟で、intelとAMDは一応は和解するのですが、その和解の条件の中に、「AMDの次期CPUでは、intelがPentiumII以降で採用するCPUバス仕様(GTL+)を用いてはならない」というのがありました。実に、これが、AMDがAthlonにおいて、独自仕様のスロット(SlotA)とEV6互換バスプロトコルを採用する原因となったのです。もちろん、この条件は他社には関係ないですから、Slot1互換CPUを作ろうというメーカが現れても不思議ではありません。実際、Cyrix(今はVIA Technologiesに買収されている)は、コードネームJoshuaという、Socket370(Celeronで採用)互換CPUをアナウンスしています。しかし、intel製CPUに対抗しうるCPUとしては、Athlonが筆頭であるのは間違いないでしょう。ちなみに、本稿執筆時点で、流通しているAthlonの最高クロックは、700MHzですが、750MHzを今年中、800MHzを来年第一四半期中に出荷開始するという見通しが発表されています。また、Comdex/Fall'99では、0.18uアルミニウム配線プロセスによる900MHz動作品のデモが行われました。

他にもいろいろありますが、他の項目にも絡みますので、そちらに譲ります。

3 intel、新チップセットi820の9月発表に失敗

AMDが高速な浮動小数点演算性能を売りにして投入してきたAthlonに、PentiumIIIのシェアを奪われるのをintelが黙ってみているはずがありません。まず、9月にはバスクロック133MHz対応のPentiumIII 533EMHzと600EMHz(Eは間違ってついているのではありません)を投入しました。

ところが、PentiumIII用の主流チップセットである440BXは、バスクロック133MHzに対応しておらず、対応マザーボードがない状態でCPUだけがある状態になっていました。intelは、9月末に外部クロック133MHz対応で、さらに440BXでサポートされていないUltraATA/66や

AGP x4 モードへの対応を行った上、新しい高速メモリ規格である Direct Rambus DRAM(以後、DR-DRAM と略記) をメインメモリとして採用することを前提とした新チップセット、i820(コードネーム Camino(カミノ)) を発表して対応する予定でした。ところが……。

発表予定日の 4 日前になって、突如リリース延期が発表されました。その原因は、「3RIMM 構成では、メモリからの読み出しに失敗して正常に動作しない場合があることがわかった」ということでした。この突然のリリース延期は、今まで intel にはあまり見られなかったことです。

このトラブルについてみておくことにします。i820 では、メインメモリに DR-DRAM を採用しますが、RIMM(DR-DRAM を搭載したメモリモジュール) は 3 スロットまで差せるようになっています。ところが、メモリをすべてのスロットに差した場合、最後のメモリチップからの読み出しに失敗するというのです。実は、このトラブルは、DR-DRAM の特徴に原因があるようです。

というのは、DR-DRAM は、600 ～ 800MHz という、とんでもない速度でメモリチップにアクセスします。これぐらいまで速度が上がると、64bit のデータに同時にアクセスする SDRAM のような方式では、それぞれのビット単位のデータの伝送速度の微妙なばらつきが大きく影響しますし、複数のメモリモジュールに並列にアクセスしていては、信号の同期が取りにくくなってしまいます。

このため、DR-DRAM では、同時にアクセスするビット数を減らし、単純な回路にすることで、データ伝送速度のばらつきの影響を押さえ、さらに、データバスをひとつながりにして、メモリモジュールごとにアクセスしないような構造になっています。

しかし、そのために、同じ容量のメモリモジュールを使う場合、データが流れる距離が長くなってしまうため、次のクロックまでにデータの転送が終わらないことがある可能性があります。これが、このトラブルの原因に繋がるわけです。

結局、intel は 9 月のリリースをあきらめたわけですが、これが大きな失敗へと繋がっていると思われます。もともと、intel は、メインメモリの SDRAM から DR-DRAM への移行を強力に推進しており、i820 をその起爆剤にする考えでした。ところが、この延期によって、その計画が崩れてしまいました。しかも、マザーボードメーカは i820 の発表に合わせてそれを採用したマザーボードを発表する算段をし、生産も行っていたのですが、あまりにも突然の延期のために、マザーボードメーカは出荷できない不良製品を抱えこまされる羽目に陥りました。あるマザーボードメーカは、2RIMM までしか差せないようにすれば出荷可能ではないかと intel に食い下がったそうですが、結局通らず、骨折り損のくたびれ儲けに終わったといいます。さらに、この問題は、ごく早い時期から intel は認識していたようで、マザーボードメーカの intel に対する感情は、少なからず悪化したものと見られます。そのうえ、DR-DRAM を供給するはずだった韓国 Samsung と米国 Micron が、相次いで DR-DRAM 供給態勢の見直しを発表しました。DRAM 業界では、メモリチップの大暴落によってどのメーカも採算が合わないため、すぐに売れないものを作る余裕はないとの判断が働いたようです。

ということで、チップセットの発表延期のために、intel はかなりジリ貧の状態に追い込まれてしまいました。ただ、11 月 15 日からの Comdex/Fall'99 において、発表が行われ、マザーボードの出荷も始まりつつあるようです。

ただ、DR-DRAM が今後の主流になるかどうかは、以前不透明です。というのは、DR-DRAM がとんでもなくコスト高なため、普及は大きく遅れそうだからです。128MB の RIMM は、定価ベースで ¥100,000 ほどもします。¥99,800 でパソコンが買えるこの現状で、そんな額の投資をたかだか 128MB のメモリごときのためにする人がいるわけがありません。しかも、RIMM はデータバスがひとつながりになっていないといけないため、使用していないメモリスロットには C-RIMM と呼ばれる配線のみの RIMM を差しておかなければなりません (これも定価ベースで ¥6000 ほ

ど)。さらに、この C-RIMM のよしあしもシステムのパフォーマンスに影響するというのですから、たまったものではありません。そこに追い打ちをかけるのが、DR-DRAM の性能評価です。DR-DRAM は、メモリ性能に依存しないソフトウェア (たとえばブラウザなど) を使用している限りにおいては、SDRAM と同程度かそれ以下の性能しか出ないというのです。また、消費電力も、メモリとは思えないほど大きく、なんとヒートシンクが必要なほど発熱するというのです。つまり、DR-DRAM は高いわりには性能がよくなく、しかも消費電力も大きく効率的でもないということが明るみに出てしまったわけです。これでは、普及しようはずもありません。なぜ、intel が DR-DRAM を普及させようとしたのか理解に苦しむところです。

ここで、なぜ DR-DRAM が高コストなのか、そのわけを付け加えておくことにします。そもそも、DR-DRAM は、米国 Rambus 社が開発し、特許を持っています。したがって、DR-DRAM を製造しようとすると、Rambus 社からライセンスの供与を受けなければなりません。このライセンシの分がまず価格に上乗せされます。

次に、DR-DRAM のテスト機器は非常に高価です。なにしろ、普通の SDRAM なら 100MHz のクロックで試験をしますが、DR-DRAM だとその 8 倍の 800MHz のクロックで試験をしなければなりません。800MHz というと、電磁波では携帯電話などに使われるマイクロ波帯になります。そんな領域での測定機器ですから、当然の事ながら高価なものになってしまいます。このテスト機器の分も価格に上乗せされます。

さらに、DR-DRAM は、設計/製造/販売のすべてのステップにおいて、Rambus/intel による合格認定 (バリデーションといいます) が必要とされます。もちろん、そのバリデーションの取得にかかる費用もメーカ負担となります。

しかし、最も根本的なのは、DR-DRAM 自体の構造の問題です。DR-DRAM は、SDRAM に比べてダイサイズが大きく、1 枚のウェハから採れる数が少ない上に、歩留まりも悪いため、この相乗効果でコスト高になります。特に、歩留まりの悪さは深刻で、800MHz にまで対応できるチップは、ほとんど採れず、最低の 600MHz 対応品ですら充分採れないといいます。しかも、市場の要求としては 800MHz 対応品のほうが格段に多く、600MHz 対応品はほとんど見向きもされないため、メモリメーカは 800MHz 対応品しか出荷できない状況に陥って、さらにコストを押し上げています。メモリメーカとしても、これほど採算の合わないメモリはないでしょうから、正直なところ作りたくないというのが本音でしょう。

ただ、このコスト高による普及遅れという問題に関しては Rambus/intel ともに危機感をもっており、600MHz 以下の規格についても早晩策定されるのではないかといわれています。とはいうものの、800MHz 対応品ですら 100MHz 動作の SDRAM に対して有意な性能的アドバンテージを持たない以上、いくら安いからといって、それより明らかに性能が落ちるものを使うユーザはいないと思われますので、結局は企画倒れに終わるのではないかと思います。

また、DR-DRAM に対抗する高速メモリ規格として、現行の SDRAM の技術を延長とした、DDR-SDRAM(Double Data Rate SDRAM) があり、こちらはライセンス不要で、SDRAM のラインから大きな変更をしないで済み、しかも性能も悪くないときています。極端な話、SDRAM のクロックの立ち上がり/立ち下がりの両方を使うようにするだけで出来てしまうのですから、赤字続きの DRAM メーカとしてはこちらに決まって欲しいところでしょう。ちなみに、こちらは台湾 VIA Technologies(以下、VIA) がサポートするとアナウンスしていますし、AMD も来夏発表予定の Athlon 用チップセット、AMD-760 でサポートを表明しています。しかも、米国の DRAM 業界団体もそれに賛同しており、状況によっては DR-DRAM がポシャって DDR-SDRAM に流れるのではないかという観測もあります。パフォーマンスについては、米国 Micron によれば、「ほかの広帯域メモリソリューションよりも高い性能を示した」(ニュースリリースを筆者の責任において

和訳) そうで、期待が持てるところです。

4 intel、新 PentiumIII を発表

去る 10 月 25 日、intel は以前より Coppermine のコードネームで呼ばれていた新 PentiumIII を発表しました。待望されていた Mobile-PentiumIII も含め、15 品種にも及ぶラインナップでした。特に目玉が 700MHz、733MHz で、733MHz の発表により、クロック数で最速の地位を Athlon から奪還しました。

ですが、この発表前後の市場は、ここしばらくの intel の新 CPU リリースのときとはずいぶん違っていました。というのは、しばらく intel の新 CPU の発表のときは、その前から製品自体は少数ながら出回っていたのです。ところが、今回は、発表前に製品が市場に流通せず、1 週間から 10 日経ってやっと流通し出すという状況でした。つまり、ペーパーリリースになっていたわけです。

もちろん、新製品の発表があり、それから出荷が開始されるというのがどの業界でも通例ですから、今回の状況が異常なわけではありませんが、最近の状況からすると異常に見えてしまいます。これには、諸説あるのですが、おそらく充分な数が準備できていない段階で発表に踏み切ったということでしょう。というのは、9 月中旬以降、AMD の Athlon700MHz の発表により intel はクロック最速の座を AMD に奪われていました。このため、一刻も早く奪回したいということでそのようなことに繋がったのでしょう。ちなみに、intel 社内で工場部門と広報部門との足並みがまったく揃っていないという情報もあります。つまり、AMD に対抗してどんどんそれより上位の CPU を発表していきたい広報と、そう右から左へポンポン作れるわけではない工場との間でギャップが生じているというのです (なお、昔の intel を知る人によると、「発表だけして品物は出ない」のは、当たり前だったそうです)。

それはさておき、Coppermine はそれまでの PentiumIII とは何が変わったのかについて見ておきましょう。一つには、製造プロセスがそれまでの 0.25u プロセスから 0.18u プロセスに変わったことです。もう一つは、L2 キャッシュのオンダイ化と、それに伴うキャッシュ廻りの改良です。

まず、製造プロセスの変更ですが、0.25u プロセスから 0.18u プロセスに変更すると、次のような利点が得られます。

- ダイサイズが小さくなる
- 配線が短くなるので、高速化しやすくなる
- 発熱を減らせる

配線の太さが細くなるということは、それだけダイサイズが小さくなります。ということは、1 枚のウェハから採れる CPU の数は増えますから、歩留まりが上がり、コストダウンに繋がります。また、配線が細くなると、配線の長さも短く出来ますから、高速化も出来ますし、同時に抵抗も減らせますから、発熱も減らせます。と、製造プロセスを微細化することは、非常にメリットが大きいものとなります (もちろん、技術開発コストは上がる)。

実際、消費電力にしても、演算性能にしても、以前の PentiumIII よりも改善しています。ですが、Athlon と比較した場合、有意なアドバンテージがあるとは言いがたい状況になっています。つまり、このベンチマーク項目では勝つがあれでは負けるといったことになっています。結局、性能としては、Athlon に並んだかどうかといったところでしょうか。少なくとも、抜き去ったといえるほどではありません (もちろん、消費電力は Coppermine の圧勝)。

ところが、です。Athlon と Coppermine が並んだといっても、比較しているプロセスが異なります。Athlon は 0.25u に対し、Coppermine は 0.18u です。Coppermine が 0.18u プロセス化によって性能向上を見たのなら、Athlon にも同じことが言えるはずです。実際の比較は、今年中にリリースするとアナウンスされている 750MHz(0.18u プロセスでの生産になるという) を待つことになるでしょうが、おそらく Athlon が上回る結果になるでしょう。さらに、AMD はドイツのドレスデンに新設した工場、Fab30 において、0.18u 銅配線プロセスを準備しています。銅は、アルミニウムよりも電気抵抗が格段に小さく、そこを流れる電子の速度もずっと速いので、更なる高速化・低消費電力化が見込まれています。対して、intel は、0.18u 銅配線プロセスは、Coppermine の後継である Willamette になってはじめて導入される予定で、しかも、Willamette の投入時期は来年後半以降であるといわれており、この点で AMD に水を空けられる可能性があります。また、Coppermine のコアは、基本的には 95 年に発表された PentiumPro のコア設計を引きずっており、いかにそれからチューニングしているとはいえ、製造プロセスの微細化によっても高クロック化に限界があるようで、一部では「800MHz まで行ければいいほうでせいぜい 766MHz どまりでは」とも言われています。また、現在最高速の 733MHz でも、収率はあまりよくないようです。もちろん、intel は全世界に多数の工場を持っているので、収率が悪くても力押しで多数調達することは可能です。

逆に、Athlon は最初から 1GHz 以上のクロックを達成することを目標にしてデザインされており、製造プロセスの微細化によって上限クロックを引き上げやすくなっています。実際、650MHz が市場に出回り出した頃から、既に 700MHz 以上も採れ出している、という情報があったほどです。これは、700MHz のリリースが予想以上に早かった (650MHz が充分出回りだしてまもなく発表があった) ことからも裏付けられています。また、歩留まりについても、AMD によれば、0.25u プロセスで生産の立ち上げ時でさえ、K6-2 や K6-III の立ち上げ時よりも 15%向上したということです。0.18u プロセスの試験生産でも、良好な成績を収めており、充分な生産量が確保できる見通しだといいます。

さらに、価格を考えると、同クロックでは明らかに Athlon の方が安価です。例えば、700MHz 同士で比較すると、Coppermine が ¥85,000 ～ 90,000 に対して、Athlon は ¥75,000 ～ 80000 であり、¥10,000 程度の差があります。もっとも、Athlon 対応マザーボードは ¥18,800 ～ 24,800 程度の水準で、PentiumIII 対応マザーより ¥7,000 ～ 10,000 高になります。したがって、マザーボード込みでの両者の価格差はほとんどないということになります。

性能面でも、価格面でもほぼ同等の両者。まさに土俵上でがっぷり四つの状態です。今後の展開が、実に楽しみです。

5 intel、卑劣な戦略

発売開始以来、ユーザの間では「速い」「高負荷でも粘る」など、概して高い評価を得ている (PentiumIII450MHz から Athlon550MHz でもかなり大きく変わったという声もある)Athlon ですが、どうも普及自体は評価の高さの割には遅いようです。特に、対応マザーボードの増えなさ具合は、異常です。パーツショップでも、かなりの売り上げをしているにもかかわらず、マザーボードが増えない原因の一つに、intel の圧力があります。

これは、中小マザーボードメーカに対して、Athlon 対応マザーボードの製造/販売をしないようにという圧力で、従わない場合はチップセットの供給停止などの措置が取られているようです。実際、真っ先に Athlon 対応マザーボードを出荷した台湾 MSI は、チップセットの供給を本当に止められたといいます (日本 MSI 社員経由の情報)。また、Asustek は、発売しようとしたところで横

槍が入り、結局 FreeWay ブランドに OEM 供給という形で販売していました。現在では、自社ブランドで出荷していますが、先頃、また出荷を一時停止したようです。他にも、台湾 FIC は、特許の侵害という名目で提訴されています (intel の特許を侵害している VIA のチップセットを使用しているため、という理由)。

このように、intel はあの手この手で Athlon の普及を阻むように行動しています。もちろん、商売ごとですから、相手の妨害も多少はありでしょうが、チップセット供給停止を盾にして Athlon 対応マザーボードを製造/販売しないように圧力をかけるというのは、明らかに独占的支配力を背景にした不当競争行為であると思われます。技術競争ではなく、このような策略で相手をどうこうしようというのは、客不在もいいところです。「客がいて、企業がある」「客が望む物を提供して、その結果として利益を得る」という基本的なことを忘れ、シェアの確保にのみ狂奔する intel の姿勢には憤りを禁じ得ません。それにしても、AMD を叩き潰したとして、どうしようというのでしょう? その時点で、独占企業のレッテルを貼られて会社分割される結果につながりかねないということを分かってやっているのでしょうか?

