インターフェイスの街角 (21)
OpenGL によるクロス・プラットホームの GUI 開発

増井俊之

最近は、UNIX システムと Windows や Macintosh とを状況に応じて使い分ける機会が増えてきたように思います。会社では UNIX ワークステーションを使い、社外では Windows のノート PC を使っている人も多いのではないでしょうか。私の場合、会社では SGI (Silicon Graphics Inc.) のワークステーションを使っていますが、社外でのデモなどでは Windows の PC しか利用できないことがよくあります。自宅にワークステーションを置いているわけではないので、新しい GUI を開発しようとする場合など、UNIX や Windows などで共通に使える GUI ツールが欲しいといつも思っています。

さまざまなプラットホームで共通に使えるツールというとすぐに Java が思い浮かびますが、私自身はこのところ OpenGL を好んで使っています。OpenGL は、もともと SGI が開発した 3 次元表示/操作ライブラリ GL を複数のプラットホームで使えるように拡張したものです。現在は、さまざまなシステム上で実装され、メニューやスライダなどの GUI をサポートするツールキットもいろいろ開発されています。

今回は、異なる環境で共通に使える OpenGL にもとづくインターフェイス開発環境を紹介します。

GL と OpenGL

SGI は、3 次元グラフィックス・ワークステーションの代表的なメーカーです。GL は SGI のシステムで共通に使われる 3 次元表示ライブラリで、図形描画やシェーディング、透視変換、隠面消去、入力/ウィンドウ処理など、3 次元処理を扱うアプリケーションの作成に必要な機能をすべて含んでいます。

リスト 1　GL によるプログラムの例

```
/* GL sample program */
#include <gl/gl.h>

main()
{
    prefposition(0,200,0,100);
    winopen("GL Example");
    RGBmode();
    gconfig();
    RGBcolor(0,0,100);
    clear();
    cmov2i(40,45);
    RGBcolor(255,255,255);
    charstr("Hello, World!");
    sleep(10);
}
```

実行結果

GL を用いた簡単なプログラムの例と実行結果をリスト 1 に示します。prefposition() で場所と大きさを指定した新しいウィンドウを winopen() で作成し、charstr() でテキストを描画しています。

GL は、もともとは SGI のワークステーション上のウィンドウ・システムで動くライブラリでした。その後、X ウィンドウ・システムと統合され、通常の X アプリケーションと一緒に使えるように拡張されました。現在は、GL ライブラリからウィンドウや入力処理など処理系に依存するものを取り除き、各種のシステムで共通して使えるライブ

リスト 2　正 8 面体を描く OpenGL プログラム

```
// octahedron.c
// OpenGLによる正8面体描画

#include <GL/glut.h>

void ohinit(void) // 初期化
{
  glClearColor(0.0,0.0,0.4,0.0);
  glFrontFace(GL_CW);
  glEnable(GL_CULL_FACE);
  glCullFace(GL_BACK);
}

void ohreshape(int w,int h)  // 再描画時
{
  glViewport(0,0,(GLsizei)w,(GLsizei)h);
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  gluOrtho2D(-1.2,1.2,-1.2,1.2);
  glMatrixMode(GL_MODELVIEW);
}
```

```
void ohdraw(void) // 正8面体描画
{
  glBegin(GL_TRIANGLE_FAN);
  glColor3f(0.0,0.0,1.0); glVertex3f(0.0,0.0,1.0);
  glColor3f(1.0,0.0,0.0); glVertex3f(1.0,0.0,0.0);
  glColor3f(0.0,1.0,0.0); glVertex3f(0.0,1.0,0.0);
  glColor3f(0.0,1.0,1.0); glVertex3f(-1.0,0.0,0.0);
  glColor3f(1.0,0.0,1.0); glVertex3f(0.0,-1.0,0.0);
  glColor3f(1.0,0.0,0.0); glVertex3f(1.0,0.0,0.0);
  glEnd();

  glBegin(GL_TRIANGLE_FAN);
  glColor3f(1.0,1.0,0.0); glVertex3f(0.0,0.0,-1.0);
  glColor3f(1.0,0.0,0.0); glVertex3f(1.0,0.0,0.0);
  glColor3f(1.0,0.0,1.0); glVertex3f(0.0,-1.0,0.0);
  glColor3f(0.0,1.0,1.0); glVertex3f(-1.0,0.0,0.0);
  glColor3f(0.0,1.0,0.0); glVertex3f(0.0,1.0,0.0);
  glColor3f(1.0,0.0,0.0); glVertex3f(1.0,0.0,0.0);
  glEnd();
}
```

