計算機圖學單元介紹

一、英文主題：

Chapter 08 : Discrete Techniques

二、中文主題：

單元08 : 其他的繪圖技巧

三、組別：第06組

四、組員：

B0729003 何妍霖；B0827213 陳昱慈；B0829011 王紹丞；

B0829015 黃聖文；B0829024 葉季儒；B0829057 沈沛錡；

五、功能簡述：

本單元內容為介紹繪圖技巧，包含生成紋理、利用反射原理來繪圖的技巧，以及線段呈現和透明物體相疊的處理方式，還有離散的繪圖技巧。

六、主要程式碼：

相關檔案：Ch_08_tm6_src1.cpp

```
#include<stdlib.h>

#include<stdio.h>

#include<time.h>

#include <GL/glut.h>


/* default data*/

/* can enter other values via command line arguments */


#define CENTERX -0.5

#define CENTERY 0.5

#define HEIGHT 0.5

#define WIDTH 0.5

#define MAX_ITER 100


/* N x M array to be generated */


#define N 500

#define M 500


float height ; /* size of window in complex plane */

float width ;

float cx; /* center of window in complex plane */
```

```
float cy;

int max; /* number of interations per point */


int n=N;

int m=M;


/* use unsigned bytes for image */


GLubyte image[N][M];


/* complex data type and complex add, mult, and magnitude functions

   probably not worth overhead */


typedef float complex[2];


void add(complex a, complex b, complex p)

{

    p[0]=a[0]+b[0]; //x1y1+y1y2+c

    p[1]=a[1]+b[1]; //x1y2+x2y1+c

}


void mult(complex a, complex b, complex p)
```

```
{
    p[0]=a[0]*b[0]-a[1]*b[1];
    p[1]=a[0]*b[1]+a[1]*b[0];
}

float mag2(complex a){
    return(a[0]*a[0]+a[1]*a[1]);//]z]^2=x^2+y^2
}

void form(float a, float b, complex p)
{
    p[0]=a;
    p[1]=b;
}

void display()
{
    glClear(GL_COLOR_BUFFER_BIT);
    glDrawPixels(n,m,GL_COLOR_INDEX, GL_UNSIGNED_BYTE, image);
    glFlush();
}
```

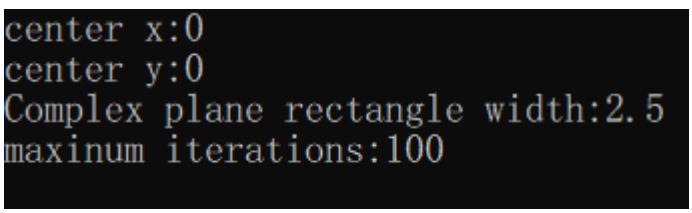
```
void myReshape(int w, int h)
{
    glViewport(0, 0, w, h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    if (w <= h)
    gluOrtho2D(0.0, 0.0, (GLfloat) n, (GLfloat) m* (GLfloat) h /(GLfloat) w);
    else
    gluOrtho2D(0.0, 0.0, (GLfloat) n * (GLfloat) w / (GLfloat) h,
                        (GLfloat) m);
    glMatrixMode(GL_MODELVIEW);
}
void myinit()
{
    float redmap[256], greenmap[256],bluemap[256];  //mapping RGB color
    int i;

    glClearColor (1.0, 1.0, 1.0, 1.0);
    gluOrtho2D(0.0, 0.0, (GLfloat) n, (GLfloat) m);//二維初始化

/* define pseudocolor maps, ramps for red and blue,
   random for green */
```

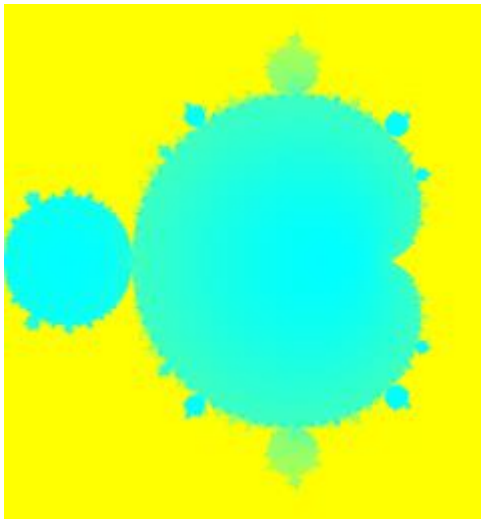
```
    for(i=0;i<256;i++)          //random color
    {
            redmap[i]=i/255.;
            greenmap[i]=rand()%255;
            bluemap[i]=1.0-i/255.;
    }

    glPixelMapfv(GL_PIXEL_MAP_I_TO_R, 256, redmap);
    glPixelMapfv(GL_PIXEL_MAP_I_TO_G, 256, greenmap);
    glPixelMapfv(GL_PIXEL_MAP_I_TO_B, 256, bluemap);
}

main(int argc, char *argv[])
{
    int i, j, k;
    float x, y, v;
    complex c0, c, d;

/* uncomment to define your own parameters */

    scanf("%f", &cx); /* center x */
    scanf("%f", &cy); /* center y */
```

```
scanf("%f", &width); /* rectangle width */

height=width; /* rectangle height */

scanf("%d",&max); /* maximum iterations */


for (i=0; i<n; i++)

        for(j=0; j<m; j++){

        /* starting point */

        x= i *(width/(n-1)) + cx -width/2;

        y= j *(height/(m-1)) + cy -height/2;


        form(0,0,c);

        form(y,x,c0);


        /* complex iteration */


 for(k=0; k<max; k++){

        mult(c,c,d);

        add(d,c0,c);

        v=mag2(c);

         if(v>4.0) break; /* assume not in set if mag > 4 */

}
```

```
                /* assign gray level to point based on its magnitude */
    if(v>1.0) v=1.0; /* clamp if > 1 256(0-255) types gray level*/
    image[i][j]=255*v;
    }



    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB );
    glutInitWindowSize(N, M);
    glutCreateWindow("mandlebrot");
    myinit();
    glutReshapeFunc(myReshape);
    glutDisplayFunc(display);


    glutMainLoop();
}
```

