109(上)電腦圖學　作業二

3D 模型預覽

資工 4 甲

406261688

陳君平

程式架構 ：

- MAIN :
  - 模型顯示 glutDisplayFunc(display)
    - 判斷 material_mode=none/color/material
      - 判斷 facet_normal=wireframe/filled
    - 模型外接方塊(bounding_box)
    - 模型基本資料
      - Vertices,Triangles,Normal,Texcoords,Groups,Materials
    - 顯示 FPS
  - 按鍵觸發 glutKeyboardFunc(keyboard)
    - w - wireframe (線框) / filled (預設:填充)
    - c - culling on / off (可看到 3D Models 內部)
    - n - face/smooth (凹凸面/平滑面)
    - b - bounding box on/off (3D Models 外接透明方塊)
    - r - Reverse polygon winding (只看到 3D Models 的內部)
    - m - Toggle color/material/none mode (顏色/

材質/無)

- p - frame rate on/off (顯示 FPS 幀數)
- s/S - Scale model smaller/larger
- t - Toggle model statistics (3D Models 的訊息)
- +/- - Increase/decrease smoothing angle(平滑程度調整)
- W - Write model to file (out.obj)(匯出 3D Models)
- q/escape - [Esc] Quit

滑鼠控制 glutMouseFunc(mouse)

- 拖移 3D Models(直接按滑鼠中鍵 or shift+滑鼠左鍵)
  - GLUT_LEFT_BUTTON = 滑鼠左鍵
  - GLUT_ACTIVE_SHIFT = 鍵盤 shift
    - GLUT_LEFT_BUTTON+ GLUT_ACTIVE_SHIFT= GLUT_MIDDLE_BUTTON 滑鼠中鍵
- 轉動 3D Models (gltbMouse(button, state, x, y))

選單 glutCreateMenu(menu)

- Menu function
  - 選擇 3D Models

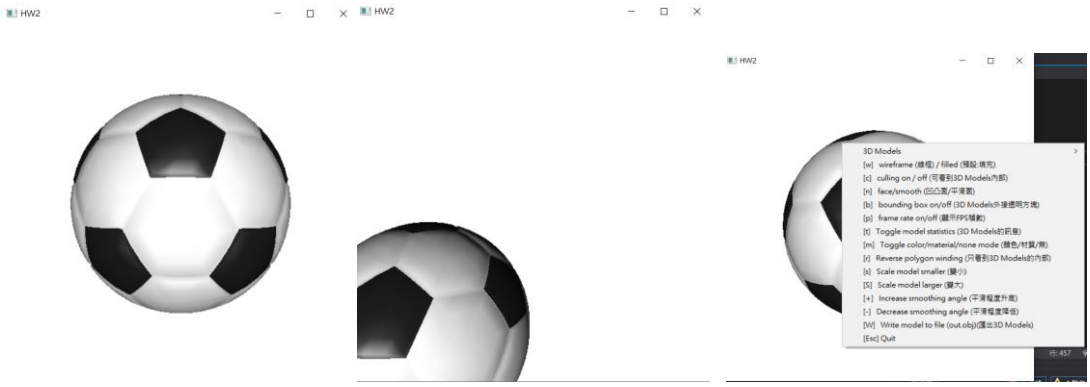
- 拿出該 Model 的基本資料
  - Vertices,Triangles,Normal,Texcoords,Groups,Materials
- (nummaterials>0)? Material: none

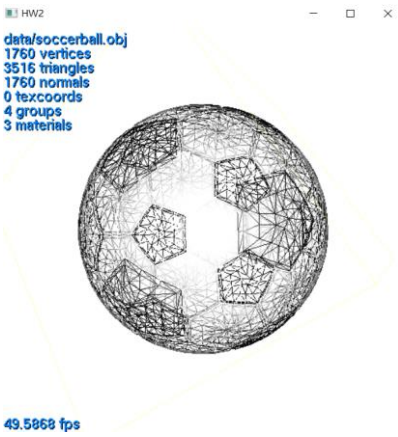
◆ 其餘選單同按鍵觸發 glutKeyboardFunc(keyboard)