ラリを規格化した OpenGL が 3 次元表示ライブラリの標準として普及しています。OpenGL は、Windows 95 や Windows NT では Microsoft の提供するライブラリによって利用できますし、X 上でもフリーな互換ライブラリ MESA[1]を介して使えます。

リスト 2 は、正 8 面体を描く OpenGL プログラムです。 `gl″で始まるのが OpenGL の基本関数、 `glu″で始まるのが拡張関数です。GL では、ウィンドウ操作関数や描画関数などがすべて同様に扱われていました。一方、OpenGL で定義されている関数は描画部分だけで、ウィンドウを開いたりユーザーの操作を検知したりといったシステムに依存する部分はシステムごとに異なります。

GLUT

複数のプラットホームで共通して使えるようにするため、OpenGL の標準化は描画ライブラリに焦点が絞られています。たとえばリスト 1 にある winopen() のような関数は OpenGL では定義されていません 。このためウィンドウの操作や入力イベント処理などの機能については、ウィンドウ・システムごとに異なる関数を呼ぶ必要があります。しかし、これでは不便なので、各種システムで共通に使えるウィンドウ/イベント操作ライブラリ GLUT[2]が開発されました。OpenGL と GLUT を併用すれば、SGI のシステムだけでなく、X が利用できる各種 UNIX システムや Windows などでもまったく同じソースコードを利用して開発を進めることが可能になります。

リスト 3 は、回転する正 8 面体を表示する OpenGL/GLUT プログラムの例です。ウィンドウ・システムの画面に表示したり、ユーザーからの操作を可能にするには、ウィンドウを開いたり入力イベントを取得する必要があります。そこで、リスト 2 で使っている正 8 面体を表示するための関数のほかに、GLUT のライブラリを用いて実現しています。 `gl″で始まるのが OpenGL の関数、 `glut″で始まるのが GLUT の関数です。

GLUT のメインルーチンは glutMainLoop() で、ウィンドウの大きさが変わったりキーボード入力があったときに呼ばれるコールバック関数が glutReshapeFunc() や glutKeyboardFunc() などで定義されています。

GLUT のおもな関数を表 1 に示します。これらの関数のうち、glut...Func() は、イベントが発生したときに呼ばれるコールバック関数を定義するもので、最後に呼ばれる GLUT のメインルーチン glutMainLoop() から呼び出されます。

リスト 3 のプログラムは、SGI のワークステーション、FreeBSD などの UNIX システムと MESA との組合せ、Windows 95/NT のいずれでも、ソースをまったく変更せずにコンパイル、実行できます。

1 FreeBSD や Linux ではパッケージとして配布されていて、インストールもごく簡単です (http://www.mesa3d.org/)。

2 http://reality.sgi.com/mjk_asd/glut3/glut3.html

リスト 3　回転する正 8 面体を描く GLUT プログラム

```
// glut.c
// 回転する正8面体をウィンドウに描くGLUTプログラム

#include <GL/glut.h>

void display(void) // 回転させながら表示
{
    static GLfloat rot = 0.0;
    glClear(GL_COLOR_BUFFER_BIT);
    glPushMatrix();
    glRotatef(rot, 1.0, 1.0, 1.0);
    rot += 4.0;
    ohdraw();
    glPopMatrix();
    glFlush();
    glutSwapBuffers();
}

void reshape(int w, int h)
{
    ohreshape(w,h);
}

void keyboard(unsigned char key, int x, int y)
{
    if(key == 0x1b) exit(0);
}

void idle(void)
{
    glutPostRedisplay();
}

int main(int argc, char** argv)
{
    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
    glutInitWindowSize(300,300);
    glutInitWindowPosition(100,100);
    glutCreateWindow(argv[0]);
    ohinit();
    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
    glutKeyboardFunc(keyboard);
    glutIdleFunc(idle);
    glutMainLoop();
    return 0;
}
```