執行結果:





七、程式說明：

A. 宣告:

宣告式	原因
#define N 500	宣告N*M array

#define M 500	宣告N*M array
GLubyte image[N][M];	OpenGL 定義式:Unsigned binary integer，用以繪製圖形。
typedef float complex[2];	賦予一個float型態陣列的新名稱complex，用以表達複數方程式。
float height ;	複數平面上一個正方格的長。
float width ;	複數平面上一個正方格的寬。
float cx;	複數平面中心座標點x
float cy;	複數平面中心座標點y
int max;	疊代最大次數
int n=N;	設定顯示窗格之二維陣列數
int m=M;	設定顯示窗格之二維陣列數

B. 函式介紹:

甲、 void add(complex a, complex b, complex p)

將繪製之長寬數值與所計算之疊代相乘結果相加，得到完整式子

p[0]=a[0]+b[0];

p[1]=a[1]+b[1];

乙、 void mult(complex a, complex b, complex p)

將數值相乘，滿足z^2+c的形式

p[0]=a[0]*b[0]-a[1]*b[1];　　虛部

p[1]=a[0]*b[1]+a[1]*b[0];　　實部

丙、 float mag2(complex a)

將其換置成｜z^2｜的型式

a[0]*a[0]+a[1]*a[1]→｜z^2｜=x^2+y^2

丁、 void form(float a, float b, complex p)

將float型態變數轉成定義之complex型態

p[0]=a; 虛部

p[1]=b; 實部

戊、 void display()

glClear(GL_COLOR_BUFFER_BIT);清除viewpoint的buffer

glDrawPixels(n,m,GL_COLOR_INDEX, GL_UNSIGNED_BYTE, image);

繪製已經計算過後的曼德博碎形數值陣列(像素)，繪製視窗為n*m。

glFlush();清空buffer後cpu不會有opengl相關事情需要處裡。

己、 void myReshape(int w, int h)

主要使用在重繪即刷新視窗之設定

parameter w、h為視窗長寬

glViewport(0, 0, w, h);根據變化重新繪製視窗矩形寬度及高度。

glMatrixMode(GL_PROJECTION);宣告對物件進行投影操作

glLoadIdentity();物件設置為單位矩陣。

if (w <= h)

gluOrtho2D(0.0, 0.0, (GLfloat) n, (GLfloat) m* (GLfloat) h /(GLfloat) w);

當視窗寬度<=高；根據線性比例重新初始化top座標。

else

gluOrtho2D(0.0, 0.0, (GLfloat) n * (GLfloat) w / (GLfloat) h,(GLfloat) m);

當視窗寬度>高；根據線性比例重新初始化botton座標。

glMatrixMode(GL_MODELVIEW);宣告對模型進行投影操作。

庚、 void myinit()

主要處理隨機上色並mapping。

三個float型態長度為256的陣列: redmap[256], greenmap[256],bluemap[256]

為新增空間放置三原色之數值。

gluOrtho2D(0.0, 0.0, (GLfloat) n, (GLfloat) m)為二維投影初始化設定；n和m為定義繪製窗框大小，前有敘述。

for(i=0;i<256;i++)重複256次賦予前述三原色陣列色像

(1)redmap[i]=i/255.;　　以線性方式添加紅色

(2)greenmap[i]=rand()%255; 以隨機方式添加綠色

(3)bluemap[i]=1.0-i/255.;　　以線性方式添加藍色

glPixelMapfv(GL_PIXEL_MAP_I_TO_R, 256, redmap);

glPixelMapfv(GL_PIXEL_MAP_I_TO_G, 256, greenmap);

glPixelMapfv(GL_PIXEL_MAP_I_TO_B, 256, bluemap);

glPixelMapfv:將像素映射(mapping)