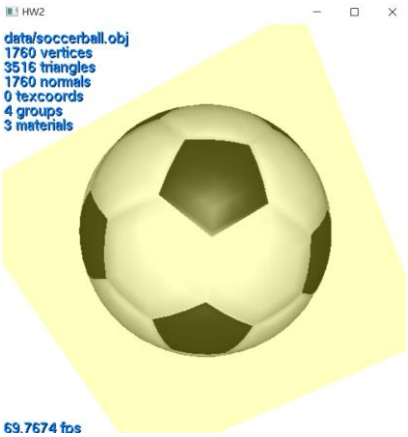
討論 ：

在本次作業中了解 3DModels 的預覽功能，從如何載入 obj 檔到每個按鍵功能都詳細了解，其中我處理最久的是自行旋轉問題，glRotatef()是使 model 轉動，分別為四個參數，第一個為轉動角度，二三四為 x,y,z 法向量，但加在 display 上程式執行只有第一次會轉動，之後需要做按鍵或是滑鼠點擊才會繼續轉動，而我又發現課本例子卻能自行轉動，因此目前雖然一開始無法自行旋轉，但經過滑鼠拖拉後能夠自行旋轉。

執行畫面 ：

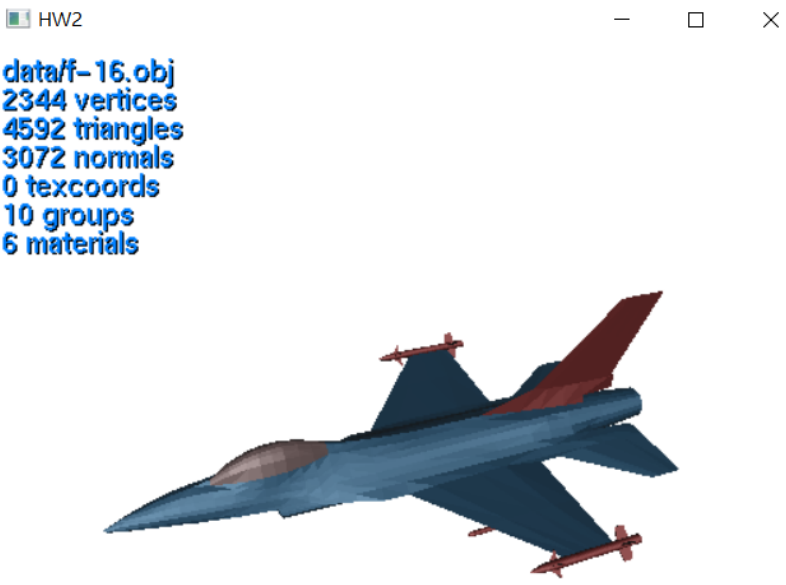



HW2
data/soccerball.obj
1760 vertices
3516 triangles
1760 normals
0 texcoords
4 groups
3 materials
49.5868 fps


HW2
data/soccerball.obj
1760 vertices
3516 triangles
1760 normals
0 texcoords
4 groups
3 materials
69.7674 fps


HW2
data/soccerball.obj
1760 vertices
3516 triangles
1760 normals
0 texcoords
4 groups
3 materials
10.8303 fps


HW2
data/dolphins.obj
855 vertices
1692 triangles
1202 normals
0 texcoords
4 groups
4 materials
60 fps


HW2
data/f-16.obj
2344 vertices
4592 triangles
3072 normals
0 texcoords
10 groups
6 materials
58.2524 fps