さらに言えば、マザーボードメーカと喧嘩して、いいことは一つもないはずです。FIC は、IBM にもマザーボードを OEM 供給 (IBM が米国市場で出している Athlon 搭載機のマザーボードを供給) するほどのメーカで、結構なシェアを持っています。それを相手取って訴訟を起こして、どうしようというのでしょうか。しかも、理由は intel の特許を侵害している VIA のチップセットを使用しているため、という、訳の分からないものです。VIA のチップセットを使用しているマザーボードメーカは、他にもかなり多数ありますし、そもそも VIA のチップセットが intel の特許を侵害しているかどうかはその 2 社間のみの問題であるはずで、そのチップセットを使って製品を供給するマザーボードメーカには何ら関係がないものであるはずです (例えば、MP3 に関しては、ドイツの Fraunhofer 社が特許を保有していますが、ある MP3 プレイヤがその特許に触れたからといって、それで音楽を聞いている人が訴えられるのと同じことです。これって、どう考えても不合理で変だと思いませんか?)。にもかかわらず、FIC のみが標的にされたのは、一つには VIA と同一資本 (実際、FIC と VIA のトップは血縁関係) であること、もう一つは、一応、公式に Athlon 対応マザーボードを供給している唯一のメーカである (他のメーカは、公式には Athlon 対応マザーボードを供給していない) ことが背景にあるという見方が支配的です。

いずれにせよ、このような不当競争行為が一刻も早く排除されることを祈るばかりです。AMD にしても、いつも訴えられる側ではなく、たまには訴える側に廻って欲しいものです。それはともかく、自由で公正な競争が行われることを期待して止みません。どちらがよいかは、メーカではなく、ユーザが決めるべきことですから。

6 終わりに

駆け足で、いろいろと見てきましたが、どうも批判 (特に intel に対して) の方が多いです。これについては、最初にお断りした通りです。ただ、出来る限り公平な見方をする用には努めたつもりです。そのあたりを汲み取っていただければ幸いです。

さて、あなたの廻りに、「PC 組みたい」といっている人はいませんか?そんな人には、「やっぱ Athlon だよ」といってあげましょう。

あ、そうそう、Athlon ユーザは自分たちのことを、なんて呼んでいるか知っていますか? Athlete(アスリート)って呼ぶんですよ。

あと、参考までに、私の Athlon マシンの環境を書いておきます。

CPU	AMD Athlon 500MHz
M/B	MSI MS-6167 (BIOS ver1.3)
MEM	日立チップ採用 128MB DIMM PC125 CL=2
VGA	ATI All-in-Wonder128 AGP/16M
HDD	IBM DTTA-371010 10GB 7200rpm
HDD	Seagate BarracudaATA ST328040 28GB 7200rpm
FDD	No Bland 2-mode 3.5"
サウンド	YAMAHA WaveForce192XG
NIC	D-Link DFE540TX 10/100-Base-TX
SCSI	Tekram DC-390U
CD-R	TEAC CD-R56SKT
MO	MELCO MOS-230
CASE	Enlight EN7237(P/S 250W ATX-1125B)
CPU ファン	ALPHA PC125CA with Heat Sink

それでは、学会の準備をするか。プログラム書かねば……交通費が……。あ～～～～う～～～～～。

11 月 29 日から新潟で行われる学会に行くのですが、関連費用のせいで
Athlon750MHz ＋マザーボードに当てようと思っていた資金が消えてしまいました……。
お金帰ってくるのは年明け……。すると、1 月末の沖縄での学会の費用に当てられる……。
だれか、こんな私に恵んでください……。

3Dテクスチャマッピング

岐津三泰

平成11年11月16日

概 要

皆さんは日頃PlayStationやNintendo64で遊んでいて、3Dポリゴンに絵をはるのってどうやってやるのだろうと思ったことはありませんか?最近はCPUが十分に速いので簡単なものならば特別なハードウェアを使用せずに実現できます。今回はその触りの部分として、長方形にテクスチャをはる方法を皆さんに紹介したいと思います。なお、この文書は高校生程度の数学の知識と、若干のC言語の知識を前提とします。ただ、数学の知識に関しては必要なものは示してあるので、少々の支障があるかも知れませんが、一応プログラムは作成できると思います。

1 3次元から2次元への投影

3次元の物体を2次元のディスプレイに描き出すには、当然座標変換が必要です。座標変換にはさまざまな方法がありますが、最もポピュラーな方法として次のような方法があります。まず、変換元の3次元座標を(x, y, z)とおき、変換後の2次元座標を(X, Y)とおきます。このとき投影する面と視点の距離をdとしたとき、

$$\begin{cases} X = \dfrac{x}{z}d \\ Y = \dfrac{y}{z}d \end{cases} \tag{1}$$

となります。つまり頂点1つ当たり2回の割算が発生することになります。3次元グラフィックが一般に速いコンピュータを必要とするということのひとつの原因がここにあります。この部分を工夫して高速化をはかっているものもあります。

2 頂点の平行移動、回転

前節で3次元→2次元の変換は出来るようになったので、後は全て3次元の世界でのお話になります。リアルタイムで動く物体を表現する上で必ず必要となるのが平行移動と回転です。リアルタイムでなくても、あると大変便利です。さて、ここでは4次の正方行列と4次の縦ベクトルのかけ算を行ないます。3次元なのになぜ4次なのかと疑問に思う方もいるかも知れませんが、簡単にいうと平行移動も行列とベクトルのかけ算で表現できるからです。行列とベクトルのかけ算については、第6節を参照して下さい。

最初にまず回転です。回転前の座標を (x,y,z)、回転後の座標を (X,Y,Z) とおき、各軸中心の回転角度を (t_x,t_y,t_z) とおくと、

$$\left\{\begin{array}{rcl} X &=& x\cos t_y\cos t_z \\ && +y(\cos t_x\sin t_z+\sin t_x\sin t_y\cos t_z) \\ && +z(\sin t_x\sin t_z+\cos t_x\sin t_y\cos t_z) \\ Y &=& -x\cos t_y\sin t_z \\ && +y(\cos t_x\cos t_z+\sin t_x\sin t_y\sin t_z) \\ && +z(\sin t_x\cos t_z+\cos t_x\sin t_y\sin t_z) \\ Z &=& x\sin t_y-y\sin t_x\cos t_y+z\cos t_x\cos t_y \end{array}\right.$$

となります。これを行列とベクトルのかけ算で表すと次のようになります。回転前の位置ベクトルを $\boldsymbol{v}=\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$、回転後の位置ベクトルを $\boldsymbol{V}=\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$ とおくと、回転行列 R を

$$R=\begin{pmatrix} \cos t_y\cos t_z & \cos t_x\sin t_z+\sin t_x\sin t_y\cos t_z & \sin t_x\sin t_z+\cos t_x\sin t_y\cos t_z & 0 \\ -\cos t_y\sin t_z & \cos t_x\cos t_z+\sin t_x\sin t_y\sin t_z & \sin t_x\cos t_z+\cos t_x\sin t_y\sin t_z & 0 \\ \sin t_y & \sin t_x\cos t_y & \cos t_x\cos t_y & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

とした時に、

$$\boldsymbol{V}=R\boldsymbol{v}$$

によって回転できます。ただしこの計算を行なう時に、かけ算を出来るだけ減らすために、次の積和の公式を用いることによって計算速度を早めます。

$$\begin{aligned} \cos t_x\sin t_z &= \frac{1}{2}\{\sin(t_x+t_z)-\sin(t_x-t_z)\} \\ \sin t_x\cos t_z &= \frac{1}{2}\{\sin(t_x+t_z)+\sin(t_x-t_z)\} \\ \cos t_x\cos t_z &= \frac{1}{2}\{\cos(t_x+t_z)+\cos(t_x-t_z)\} \\ \sin t_x\sin t_z &= -\frac{1}{2}\{\cos(t_x+t_z)-\cos(t_x-t_z)\} \end{aligned}$$

次に平行移動です。平行移動はとても簡単で、各軸に平行に $-\Delta x$、$-\Delta y$、$-\Delta z$ だけ平行移動させる時、

$$\left\{\begin{array}{rcl} X &=& x+\Delta x \\ Y &=& y+\Delta y \\ Z &=& z+\Delta z \end{array}\right.$$

となり、行列表現では、平行移動行列 T を

$$T=\begin{pmatrix} 1 & 0 & 0 & \Delta x \\ 0 & 1 & 0 & \Delta y \\ 0 & 0 & 1 & \Delta z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

とした時に、

$$\boldsymbol{V} = T\boldsymbol{v}$$

となります。従って、回転と平行移動を同時にする場合、

$$\boldsymbol{V} = TR\boldsymbol{v}$$

となりますが、

$$TR = \begin{pmatrix} \cos t_y \cos t_z & \cos t_x \sin t_z + \sin t_x \sin t_y \cos t_z & \sin t_x \sin t_z + \cos t_x \sin t_y \cos t_z & \Delta x \\ -\cos t_y \sin t_z & \cos t_x \cos t_z + \sin t_x \sin t_y \sin t_z & \sin t_x \cos t_z + \cos t_x \sin t_y \sin t_z & \Delta y \\ \sin t_y & \sin t_x \cos t_y & \cos t_x \cos t_y & \Delta z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

を使うことによって、一回の行列のかけ算で回転と平行移動が同時に行なえるわけです。

もちろんこの部分の計算もかなり遅いですが、高速化はこの部分では出来ないと思います。

3 テクスチャをはる

ここまでは3次元を扱う上での一般的なお話でした。ここからがいよいよ本番です。といってもここまで来れば後は難しいことはほとんどありません。

まず、2次元の長方形の絵が3次元空間に浮かんでいるとします。その絵が図3の左のようにスクリーンに投影されたとします。これをディスプレイのY座標を固定して横方向

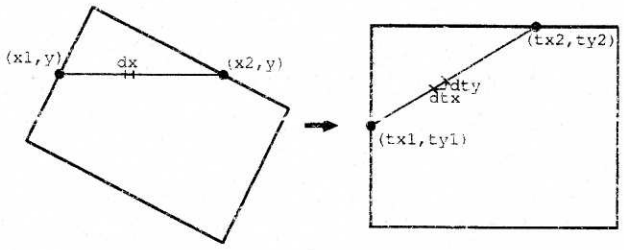


図 1: 空間に浮かぶ絵

に1ラインずつ描く時、まず $(x1, y)$、$(x2, y)$ に対応する絵の上の点 $(tx1, ty1)$、$(tx2, ty2)$ を求め、ディスプレイ上で x を1ドット $(= dx)$ 動かした時、絵の上で x と y がどれだけ変化するか $(= dtx, dty)$ を、

$$dtx = \frac{tx2 - tx1}{x2 - x1}$$
$$dty = \frac{ty2 - ty1}{x2 - x1}$$

という式によって求め、絵の上のその点の色をディスプレイ上に描いていけば、見事に空間に浮かぶ絵の出来上がりです。ただし、プログラムするに当たって、逐一点を描画すると非常に遅くなりますので実際は次のようなコードになります。

```
char scrn[640 * 480];  /* 640x480 の画面の各ピクセルを格納する */
char tex[1 << 8 << 8];  /* 256x256 のテクスチャの各ピクセルを格納する */

for(y = 0; y < 480; y++)
  {

    char *img;

    img = scrn;

    /* ここで、x1, x2, tx1, tx2, ty1, ty2, dtx, dty を計算 */

    for(x = x1, tx = tx1, ty = ty1; x <= x2; x++, tx += tdx, ty += tdy)
      {

        img[x] = tex[tx + ty << 8];

      }

    img += 640;

  }

bitblt(scrn);  /* scrn を画面に一括転送 */
```

たったこれだけ?と思われるかも知れませんが、テクスチャマッピングで重要な部分はこれだけです。

4　さらに発展させるには

今回説明した部分は本当に中心の部分だけで、実際に使うとなるともっと色々な機能が必要です。

まず、式(1)において、Z座標が0になってはいけないのはお分かりだと思います。そもそも、Z座標が0以下になる時はすでに見えないはずですから、Z座標がある値より下回った場合は、そこでポリゴンをクリップしなければなりません。

次に、複数の物体がある場合、奥にある物体を手前の物体が隠すような処理を行なう必要があります。最も簡単な方法は、あるZ座標をキーにソートし、奥にある物体を先に描く方法です。ただしこの方法は、複数のZ座標について調べなければ正しく判定できませんし、そもそも面を交差させることができません。そこで、Zバッファリングという手法があります。最近はこの機能がハードウェアで実現されています。

それ以外にも、ここで述べたテクスチャをはる方法はZ座標を全く考慮していないので、物体がスクリーンに近付き過ぎると激しく歪みます。それを防ぐには、ポリゴンをZ座標

で分割しなければいけません。

5 最後に

駆け足で3Dテクスチャマッピングの説明をしましたが、お分かりいただけたでしょうか? 今ハードで行なわれている全ての効果をソフトウェアだけで再現するのはほとんど不可能だと思いますが、一部だけならなんとか再現できます。Direct3DやOpenGLがある今時こういうことをして意味があるのか?と思われる方もいると思いますが、これは3Dグラフィック技術の基礎の基礎の部分なので、知っておく価値はあると思います。

学祭中は、サンプルプログラムをコンピュータ部の展示場所(東1号館2階205講義室)にて展示していますので、興味のある人は言っていただければソースコードをお見せします。また、来年の展示に向けて3Dダンジョン RPG を製作中なので、そちらの方もよろしくお願いします。

