Hyakumangoku

2000
Autumn

祝！
調布祭
２０００！

部長挨拶ほか
充実のMMA電脳執筆陣が
秋の新作リリース
Fleshers:oka,sano
Veterans:uirou 他

冬直前！
超ホットラインナップ！！

by K.T.Omada-

部長挨拶

MMA 部長 本間 祐介 (shimiz98)[1]

2000 年 11 月

MMA の出展へようこそ。MMA は Microcomputer Making Association (マイクロコンピュータを作る会) の略です。昔はコンピュータセンターが買ったマシンより速いマシンを秋葉原で買った IC を半田付けして作っていたりしたそうですが、いまではソフトウエア開発など各自が興味のある事をしています。そして活動報告としてこの百萬石を春の新入生歓迎期間と秋の調布祭に発行しています。出展と合わせて、どうぞお楽しみ下さい。

私の3D lib

青木 健太郎 (aoki)[1]

2000 年 11 月

青木といいます。グラフィック関係の事が好きです。グラフィック回りの事で興味のある事を自分で実装してみたいと思っています。

今回は、私が4年の卒業研究で作る事になった3 DCG ライブラリ[2] について簡単な紹介の記事を書こうと思います。

3 DCG に興味があるのだけれど、自分で全部プログラムするのは大変そうと思っている人が この記事を読んで、 自分にも3 DCG のプログラムはできそうだと思ってもらえたら大変嬉しいです。

現段階では、自分の作ったプログラムはライブラリと呼べるような代物ではなく、技術的な未熟さ、プログラムの設計の悪さなどの理由により、自分でも使うのに苦労する程です。

また、機能の方もいろいろな事ができる訳ではありません。ですから、売られている3 DCG ソフトのようなものを期待されている方にとっては、期待はずれに終る事でしょう。

今はまだ未完成であり、年末あたりをメドにまとめるつもりでいます。まだできていないものをどうやって解説しようか迷ったのですが、開発の過程に於いて、つまずいた事や気づいた事を中心に述べて行こうと思います。

以下では3 DCG ライブラリを可視化ライブラリと呼び、可視化ライブラリとは可視化プログラム開発者が可視化プログラムの作成を容易にするプログラム部品であるとします。

可視化ライブラリには、いくつかのアルゴリズムとその実装が含まれます。そのアルゴリズムは、３次元の回転変換、視野変換、透視変換、ビューポート変換などが基本的なものです。以上の基本的な変換を用いる事により、３次元のワールド座標上に与えたデータをスクリーン上の２次元平面に投影して表示する事ができます。

おおまかな流れを説明すると、表示する元になるデータ (オブジェクト) をモデリング座標系で定義する。ここで言うモデリング座標系は、データ (オブジェクト) を与える際に用いる便宜上の座標系であるとします。この座標系の取り方はどのようなものでも構いません。

次にモデリング座標系で定義されたデータをワールド座標系に配置します。

オブジェクトが人間、動物、車などに対応し、ワールド座標系がオブジェクトが存在する地球に対応すると考えると分りやすいかもしれません。次に視点の設定を行ないます。視点は視線ベクトルを与える事で、設定を行ないます。視線ベクトルは視点、視点が見ている先の点 (目標点と呼ぶことにする) を結んだベクトルであると考える事ができます。この他、視点とスクリーンまでの距離、視野角などを設定して中心投影を行なうことにより、３次元上のデータをスクリーンに投影します。

==========

1 ＜ディレクトリ構成＞

dat データのファイル入出力用

doc ドキュメント用 buhin まだ整理されていない関数群 garakuta まだ整理されてない関数群 etc 等高線のアルゴリズム用の双方向リンクリストを用いた関数群まだ、ライブラリの本体と合わせて動くようにはなってない。調整が必要。myhead params.h マクロの設定 global_var.h グローバル変数の設定

＜その他のヘッダ： データ型の定義＞ dual_list.h 双方向リンクリスト 等高線のアルゴリズム実現の為のデータ型 eye.h 視線ベクトルの設定用のデータ型 grid_line.h ピックスマップに表示する為のグリッドの設定情報 window_attr.h ウィンドウの設定用 pixmap_attr.h ウィンドウに表示するピックスマップのデータ型 (親と子の階層構造を持つ) 親はウィンドウに直接マップする親は子を複数持ち、レイアウト決定機構 (未実装) によりレイアウトを行なう。polygon.h 頂点数 n のデータがらポリゴンを構成する。現時点の実装では、３頂点以上の平面の場合のねじれのチェックを行なっていない。また、複数のポリゴンからオブジェクトを構成する。