程式碼 ：

```
#include <math.h>

#include <stdio.h>

#include <stdlib.h>

#include <assert.h>

#include <stdarg.h>

#include <sys/timeb.h>

#include <GL/glut.h>

#include "gltb.h"

#include "glm.h"

#include "dirent32.h"


#pragma comment( linker, "/entry:\"mainCRTStartup\"" )    //

set the entry point to be main()


#define DATA_DIR "data/"

#define CLK_TCK 1000


char* model_file = NULL;        /* name of the obect file */
```

```
GLuint          model_list = 0;                 /* display list for object
*/
GLMmodel* model;                                /* glm model data
structure */
GLfloat         scale;                          /* original scale factor */
GLfloat         smoothing_angle = 90.0;/* smoothing angle */
GLfloat         weld_distance = 0.00001;     /* epsilon for welding
vertices */
GLboolean    facet_normal = GL_FALSE;   /* draw with facet
normal? */
GLboolean    bounding_box = GL_FALSE; /* bounding box on?
*/
GLboolean    performance = GL_FALSE;    /* performance
counter on? */
GLboolean    stats = GL_FALSE;          /* statistics on? */
GLuint          material_mode = 0;        /* 0=none, 1=color,
2=material */
GLint           entries = 0;                    /* entries in model menu
*/
```

```
GLdouble      pan_x = 0.0;
GLdouble      pan_y = 0.0;
GLdouble      pan_z = 0.0;



float elapsed(void)
{
        static long begin = 0;
        static long finish, difference;

        static struct timeb tb;
        ftime(&tb);
        finish = tb.time * 1000 + tb.millitm;

        difference = finish - begin;
        begin = finish;

        return (float)difference / (float)CLK_TCK;
```

```
}

void shadowtext(int x, int y, char* s)    //顯示文字
{
        int lines;
        char* p;

        glDisable(GL_DEPTH_TEST);
        glMatrixMode(GL_PROJECTION);
        glPushMatrix();
        glLoadIdentity();
        glOrtho(0, glutGet(GLUT_WINDOW_WIDTH),
                0, glutGet(GLUT_WINDOW_HEIGHT), -1, 1);
        glMatrixMode(GL_MODELVIEW);
        glPushMatrix();
        glLoadIdentity();
        glColor3ub(0, 0, 0);
        glRasterPos2i(x + 1, y - 1);
        for (p = s, lines = 0; *p; p++) {
```

```
            if (*p == '\n') {
                lines++;
                glRasterPos2i(x + 1, y - 1 - (lines * 18));
            }
            glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18,
*p);
        }
        glColor3ub(0, 128, 255);
        glRasterPos2i(x, y);
        for (p = s, lines = 0; *p; p++) {
            if (*p == '\n') {
                lines++;
                glRasterPos2i(x, y - (lines * 18));
            }
            glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18,
*p);
        }
        glMatrixMode(GL_PROJECTION);
        glPopMatrix();
```

```
	glMatrixMode(GL_MODELVIEW);
	glPopMatrix();
	glEnable(GL_DEPTH_TEST);
}

void lists(void)
{
	GLfloat ambient[] = { 0.2, 0.2, 0.2, 1.0 };
	GLfloat diffuse[] = { 0.8, 0.8, 0.8, 1.0 };
	GLfloat specular[] = { 0.0, 0.0, 0.0, 1.0 };
	GLfloat shininess = 65.0;

	glMaterialfv(GL_FRONT, GL_AMBIENT, ambient);
	glMaterialfv(GL_FRONT, GL_DIFFUSE, diffuse);
	glMaterialfv(GL_FRONT, GL_SPECULAR, specular);
	glMaterialf(GL_FRONT, GL_SHININESS, shininess);

	if (model_list)
		glDeleteLists(model_list, 1);
```

```
	/* generate a list */
	if (material_mode == 0) {	//material_mode = 0 = none
		if (facet_normal)	//判斷顯示平滑或凹凸
			model_list = glmList(model, GLM_FLAT);
		else
			model_list = glmList(model, GLM_SMOOTH);
	}
	else if (material_mode == 1) {	//material_mode = 1 =
color
		if (facet_normal)	//判斷顯示平滑或凹凸
			model_list = glmList(model, GLM_FLAT |
GLM_COLOR);
		else
			model_list = glmList(model, GLM_SMOOTH |
GLM_COLOR);
	}
	else if (material_mode == 2) {	//material_mode = 2 =
material
```

```
            if (facet_normal)      //判斷顯示平滑或凹凸
                model_list = glmList(model, GLM_FLAT |