6 Appendix - 行列とベクトルのかけ算

行列とベクトルのかけ算は大学の線形代数で習うことなので、必要な数学は示すと書いたからには一応計算方法を述べておきます。

4次の正方行列 A と、4次の縦ベクトル $\boldsymbol{V}$(高校生の方は $\vec{V}$ の方が馴染み深いかと思いますが、大学ではベクトルを太字で表します) を次のように定義します。

$$A = \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{pmatrix}, \qquad \boldsymbol{V} = \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix}$$

このとき、

$$A\boldsymbol{V} = \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} = \begin{pmatrix} ax+by+cz+dw \\ ex+fy+gz+hw \\ ix+jy+kz+lw \\ mx+ny+oz+pw \end{pmatrix}$$

となります。

Mitsuhiro KIZU -*- Kyoto Institute of Technology
岐津三泰 @ 電子情報工学科2回生. 京都工芸繊維大学
murakumo@mail2.dddd.ne.jp kidu8m@djedu.kit.ac.jp
murakumo@pdx.ne.jp (PHS; up to 1000 charactors)

==========

DNS の仕組み

- DNS ってなんだろう？ -

Written by 大宮広義
Nov.6, 1999

==========

- 目次 -

1.はじめに

このドキュメントは、

(1) TCP/IPの基本的な知識
(2) Networkの基本的な知識

が前提となります。これらについては、関連書籍等を参照してください。

普段わたしたちが、インターネット上のWebページを閲覧したり、電子メールを送ったり、telnet や ftp を使って別のドメインにアクセスするときには、必ずドメインネームシステムのお世話になっています。このドキュメントでは、それほど人々が気にせずに使っているが、実は非常に大きな役割を担っている、いわゆるインターネットにおける「縁の下の力持ち」的なドメインネームシステムについて解説します。

2. DNSとは？

DNS(Domain Name Server)の略です。

ここで非常に簡単ですが、説明しておきますと、基本的に

正引き　　[ドメイン名]　⇒　[IPアドレス]
逆引き　　[IPアドレス]　⇒　[ドメイン名]

という変換をしてくれるのがDNSです。

たとえば、

[IPアドレス] 212.164.70.24

に対して、

[ドメイン名]　www.yahoo.co.jp

という具合に変換をしてくれます。

3. インターネットの歴史

インターネットの歴史は、DNSが開発されるキッカケとなった背景を知る上で重要です。
1960年代の終わり頃、米国国防総省高等研究計画局(ARPA : Advanced Research Project Agency、後にDARPAと改称)は、ARPANET という米国全土に渡る広域コンピュータネットワークの実験的運用を開始しまし

た。当時、コンピュータは高価で希少な代物でしたから、政府の研究を請け負う業者にその数少ないコンピュータを共有させようとしたわけです。「コンピュータ高いんだよ、てめーらみんなで仲良く分け合って使えよ」ってなもんでしょうか。(笑)コンピュータどうしがつながっていれば、ファイルやソフトウェアを有効に共有できますからね。

1980年代のはじめ頃、TCP/IP(Transmission Control Protocol/Internet Protocol)が開発され、すぐにARPANETの標準的なネットワークプロトコルとなりました。そして、当時大学に事実上無償で配布されていた、カリフォルニア大学バークレー校のBSD UNIXオペレーティングシステムにTCP/IPが組み込まれたのです。これによって、BSD UNIXを使っている組織および大学はARPANETに接続できるようになったのです。しかも、組織や大学内でも独自のネットワークを組んでいたので、そのネットワーク内に接続されている他のコンピュータもARPANETへの接続が可能となったのです。

はじめ数台のホストからはじまったARPANETは、一度に数万台のホストがつながる(当時としては)非常に大きなネットワークになったわけです。

しかしながら、1988年にDARPAは実験の終了を決定し、国防総省はARPANETの解体を始めたのでした。じゃあ、これで何もかも終わったんじゃんって思うかもしれませんが、ここで全米科学財団(NSF:National Science Foundation)が運営するNSFNETがこれを引き継いだわけです。

そして、1995年の春以降、インターネットは、NSFNETから複数の商用ネットワークからなるバックボーンへと引き継がれました。

まぁ、こんなところが簡単なインターネットの歴史でしょうか。

4. ドメインネームシステムの歴史

1970年代のARPANETは、数百台のホストがつながった、こじんまりとしたネットワークでした。これらのホストの情報(ホスト名とIPアドレスの対応)はすべてHOSTS.TXTというファイルで管理されていました。UNIXの/etc/hostsはこれに由来します。

そして、そのHOSTS.TXTはSRI(Stanford Research Institute)のNIC(Network Infomation Center)が管理し、SRI-NICというホストから配られていました。ARPANETの管理者は、ホスト情報に変更があると、その旨をNICにいちいち電子メールで報告し、定期的にSRI-NICにftp接続して最新のHOSTS.TXTをダウンロードしていました。しかし次第にネットワークにつながるホストが増えてくると、当然この方法ではうまく行かなくなってきました。理由は簡単ですね。ホストの増加に比例してHOSTS.TXTのサイズが大きくなる上に、更新処理のためにネットワークのトラフィック量も増加したのです。ホストが増えるということは、SRI-NICから最新情報を得ようとするホストも増えるということですからね。そしてARPANETがTCP/IPの登場とともに巨大なものに成長しましたからもう大変です。HOSTS.TXTによる管理では到底、対処できなくなってしまいました。

問題(1) ネットワークトラフィックとプロセッサの負荷が増加し、SRI-NICの負担が耐えきれないものになってしまいました。

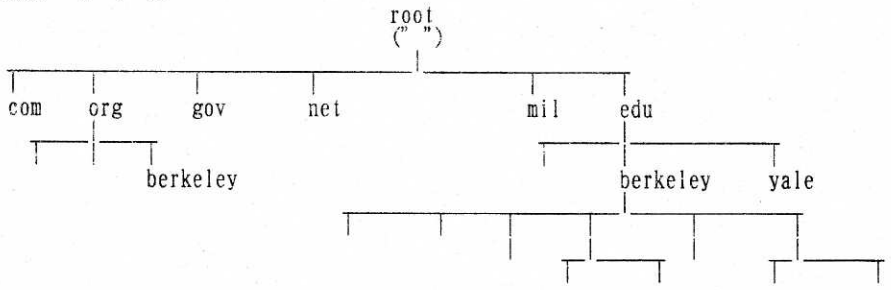
問題(2) NICでは、IPアドレスの一意性を保証していたのですが、ホスト名の割り当てについては、なんら権限を持っていませんでした。ですから、すでに登録されているホスト名を誰かが勝手にHOSTS.TXTに登録してしまうことを防止できなかったというわけです。ですから、「名前の衝突」という問題が発生してしまいました。

問題(3) ホストが絶え間なく増加しているわけですから、NICの管理者がHOSTS.TXTを更新している間にもうホストのIPアドレスが変わっていたり、あらたなホストが追加されているのです。

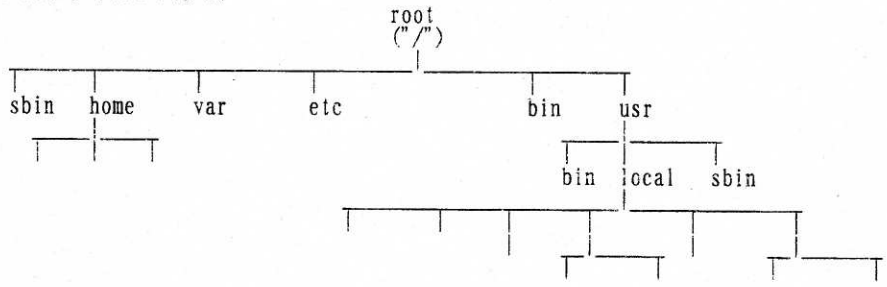
こういった問題がますます浮き彫りになってくると、管理を分散させて、ローカルでそれぞれ管理し、かつ、その管理データを全体でも参照できるようなシステムに変更しようということになりました。

考え出されたDNSデータベースの構造はUNIXファイルシステムに類似し、一連のツリー構造を成しています。

DNSデータベース



UNIXファイルシステム



UNIXでは、ルートは"/"(スラッシュ)で表されますが、DNSでは、ルートは""(空ラベル)で表され、テキスト表記では、"."(ドット)で表されます。

上図を参照しながら、

[DNSデータベース]	[UNIXファイルシステム]
ドメイン (com, net, org など)	ディレクトリ (usr, bin, etc など)
サブドメイン (berkeley, yale など)	サブディレクトリ (bin, local, sbin など)

DNSデータベースの「ドメイン」が、UNIXファイルシステムの「ディレクトリ」に相当します。このようなツリー構造をとれば、「名前の衝突」は起こりません。

```
berkeley.edu
berkeley.org
```

は同じberkeleyという名前のノードですが、eduドメインに属しているberkeleyなのか、orgドメインに属しているberkeleyなのかで区別ができます。
また、たとえば、berkeley.eduドメインの管理ををカリフォルニア大学バークレー校に任せることによって(これを「委任」と呼ぶ)、管理を分散させることができ、これまでの問題点がうまく解決できたわけです。

ちなみに、rootの直下にあるドメインをトップレベルドメインといいます。ですから、jpドメインもトップレベルドメインです。

5. DNSの仕組み

[1] ネームサーバ

ドメイン空間に関する情報を扱うプログラムをネームサーバ(name server)といいます。一般に、ネームサーバには、ゾーン(zone)と呼ばれるドメイン内の一部分の完全な情報が格納されています。ドメイン空間の一部分という点についてすこし説明しておきましょう。ネームサーバは自分の管理する以外のドメインの情報を別に知っておく必要はありません。Fig.1で説明しますと、たとえば、on.orgドメインを別の組織に委任しておけば、orgドメイン管理者はon.orgドメインの情報をorgのネームサーバに登録する必要はありません。知りたければon.orgの管理者が管理するネームサーバに問い合わせを出せばいいのですから。ですからorgドメイン管理者はon.orgドメイン以外、つまりab.orgドメインとqc.orgドメインの完全な情報をネームサーバに登録しておけばいいのです。このab.orgドメインとqc.orgドメインにあたるのが、orgのネームサーバが管理するゾーンなわけです。だから「ドメイン空間の一部分」なわけです。そして、このとき「ネームサーバはゾーンに対して権威を持っている」といいます。また、複数のゾーンを管理することも可能です。

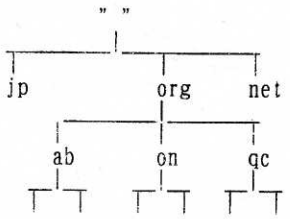


Fig. 1

[2] ネームサーバの種類
DNSの仕様では、プライマリマスタ(primary master)とセカンダリマスタ(secondary master)という2種類のネームサーバが定義されています。

プライマリマスタ：自分が権威を持っているゾーンのデータを、自分が動作しているホストから取得するネームサーバ
セカンダリマスタ：ゾーンの権威を持っている自分以外のネームサーバからゾーンのデータを取得するネームサーバ

実は両者とも同じゾーンの権威を持っています。そしてセカンダリマスタは複数存在し得ます。セカンダリマスタは、定期的にプライマリマスタからゾーンデータを問い合わせて、更新されていれば自分の所に持ってきます。このことをゾーン転送といいます。どうして同じデータを持った2種類のネームサーバが必要であるかというと、ゾーンデータに冗長性を持たせることができるますし、ネームサーバへの負荷を分散させることもできます。また、ゾーン内の各ホストの手近にネームサーバを持たせることもできます。また、セカンダリマスタは、プライマリマスタが停止してしまったときのスペアになるわけです。

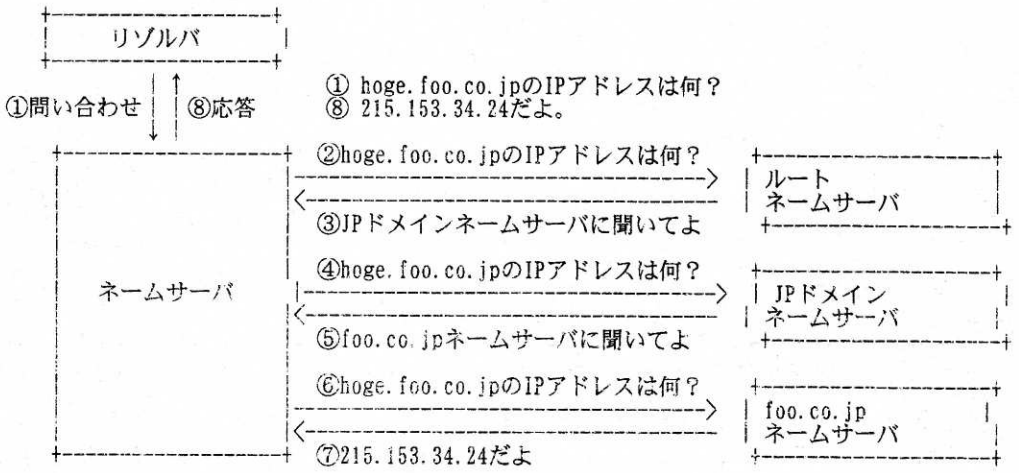
[3] リゾルバ
リゾルバ(resolver)とは、ネームサーバにアクセスするクライアントのことです。ドメイン名空間からの情報を必要するプログラムは、リゾルバを使用します。リゾルバは次のことを行います。

(1) ネームサーバへの問い合わせ
(2) 応答の解釈
(3) 要求の側プログラムへの情報の転送

リゾルバは、単なるライブラリルーチンの集まりで、telnet や ftp などのプログラムに組み込まれていて、独立したプロセスではありません。リゾルバは、問い合わせ要求を組み立てて、それをネームサーバに送信して応答を待ち、応答が帰ってこないと要求を再送信するだけでほかになにもできません。

[4] ルートネームサーバ
ルートネームサーバは各トップレベルドメインのネームサーバがどこにあるか知っています。ですから、ルートネームサーバはドメイン名の問い合わせについて、最低でも、そのドメイン名に含まれるトップレベルドメインの権威を持つネームサーバの名前とIPアドレスを返すことができます。ドメイン名しか情報が提供されていなければ、ルートネームサーバへの問い合わせから始めなければなりません。ですから、ルートネームサーバはDNSにおいて非常に重要な位置を占めています。これが停止してしまうとインターネットが使えなくなってしまいます。こういった事態を避けるために、インターネット上には9つのルートネームサーバが設けられていて、いろいろなところに散在しています。米軍が管理しているMILNETにもありますし、NASAのSPAN、ヨーロッパにもあります。 ルートネームサーバはとにかく忙しい。いくら9つあるといっても膨大な問い合わせが毎日のように寄せられます。1992年にカリフォルニア大学がおこなった調査によると、同大学のルートネームサーバは1時間に20,000件の問い合わせを受けています。1秒間に6件程度の割合です。また、InterNICのルートネームサーバであるns.internic.netは、1時間に255,600件もの問い合わせを受けています。こちらは1秒間になんと71件の割合です！しかし、これほど多くの問い合わせにも関わらず、インターネットでの名前の解決は順調に行われています。すごいことですね。

[5] ドメイン名⇒IPアドレス変換



このように解決がおこなわれます。説明不要ですね。
なお、最初に問い合わせるネームサーバが知っている場合は、その場で解決し、ネームサーバは他のネームサーバに問い合わせしません。