[2] http://www.mma.gr.jp/~ aoki/3Dlib.tar.gz に archive があるので、興味のある方は、どうぞ。(編)

vector_and_matrix.h ３次元の座標変換に関するデータ型 lib とりあえず、今できているライブラリ。だが、まだ動くように調整していないものもある。color.c 色に関するもの create_window.c ウィンドウを作成 data_read.c データをファイルから読み込む為のもの draw_in_grid.c グリッドに線を引く draw_point_to_point.c ２頂点のデータを与えて、線を引く global_var.c グローバル変数の実体の確保 perspective_and_view.c 透視変換とその結果をウィンドウ上に写像するためのもの print_token.c null で区切られた文字列の表示用ファイル入出力の際に、デバッグに用いる。read_eye.c ファイル入出力、視野ベクトルの設定値よみとり read_object.c オブジェクトの設定値読み取り read_pixmap.c ピックスマップの読み取り read_window.c ウィンドウの読み取り vector_and_matrix.c ３次元の回転変換、それに関する回転行列を扱ったもの

==========

2 ＜データの型＞

dual_list.h 双方向リンクリスト (等高線のアルゴリズム用)

typedef struct dual_node{

int flag; 頂点間に断面が存在するかどうかのフラグ double x, y, z; 頂点の座標値 struct dual_node *South, *West, *East, *North;

頂点の４方向の接続関係を示す

} Dual_Node, Dual_Node_ptr;

eye.h 視線ベクトル

typedef struct {

int total_num; 視線ベクトルの数 xyz view_p; 視点の座標値 xyz target_p; 目標点の座標値 double view_angle_x, view_angle_y; 視野角 double distance; スクリーンまでの距離 char *gc_vector_color_name; 視線ベクトルの色 GC gc_vector; 視線ベクトルの GC

} EYE, *EYE_ptr;

grid_line.h グリッド

typedef struct{

int grid_num; グリッドの数 int grid_offset; グリッドの間隔 GC gc_grid, gc_sub_grid, gc_border_x, gc_border_y; グリッドの GC char *dash_pattern; ダッシュパターン char *grid_color, *sub_grid_color; グリッドの色

} Grid_line;

pixmap_attr.h ピックスマップの設定用親と子の階層構造を持つ。親はウィンドウに直接マップする。複数の子は親にマップされるが、その際、レイアウト管理機構 (未実装) によりレイアウトが決められる。

typedef struct PIXMAP_ATTR{

int O_x, O_y; ピックスマップの原点 int width,height; 幅、高さ int MAX_width, MAX_height; 最大の幅、高さ int parent_num; 親の数 int child_num; 子の数 Pixmap pix; ピックスマップの実体 struct PIXMAP_ATTR *parent; 親へのポインタ struct PIXMAP_ATTR *child; 子へのポインタ char *foreground_color, *background_color; フォアグラウンド、バックグラウンドの色

} Pixmap_attr;

polygon.h ポリゴン、ポリゴンからなるオブジェクト

typedef struct { ポリゴン int point_num; 頂点数 xyz_ptr p; 頂点の座標値 long color_pix; 色

} Polygon, * Polygon_ptr;

typedef struct OBJECT{ オブジェクト

int num; オブジェクトの数 Polygon *polygon; オブジェクトを構成するポリゴンへのポインタ struct OBJECT *child_obj; 階層構造用のポインタ xyz origin; オブジェクトを定義する為のモデリング座標系の原点

} Object, *Object_ptr;

vector_and_matrix.h ３次元の回転変換

#define AXIS_X 0 回転軸の回転順序を設定する為のマクロ#define AXIS_Y 1 #define AXIS_Z 2

typedef struct{ 頂点の座標値

double x, y, z;

} xyz, * xyz_ptr;

typedef struct{ X,Y,Z それぞれの回転角

double phiX, phiY, phiZ;

} Angle, * Angle_ptr;

typedef double Matrix[3][3]; 行列

window_attr.h ウィンドウ

typedef struct{

int O_x, O_y; ウィンドウの原点 int width,height; 幅、高さ int border_width; 枠の幅 int total_win; ウィンドウの総数 Window win_id; ウィンドウの識別子 char *border_color, *foreground_color, *background_color; フォアグラウンド、バックグラウンド、ボーダーの色 char *Event_string; イベントマスクの設定用