表 1　GLUT のおもな関数

glutInit()	初期化
glutInitWindowSize()	ウィンドウの大きさの設定
glutInitWindowPosition()	ウィンドウの位置設定
glutInitDisplayMode()	表示モード（ダブルバッファを使うか否かなど）の設定
glutCreateWindow()	ウィンドウの生成
glutDisplayFunc()	表示用コールバック関数の定義
glutReshapeFunc()	ウィンドウサイズ変更時のコールバック関数の定義
glutKeyboardFunc()	キーボード入力用コールバック関数の定義
glutIdleFunc()	無入力時のコールバック関数の定義
glutMainLoop()	メインループ
glutSwapBuffers()	ダブルバッファの切替え
glutPostRedisplay()	表示予約

さきほど説明したように、OpenGL はもともとは 3 次元表示のためのライブラリですが、2 次元表示しか扱わないプログラムでも、共通のソースコードを使えるという特徴を活かしてクロス・プラットホームでの GUI 開発に利用できます。

各種システムでのコンパイル例

SGI のシステムの場合

GL ライブラリは SGI のワークステーションには標準で付属しているので、GLUT ライブラリを追加インストールすればリスト 3 のプログラムを動かすことができます。GLUT は、前述の URL から入手できます。

```
# Makefile for SGI / X11
CFLAGS = -Xcpluscomm
LIBS = -lglut -lGLU -lGL -lXmu -lXi -lX11

glut: glut.o
        $(CC) $(CFLAGS) -o $@ $@.o $(LIBS)
```

FreeBSD/X11 と MESA を使う場合

FreeBSD などで OpenGL/GLUT のプログラムを開発する場合は、GLUT ライブラリのほかに MESA など

図 1　OpenGL/GLUT による 8 面体回転画面 (FreeBSD)

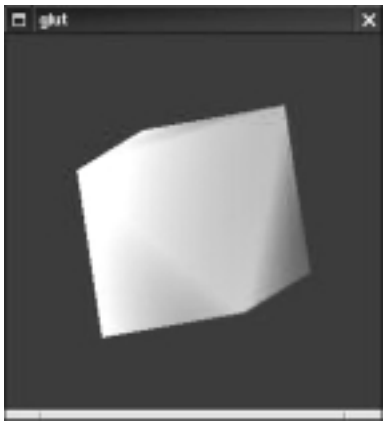
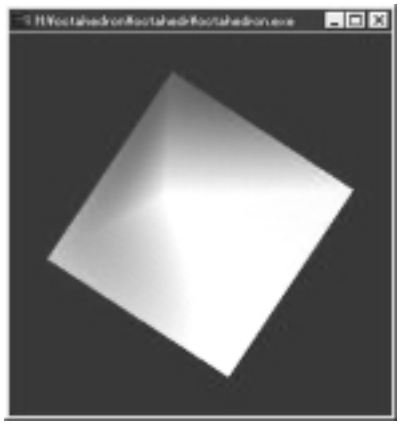


図 2　OpenGL/GLUT による 8 面体回転画面 (Windows)



の OpenGL 互換ライブラリを利用する必要があります。FreeBSD では MESA も GLUT もパッケージになっているため、sysinstall コマンドで簡単にインストールできます。最新版も前述の URL で手に入ります。

```
# Makefile for MESA / X11
CFLAGS = -I/usr/X11R6/include -L/usr/X11R6/lib
LIBS = -lGL -lglut -lGLU -lX11 -lXext -lXi \
       -lXmu -lm

glut: glut.o
        $(CC) $(CFLAGS) -o $@ $@.o $(LIBS)
```

Windows 95/NT の場合

Windows 95/98/NT には、Microsoft が提供する OpenGL のランタイム・ライブラリ opengl32.dll と拡張ライブラリ glu32.dll が最初から入っているので、そのまま OpenGL アプリケーションを動かすことができます。OpenGL 対応のグラフィックス・アクセラレータのなかには独自の opengl32.dll をもつものがあり、それを使えば OpenGL の実行は格段に高速化されます。

Visual C++などで OpenGL のプログラム開発をおこなう場合は、これらに加えてライブラリ定義ファイル opengl32.lib と glu32.lib、およびヘッダファイルが必要になります。これらは、Microsoft の FTP サーバー[3]からダウンロードできます。Windows 用の GLUT は、Nate Robins 氏の Web ページ[4]から入手できます。