GLM_MATERIAL);
            else
                model_list = glmList(model, GLM_SMOOTH |
GLM_MATERIAL);
        }
}

void init(void)
{
        gltbInit(GLUT_LEFT_BUTTON);

        /* read in the model */
        model = glmReadOBJ(model_file);
        scale = glmUnitize(model);
        glmFacetNormals(model);
        glmVertexNormals(model, smoothing_angle);
```

```
        if (model->nummaterials > 0)
                material_mode = 2;

        /* create new display lists */
        lists();

        glEnable(GL_LIGHTING);
        glEnable(GL_LIGHT0);
        glLightModeli(GL_LIGHT_MODEL_TWO_SIDE, GL_TRUE);

        glEnable(GL_DEPTH_TEST);

        glEnable(GL_CULL_FACE);
}

void reshape(int width, int height) //視窗初始設定
{
        gltbReshape(width, height);
```

```
    glViewport(0, 0, width, height);


    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(60.0, (GLfloat)height / (GLfloat)width, 1.0,
128.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glTranslatef(0.0, 0.0, -3.0);
}


#define NUM_FRAMES 5
void display(void)
{
    static char s[256], t[32];
    static char* p;
    static int frames = 0;


    glClearColor(1.0, 1.0, 1.0, 1.0);
```

```
        glClear(GL_COLOR_BUFFER_BIT |
GL_DEPTH_BUFFER_BIT);

        glPushMatrix();
        glTranslatef(pan_x, pan_y, 0.0);
        gltbMatrix();

#if 0      /* glmDraw() performance test */
        if (material_mode == 0) {      //material_mode = 0 = none
                if (facet_normal)      //判斷顯示平滑或凹凸
                        glmDraw(model, GLM_FLAT);
                else
                        glmDraw(model, GLM_SMOOTH);
        }
        else if (material_mode == 1) {    //material_mode = 1 =
color
                if (facet_normal)      //判斷顯示平滑或凹凸
                        glmDraw(model, GLM_FLAT | GLM_COLOR);
                else
```

```
                glmDraw(model, GLM_SMOOTH | GLM_COLOR);
        }
        else if (material_mode == 2) {    //material_mode = 2 =
material
                if (facet_normal)      //判斷顯示平滑或凹凸
                        glmDraw(model, GLM_FLAT | GLM_MATERIAL);
                else
                        glmDraw(model, GLM_SMOOTH |
GLM_MATERIAL);
        }
#else
        glCallList(model_list);
#endif


        glDisable(GL_LIGHTING);
        if (bounding_box) { //顯示模型外方塊
                glBlendFunc(GL_SRC_ALPHA,
GL_ONE_MINUS_SRC_ALPHA);
                glEnable(GL_BLEND);
```

```
            glEnable(GL_CULL_FACE);
            glColor4f(1.0, 1.0, 0.0, 0.25);
            glutSolidCube(2.0);
            glDisable(GL_BLEND);
        }

        glPopMatrix();

        if (stats) {        //模型基本資料
            /* XXX - this could be done a _whole lot_ faster... */
            int height = glutGet(GLUT_WINDOW_HEIGHT);
            glColor3ub(0, 0, 0);
            sprintf(s, "%s\n%d vertices\n%d triangles\n%d
normals\n"
                    "%d texcoords\n%d groups\n%d materials",
                    model->pathname, model->numvertices, model-
>numtriangles,
                    model->numnormals, model->numtexcoords,
model->numgroups,
```

```
                model->nummaterials);
        shadowtext(5, height - (5 + 18 * 1), s);
    }

    /*顯示幀數*/
    frames++;
    if (frames > NUM_FRAMES) {
        sprintf(t, "%g fps", frames / elapsed());
        frames = 0;
    }
    if (performance) {
        shadowtext(5, 5, t);
    }

    glutSwapBuffers();
    glEnable(GL_LIGHTING);
}

void keyboard(unsigned char key, int x, int y)
```

```
{
	GLint params[2];


	switch (key) {
	case 'h':
		printf("help\n\n");
		printf("w		-	Toggle wireframe/filled\n");
		printf("c		-	Toggle culling\n");
		printf("n		-	Toggle facet/smooth
normal\n");
		printf("b		-	Toggle bounding box\n");
		printf("r		-	Reverse polygon winding\n");
		printf("m		-	Toggle color/material/none