[5] IPアドレス⇒ドメイン名変換

本来IPアドレスからドメイン名を見つけることは難しいです。ドメイン名空間では、アドレスを含むすべてのデータは、ドメイン名を索引として整理されています。ですからドメイン名からドメインのIPアドレスを見つけることは簡単なのですが、その逆は非常に難しくなります。しかし、実際には賢く、効率的な方法でアドレスからドメイン名を見つけることができます。「名前 ⇒ IPアドレス」は簡単なのですから、IPアドレスそのものをアドレスとして使用する別の空間を作ってしまえばいいわけです。インターネット上のドメイン名空間には、in-addr.arpaというドメインがあり、このドメインではIPアドレスを名前として整理しています。
in-addr.arpaドメインでは、IPアドレスのようなドットつきオクテット表記中の数字によってノードに名前をつけています。IPアドレスをin-addr.arpaドメインとして読む場合は、最後のオクテットから読まなければなりません。たとえば、hoge.foo.co.jp のIPアドレスが215.153.34.24であれば、そのin-addr.arpaサブドメイン名は、24.34.153.215.in-addr.arpaとなります。これによってIPアドレスがドメイン名のようにツリー構造を成し、ドメイン名と同じように扱うことが可能なわけです。

[6] キャッシュ

ネームサーバに一度問い合わせがあったとします。そしてもう一度同じ問い合わせが来たときに、ネームサーバはまたもう一度ルートネームサーバから問い合わせるのでしょうか？ いえ、そうではありません。一度問い合わせを受けると、その情報をキャッシュします。次に問い合わせをされたときに備えて、データを保存しておくわけです。またネガティブキャッシュというのもあって、指定されたドメインは存在しないというデータが帰ってきた場合には、その不在情報もキャッシュします。こうすることで少しでも無駄をなくし、効率を良くしているのです。
キャッシュは、ルートネームサーバに問い合わせるということを防ぎ、名前の解決を高速化しています。これによってルートネームサーバへの負荷を軽減しているわけですね。
ただし、このキャッシュが残ったままだと、そのデータが更新されていても古いデータを使い続けてしまいます。それを防ぐために生存時間というものを設けて、その時間が経過するとネームサーバはそのキャッシュを捨ててしまいます。

6. さいごに

DNSに関する歴史や、ネームサーバの名前解決について述べました。DNSはこれだけ重要な役割を果しているのですが、実はあまりメジャーなものではありません。しかしこのドキュメントを読んでいただいた方はDNSについてかなりの知識がついたわけですから、普段ネットサーフィンをしているときや、電子メールを送信しているときにでも、ネームサーバがやっていることを想像してみてください。むなしいだけかもしれませんが、これを知らない人よりはましです。このドキュメントが、あなたのネットワークに

関する知識の足しになれば光栄です。

- 参考文献 -

Paul Albitz, Cricket Liu, "DNS & BIND" Second Edition.

CDをデジタルで再生させる

横川龍雄

PC/AT互換機でCDを再生するとき、一般的にはCD-ROMドライブでD/A変換されたデータをサウンドカードに送り、それをサウンドカードのミキサーを通してから再生するため少々データにノイズが乗ることになってしまいます。そこで私は最近Windows上でCDから音声データをデジタルデータのまま読み出し、それをPCMデバイスを使って再生するプログラムをつくりました。これだと多少の音質劣化は減るはずです。ここではこれが一通り出来あがるまでの簡単な過程を書かせていただきます。

まず最初に実際にPCMデバイスを使って再生するCDプレーヤーを作るには次の機能は必要です。

① CD-ROMドライブの検出
② メディアの挿入チェック
③ トラック情報の取得
④ CDからのオーディオデータの読み出し
⑤ 読み出したオーディオデータの再生

このうち、④以外についてはWindowsにある機能だけでも十分実現はできます。このうち、⑤についてはDirect Soundを使用しました。④のデータの読み出しですがWindowsにはこのような機能はありませんので、SCSI機器を直接制御するASPI for Win32を使わなければいけません。ASPIは米国のADAPTEC社がMS-DOSでSCSI機器を制御する標準インターフェイスとして提案したもので、PC/AT互換機では標準的なものでした。Windows3.1の時に同社がWindowsのAPIとしてインプリメントし、さらにWindowsNTで32ビット化されました。そしてWindows95でマイクロソフトに正式に採用され、SCSIコントローラを導入すれば自動的にASPIが導入されるようになりました。またこれを利用すれば直接SCSIコマンドを送ってWindowsがサポートしていないデバイス特有の機能も利用できます。また、ASPIでは一定の範囲内であればATAPI接続のCD-ROMも制御することが可能です。そして、SCSIコマンドを利用してももちろん①～④の機能は実現できますので、④だけをSCSIコマンドを送って使うというようなかえって面倒くさいことはしません。

実際のプログラムですが最初にASPIを使えるようにするためWindowsのLoadLibrary APIでwnaspi32.dllをロードします。そして、GetASPI32SupportInfo APIとSendASPI32Command APIをロードします。つぎにGetASPI32SupportInfo APIを実行してホストアダプタの数を取得します。そして各ホストアダプタの全ターゲットに対して、SendASPI32Command APIを使いデバイス情報を取得するコマンドを送りCD-ROMドライブであるかどうかを確認します。こうしてCD-ROMを検出したら、つぎに共通SCSIコマンドのTest Unit Readyコマンドを発行してメディアが挿入されているか、いないかをチェックします。メディアの挿入を確認したら、つぎはCDのトラック情報が格納されているTOCデータの読み出しです。これは汎用のReadコマンドでは読み出すことが出来ないので専用のReadTOCコマンドを使って読み出します。ここで返ってくるデータには各トラックの先頭アドレスしか含まれませんのでので再生させるときは再生したいトラックのアドレスと次のトラックのアドレスとの差を求めて再生させます。また、ここでは説明を省きますが最終トラックを再生させるにはリードアウトトラック(最終トラックの次にある真の最終トラックとでも思っておいてください)と呼ばれるトラックのアドレスも取得しておかなければなりません。

ここまでくれば、後はデータを読み出し演奏するだけです。しかし、オーディオデータの読み込みでは少し重要なことがあります。それは、オーディオデータではデータ位置を間違えて読み出す可能性があるということです。これを防ぐためにCD-Ripperなどではジッターコレクション(ふらつき補正)を行ってデータの連続性を保っています。もちろん、直接読み出して再生する場合にもこれを行うことは必要です。具体的には読み込み速度を落として読み込みを安定させたり、前回読み出した領域と今回読み出した領域とを比較して、データの一致するところを探して重ね合わせたりします。こうして、補正されたデータを再生させれば音質劣化の少ないCDプレーヤが出来あがるはずです。

ASPIを使った所(①～④)の例を以下に載せておきます。

※ASPIの使い方の詳細はASPI SDK(http://www.adaptec.com/adaptec/developers/)等を参照してください。

```
/* デバイスのタイプを取得する */
WORD GetDeviceType(Adapter,ID,LUN)
int Adapter,ID,LUN;
{
  SRB_GDEVBlock  SRB;
  DWORD ASPI_Stat;

  memset(&SRB,0,sizeof(SRB));
  SRB.SRB_Cmd=SC_GET_DEV_TYPE;
  SRB.SRB_HaId=Adapter;
  SRB.SRB_Target=ID;
  SRB.SRB_Lun=LUN;
  ASPI_Stat=SendASPI32Command(&SRB);

  if(ASPI_Stat==SS_COMP)  return  (WORD)SRB.SRB_DeviceType;
  else                    return  0xffff;
}

/* メディアの挿入チェック */
BOOL CheckMedia(Adapter,ID,LUN)
int Adapter,ID,LUN;
{
  SRB_ExecSCSICmd SRB;
  DWORD ASPI_Stat;

  memset(&SRB,0,sizeof(SRB));
  SRB.SRB_Cmd=SC_EXEC_SCSI_CMD;
  SRB.SRB_HaId=Adapter;
  SRB.SRB_Target=ID;
  SRB.SRB_Lun=LUN;
  SRB.SRB_Flags=0;
  SRB.SRB_BufLen=0;
  SRB.SRB_BufPointer=NULL;
  SRB.SRB_SenseLen=SENSE_LEN;
  SRB.SRB_CDBLen=6;
  SRB.CDBByte[0]=0x00;/* TestUnitReadyコマンド */

  ASPI_Stat=SendASPI32Command(&SRB);
  while(SRB.SRB_Status==SS_PENDING);

  if(SRB.SRB_Status==SS_COMP)  return TRUE;
  else                         return FALSE;
}

/* トラック情報の取得 */
typedef struct _TrackInfo
{
  BOOL  audio;
  DWORD add;
}TrackInfo;

BOOL ReadToc(Adapter,ID,LUN,tracks,nMaxTracks)
int Adapter,ID,LUN;
TrackInfo  *tracks;    /* トラック情報を格納する構造体の配列へのポインタ */
int        *nMaxTracks;/* 最大トラック数を格納する変数へのポインタ       */
{
  SRB_ExecSCSICmd SRB;
  DWORD ASPI_Stat;
```

```
  BYTE  buf[12];
  int   i=1;

  while(1){
    memset(&SRB,0,sizeof(SRB));
    SRB.SRB_Cmd=SC_EXEC_SCSI_CMD;
    SRB.SRB_HaId=Adapter;
    SRB.SRB_Target=ID;
    SRB.SRB_Lun=LUN;
    SRB.SRB_Flags=SRB_DIR_IN;
    SRB.SRB_BufLen=12;
    SRB.SRB_BufPointer=buf;
    SRB.SRB_SenseLen=SENSE_LEN;
    SRB.SRB_CDBLen=10;
    SRB.CDBByte[0]=0x43; /* Read TOCコマンド      */
    SRB.CDBByte[6]=i;    /* 読み出すトラック番号 */
    SRB.CDBByte[8]=12;

    ASPI_Stat=SendASPI32Command(&SRB);
    while(SRB.SRB_Status==SS_PENDING);
    if(SRB.SRB_Status!=SS_COMP)  return FALSE:
    *nMaxTracks=buf[3];

    if(buf[5]&4)  tracks[i].audio=TRUE;
    else          tracks[i].audio=FALSE;

    tracks[i].add|=buf[8]<<24;
    tracks[i].add|=buf[9]<<16;
    tracks[i].add|=buf[10]<<8;
    tracks[i].add|=buf[11];

    i++;
    if(i>(int)buf[3]) break;
  }

  memset(&SRB,0,sizeof(SRB));
  SRB.SRB_Cmd=SC_EXEC_SCSI_CMD;
  SRB.SRB_HaId=Adapter;
  SRB.SRB_Target=ID;
  SRB.SRB_Lun=LUN;
  SRB.SRB_Flags=SRB_DIR_IN;
  SRB.SRB_BufLen=12;
  SRB.SRB_BufPointer=buf;
  SRB.SRB_SenseLen=SENSE_LEN;
  SRB.SRB_CDBLen=10;
  SRB.CDBByte[0]=0x43;
  SRB.CDBByte[6]=0xaa; /* リードアウトトラック */
  SRB.CDBByte[8]=12;

  ASPI_Stat=SendASPI32Command(&SRB);
  while(SRB.SRB_Status==SS_PENDING);
  if(SRB.SRB_Status!=SS_COMP)  return FALSE;

  if(buf[5]&4)  tracks[buf[3]+1].audio=TRUE;
  else          tracks[buf[3]+1].audio=FALSE;

  tracks[i].add|=buf[8]<<24;
  tracks[i].add|=buf[9]<<16;
  tracks[i].add|=buf[10]<<8;
  tracks[i].add|=buf[11];

  return TRUE;
}
```

```
/* CDからのオーディオデータの読み出し */
BOOL ReadCD(Adapter,ID,LUN,buf,lba,sec)
int    Adapter,ID,LUN;
BYTE   *buf;
DWORD  lba;/* 読み込みアドレス */
DWORD  sec;/* 読み込むセクタ数 */
{
  SRB_ExecSCSICmd SRB;
  DWORD ASPI_Stat,WAIT_Stat;
  HANDLE  hEvent;

  hEvent=CreateEvent(NULL,FALSE,FALSE,NULL);
  if(!hEvent)  return FALSE;

  memset(&SRB,0,sizeof(SRB));
  SRB.SRB_Cmd=SC_EXEC_SCSI_CMD;
  SRB.SRB_HaId=Adapter;
  SRB.SRB_Target=ID;
  SRB.SRB_Lun=LUN;
  SRB.SRB_Flags=SRB_EVENT_NOTIFY|SRB_DIR_IN;
  SRB.SRB_PostProc=hEvent;
  SRB.SRB_BufLen=sec*2352;
  SRB.SRB_BufPointer=buf;
  SRB.SRB_SenseLen=SENSE_LEN;
  SRB.SRB_CDBLen=12;
  SRB.CDBByte[0]=0xbe;/* Read CDコマンド */

  SRB.CDBByte[2]=(BYTE)(lba>>24)&0xff;
  SRB.CDBByte[3]=(BYTE)(lba>>16)&0xff;
  SRB.CDBByte[4]=(BYTE)(lba>>8)&0xff;
  SRB.CDBByte[5]=(BYTE)lba&0xff;

  SRB.CDBByte[6]=(BYTE)(sec>>16)&0xff;
  SRB.CDBByte[7]=(BYTE)(sec>>8)&0xff;
  SRB.CDBByte[8]=(BYTE)sec&0xff;

  SRB.CDBByte[9]=0x10;

  ASPI_Stat=SendASPI32Command(&SRB);

  if(ASPI_Stat==SS_PENDING){
    WAIT_Stat=WaitForSingleObject(hEvent,2000);
    if(WAIT_Stat==WAIT_TIMEOUT){
      ServiceAbort(dev.Adapter,&SRB);
    }
  }

  CloseHandle(hEvent);

  if(SRB.SRB_Status==SS_COMP)  return  TRUE;
  else                         return  FALSE;
}
```

なお、私が作ったプログラムはhttp://www5.freeweb.ne.jp/computer/yokogawa/で公開しています。

参考文献:「Inside Windows '98年12月号 第1特集 SCSIデバイス制御」

格ゲーアクションスクリプトVer 0.1 - 安達洋明

始めに

展示されている(たぶん)格闘ゲームの動きはプログラム内に組み込まず、データで後から定義できるようになっています。この文章はその定義の仕方の一部を説明してるわけですが、一般の方々が読んでも全然おもしろくないです。読み飛ばすことをおすすめします。

(プログラムの方で時間をとられたため他にネタがありませんでした。)

概要みたいなもの

対戦型格闘ゲームのキャラクターデータのうち、プログラム的なもの(技の動き、アニメーション全般)を定義できるようにしたスクリプトです。データ的なもの(ジャンプ軌道、グラフィック、当たり判定)などは個別に専用ツールで作成します。

構文とか

- 一部命令を除き、文の最後は';'をつけ、一行につきいくつでも式文が書ける。
- ラベルの定義は行の初めに':'をつけ、そこから改行まではすべてラベルと見なされる。
- 特別な用途に使うラベルは、'@'を後ろにつけてタイプを指定する。
- //から改行まではコメントになる。
- 今のところifとincludeのネストはできない。
- 命令によっては効果の対象を指定する場合があり、その対象は<object>[*NN*].<*command*>で指定する。NNは添字で1-99までが使用でき、省略または0の場合は最後に使った番号が適用される。その最終適用番号は各スレッドが固有に持っている。