3 ftp://ftp.microsoft.com/Softlib/MSLFILES/Opengl95.exe

4 http://www.xmission.com/~nate/glut.html

まず、これらのファイルを Visual C++のライブラリ・ディレクトリ（C:¥Program Files¥DevStudio¥VC¥lib など）とヘッダファイル用ディレクトリ（C:¥Program Files¥DevStudio¥VC¥include¥gl など）に格納します。そして、Visual C++のプロジェクト設定パネルで opengl32.dll、glu32.dll、glut32.dll を追加すれば、リスト 3 のようなアプリケーションが作成できます。図 2 は、Windows 95 上でリスト 3 のプログラムを動かしたところです。

OpenGL 対応のツールキット

GLUT を使えば、とりあえず OpenGL 上でウィンドウやマウスなどの処理ができるようになります。しかし、GLUT にはメニューやスライダなどの GUI 部品は用意されていないので、これらの高度な GUI 部品を利用するには不十分です。

X ウィンドウ・システムの開発がさかんだったころは UNIX 上の GUI ツールの開発も活発で、各種のツールキットが提唱されていました。その後、これらのシステムの開発は一時下火になっていました。しかし、最近のフリー UNIX の流行にともなって数多くの GUI 開発ツールが発表され、OpenGL が使えるもの、あるいは Windows でも利用できるものなどが出てきました。各種のツールキットについては、Li-Cheng (Andy) Tai 氏のページ[5]に詳しい一覧があります。このうち、OpenGL が使えるもの

5 http://home.pacbell.net/atai/guitool/

図 3　MUI の部品の例

をいくつか紹介しましょう。

MUI

MUI[6]は、GLUT のみで作られたツールキットです。MUI では、GLUT で使われる低レベルのイベントやコールバック関数の代わりにボタンやスライダなどの GUI 部品を利用することができます（図 3）。

MUI は OpenGL と GLUT だけで構築されているため、これらが動作する環境であれば手軽に使えるという利点はあります。しかし、機能や見栄えの点ではほかのツールキットにはまだ及ばないようです。

Forms/XForms

オランダの Mark Overmars 氏は、かつて SGI の GL 上で動く Forms というシンプルで便利なツールキットを配布していました。Forms は OpenGL に対応しておらず、現在はサポートされていません。これに代わり、T.C.Zhao 氏が Forms を拡張して普通の X で使えるようにした XForms[7]というライブラリを公開しています。Forms/XForms は独自の描画プリミティブをもっていますが、OpenGL との併用も可能です。XForms はその名が表すとおり、UNIX の X 上で動作します。

XForms を用いて正 8 面体を回転表示するアプリケーションのプログラム例とその表示結果をリスト 4 に示します。このプログラムは、4 個のスライダを生成してウィンドウ上に配置し、8 面体の大きさと回転を制御できるようにするものです。

オリジナルの Forms ライブラリは、どちらかといえば GL 上でボタンなどの GUI をとりあえず使えるようにするといった簡単なものでした。これに対し、XForms は機能がかなり拡張され、関数の数などが増えています。

FLTK

Forms/XForms とは別に、Bill Spitzak 氏が Forms に触発されて開発した C++ベースの FLTK[8]というライブラリもあります。FLTK は UNIX だけでなく Windows にも対応し、GLUT や Forms との互換性も考慮されています。

FLTK を用いて正 8 面体を回転表示するアプリケーションのプログラムと、その表示結果を末尾のリスト 5 に示します。

Qt

Qt[9]はノルウェーの Troll Tech が開発した製品版の GUI ツールキットで、さきほど紹介した Li-Cheng (Andy) Tai 氏のページでは最良のツールキットとして紹介されています。Qt も FLTK と同様に C++をベースに開発され、UNIX と Windows に対応しており、拡張機能として OpenGL も利用できます。今回紹介したほかのシステムと異なり、Qt は有償の製品ですが、UNIX/X11 用にはフリーのバージョンもあります。図 4 は、OpenGL で描いた直方体の回転を Qt のスライダ部品を用いて制御している様子です。

商品として市販されていることもあり、Qt はほかのツールキットとくらべて機能や見栄えが優れています。たとえば、図 5 のような Windows に似たファイル選択ダイアログ画面も使えます。