} Window_attr, *Window_attr_ptr;

PXEでFDDとオサラバしよう

飯村 卓司 (uirou)[1]

2000年11月20日

abstract

PXE[2] というものがあります。こいつは、CD-ROM Boot 等のような感じで、NIC が boot に必要なバイナリを適当に Network 越しに持って来て、OS を boot させたり、BIOS の update をさせたり、OS の install をさせたり、まぁ、とにかく、いろいろ出来ますよ、という物です。今までは、NIC に ROM を刺すためのソケットがついていて、そこに、適当な boot rom と呼ばれる boot に必要なバイナリの書き込まれた ROM を差し込んで、そこから boot させていました。これが、PXE に対応の NIC(と、BIOS?) を使うと、NIC が DHCP を喋り、TFTP Server 等を見つけ出し、自分から boot rom を拾って来て、boot します。要するに、boot rom をいちいちソケットに刺す必要は無く、適当に TFTP Server 側の設定を変えるだけで、boot する物自体を変えることが出来るわけです。これと、Wake on LAN の機能を組み合わせれば、夜中の間に会社の全てのマシンの BIOS を update するだとか、うまく作れば、OS の再インストールさえ自動化出来てしまうわけです。

まぁ、そういう PXE という物を使って、いろいろ遊んでみたので、その報告です。

PXEで遊ぶには

とりあえず PXE で遊ぶためには、以下の物が必要です。

- PXE 対応の NIC
- PXE で boot する client マシン
- DHCP Server となるマシン
- TFTP Server となるマシン

このうち、Server 側一台、client 側一台の合計二台は最低必要です。PXE client 側のマシンでは、PXE boot が行えるように設定しておいてください。だいたい、boot 時に、PXE 対応 NIC が「`Ctrl-S` を押す事によって設定できるよー」とか、そんな感じで言ってくれると思います。で、boot する時に、「DHCP で IP もらってまーす…… もらえませぇん (涙」とか、言ってくれていれば、動いていると思います。DHCP Server 側には NetBSD[3] を使用し、DHCP Server には ISC[4] の物を使いました。そこで、この文書ではこれを使った config とかを書く事にします。まぁ、他の DHCP Server でも、同じような設定を行えば問題無いと思います。TFTP Server はとりあえずどんなものでも動くようなので、最初から入っている物でも使えば良いでしょう。

DHCP Server の設定

ここから先は、ほとんどが Server 側の設定だけになります。まず、client 側の MAC Address を手に入れると良いでしょう。これは、`/var/db/dhcpd.leases` とかに残っていたりすると思いますし、`arp(1)` を使っても手に入れる事が出来ます。それから、client 側で OS が上がれば、それで知る事も出来ますね。そうしたら、次のようなブロックを `/etc/dhcpd.conf` に追加します。

```
host PXEclient
  hardware ethernet 00:d0:b7:83:53:b9;
  fixed-address pxeclient.localdomain.example;
  next-server tftp.localdomain.example;
```

[2] Preboot eXecution Environment
[3] http://www.netbsd.org/
[4] http://www.isc.org/

```
filename "pxeboot.rom";
option vendor-class-identifier "PXEClient";
option vendor-encapsulated-options 01:04:00:00:00:00:ff;
```

これを上から説明して行くと、

`host PXEclient {`
: とりあえず、client 用に一つのブロックを区切っています。まぁ、この下の MAC Address でくくりたいからという話です。

`hardware ethernet 00:d0:b7:83:53:b9;`
: 対象となる client をこの MAC Address のもの一つに絞っています。他の DHCP client に迷惑をかけたくないというだけです。

`fixed-address pxeclient.localdomain.example;`
: 毎回 DHCP で固定の IP address が割り当てられるように `fixed-address` を指定しています。

`next-server tftp.localdomain.example;`
: PXE が TFTP で boot ROM を取りに行く Server を指定しています。

`filename "pxeboot.rom";`
: PXE が TFTP で取るための boot ROM の名前を指定しています。実際に拾いに来る ROM の名前を指定して下さい。

`option vendor-class-identifier "PXEClient";`
: この `vendor-class-identifier` で、PXE に boot ROM を取って来るように指定します。PXE はこの field があることを確認して boot ROM を拾いに行きます。