対象オブジェクト

thread

スレッド

sprite

スプライト

loop

ループ

通常命令のうち仕様がだいたい固まってる物の一部

sleep(*frame*)

スクリプトエンジンの処理を中断し他の部分(描画、当たり判定など)に処理を移します。

frame	スクリプトエンジンの処理を休ませるフレーム数。 0で中断のみ、1以上でその回数分スルーします。

対象　　標準

setgraphic(*graphicnumber*)
setgrp(*graphicnumber*)

オブジェクトに画像番号をセットします。

graphicnumber	セットする画像番号。 +500する事で左右反転も指定できますが、当たり判定が関わってくる場合はsetdirectionを使ってください。

対象　　標準、スプライト

move(*x,y*[,*sleepframe*])
mov(*x,y*[,*sleepframe*])

x	x増分 (-255 < x < 255)
y	y増分 (-255 < x < 255)
sleepframe	0より大きい値をセットするとsleep(*sleepframe-1*)を後ろに挿入したのと同じ効果があります。

対象　　標準、スプライト

show

オブジェクトを表示します。

対象　　スプライト

hide

オブジェクトを非表示にします。

対象　　スプライト

drop(*flag*)

スクリプト実行中は落下を含めた自分の動きが停止しますが、この命令によって本来のジャンプ、落下の運動を再開させます。

flag	0	落下停止します
	1	落下します。

対象　　標準

jumpreset

ジャンプ、落下の軌道を初期化します。

対象　　標準

setdirection(*flag*)

キャラクターの向きを変えます。

flag	0	本来の向きにします。(前向き)
	1	現在と逆の向きに向きます。(後ろ向き)
	2	強制的に右向きにします。
	3	強制的に左向きにします。

対象　　標準

jumpattack(*number*)

ジャンプ中(滞空中)に攻撃できる回数を指定します。
ジャンプ時に1に初期化されます。

number	攻撃できる回数。

対象　　標準

jump(*label*)

指定ラベルに処理を移します。

label	対象のラベル。

対象　　標準

call(*label*)

指定ラベルに処理を移します。call先でreturnを実行することで次の行に復帰できます。

label	対象のラベル。

対象　　標準

return

最後にcallで呼び出された行の次行に処理を復帰します。callで呼び出されておらず復帰できない場合はendとほぼ同様の処理を行います。

対象　　標準

end

処理中のスクリプトスレッドを終了します。

対象　　標準

loop(*label,number*)

指定回数分指定ラベルに処理を移します。
ループオブジェクトの添字を変えることでネスト可能。

label	さかのぼる対象のラベル。
number	ループする回数。 0だと素通りします。

対象　　ループ

loopl(*line,number*)

line	さかのぼるライン数。 中間コード時の行数で指定してください。

対象　　ループ

create

オブジェクトの作成(初期化)を行います。

対象　　スプライト

create(*label*) *for thread*

オブジェクトの作成(初期化)を行います。(スレッド用)

label	スレッドを開始するラベル。

start　　スレッドを開始します。
destroy　　スレッドを破棄します。
stop　　スレッドを停止します。
resume　　スレッドを再会します。

対象　　スレッド

if文

if[*keyword*](*parameter*):<*syntax*> [<*syntax*>...] [else:;<*syntax*>[<*syntax*>...]]

if[*keyword*](*parameter*):

　　<*syntax*>

[else

　　<*syntax*>]

endif

keyword	条件のキーワード 。
parameter	*keyword*に対する引数。

キーワード一覧

key(*keycommand*)

　　キーの入力判定を行います。

keycommand	それぞれの方向キー 7 8 9 1,2,3,4 ,6,7,8,9 4 6 1 2 3 後<->前	
	0	ニュートラル
	a-f	それぞれのボタン
	+	同時押し
	/	曖昧判定モードにする。 （よけいなキーを押していても成立します）
	#NN	NNを入力受付フレーム数にする 後ろに他のコマンドを続ける場合は,で区切る
	~	判定が成立した場合キーバッファをクリア
	^	空中
	_	屈み
	-	地上

Ex.

　　if[key](-623+a,#50):call(shoryuken);end;

　　compare(*value1* <*cmp*> *value2*)

変数、内部変数との比較を行います。

value1,value2	即値の場合はそのまま、標準変数の場合は$NN(NN=00-99) 内部変数一覧 ($は省略) x　x座標 y　y座標 dr / direction　向き 0-右向き 1-左向き jump　ジャンプ 0-なし 1-上昇中 2-落下中 life　ライフゲージ special　スペシャルゲージ
<cmp>	関係演算子を指定します。 (==,!=,<>,<,>,<=,>=)

Ex.

```
if[cmpare](%y<=0):return;
```

その他

;はつけない

#include=*includefile*

*includefile*を読み込んで#includeがある位置に挿入する。

#replace:*string1=string2*

単純に*string1*を*string2*に置き換える。
どこで宣言しても全体に対して有効。

FreeBSD インストール奮闘記

機械システム工学科　1回生　米田　裕

0.FreeBSD とは？

別に解説なんてしなくてもよいと思うのですが、念のために書いておきます。ちなみに下の文章は The FreeBSD Project (Japan)のページから拝借しました。

FreeBSD は "PC 互換機" コンピュータ用の先進的な BSD UNIX オペレーティングシステムであり，多くの人たち によって によって保守・開発されています.

1.きっかけ

某月某日 FreeBSD3.1-release 版を手に入れたので、新しいパソコンを買ったために使い道のなくなっていた PC-9801BX4 にインストールすることを決意する。また夏に PC-9821Nw133 を購入。ハードディスクの内 500MB を FreeBSD 領域に割り当てることにする。

2.インストールに先立ち

手始めに PC-9801BX4 のハードディスク（外付）が 240MB しかないので、日本橋で新たに 320MB のハードディスクを購入するも、X Window System まで入れるのはしんどいと言われたため、高校の部室から 500MB のハードディスクを強奪。かわりに 320MB のハードディスクを置いていく。これでとりあえず環境が整ったため、インストールを開始する。しかし自分は MS-DOS と Windows 以外の OS を扱うのは初めてなので、何がなんだかさっぱりわからない。そこで、わからないところは人に聞きまくってインストールしていくことを決意する。

ちなみにインストール環境は、PC-9801BX4 の方は PC-9801BX4 の元のスペックに 2 倍速内臓 CD-ROM ドライブとメモリ 12MB を追加してあり、ハードディスクは元からあった 240MB のやつと高校の部室から強奪した 500MB のハードディスク。ID ナンバーは 240MB のハードディスクの方が先。ディスプレイはかなり昔（たしか PC-9801U2 の時あたりから使っていた記憶がある）から使っているもののため詳しいスペックは不明。ただ、画面の表示サイズは 640x480 が限界だという事だけはわかっている。マウスは NEC の純正マウス（どうでもいいことではあるが、自分は NEC 純正マウスのクリックしたときのあの独特の感覚がすごく好き）。PC-9821Nw133 は PC-9821Nw133 の元のスペックにメモリ 32MB を追加してあって（中古で購入したため）合計で 64MB。ハードディスク 1.3GB 中 500MB を FreeBSD 領域に割り当てています。

ノートパソコンの場合

とりあえずkernフロッピをドライブに入れてパソコンを起動する。しばらくするとmfsrootフロッピを入れろと言われたのでフロッピを交換してretarnキーを押す。するとまたしばらくして、今度は

Skip kernel configuration and continue with installation
Start kernel configuration in full-screen visual mode
Start kernel configuration in CLI mode

というメニューが出てきた。どうやらこのメニューは、組み込むデバイスドライバの選択、有効化、無効化、設定を行うためのもので、上から

この設定画面をスキップし、標準のカーネルを使ってインストーラを起動する。
ビジュアルモードで設定する。
コマンドラインモードで設定する。

ということらしいが、よくわからないし面倒くさいのでとりあえずスキップする。インストールメニューが起動したのでUsageもDocも読まずにCustomを選択する。初心者なのにCustomなんか選択してもよいのかとも思うが、わからなかったら聞けばいいやと開き直る。というわけで、カスタムインストールメニューが開いたのでまずPartitionでインストールするドライブを選択…する必要はないので、とりあえず最初に用意した500MBの領域でunusedになっている部分をすべてFreeBSDの領域に割り当てる。次にLabelでFreeBSDのパーティションを設定する。まず"A"を押してパーティションを自動設定してみる。ちょっとswapに割り当てられた容量が多かったのでそこだけ手動で設定する。やっとDistributionにうつる。本当はX-Developperが選択できれば一番いいのだが、いかんせん500MBの容量では無理なのであきらめてCustomを選択してインストールするファイルを自分で選択することにする。選択したファイルはとりあえずbinと98bin、あとdict、98doc、info、man、jcatman、profilbs最後にXFree86のBasicからbin、cfg、doc、lib、man、prog、set98、それとServerはPC98のTGUI、Fontsはfnts、f100、fscl、nonを選択。念のため次のページにDistributionsの内容を書いておきます。というわけで配布ファイルの選択が終了したのでCommitでインストールした後、Configureで各種の設定を行う…といっても設定したのはRoot PasswordとUser ManegementとTime ZoneとMouseだけですが…とりあえずこれでインストール時に行う設定作業が終了したので、インストーラを終了して、ディスクを抜いてrebootする。さて、次はX Window systemの設定とあとportsとpackagesをインストールしなければ…。

・Distributions（配布ファイル）の内容

bin	バイナリ基本配布ファイル（必須）
98bin	PC98　基本バイナリ配布（必須）
compat1x	FreeBSD 1.x バイナリ互換性
compat20	FreeBSD 2.0 バイナリ互換性
compat21	FreeBSD 2.1 バイナリ互換性
compat22	FreeBSD 2.2.x と 3.0 a.out バイナリ互換性
DES	DES 暗号化コード
dict	スペルチェッカー辞書ファイル
doc	種々の FreeBSD オンラインドキュメント
98doc	FreeBSD（98）ハンドブックとオンラインドキュメント
games	ゲーム（非商用）
info	GNUinfo ファイル
man	システムマニュアルページ － 推奨
catman	フォーマット済みシステムマニュアルページ
jcatman	日本語フォーマット済みシステムマニュアルページ
proflibs	プロファイル版のライブラリ
98src	PC98 用全てのソース
src	DES を除く全てのソース
ports	FreeBSD Ports コレクション
XFree86	XFree86 3.3.3.1 配布ファイル

PC－9801BX4 の場合

こちらもノートパソコンとほとんど同じ設定でインストールしたのだが、なぜか reboot した時に

error 6: panic: cannot mount root (2)

というエラーメッセージが出てきて、また reboot されてしまった。なぜかわからなかったか、とりあえず正常に boot させようと何度も試みるもいまだになおらず。現在原因究明中。

結局たいしたことは書けなかったので、次回はもう少し人に見せられるものを書けたらいいなと思う。

ゲームについて

谷尾元聡

ノートPCを購入してから一番よくやっていたことといえばゲームでしょう。

ほとんどPCにさわったことのない初心者だったので、何をするにも戸惑っていました。初心者には、ゲームなどのインストールなどをさせるといいのだそうです。(ファイル、フォルダの操作が一通りできるようになる、そういうことほど一生懸命になる)という訳で、現在までたいしたことはやっておりません。以下自分で良かったなと思ったフリーウェアのWindowsのゲームについて書きます。

魔法の塔 for Windows

パズルゲームです。迷路があって、そのなかに配置された敵キャラを倒し、(といっても戦闘時に画面が切り替わったりはせず、画面の横の部分に敵のステータスが表示され、自動的に結果が計算されるという形で)扉をあけたりしながら、塔の上階へお姫様を助けにいく(このあたりは結構意外かもしれない)という内容です。アイテムや敵キャラも多彩です。面白いです。ただし難しいです。適当にやっていると、先へ進めなくなります。難しすぎると思ったら、「魔法の塔Jr. for Windows」というのもあります。(30分でクリアできるとある)英語版とか3D版などもあるらしいです。ただし3Dバージョンはシェアウェアです。ベクター(http://www.vector.co.jp)からダウンロードできます。

作者のページ：(http://plaza.across.or.jp/~furano/)&(http://www.top.or.jp/~kenichi/)

AMEL BROAT

ドラクエなどと同じRPGです。かなり前からあるもののようですし、知っている人も多
いと思います。今のゲームを見慣れている人にはグラフィックは大したことがないように見えるかもしれないですが、(現に家でやっていてけなされた)それでも面白いです。台詞も人によっては難かもしれませんが、(誤植もあったような)そんなことを十二分に補えるゲームだと思います。隠しアイテムあり、隠しイベントあり、(Windowsバージョンで追加されたらしい)強いボスキャラありのRPG好きにはお勧めのゲームだと思います。BGMも多彩です。エンディングも何通りかあるようで、イベントもいろいろとあり楽しいゲームです。話自体も楽しめると思います。

作者のページ(http://member.nifty.ne.jp/shogokawai/)にあります。ゲームについてもいろいろあります。

GNU Chess for Windows

コンピュータと対戦できるチェスのアプリケーションです。何より強いです。軽い気持ちでやっていたところ、負けつづけて、１０数回やってやっと引き分けました。あからさまなミスはしませんし、そつがありません。目下負けつづけております。強い人なら楽勝なのかもしれませんが…。ベクター(http://www.vector.co.jp)で入手できます。作者は日本の方ではありません。出てくる文字はすべて英語ですが操作は簡単です。ベクターにはほかにもチェスのアプリケーションがあります。

以上いいと思ったゲームです。自分の趣味です。ほかにもいろいろいいゲームはあるとおもいます。

何千円、何万円払ってするゲームもいいですが、フリーウェアのゲームもいいものがたくさんあります。PC を買っても特にすることがないとか(意外にある)、時間のある人は、こういうゲームはお勧めです。

阪急電鉄（株）80年の歴史

奥谷功一

1.1 阪急の歴史ここから始まる

年代	沿線開発	車両開発	その他
1910年3月10日	梅田～宝塚、石橋～箕面間竣工	1形	
1916年		34形	
1918年			社名を阪神急行電鉄㈱に変更
1920年	梅田～神戸上筒井、伊丹線営業開始	51形、47形	
1921年	西宮北口～宝塚間営業開始	37,63形	
1921年4月		1形(P-1)	北大阪電気鉄道㈱営業開始
1923年		81,40形	
1924年	甲陽支線営業開始	500(300)形	
1925年	新京阪鉄道，天神橋～淡路間営業開始	510,700(310)形10,50形	
1926年	梅田～十三複々線、西北～今津間開通	600 系 90,151形	
1927年		500(1500)，100形(P-6)	
1928年	嵐山線、淡路～西院（仮）間開通		
1930年		900形	
1931年	京阪電気鉄道㈱西院～京都（大宮）開通		
1934年		920系	
1935年		320形	
1936年	西灘～神戸（三宮）間新設	380形	
1937年		200系	
1938年		500形	
1940年	上筒井線運輸営業を廃止	96形	
1943年		300(130)形	社名を京阪神急行電鉄㈱に変更
1948年		550,700系	
1949年	北野線運輸営業休止	700(800)系 500(1550)形	京阪電気鉄道を設立営業の一部を譲渡
1950年		810,710系	
1953年		610系	
1954年		1000系	貨物運輸営業を廃止
1956年		210,1010,1100,1200系	
1957年		1300,1600系	
1959年	梅田～十三間複線増設工事竣工。三複線開通		