6 http://reality.sgi.com/mjk_asd/tips/mui/mui.html
7 http://bragg.phys.uwm.edu/xforms/
8 http://www.fltk.org/
9 http://www.troll.no/products/qt.html

リスト 4　正 8 面体を回転する XForms プログラムとその表示

```
#include <stdio.h>
#include "GL/gl.h"
#include "forms.h"

void ohinit(void);
void ohreshape(int w, int h);
void ohdraw(void);

FL_FORM *form;
FL_OBJECT *size, *rx, *ry, *rz;
FL_OBJECT *glcanvas;

void
slider_cb(FL_OBJECT *ob, long data)
{
    float scale;

    fl_activate_glcanvas(glcanvas);

    ohinit();
    ohreshape(300,300);

    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glPushMatrix();
    glRotatef(fl_get_slider_value(rx),1,0,0);
    glRotatef(fl_get_slider_value(ry),0,1,0);
    glRotatef(fl_get_slider_value(rz),0,0,1);
    scale = fl_get_slider_value(size);
    glScalef(scale,scale,scale);
    ohdraw();
    glPopMatrix();
    glXSwapBuffers(fl_display, fl_get_canvas_id(glcanvas));
}

int
main(int argc, char *argv[])
{
    fl_initialize(&argc, argv, "FormDemo", 0, 0);
    form = fl_bgn_form(FL_UP_BOX,500,340);
        size = fl_add_slider(FL_VERT_SLIDER,340,20,30,300,"");
        fl_set_slider_bounds(size,1,0.01);
        fl_set_slider_value(size,1);
        fl_set_object_callback(size,slider_cb,NULL);
        rx = fl_add_slider(FL_VERT_SLIDER,380,20,30,300,"");
        fl_set_slider_bounds(rx,360,0);
        fl_set_slider_value(rx,0);
        fl_set_object_callback(rx,slider_cb,NULL);
        ry = fl_add_slider(FL_VERT_SLIDER,420,20,30,300,"");
        fl_set_slider_bounds(ry,360,0);
        fl_set_slider_value(ry,0);
        fl_set_object_callback(ry,slider_cb,NULL);
        rz = fl_add_slider(FL_VERT_SLIDER,460,20,30,300,"");
        fl_set_slider_bounds(rz,360,0);
        fl_set_slider_value(rz,0);
        fl_set_object_callback(rz,slider_cb,NULL);
        glcanvas = fl_add_glcanvas(FL_NORMAL_CANVAS,20,20,300,300,"");
    fl_end_form();
    fl_show_form(form,FL_PLACE_CENTER,FL_NOBORDER,"Slider");
    slider_cb(NULL,NULL);
    fl_do_forms();
    return 0;
}
```

表示結果

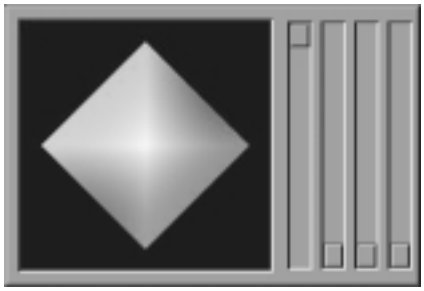
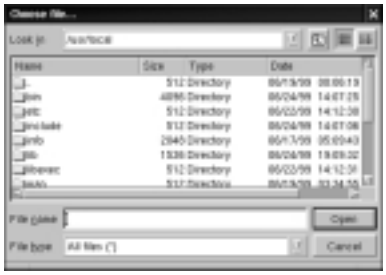


図 4　Qt による OpenGL 制御

図 5　Qt によるファイル選択画面



おわりに

今回は、マルチ・プラットホーム GUI の開発という観点から OpenGL を紹介しました。従来、GL/OpenGL は SGI のマシン上でしか使えなかったため、GL/OpenGL にもとづくツールキットなどは Forms を除けばほとんどありませんでした。しかし、最近は X11 上の MESA や Windows の OpenGL サポートなどにより、OpenGL を用いたツールキットやアプリケーションが増えてきました。GLUT-3.7 では OpenGL でフルスクリーン描画もできるようになり、ウィンドウ・マネージャーから端末ウィンドウまで、あらゆるアプリケーションを OpenGL だけで実装することも不可能ではないかもしれません。