mode\n");
		printf("p		-	Toggle performance
indicator\n");
		printf("s/S		-	Scale model smaller/larger\n");
		printf("t		-	Show model stats\n");
		printf("+/-		-	Increase/decrease smoothing
```

```
angle\n");
            printf("W               -    Write model to file
(out.obj)\n");
            printf("q/escape    -    Quit\n\n");
            break;

        case 't':
            stats = !stats;
            break;

        case 'p':
            performance = !performance;
            break;

        case 'm':
            material_mode++;
            if (material_mode > 2)
                material_mode = 0;
            printf("material_mode = %d\n", material_mode);
```

```
            lists();
            break;

        case 'd':
            glmDelete(model);
            init();
            lists();
            break;

        case 'w':
            glGetIntegerv(GL_POLYGON_MODE, params);
            if (params[0] == GL_FILL)
                glPolygonMode(GL_FRONT_AND_BACK,
GL_LINE);
            else
                glPolygonMode(GL_FRONT_AND_BACK,
GL_FILL);
            break;
```

```
case 'c':
        if (glIsEnabled(GL_CULL_FACE))
                glDisable(GL_CULL_FACE);
        else
                glEnable(GL_CULL_FACE);
        break;

case 'b':
        bounding_box = !bounding_box;
        break;

case 'n':
        facet_normal = !facet_normal;
        lists();
        break;

case 'r':
        glmReverseWinding(model);
        lists();
```

```
        break;

case 's':
        glmScale(model, 0.8);
        lists();
        break;

case 'S':
        glmScale(model, 1.25);
        lists();
        break;

case '-':
        smoothing_angle -= 1.0;
        printf("Smoothing angle: %.1f\n", smoothing_angle);
        glmVertexNormals(model, smoothing_angle);
        lists();
        break;
```

```
        case '+':
                smoothing_angle += 1.0;
                printf("Smoothing angle: %.1f\n", smoothing_angle);
                glmVertexNormals(model, smoothing_angle);
                lists();
                break;

        case 'W':
                glmScale(model, 1.0 / scale);
                glmWriteOBJ(model, "out.obj", GLM_SMOOTH |
GLM_MATERIAL);
                break;

        case 'R':
        {
                GLuint i;
                GLfloat swap;
                for (i = 1; i <= model->numvertices; i++) {
                        swap = model->vertices[3 * i + 1];
```

```
                model->vertices[3 * i + 1] = model->vertices[3 * i
+ 2];
                model->vertices[3 * i + 2] = -swap;
            }
            glmFacetNormals(model);
            lists();
            break;
        }

        case 27:
            exit(0);
            break;
        }

        glutPostRedisplay();
}

void menu(int item) //右鍵選單以及一些模型的初始設定
{
```

```
int i = 0;
DIR* dirp;
char* name;
struct dirent* direntp;


if (item > 0) {
	keyboard((unsigned char)item, 0, 0);
}
else {
	dirp = opendir(DATA_DIR);
	while ((direntp = readdir(dirp)) != NULL) {
		if (strstr(direntp->d_name, ".obj")) {
			i++;
			if (i == -item)
				break;
		}
	}
	if (!direntp)
		return;
```

```
            name = (char*)malloc(strlen(direntp->d_name) +
strlen(DATA_DIR) + 1);
            strcpy(name, DATA_DIR);
            strcat(name, direntp->d_name);
            model = glmReadOBJ(name);
            scale = glmUnitize(model);
            glmFacetNormals(model);
            glmVertexNormals(model, smoothing_angle);

            if (model->nummaterials > 0)
                  material_mode = 2;
            else
                  material_mode = 0;

            lists();
            free(name);

            glutPostRedisplay();
      }
```

```
}

static GLint		mouse_state;
static GLint		mouse_button;

void mouse(int button, int state, int x, int y)
{
	GLdouble model[4 * 4];