`option vendor-encapsulated-options 01:04:00:00:00:00:ff;`
: PXE に対して何をするかのオプションを指定できるのですが、ここで指定しているのは何のオプションも無いという意味のオプションです。

と、いうことで、dhcpd を restart させてから、PXE client を上げてみましょう。多分、「TFTP で file が get 出来ませんでしたぁ。(涙」と、言うと思います。

TFTP Server の設定

ということで、TFTP Server 側に file を用意します。で、ここで用意する file が問題です。ここに用意する file は PXE に対応した boot ROM を用意してあげなければいけません。で、PXE に対応した boot ROM は、以下の物があるようです。

GNU GRUB
: GNU の手掛けている GRUB という boot selector です。かなり煮詰まった boot selector ですが、`pxegrub` という物が入っていて、これを使えば GRUB を PXE を用いて使う事が出来ます。boot 出来る OS は、Linux, NetBSD, OpenBSD, FreeBSD と、Windows や MS-DOS 等です。Windows 等はちょっと変則的な boot をしなければいけないんで、PXE で boot させるのは難しそうです。

BpBatch
: MS-DOS や Windows を remote boot させられる PXE 対応の boot selector です。これもかなり煮詰まった boot selector で、1024x768 の解像度で Window を開いて progress bar を表示出来たりします。というか、Windows のシステムをまるまる全部 TFTP 越しに持って来て boot なんていう荒技も出来るようです。

netboot

こいつも PXE に対応しているようです。boot 出来るのは、Linux,*BSD のようです。というか、GRUB はこいつの NetBoot の部分をもらって来ているようです。

FreeBSD の `loader(8)`

FreeBSD-4.*以降の boot loader である `loader(8)` は、PXE に対応しています。もちろん、FreeBSD kernel を TFTP で持って来て直接 boot させる能力があります。

SYSLinux

Linux kernel を NetBoot させるためのものですが、これの中にも PXELinux という PXE に対応した boot loader があります。これは、Linux kernel を TFTP で持って来て boot させる事が出来ます。

これのどれもが TFTP で kernel(または FloppyImage 等) を持って来て、それを使って boot します。今回は、NetBSD と MS-DOS を PXE 越しに boot させてみたかったので、FreeBSD の `loader(8)` や、PXELinux は使わずに、GNU GRUB と BpBatch を使ってみました。

GNU GRUB

GNU GRUB は、`http://www.gnu.org/software/grub/grub.ja.html` にあります。まだ、公式にリリースされてはいないようですが、テスト版が入手可能です。`pxegrub` は compile された状態の物には入ってないので、source を持って来て compile する必要があります。compile には GNU Binutils の 2.9.1.0.23 以降が必要となります。適当に入手して、compile して下さい。`stage2/pxegrub` が出来上がったら、それを使います。

BpBatch

BpBatch は `http://www.bpbatch.com/` にあります。これは、既に compile された状態で配布されているので、持って来て利用するだけです。PXE boot に使える file は、`bpbatch.P` です。

これらを、`/tftpboot` 等の TFTP のトップディレクトリに、さっき `dhcpd.conf` で指定した `pxeboot.rom` といった名前に変えて、置いておきます。

boot loader 個別の設定

GNU GRUB

こちらは、PXE 越しに config file を読ませるには、`dhcpd.conf` にもう少しオプションをつける必要があります。以下のオプションを `dhcpd.conf` に加えて下さい。

```
option-150 "hoge";
```

本当ならば、この `option-150` で指定した file を config file として読んでくれるのが正しそうな気がするのですが、どうも source に直に書かれてしまっているようで、`boot/grub/menu.lst` しか読み込んでくれません。また、この file を用意しない場合は、client 側にプロンプトが出るので、そこから適当なコマンドを打ち込んでも問題ありません。例えば、NetBSD を boot させたい場合、プロンプトでは、

```
grub> root (nd)
grub> kernel --type=netbsd /netbsd.INSTALL.gz
grub> boot
```

と、いった感じで、boot させる事になります。(この場合は、install に使う kernel を読み込ませています) ここで、menu を使った場合の config file の例を示しておきます。これは、grub のアーカイブの `doc/menu.lst` にあるものを簡略化し