今回紹介したようなシステムでは、X11 や Windows の描画プリミティブの上に OpenGL を実装しているため、どうしても速度が問題になります。しかし、OpenGL のみを基本描画プリミティブとした十分に高速なシステムであれば、あらゆるインターフェイスを OpenGL にしても実用に耐えるのではないでしょうか。

X などのウィンドウ・システムが普及するまでは、グラフィックス・システムの上にウィンドウ・システムを作るか、それともウィンドウ・システムの上にグラフィックス・システムを作るかといった問題がよく論じられていました。コンピュータ・グラフィックスの研究者のあいだでは、どちらかというと前者の考え方が優勢でしたが、グラフィックス・システム上に実用に耐えるウィンドウ・システムを構築できないうちに、X のような特殊な描画プリミティブをもつシステムが普及してしまったのが実状です。現在は、安価な計算機でも十分に高速な表示が可能ですから、高機能な描画システムにもとづくウィンドウ・システムをもう一度考えてもよいのではないでしょうか。

OpenGL の詳細と動向は、http://www.opengl.org/ や http://tech.webcity.ne.jp/~andoh/opengl/opengl-faq.html を参照してください。

（ますい・としゆき　ソニー CSL）

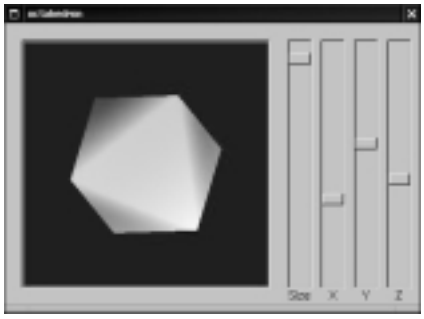
リスト 5　正 8 面体を回転する FLTK プログラムとその表示

```
#include <FL/Fl.H>
#include <FL/Fl_Window.H>
#include <FL/Fl_Box.H>
#include <FL/Fl_Slider.H>
#include <FL/Fl_Gl_Window.H>
#include <FL/gl.h>

void ohinit(void);
void ohreshape(int w, int h);
void ohdraw(void);

class oh_window : public Fl_Gl_Window {
    void draw();
public:
    oh_window(int x,int y,int w,int h,const char *l=0)
```

表示結果



```
        : Fl_Gl_Window(x,y,w,h,l) { }
};

Fl_Window *form;
Fl_Slider *rx, *ry, *rz, *hsize;
oh_window *oh;

void oh_window::draw() {
    if (!valid()) {
        ohinit();
        ohreshape(w(),h());
    }
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glPushMatrix();
    glRotatef(rx->value(),1,0,0);
    glRotatef(ry->value(),0,1,0);
    glRotatef(rz->value(),0,0,1);
    glScalef(float(hsize->value()),float(hsize->value()),
        float(hsize->value()));
    ohdraw();
    glPopMatrix();
}
void callback(Fl_Widget*, void* v) {
    oh->redraw();
}

void makeform(const char *name) {
    form = new Fl_Window(500,340,name);
    new Fl_Box(FL_DOWN_FRAME,20,20,300,300,"");
    hsize = new Fl_Slider(FL_VERT_SLIDER,340,20,30,300,"Size");
    rx = new Fl_Slider(FL_VERT_SLIDER,380,20,30,300,"X");
    ry = new Fl_Slider(FL_VERT_SLIDER,420,20,30,300,"Y");
    rz = new Fl_Slider(FL_VERT_SLIDER,460,20,30,300,"Z");
    rx->callback(callback);
    ry->callback(callback);
    rz->callback(callback);
    hsize->callback(callback);
    oh = new oh_window(23,23,294,294, 0);

    Fl_Box *b = new Fl_Box(FL_NO_BOX,oh->x(),hsize->y(),
        oh->w(),hsize->h(),0);
    form->resizable(b);
    b->hide();
    form->end();
}

main(int argc, char **argv) {
    makeform(argv[0]);
    hsize->bounds(1,0.01);
    hsize->value(1.0);
    rx->bounds(360,0);
    ry->bounds(360,0);
    rz->bounds(360,0);
    form->label("octahedron");
    form->show(argc,argv);
    oh->show();
    Fl::run();
    return 0;
}
```