1.2 登場！現代の車両 2000 系

1960年		2000,2300系	
1962年		2100系	
1963年	千里山～新千里山間、四条大宮～河原町間営業開始		
1964年		2800,300,3100系	
1967年	南千里（新千里山）～北千里間営業開始	3300系	
1968年	高速神戸鉄道㈱開通。阪急・山陽相互直通運転開始	5000系	
1969年	地下鉄堺筋線と相互直通運転開始		
1970年		5200系	
1971年		5100系	
1972年		5300系	
1975年		2200,6300系	
1976年		6000系	
1980年		7000系	
1982年		7300系	
1987年		8000系	
1989年		8300系	
1995年		8200系	

2.1 ここから車両は始まった

1形 M43～S36	創業時に川崎造船所に発注した木造ボギー電車、1910年の開業にあわせて計18台が製造された。その後15両が追加されて、計33台が運用される。妻戸の変形や腰板の補強などいろいろな改造を行って1956年から順次廃車となり今では正雀工場に一両と宝塚ファミリーランドに一両あるだけとなっている。
34形 T5～S25	箕面線用として南海電鉄軌道線から譲り受けた路面電車の木造ボギー車プラットホームの高さに客室の高さを改造して使っていた。昭和8年からは今はなき北野線を走った。
37形 T10～S24	伊丹線専用として造られた丸みのある小型の木造ボギー車。小型車にしては珍しい高速運転が可能で後に箕面線、甲陽線を走った結果、能勢電鉄に譲渡された。
51形 T9～S35	神戸線の開通に備えて12両が製造された。宝塚線の1形をひと回り大きくした3枚扉で1形に比べ高速運転を可能にしている。大正15年に600系が導入されるまで神戸線で使用されてその後宝塚線に転属となった。
63形 T20～S31	神戸線を走る51形の増結用として12両が作られた。51形と同じ機器を使っているが、正面は丸形から平型に変わるとともに3枚窓に変更された。神戸線と宝塚線で2連運転で使われた。900形が登場すると宝塚線に転属。
151形 T15～S14	大阪市内の高架が開通すると従来の路面軌道のうち北野以南0.8km区間は北野線として残された。この北野線で使う電車として大阪市電から譲り受けた。手動ブレーキの四輪車で長崎電軌に譲渡されるまで使われた。
500系 T13～S42	阪急最初の半鋼製車両として神戸線でデビュー。車両形状は63形を継承しており、81形にも似ているが丸みがなく角張った印象を与える。後に300系に改番された。

2.2 戦前そして戦後へ

510形 T14～T15	実は知る人ぞ知る阪急の伝説車両の一つ。我が国では初の全鋼製車両として作られたが、十三で衝突事故に遭ってしまい登場1年で廃車となる。
320形 S10～S48	神戸線用に使われていた900形を小型化することにより宝塚線用として製造した全鋼車。自動空気ブレーキを採用しており宝塚線の近代化に献上した。
380形 S11～S52	320形のスタイルに引き継いでいるが車体幅は50mm広くなっている。戦後は4両が連合軍専用車として使われ、別の1両もアメリカ博開催の時に黄色に塗られた。その後は能勢電鉄で活躍をしていた。
500形 S13～S52	宝塚線用として320・380に次いで作られた。幅広貫通路を取り入れた2両固定編成で、偶数車が大阪向き、奇数車が宝塚向きとなっていた。後に8両が広島電鉄に譲渡され、23両が能勢電鉄に貸与され譲渡された。
600系 T15～S50	510形が試験的に作られたのに対しM車の600形とT車の800系は我が国最初の本格的な全鋼製車両として開発。十三-梅田間が複々線化したのに伴い神戸線に登場した。3扉で下降窓、深い屋根の形状は他社にも採用された。
900形 S5～S53	ダイヤ改正の特急としてデビュー。600系の重厚なイメージから脱皮、3扉から2扉に変わりクロスシートを備え洗練された軽快な車両して登場した。全鋼製車体を採用するとともに軽量化を考慮して設計されている。車体形態は以後の阪急の基本となった。今ではほぼ原型の姿に復元され正雀工場に静態保存されている。
920系 S9～S57	900形をベースとして作られた戦前の阪急を代表する車両。主電動機は当時最大昭和54から順次廃車となっているが950形の4両は今でも救援車として活躍している。
100形 (P-6) S2～S48	大阪～京都間の本線が開業するのに備えて、当時の技術の粋を集結して製造された。長距離高速電車の草分けになった画期的な車両で国鉄と併走する区間では特急「燕」を追い抜いたことで注目を集めた。合計73両が作られて車両によってはクロスシートやロングシートなどバラエティーに富んでいる。さらに116号車は技術遺産として正雀工場で動態保存されていて毎レールウェイフェスティバルに展示され、その後には毎回総点検が行われている。
550系 S23～S44	ナニワ工機(現アルナ工機)が初めて作った車両で宝塚線を走った。この車両以降阪急電車はすべてナニワ工機製になっている。アルナ工機では550号車の車体先頭部分を記念として保存している。
700系 S23～S51	宝塚線の550系を一回り大きくした車両。主に千里線で使われるため大きな力が出せるため歯車比を4:3にしている。車番は阪急方式のM車は700形、T車は+50して750形となっている。
800系 S24～S57	神戸線用として開発された車両で、車体は半鋼製。スタイルは920形を引き継いでいるが車体幅が50mm広くなっている。さらに京都-神戸間の直通特急として600-1500Vの複電圧車となり、両区間を70分間で走った。
1000系 S29～S59	昭和20年代の鉄道車両界に起こった高性能化に対応して開発された。戦後新造100両目の記念車両として神戸線にデビュー。軽量車体や発電ブレーキなど多くの新機構を取り入れている。このころから阪急では000が神戸線、100が宝塚線、300が京都線というように分けるようになってきた。この電車はひょっとしたら今でも能勢電鉄で走っているかもしれない。
1100系 S31～H1	1010形と同じ宝塚専用車両。MTで1ユニットとして歯車比を大きくしている。晩年は千里線、箕面線、嵐山線などで使われていたが平成元年とうとう廃車となった。
1300系 S32～S62	高性能車1010・1100の京都線版として登場。特に変更点などはないがブレーキやモーターに仕組みが1010・1100とは異なり高性能になっている。

2.3 現代の原型となった車両

2000系 S35～	阪急電鉄の印象を変えるような新型車で直線的で平面になり角張った印象を与える今ではすべてがこの形になっている。昭和36年には2300系とともにローレル賞を受賞していて今でも現役活動している一番古い電車。しかし客からは「乗り心地が悪い」と言われ、運転手からは「加速、減速の性能が悪い」とそしてバイトからは「方扉がいや」などと言われたい放題言われていて今ではかわいそうな電車である。もしも阪神大震災がなかったら現在は走っていないはず。
2300系 S35～	2000系が神戸線用であるのに対し2300系は京都線用。2000系と全く同じデザインにも関わらず制御方式は大きく異なっている電圧の違いから2300系は主電動機を4台直列につないでいる。
2100系 S37～S60	2000系の宝塚線用として作られ、当時の最高時速80km/hと高性能が要求されないためモーターは100kwと小さく歯車比も小さくなっている。今は廃車になって能勢電鉄で走っているかもしれない。
3000系 S39～	神戸線が600Vから1500Vに昇圧されるのに対応し、昇圧転換に即応できるように開発された。車体などは2000系を継承している。
3100系 S39～	昇圧に備え、3000系の宝塚線用として作られた。2000と2100の関係と同様に3000よりもモーター出力は小さくしてあり半面歯車比を高くした。
3300系 S42～	京都線が大阪市営地下鉄と相互直通運転を始めるのに対応して作られた車両。車体は標準車体に比べて100mm短く幅は100mmひろくなった。
5000系 S43～	神戸線昇圧ご最初に投入された1500V専用車。車体事態は変化がないが1500V専用としたため電動車2両分の8個のモーターを1台の制御器でコントロールする1C8M方式を採用。この電車から車掌台にしきり窓が取り付けられた。
5200系 S45～	阪急最初の冷房付き新造車。冷房装置のダクトを配備したため5000系より屋根が70mmたかくなった。それ以外は何も変わらない。
5100系 S46～	5000・5200をベースに車体や機器を改良して製造された。神戸・宝塚線共通運用を可能とし、新造時は京都線でも活躍した。
5300系 S47～	5100系の京都線用として作られた。パンタグラフは当初から2基搭載され、全電気指令式ブレーキを採用した。
6300系 S50～	京都線を走っていた2800系の特急電車の後継者として誕生しした。電気機器や台車は5300系と同じだが、T形ワンハンドルマスコンを導入。扉も3扉から2扉に減り車体も一回り大きくなった。
6000系 S51～	2200系と5100系の特徴を備えた神戸・宝塚専用車両。2200系からはこの車種に採用された新車体を使い電気指令式ブレーキを搭載。モーターや電気機器は5100系を引き継いでいる。ものによっては全自動密着連結器を備えているものもある。
2200系 S50～	電子技術を使ったチョッパ制御装置を初めて導入した。VVVF制御装置や交流主電動機などの長期使用のために使用された。阪神大震災によりこの形式は廃止された。
7000系 S55～	6000系をベースに改良を加えた神戸・宝塚線用の車両で、新たに界磁チョッパ方式の制御装置を採用している。制御器に電子部品を多く使い、低速まで回生ブレーキを活用して架線に電力を帰す。
7300系 S57～	7000系を京都線用として開発された車両。7000系とほとんど変わらない。7000系とこの電車の謎な点として、なぜ上部にアイボリー塗装したかは不明。7300系はものによってはVVVF装置を搭載しているものもある。

2.4 今技術は革新した。そして未来へ

8000系 S63～	阪急80周年を記念して開発された車両で、神戸・宝塚線で活躍している。省エネと保守の軽減を測り経済性を高めるVVVF制御を本格的に導入。速度計や圧力計はデジタル表示とし、機器の動作状態を監視するモニタ装置も本格的に導入した。さらに客室の窓にはパワーウィンドウを採用そして8004以降の新車両には冬季の窓の曇りが防止がされた。4期にわたり製造され第1期は車番が正面扉の下の方に書かれており、第2期ではそれが左窓のすぐ下に移動、第3期では左妻窓の中に移動、そして第4期では車番の位置は3期の窓の中と同じだがシングルパンタを採用。といったようにいろいろとバラエティに富んだ電車でもある。
8300系 H1～	神戸・宝塚線の8000系とほぼ同じ車両の京都線版。8315-8415ユニットには今までのよろい戸（日よけ）がフリーストップ式のカーテンに変わり、8304-8404ユニットは曇り止めガラスが真中の固定窓だけ、とおもしろいことになっているものもある。
8200系 H2～	関西初の座席収納車両として神戸線に登場。車内の混雑の解消する切り札として4両が製造された。VVVFインバータ、ＡＢＳをはじめ様々な先端技術を導入している。車内にはカラー液晶モニタがあり、沿線情報や天気、スポーツ情報などを提供している。しかし今では朝の通勤急行としてしか使われていないのでお目にかかれるのはかなりまれ。レールウェイフェスティバルに行くと見ることができる。

3.阪急バイト経験日誌

H11 年 11 月 13 日

やめましょう飛び込み

今日は穏やかな日でバイトももうすぐ終わると思った 8 時半頃、上牧駅にていきなり線路に飛び降りる高校生を発見！！たまたま電車が着ていなかったからよかったものの『電車が着たらどうするねん』と思った。線路は以外と高くて私の身長(165cm)では目の高さほどある。いざというときにはすぐに上れないのが事実である。しかも普通の人ではいざというときの隠れ場所（普段は線路点検時に使用する避難場所）の位置を知らないだろう。もしも降りたいのなら何か一言ぐらい言ってから降りてください。たぶん誰もいいとは言わないと思いますが…。

H11 年 11 月 15 日

いつもの遅刻者

どこのバイトでもいると思うが、ここでもいるのだ。遅刻の常習犯が。しかも今日のパートナーはそいつ(N)だった。待つこと電車 2 本やっと N が到着した。無論私は怒った。しかしNはいつものことなのであまり気にしていなかったみたいである。しかしこれは私にとっては迷惑この上ない。まだ通勤時間が終わって登場するよりもよっぽどましだが…。とにかくこの遅刻者に対しては何か対策をとるべくではないでしょうか。

H11 年 11 月 16 日

普段のストレス解消法 in 阪急バイト

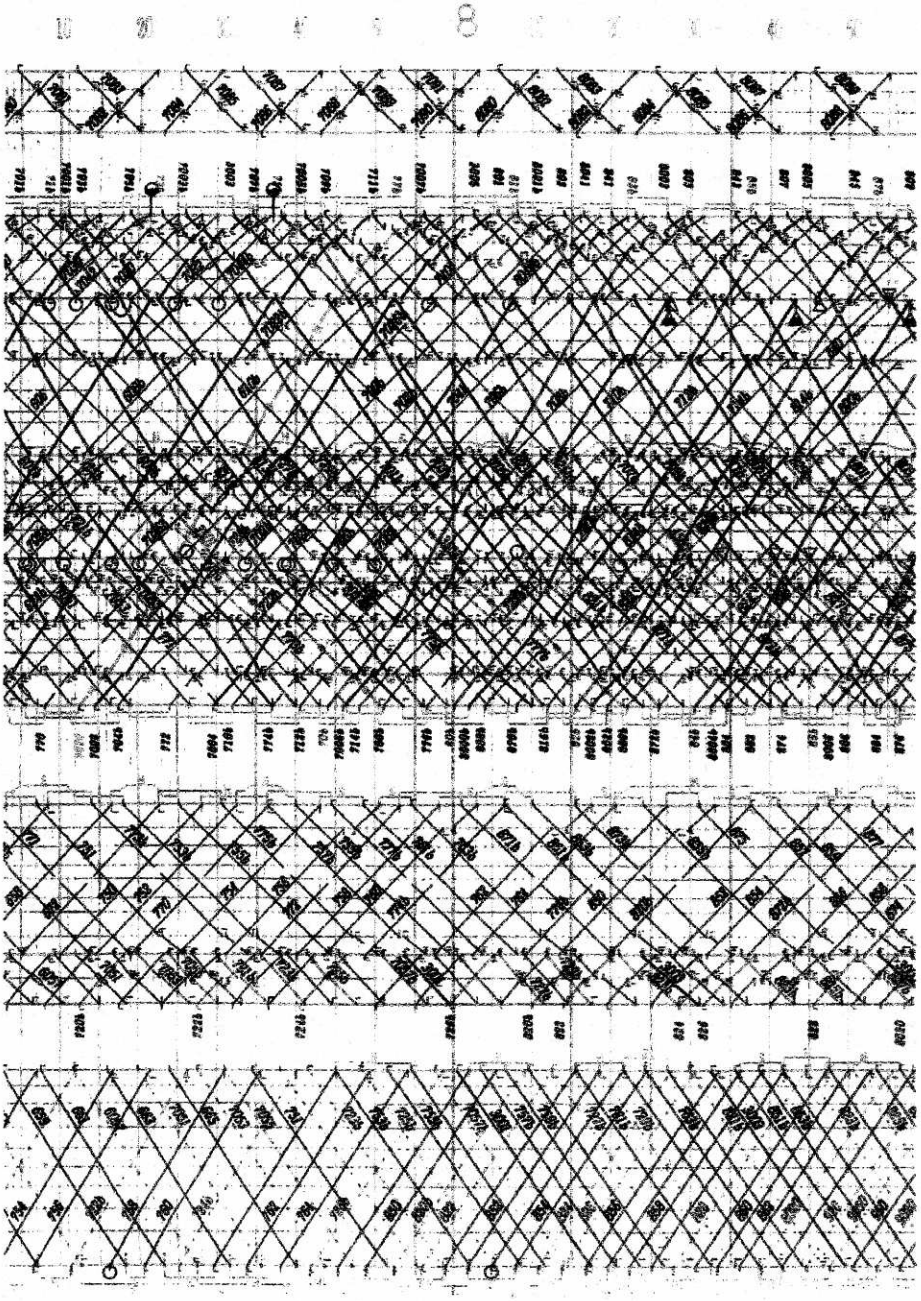
今日は珍しく高槻市駅でバイトをしていた。しかも下り後ろ(大阪方面後方エスカレーター付近)。ここは普段乗降客も多いので、駆け込んでくる人もかなり多い。しかものんびりと歩く人もいる。そんなことなどがあり結構ストレスがたまったりする。しかしそんなときに嬉しいのが知り合いの嫌いな人が駆け込んできたとき。この日、特急は 1 分遅れて到着した。通勤時間帯だと当たり前のことなのだが、これ以上遅らせてしまうと乗り継ぎなどの問題に支障が出るのでいち早く出さなくては行けない。そんなことを考えながら、扉付近の人がみんな乗車したのを確認して手を挙げて、扉を閉めるサインを出した。そこに私の知り合い H さんが駆け込もうとしてエスカレーターを駆け上がってきた。これはチャンスと思い扉を持った。むろん私が手を挙げたので扉はもう閉まりかけているが、足で支えればそんなに重いものではない。扉を支えること数秒。H さんは安心したに違いない。しかし世間はそんなに甘くない。私は H さんの目の前で扉を閉めた。むろん H さんは入ることができない。そして H さんは私の方をにらんできた。これこそがストレス解消である。嫌いな人に嫌がらせをする。究極のストレス解消ではないか。しかも、もしも相手が何か文句を付けてきたら他人の振りをして「我が社では駆け込み乗車は認めていませんのでご了承ください。」とでも言っておけばいい。いつもこんなことばっかりしてストレスを解消しているのだが本当にこんなことって認められるのだろうか？まあいいやバイトだし…。