たものなので、もっと詳しい説明は、grub のアーカイブに入っている `doc/menu.lst` か、info file を参照して下さい。後は、普通に TFTP に置いてある file を kernel として読み込ませれば問題ありません。

```
timeout 30
default 2
fallback 1

# For booting the NetBSD installer
title  NetBSD-install
root   (nd)
kernel --type=netbsd /netbsd.INSTALL.gz

# For booting the NetBSD
title  NetBSD netboot
root   (nd)
kernel --type=netbsd /netbsd.nfsroot.gz

# For booting Windows95
title Windows(or first HDD)
rootnoverify (hd0,0)
chainloader  +1
```

このような menu を置いておく事で、GRUB が boot selector としても機能します。また、netboot させて、NFSROOT 等を使って diskless な運用をしたいと思った場合は、BOOTP なり NFSROOT なりを使って、kernel に IP Address を指定したり、root filesystem をなんとかする必要がありますね。このあたりは、`diskless(8)` 等のマニュアルを読んで下さい。

BpBatch

BpBatch は、TFTP で load された後、default で同じ file 名の拡張子を変えた file を二つ拾いに来ます。今回の例のような `pxeboot.rom` という名前にした場合には、`pxeboot.ovl` と、`pxeboot.bpb` を読みに来ます。このうち、`.ovl` の方は、BpBatch の help 用の file ですので、`bpbatch.ovl` をそのまま copy したり、link したりして下さい。`.bpb` の方が config file ですので、これを編集する事により、動作を変える事が出来ます。この config file の書き方は、BpBatch の WebPage に詳しく解説してあるのですが、ここで一つ注意して欲しい所があります。example として書かれている、`SetPartitions` は、書かない方が良いです。と、いうのも、これは、NetBoot をした後に、簡単に OS の install が出来るようにするための、簡易 fdisk 機能として提供されているコマンドで、ここに書かれたように、HDD のパーティションテーブルを書き換えてしまいます。また、BpBatch は default で local についている HDD を cache として使用しようとして、パーティション的に余っている HDD を勝手に利用します。なので、`set CacheNever = "ON"`を、必ず入れておく事をお勧めします。で、この boot loader は、floppy image を load して boot する事が出来るので、MS-DOS の `fdisk.exe` や `format.exe` が使いたい場合に、便利です。floppy image は、`dd(1)` を使って作る事が出来ます。`rawread.exe` というものもあるようなので、Windows(や DOS) を使っている人も、問題ないでしょう。

と、いうことで、BpBatch を使って、DOS の便利コマンドが沢山詰まった FloppyImage[5] からの boot と、Memtest86[6] 、それから local の HDD からの boot を選択する事の出来る BpBatch 用の config file のサンプルを示しておきます。

```
ShowLog
```

[5] MS-DOS の floppy image は、~/.mtoolsrc に、drive x: file="/tmp/benri.img" とか書いておくと、% mdir x: とかすることによっていろいろいじれるので便利ですヨ。

[6] http://reality.sgi.com/cbrady/memtest86/

```
set CacheNever = "ON"

OnKey "m" goto MemTest86
OnKey "f" goto benriFD

Echo "(M) MemTest86"
Echo "(F) benriFD"
Echo "HIT ANY KEY!"
Beep

WaitForKey 10 goto HDDBoot
goto HDDBoot

:MemTest86
GetPatitions 2
GetBootPart 2
LoadRamDisk "memtest.bin"
FloppyBoot

:benriFD
GetPatitions 2
GetBootPart 2
LoadRamDisk "benri.img"
FloppyBoot

:HDDBoot
HdBoot
```

簡単に説明しますと、`Echo` コマンドで画面に「何を押したら何が起こるよー」と表示しておき、`OnKey` コマンドでそれを受けて、適当に boot 用の file を `LoadRamDisk` で読み込み、boot させていて、`WaitForKey` で 10 秒待ち、何か `OnKey` で設定されていないキーが押されるか、押されなかった時には `HdBoot` で local の HDD から boot させています。

実は、BpBatch のマニュアルを真面目に読んでいないので、もっと美しく選択メニューが Window で開けたりするのでしょうけれど、僕的に実用に耐え得る config が書けてしまったのでこの程度で勘弁して下さい。BpBatch の WebPage には、GIF file が load 出来たり、画面の解像度を 1024x768 とかに変えて、Window を開いたりするとか、いろいろ楽しそうな事が書いてあります。いろいろ遊んでみると面白いかと思います。