	GLdouble proj[4 * 4];
	GLint view[4];

	/* fix for two-button mice -- left mouse + shift = middle
mouse */
	if (button == GLUT_LEFT_BUTTON && glutGetModifiers() &
GLUT_ACTIVE_SHIFT)
		button = GLUT_MIDDLE_BUTTON;

	gltbMouse(button, state, x, y);
```

```
	mouse_state = state;
	mouse_button = button;
	//如果滑鼠按下或放開
	if (state == GLUT_DOWN && button ==
GLUT_MIDDLE_BUTTON) {
		glGetDoublev(GL_MODELVIEW_MATRIX, model);
		glGetDoublev(GL_PROJECTION_MATRIX, proj);
		glGetIntegerv(GL_VIEWPORT, view);
		gluProject((GLdouble)x, (GLdouble)y, 0.0,
			model, proj, view,
			&pan_x, &pan_y, &pan_z);
		gluUnProject((GLdouble)x, (GLdouble)y, pan_z,
			model, proj, view,
			&pan_x, &pan_y, &pan_z);
		pan_y = -pan_y;
	}

	glutPostRedisplay();
}
```

```
void motion(int x, int y)
{
	GLdouble model[4 * 4];
	GLdouble proj[4 * 4];
	GLint view[4];

	gltbMotion(x, y);
	//如果滑鼠按下或放開
	if (mouse_state == GLUT_DOWN && mouse_button ==
GLUT_MIDDLE_BUTTON) {
		glGetDoublev(GL_MODELVIEW_MATRIX, model);
		glGetDoublev(GL_PROJECTION_MATRIX, proj);
		glGetIntegerv(GL_VIEWPORT, view);
		gluProject((GLdouble)x, (GLdouble)y, 0.0,
			model, proj, view,
			&pan_x, &pan_y, &pan_z);
		gluUnProject((GLdouble)x, (GLdouble)y, pan_z,
			model, proj, view,
```

```
                &pan_x, &pan_y, &pan_z);
            pan_y = -pan_y;
        }

        glutPostRedisplay();
}

int main(int argc, char** argv)
{
        int buffering = GLUT_DOUBLE;
        struct dirent* direntp;
        DIR* dirp;
        int models;

        glutInitWindowSize(512, 512);
        glutInit(&argc, argv);

        while (--argc) {
            if (strcmp(argv[argc], "-sb") == 0)
```

```
            buffering = GLUT_SINGLE;
        else
            model_file = argv[argc];
    }

    if (!model_file) {
        model_file = "data/soccerball.obj";        //預設使用
的模型
    }

    glutInitDisplayMode(GLUT_RGB | GLUT_DEPTH |
buffering);
    glutCreateWindow("HW2");       //視窗名稱

    glutReshapeFunc(reshape);     //視窗大小設置
    glutDisplayFunc(display);     //模型顯示
    glutKeyboardFunc(keyboard); //案件觸發
    glutMouseFunc(mouse);           //滑鼠起始點
    glutMotionFunc(motion);        //滑鼠終點
```

```
        models = glutCreateMenu(menu);
        dirp = opendir(DATA_DIR);
        if (!dirp) {
                fprintf(stderr, "%s: can't open data directory.\n",
argv[0]);
        }
        else {
                while ((direntp = readdir(dirp)) != NULL) {
                        if (strstr(direntp->d_name, ".obj")) {
                                entries++;
                                glutAddMenuEntry(direntp->d_name, -
entries);
                        }
                }
                closedir(dirp);
        }

        glutCreateMenu(menu);      //3D 模型選單
```

```
glutAddSubMenu("3D Models", models);
glutAddMenuEntry("[w]  wireframe (線框) / filled (預設: 填充)", 'w');
glutAddMenuEntry("[c]  culling on / off (可看到 3D Models 內部)", 'c');
glutAddMenuEntry("[n]  face/smooth (凹凸面/平滑面)", 'n');
glutAddMenuEntry("[b]  bounding box on/off (3D Models 外接透明方塊)", 'b');
glutAddMenuEntry("[p]  frame rate on/off (顯示 FPS 幀數)", 'p');
glutAddMenuEntry("[t]  Toggle model statistics (3D Models 的訊息)", 't');
glutAddMenuEntry("[m]  Toggle color/material/none mode (顏色/材質/無)", 'm');
glutAddMenuEntry("[r]  Reverse polygon winding (只看到 3D Models 的內部)", 'r');
glutAddMenuEntry("[s]  Scale model smaller (變小)", 's');
```

```
        glutAddMenuEntry("[S]      Scale model larger (變大)", 'S');
        glutAddMenuEntry("[+]      Increase smoothing angle (平滑
程度升高)", '+');
        glutAddMenuEntry("[-]      Decrease smoothing angle (平
滑程度降低)", '-');
        glutAddMenuEntry("[W]      Write model to file
(out.obj)(匯出 3D Models)", 'W');
        glutAddMenuEntry("[Esc] Quit", 27);
        glutAttachMenu(GLUT_RIGHT_BUTTON);

        init();

        glutMainLoop();
        return 0;
}
```