H11 年 11 月 17 日

何故？

今日はとてつもなく寒かった。しかも逃げるところの一切無い上牧だったので風がさらに吹き付ける。そんなときでも 8 時をすぎると気温の上昇により暖かくなってくる。 しかし、熱くなりすぎる人もいた。K 高校の人達が S 高校の人をからんでいた。むろん私は止めなければならないのだが、ただの言い争いだったのであえて止めようとはしなかった(無論喧嘩になれば止めに入る)。人数は K 高校 5～6 人に対し S 高校 1 人。あまりに多勢に無勢ではないかと思いながらいつでも走り出せる用意をしていた。しかし、しばらくしたら S 高校の人は手を出した記憶はないのだが、K 高校の人たちが「今度 OB 連れてくるからな」と言葉をはいて逃げていった。『何故？』何でこうなるのか事実が全くつかめなかった。ふつうこういう場合は S 高校の人がこういう捨て台詞をはいて逃げるのではないでしょうか？いったい何があったのやら‥‥。

4.阪急京都線通勤時間帯ダイヤグラム



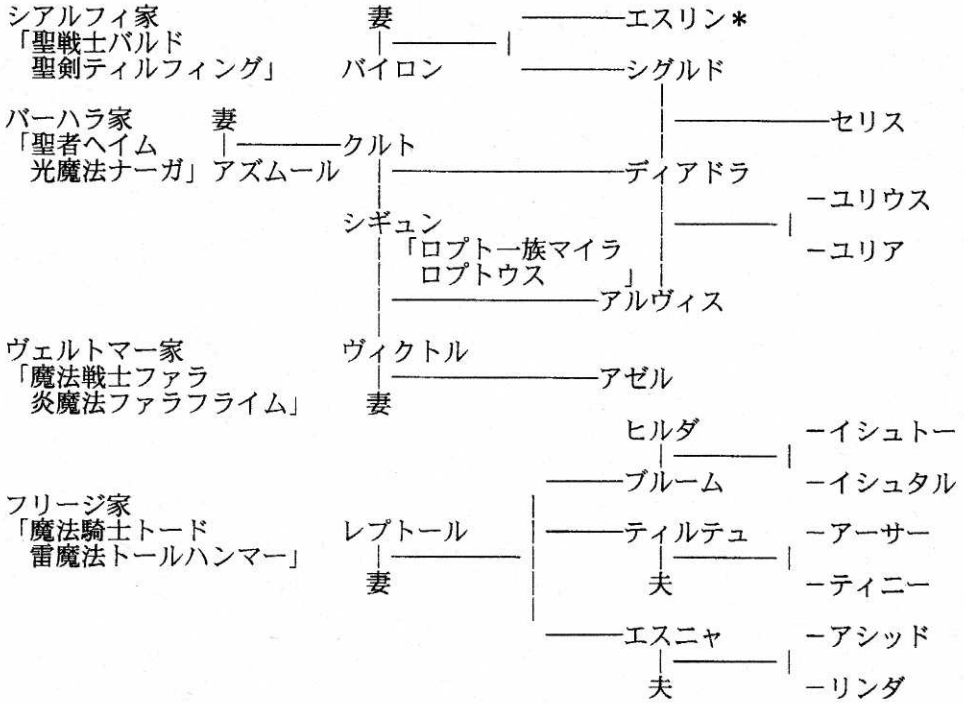
5.平成 11 年 11 月 11 日の入場券

FIRE EMBLEM -聖戦の系譜-

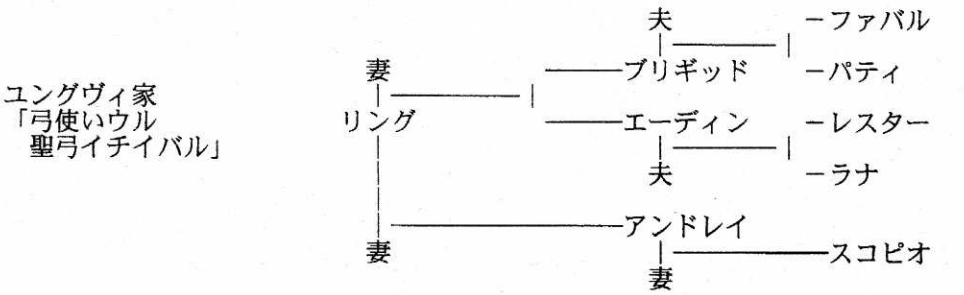
野川 博司

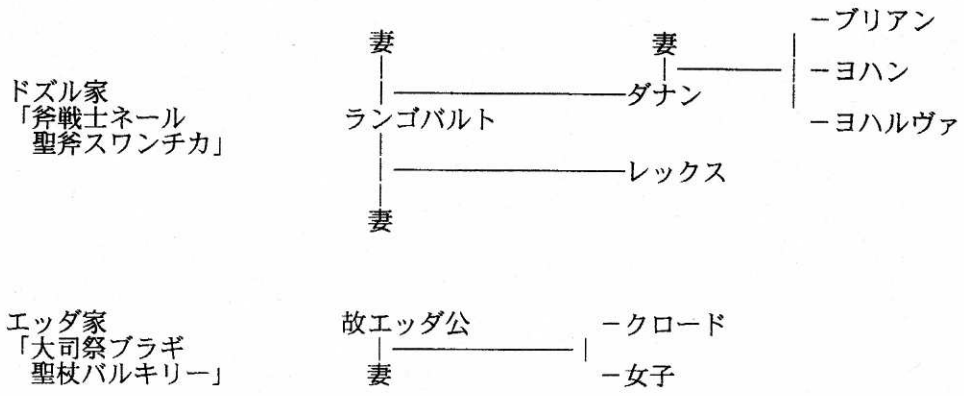
The Kingdom of GRANDBELL
グランベル王国　　主な産業：農業

大陸の中央に位置し、長い歴史を有する大国。この地はグラン王国発祥の地であり、土壌にも恵まれていたため文明の発達は早かった。途中ロプト教団の支配を受けて一時停滞したが、その後復興を果たし、大陸一の大国の座を得るに至る。現在の王国はバーハラ王家以下、６つの公爵家が統治する７つの公国から構成されている。政治、文化、軍事のいずれにおいても高い成熟度を誇っている。



＊複数の子がいる場合上から年齢順に記しているが、シアルフィ家のシグルド、エスリンのみ婚姻の関係から順序が逆になっている。





＊キャラクター（第１部）
　シグルド：本編第１部の主人公。序盤から初期値と成長率が高いので、とても使えるキャラ。聖剣ティルフィングを手に入れると、魔防が大幅に強化され、魔法使いに対しても有利に戦えるようになる。
ストーリー上では、敵国の王子であるシャナンとアイラをかくまった為、母国から反逆者として追われる身となる。ディアドラと熱愛電撃結婚するが、マンフロイによってディアドラはさらわれ、結局アルヴィスに取られてしまう。そして最後にはアルヴィスに罠にかけられ、死地に追い込まれた時、妻が奪われた事実を知らされ、絶望の中死んでいく。悲劇の主人公。

　エスリン：シグルドの妹であり、キュアンの妻。序盤唯一のトルバドールなので、機動力を活かした杖による回復には助けられる。さらに、キャラクターに似合わない必殺のスキルを持っている。あまり前線に出るタイプのキャラではないのでせっかくのスキルが勿体無く感じてしまうが、二人の頼りになる子へと受け継がれるので、頼もしく感じる。
グランベル軍が大遠征の隙を突いてヴェルダンの軍がユングヴァ公国に攻め込んできた折、わずかな兵を率いて戦いを挑んでいる兄シグルドを助けるため夫のキュアン、その部下フィンと共に援軍として駆け付けてくる。途中夫とフィンと共にレンスターに引き返すが、後にキュアンが地槍ゲイボルグを手に援軍を連れて助けに来る際、見送りのため幼い娘アルテナを連れ、途中までついてくるが、その時、敵国であるトラキアの王トラバンド率いる軍に襲撃され、殺されてしまう。そして連れていた娘を盾にとられたため、キュアンはゲイボルグを捨てトラバンドに殺されることになる。その際ゲイボルグとアルテナはそのままトラキアに連れ去られてしまう。

　バイロン：シグルドの父親。ゲーム中３人しかいないマスターナイトの一人。もうそれだけで仲間に欲しいのだがNPCキャラとしてのみ登場しシグルドに聖剣ティルフィングを渡し死亡する。
クルト王子暗殺という無実の罪にかけられ終われる身となり、その逃走中に息子に出会い聖剣を渡し死亡。

　ディアドラ：シグルドの妻となり光皇子セリスを、アルヴィスの妻となり闇皇子ユリウス、ユリアを産むキーキャラクター。光魔法オーラを持ち魔力と魔防の成長率がずば抜けて高い。HPの伸びさえよければ、前線でも通用するキャラである。しかし、すぐにさらわれるし、子供の能力に一切関与しないので育てなくても気にしなくてもいい。
聖者ヘイムの直系クルト皇子とロプト一族マイラの血（直系かどうか不明、おそらく直系ではない、なぜならロプト一族の祖はガレであり、マイラはロプトの中の派閥であるマイラ派の祖であるから）を継ぐシギュンの間の子である。しかし、シギュンはヴィクトルの本妻であった。シギュンとクルト王子の関係を知ったヴィクトルは自殺してしまう。それを苦にしてシギュンはヴェルダンの精霊の森で隠遁生活を始めてしまう。そこで、成長したディアドラは偶然？シグルドと出会い、結ばれる。それが、悲劇の始まりである。

　アルヴィス：ヴィクトルとシギュンとの間の子、魔法戦士ファラの直系。仲間になることはないが、中立軍として序章で現れる。パラメーターでは全キャラ中最強だと思われる。第１部で直接戦うことはないが、クルト王子やその重臣達の暗殺を影で操り、主人公シグルドを殺害する事により大陸全土を支配する覇王となるなど、一見悪玉キャラであるが、第２部では息子ユリウスが行う子供狩りを阻止しようとしたり、自分を殺させるため主人公セリスに聖剣ティルフィングを渡したりと、かなり良い所をもっていちゃってます。実

際には全て民のために、争いを無くそうとしての行動だったのだが、マンフロイに利用されていたため、ほとんどの行動が裏目に出ている。一番のミスはマンフロイの策略で、記憶を失った異父妹ディアドラを妻に迎え入れた事である。しかし、個人的には敵キャラで一押しのキャラ。

アゼル：ヴィクトルとシギュンの下女だった娘との間の子。アルヴィスの異母弟。兄に比べなんと戦闘力が低いのだろう。しかし序盤戦唯一の魔法使いなので重宝する事もあるだろう。僕には冴えない印象しかないが。
ユングヴァ公国がヴェルダン軍に襲われたという事実を知り、恋焦がれるエーディンの身をあんじ、兄アルヴィスの元を飛び出しレックスと共にシグルド軍に援軍として合流する。最後にはシグルドと同様、兄アルヴィスの罠にかかり死亡。

レプトール：実質的には第１部のラスボスにあたる。プレイ中幾度となく行く手を阻む伝説の武器、雷魔法トールハンマーの継承者、圧倒的な命中率を誇る。
クルト王子暗殺の実行犯。結局アルヴィスに裏切られ主人公達に殺されてしまう。哀れな奴だが僕にとってその印象は限りなく薄い。

ティルテュ：レプトールの長女。HPが半分以下だと必ず必殺を出し続けるという恐ろしいスキル、怒りを持つ。その割にはパラメーターが低いため、使いづらい。僕にはアゼル同様冴えない印象しかない。
クロードを慕ってシグルド軍に参加。父親を敵（もうこのゲームじゃ父親、母親、兄、叔父、叔母殺すなんて日常茶飯事）とすることに思い悩んでいる。アルヴィスの罠にかかり死亡。

ブリギッド：聖弓イチイバルの継承者。登場時レベル１２にも関わらず初期能力がこれでもかというくらい高くHPと運の成長率がずば抜けて高い。しかもイチイバルは最強の攻撃力と毎ターンHP回復という驚異的な能力を叩き出している。しかし個人スキルがないのがイタイ、兵種スキルの追撃だけではその能力を存分に発揮できない。せめて連続があれば、と思わせられる。あとブリギッド本人の命中率は良いのだが、イチイバル自体の命中率が悪く、重いので思ったほど活躍しない、ジャムカの相棒どまりである。
アグストリアの北方で活動する、オーガヒルの海賊の頭目として登場するが、実は行方不明になっていたユングヴァ公国の長女で聖弓イチイバルの継承者である。シグルド軍と交戦中に妹のエーディンに出会いその事実を知らされ、その後シグルド軍に参加する。アルヴィスの罠により死亡。

エーディン：もてもて姫。回復役以外の何者でもない。運のみ高い。シスターなんてそんなものだ。しかしなぜか男キャラにモテル、アゼルに、ミデェール、ジャムカ、ガンドルフ（？）、ことごとく虜にしている。僕はあまり好きではない。お前はミデェールとでも結ばれてろ。あとイロイロな武器を隠し持っている。いろいろなイベントがある度に小出ししていく。恋人には勇者の弓、姉にはイチイバル、エスリンにはワープの杖といった感じだ。ある意味超重要キャラである。
序章でヴェルダン軍にさらわれてしまう。この事件こそ、この物語の始まりである。しかし、ヴェルダンの第３王子ジャムカによって解放されシグルド軍と合流し共に戦っていく事になる。余談だが代々弓騎士の家系であるユングヴァで彼女だけシスターなのは、行方不明の姉の無事を祈るためだとか。