FDD が無い？大丈夫。僕等には PXE がある

と、いうことで、GNU GRUB や BpBatch を使えば、*BSD だろうが、Linux だろうが、MS-DOS だろうが、Windows だろうが、何でも PXE 越しに boot させる事が出来ます。と、いうことで、全国の自宅 server を持っている皆さん、もう FDD は無くてもあんまり問題ではありません。もちろん、PXE に対応している note パソコンであれば、FDD が最初からついていない企業モデルでも安心して買う事が出来ます。さぁ、あなたも PXE で FDD レスの楽しい生活をしましょう！

おまけ。Windows2000 による PXE を使った Windows の install

どうも、Windows2000 には、最初から「リモートインストールサービス」という名前で、PXE を使った Windows のリ

モートインストールをする機能があるようです。僕は全く試していないのでさっぱりわからないのですが、Windows2000をサービスパックが当たった状態でインストールする事も出来るようです。Windows2000ユーザのあなたは、試してみるのも面白いかと思います。

FreeBSDによる、IPv6クライアント構築

櫻井 謙司 (sakura)[1]

2000年11月

1　はじめに

初めは、ｖ６ルーターを構築する予定でしたが、ＰＣＩバスが一つしかなく、ＩＳＡのＮＩＣを使うのは嫌だったので、ＮＩＣを２枚挿すのをあきらめ、クライアントを構築することにしました。

構築環境　本体：　ＩＢＭ　ＰＣ　７２０－Ｐ１００ＨＤＤ：　８５０Ｍｂｙｔｅｓ　（ＩＤＥ）ＣＰＵ：　Ｐｅｎｔｉｕｍ　１００ＭＨｚ　ＯＳ：　ＦｒｅｅＢＳＤ　３．５．１－ＲＥＬＥＡＳＥ　ＮＩＣ：　3Com 3c905-TX Fast Etherlink XL　ＭＭＡのプライベートネットワーク（ＬＡＮ）に接続されている。

以上のような環境で、ＫＡＭＥを使ってカーネルをｖ６化しました。　これから、具体的な方法を述べたいと思います。

2

http://www.kame.net から、ＦｒｅｅＢＳＤ用のＫＡＭＥのスナップショットをダウンロードします。

3

適当なディレクトリ（/usr/tmp 等）に tar を使って展開します。

4

もしすでに kame が入っていたら少なくとも /usr/local/v6/lib は消し　　てください。。

5

ちゃんとファイルが揃っているか確認。

6

freebsd3 に入れるので kame/ で % make TARGET=freebsd3 prepare とします。

7

カーネルと /usr/include 以下が書き換えられるのでバックアップしてください。

8

カーネルをコンパイルします。 kame/freebsd3/sys/i386/conf で、カーネル　　設定ファイルを書きます。よくわからない人は GENERIC.v6 をコピーしてその　　まま使ってください。

9

hoge.v6 というファイルに書いたとしたら% config hoge.v6 とします。 kame/freebsd3/sys/compile/hoge.v6 というディレクトリへ移動して % make depend % make % make instal これで再起動すると v6 対応のカーネルで動作します。

10

kame/freebsd3 に居る事を確認してください。% make includes % make install-includes % make % make install 再起動

11

設定ファイル /usr/local/v6/etc/rc.net6.sample を rc.net6 とコピーして編集 してください。# manual configurations - in case ip6router=NO # you can configure only single interface, as specification assumes that # autoconfigured host has single interface only. # iface="ed0" というところでインターフェースの名前を指定します。ne0 だったら #iface="ed0" iface="ne0" とします。 /etc/rc.local を書き換えて、起動時に /usr/local/v6/etc/rc.net6 を実行する ようにします。

12

PATH に /usr/local/v6/bin を /bin /sbin /usr/bin /usr/sbin よりも 先に指定してください。

13 まとめ

以上で、ｖ６化の完了です。本当にｖ６化できているのか確かめてみましょう、ｐｉｎｇ６ （ｖ６対応の）ホスト名と入力して応答があれば成功です。 あとは、ｖ６対応のＯｐｅｎｓｓｈや、ｗ３ｍなどを入れて楽しみましょう。

くるしまぎれ

佐野 浩司 (sano)[1]

2000 年 11 月

今年入った 1 年生の sano です。
ネットワークなどについてはまったくの素人ですが、これから勉強していきます。
この写真のブツは秋葉の路上で一個３５００円で売ってました。
タイトル：有機コンピュータってこんな感じ?