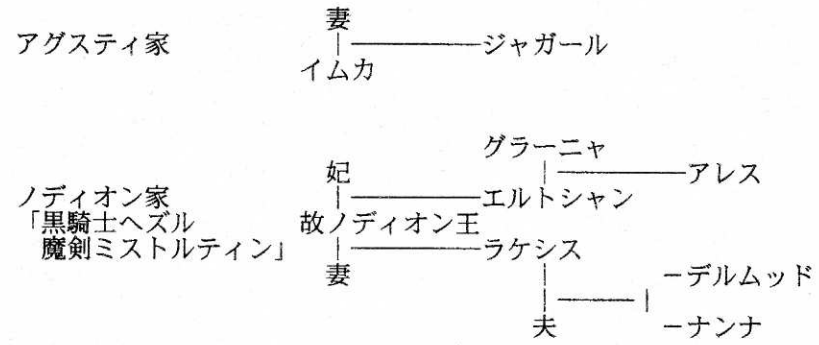
アンドレイ：誰それといった感じ。ただの雑魚キャラ。確かシレジアに弱い弓兵部隊を率いてやって来たような。

ランゴバルト：アルヴィスやレプトールと結託してクルト王子暗殺に関与した野心家。直間両用の聖斧スワンチカを持つ。防御と魔防は大きく強化されるが、いかんせん命中率と重さが異常でまるで使い物にならない。伝説の武器をもつボスキャラとして冴えないとしかいいようがない。

レックス：ランゴバルトの次男。エリートのスキルを持つため経験値が２倍得られるのでレベルアップが速く、かつ力と守備力の成長率が悪くないので使いやすいキャラ。しかし、それも勇者の斧を持てばこそ、それがなきゃ単発攻撃しか使えない冴えないキャラに。個人的にはさくさくレベルが上がるので、結構好きなキャラ。
仲の良いアゼルの頼みによりしかたなく、といった感じでアゼルと共に援軍に駆けつけて来る。後にシグルド軍はランゴバルトと敵対することになるが、もともと野心的な父親を快く思っていなかったのでそのまま敵対することになる。当然のごとく彼もアルヴィスに殺される。

クロード：聖杖バルキリーを使うハイプリースト。初期レベル２０とかなり高いレベルで登場するが、魔力と魔防ほぼ完成された高さを誇るのであまり苦にならない。基本的に杖しか使わないので、パラメーターはさほど気にしなくて良い。死者を甦らせるバルキリーは要る人には大変重宝されるものだと思う。
クルト王子暗殺に関する真実を確かめるべく、ブラギの塔を参拝し、ブラギ神からの信託を受け、暗躍する強大な闇の力を知り、ティルテュと共にシグルド軍に加わる。そしてアルヴィスの手により死亡。

Dominion of the loads AGUSTRIA
アグストリア諸公連合　　主な産業：農業
ユグドラル大陸西部に位置し、５つの小国からなる連合国家。国家の主産業は、肥沃で広大な土地を活かした農林業で、小麦をはじめとする各種の穀物は外貨獲得の貴重な手段となっている。またオーガヒル海賊が出没する北部の未開拓地を除く中央および南部は開発が進んでおり、平野部には大規模な田園や森林が広がり、一大穀倉地帯を形成。国民の多くは農業に従事しており、その生活レベルは比較的高い。
また近年、国家が指導する独自の農業政策の成功により、極めて生産性の高い農業を行うことが可能となった。これに伴って国力は上昇の一途をたどり、近年ではグランベル王国と比肩しうるまでの大国に成長しつつある。
国家の政務は、ヘズル直系のアグスティ家によって執り行われており、ノディオン家、ハイライン家、マッキリー家、アンフォニー家がこれに従い、隣国のグランベルと同様、主従関係の上に成り立っている。隣接国が大陸の覇権を争う大国グランベル王国だけに外交に関してはことさら慎重な姿勢を貫いており、現在は休戦状態にある。
その他の周辺諸国との間には不可侵条約を結ぶことで、自国の平和を保っている。



ハイライン家、マッキリー家、アンフォニー家はゲームをプレイすれば、敵として登場するが、まったく重要な王家ではないので家系図は、はっきりと僕には分からない。

＊キャラクター（第1部）
ジャガール：父親イムカ王を暗殺して王位についた実力なき王。自分に逆らったエルトシャンを幽閉していたにもかかわらず、シグルド軍が攻めてくると、エルトシャンに頼ろうとする様は悪役そのもの。結局シグルド軍との和平を薦めるエルトシャンを処刑してしまう。愚かな行動により最高の手駒をなくした王、さしずめクイーンを失ったキングはシグルド軍の攻撃により死亡する。

エルトシャン：獅子王の異名を持つ魔剣ミストルティンを操る若き王。シグルドの親友であるが忠義のためシグルドと戦うことになる。エルトシャンが率いている大陸最強と言われている（なにが最強なのかよくわからないが、エルトシャンの指揮能力が５つ星に加え、統率がとれているので非常に戦いづらいのは確かだ）クロスナイツもさることながら、エルトシャン本人が強い強い。間違いなく第１部前半の難所である。このゲームのカッコイイ敵キャラの代表だ。

ラケシス：エルトシャンの異母妹。１部で唯一マスターナイトになれる仲間である。しかし成長率があまり高くないのであまり攻撃向きのキャラではない。どちらかと言えば、前線でも安心して使える回復役といったところだろう。エルトシャンから貰える大地の剣でならば前線でも攻撃役として使える。
エルトシャンが大遠征中で手薄なグランベルに攻め込もうとするジャガール王をいさめるために王都アグスティに向かっている間にラケシスの居るノディオン城にハイライン軍が

攻め込んでき来る。その時シグルド軍に助けられたラケシスは兄を救うためシグルド軍に参加することのなる。シグルド軍に攻めてくる兄に、もう１度王に和平を薦めるよう説得する。そのためにエルトシャンは死んでしまう事になる。

The Kingdom of THRACIA
トラキア王国　　主な産業：漁業、狩猟、採掘
大陸南東に位置するトラキア半島の南部を占める国家。１２聖戦士の一人、竜騎士ダインによって建国された。国土の周囲に"ユグドラルの天井"と呼ばれる数千メートル級も山々が連なる山脈が走っているため、住居可能な地域は面積の割に少ない。天に突き刺さるかのごとくそびえる数々の岩山は、他国の侵入を防ぐ天然の要塞としての役割をも担っている反面、「陸の孤島」と表現されることもある。国民の多くは山の中腹や谷間に小さな集落を作り、畑で採れるわずかな穀物を頼りにひっそりと生活している。また鉱山から鉄を掘り出し、外貨に替えるなどしているが、その量も多いわけではなく、特にこれといった特産物もないため、国民の生活はけっして裕福とはいえない。さらには、国を支えるべき騎士たちが、各国の傭兵として日銭を稼ぐといったことも、近年ではごく日常的に行われており、貧困の度合いは深刻さを増す一方だ。

トラキア家　　トラバンド
「竜騎士ダイン　　┣━━━━━━━━アリオーン
天槍グングニル」　　妃

The Kingdom of LENSTER
レンスター王国　　主な産業：農業、工業、漁業
トラキア半島の北部、マンスター地方の一小国。同一の半島にありながら、トラキア王国とは対象的に牧歌的な田園地帯が広がる。穏やかな気候に恵まれた国家である。国民の生活は比較的裕福で、農村部では農業や牧畜、海岸部では海運や漁業、都市部では小規模な手工業による焼き物や木工品などの生産が行われ、高度な技術を用いて作られた各種の工芸品は、諸外国から高い評価を得ている。文化面では西のグランベル王国の影響を受けて年に一度、居城の中庭が民衆に解放され、音楽会や演劇などが、開催されており、国民の歓心を得ている。なおグランベルとは軍事面での結びつきも強く、南のトラキア対策として群議が結ばれている

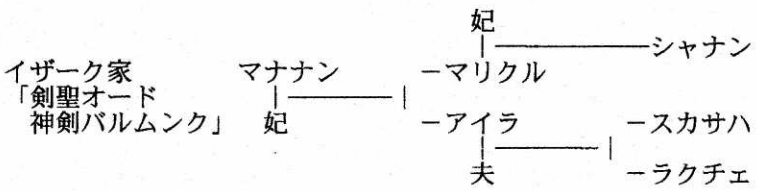
レンスター王
レンスター家　　┣━━━━━━━━キュアン　　－アルテナ
「槍騎士ノヴァ　　妃　　┣━━━━━━┫
地槍ゲイボルグ」　　エスリン　　－リーフ

＊キャラクター（第１部）
トラバンド：トラキア王国の王。キュアンとエスリンを殺害した本人。キャラの特徴は完成されたキュアンといった感じ。彼もアルヴィス同様悪玉にしか見えないが、内面ではなによりも自国の民の事を大事に思っている。個人的にはアルヴィスに並ぶくらい好きな敵キャラである。

キュアン：高い攻撃力と防御力を持つ。だだスキルが連続だけなので単発攻撃になりがちで、いまいち使いづらい。しかし帰国するほんの少しまえに、ゲイボルグを手にし、やりたい放題の強さを得るが、すぐ帰国し、その後仲間になることなく死んでしまう。大変惜しいキャラ。

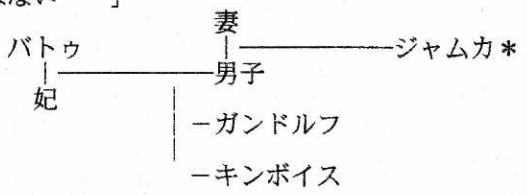
The Kingdom of ISSAC
イザーク王国　　主な産業：牧畜
大陸中央に横たわるイード砂漠の東方に位置する辺境の国。国土の大半を占めるのは、荒涼とした茶褐色の大地で、緩やかな丘陵地帯が地平の彼方まで続いている。イザーク王国はそんな牧歌的な風景が日常的に繰り広げられる遊牧民の国であり、その生活は自由であると同時に厳しい。人々は古来より遊牧を生業とし、厳しい自然の中で馬や山羊、ラクダなどによる牧畜を行っている。かつて、この土地にはれっきとした国家は存在しなかったが、グランベル王国成立後、剣聖オードの血を引くイザーク王家が国土の統一

に成功、ようやく国の体制が整った。とはいえ、一部の好戦的な部族は相変わらず抗争や略奪を繰り返しており、国家の威光は完全には浸透していない。



The Kingdom of VERDANE
ヴェルダン王国　　主な産業：漁業、狩猟
大陸の西南部に位置し、鬱蒼とした森に囲まれた蛮族の土地、それがヴェルダン王国である。ユグドラル大陸屈指の大湖を国の中心に抱き、住人たちは漁業や狩猟を中心とした生活を営んでいる。湖の西側に突き出した岬とその周辺に広がる精霊の森と呼ばれる一帯は古より聖域とされ、神秘的な出来事に関する逸話は枚挙にいとまがない。

ヴェルダン
「ヴェルダンには１２聖戦士
の血を引く王家はない　」



＊バトゥ王の長男の息子であったが、この夫婦が他界したため三男として養子となる。

＊キャラクター（第１部）
アイラ：イザークの女剣士。初期能力が高い割に、技と素早さの成長率が異様に高い。スキルも追撃と見きりと流星剣とそろっている。特筆すべきは五回攻撃ができる流星剣である。魔防以外は全く心配はない。勇者の剣を持った時点で敵はない、１部のラスボス、レプトールでさえ個人で倒すことが可能である。
イザークがグランベル軍に攻め込まれたため、後継者のシャナンを逃がすために、現王子マリクルの妹であるアイラがシャナンを連れグランベル軍の手の届かないヴェルダンに逃亡した。そこで、シャナンがキンボイスに捕らわれしかたなく、キンボイスに従いシグルド軍と交戦するが、シグルドがシャナンを救出したので恩に報いるために、シグルド軍に加わる。結局アイラも他の仲間と同様の最後を迎える。

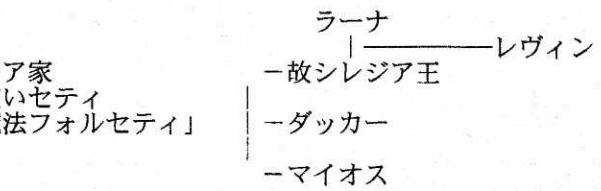
ジャムカ：個人スキルに連続、突撃と兵種スキルに追撃と攻撃系のスキルが揃い踏みである。キラーボウを持つので必殺のスキルもはいるので異常な攻撃回数と攻撃力を誇る。各種パラメーターも高いので、弓兵にかかわらず前線に立たせて囲まれてもぜんぜん怖くない。戦闘能力はアイラに匹敵する。
バトゥ王の三男であり、上の二人の兄とは違いグランベル王国への侵攻は反対でガンドルフが捕らえてきたエーディンを解放しサンディマにそそのかされているバトゥ王に反抗していたが、父の気が変わらないと知ると仕方なく協力する。そして、シグルド軍と戦うが、エーディンの説得によりシグルド軍に協力する事になる。結局アルヴィスに殺される

バトゥ王：ロプト教のサンディマにそそのかされグランベル王国に侵攻を企てるが、結局サンディマに殺される。

ガンドルフ：グランベルのユングヴァ公国に侵略した張本人。エーディンをさらったこと以外印象はない。だだの三下。

キンボイス：シャナンを人質にしてアイラを指揮下においてシグルド軍に襲いかかって来る。アイラは怖いが、キンボイス自体ぜんぜんたいした事はない。無論三下。

The Kingdom of SILESIA
シレジア王国　　主な産業：漁業、狩猟
ユグドラル大陸最北端に位置するノイマン半島を統べる国家。１２人の聖戦士の一人、風の聖戦士セティによって建国された。国土の大部分が氷山と深雪に覆われており、最北端に広がる壮大な氷河地帯は、太古より魂の宿る場所として有名である。王都シレジア城を筆頭に、セイレーン城、ザクソン城、トーヴェ城、リューベック城の５つが存在する居城はいずれも非常に堅牢で、厳しい風雪にも耐えられるように設定されている。人々は主に、小規模な狩猟や漁業、そして毛皮産業などによって日々の糧を得ており、あまりに冷涼すぎる気候のため、農業はさほど盛んでない。



＊キャラクター（第１部）
ラーナ：レヴィンの母親で、現王妃。レヴィンに王位をついで欲しがっているが、当のレヴィン本人は王位というかたぐるしさを嫌い。バードとして各国を放浪していた。が、シグルドと出会い、考えを改めファルセティを継承し、王位につくことを決心したが、その後すぐに、アルヴィスの罠にはまる。しかし、それに生き残り単身マンフロイに挑み、殺される。が、聖戦士セティにより命を吹き返す。そして、次の世代の者を導く存在となる。

レヴィン：ほぼ無敵のキャラクター。ホルセティを持てばなおさらだが、エルウインドでも十分強い。とにかく速さの初期値が高いにもかかわらずHPと速さの成長率がずば抜けているので、自然と速さ３０になる。とにかく１部の最強キャラである。

ダッカー：レヴィンの叔父。前王が死んだ今、自分がシレジアの王になろうと王妃のラーナを排斥しようと考えている。

マイオス：レヴィンの叔父。上に同じ。

以上、にわかFEファンの戯言でした。