GL_PIXEL_MAP_I_TO_R →將各個index值放入紅色元件

GL_PIXEL_MAP_I_TO_G →將各個index值放入綠色元件

GL_PIXEL_MAP_I_TO_B →將各個index值放入藍色元件

C. 主程式:

讓使用者定義複數平面的初始化數值(center x,y,rectangle width,max iteration)

for (i=0; i<n; i++)

for(j=0; j<m; j++){

(1)x= i *(width/(n-1)) + cx -width/2;

視窗座標*(複數平面窗格寬/總繪製視窗數=平均分配複數平面的1個繪製視窗數寬度

+複數平面中心座標軸-複數平面大小/2(始於座標中心，寬度為一半)=繪製寬度

(2)y= j *(height/(m-1)) + cy -height/2;

呈(1)述，y亦同。

form(0,0,c);賦予初始值至complex型態的陣列裡

form(y,x,c0)；賦予每一次計算出的繪製長度至complex型態的陣列裡

for(k=0; k<max; k++){

mult(c,c,d);呼叫函式與自身相乘，滿足z^2

add(d,c0,c);呼叫函式與繪製長寬相加

滿足$z^2+c= z^2+x+iy$=係數*實部x+係數*虛部y

v=mag2(c);將其算出｜z^2｜=x^2+y^2

if(v>4.0) break;}太大的數值暫不考慮。

if(v>1.0) v=1.0; 灰階有256種type(0-255)，當v大於1，即視為1繪製

image[i][j]=255*v

以向量大小賦予繪製陣列其點的灰階採樣顏色，之後利用myinit()上色。

glutInit(&argc, argv);

glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);　　單緩衝

glutInitWindowSize(N, M); 視窗大小

glutCreateWindow("mandlebrot");建立視窗與名稱

myinit();呼叫上色函式

glutReshapeFunc(myReshape);呼叫重繪與刷新函式

glutDisplayFunc(display);呼叫顯示物件函式。

glutMainLoop();glut事件中處理循環。

八、延伸應用程式碼：

相關檔案：Ch_08_tm6_src2.cpp

```
#include<stdlib.h>
#include<stdio.h>
#include<time.h>
#include <GL/glut.h>

GLfloat planes[]= {-1.0, 0.0, 1.0, 0.0};
GLfloat planet[]= {0.0, -1.0,  0.0, 1.0};
GLfloat vertices[][3] = {{-1.0,-1.0,-1.0},{1.0,-1.0,-1.0},//八個頂點
                         {1.0,1.0,-1.0}, {-1.0,1.0,-1.0}, {-1.0,-1.0,1.0},
                         {1.0,-1.0,1.0}, {1.0,1.0,1.0}, {-1.0,1.0,1.0}};
GLfloat colors[][4] = {{0.0,0.0,0.0,0.5},{1.0,0.0,0.0,0.5},//頂點區塊顏色
                       {1.0,1.0,0.0,0.5}, {0.0,1.0,0.0,0.5}, {0.0,0.0,1.0,0.5},
                       {1.0,0.0,1.0,0.5}, {1.0,1.0,1.0,0.5}, {0.0,1.0,1.0,0.5}};

void polygon(int a, int b, int c, int d) //繪製一個面的function
{
   glBegin(GL_POLYGON);
   glColor4fv(colors[a]);
   glTexCoord2f(0.0,0.0);
   glVertex3fv(vertices[a]);
   glColor4fv(colors[b]);
   glTexCoord2f(0.0,1.0);
   glVertex3fv(vertices[b]);
   glColor4fv(colors[c]);
```

```
    glTexCoord2f(1.0,1.0);
    glVertex3fv(vertices[c]);
    glColor4fv(colors[d]);
    glTexCoord2f(1.0,0.0);
    glVertex3fv(vertices[d]);
    glEnd();
}

void colorcube()//繪製cube總共六個面
{
/* map vertices to faces */
    polygon(0,3,2,1);
    polygon(2,3,7,6);
    polygon(0,4,7,3);
    polygon(1,2,6,5);
    polygon(4,5,6,7);
    polygon(0,1,5,4);
}

static GLfloat theta[] = {0.0,0.0,0.0}; //xyz軸的旋轉角度
static GLint axis = 2;

void display()
{
```

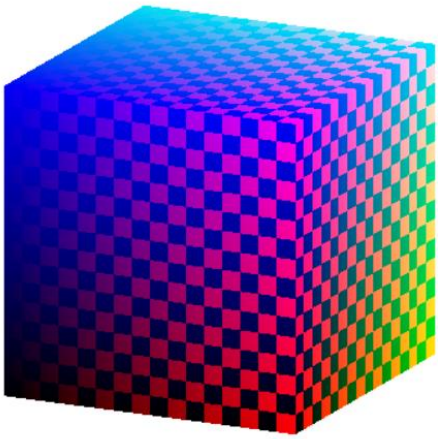
```
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();
    glRotatef(theta[0], 1.0, 0.0, 0.0);
    glRotatef(theta[1], 0.0, 1.0, 0.0);
    glRotatef(theta[2], 0.0, 0.0, 1.0);
    colorcube();
    glutSwapBuffers();
}

void spinCube() //旋轉cube
{
    theta[axis] += 0.1;
    if( theta[axis] > 360.0 ) theta[axis] -= 360.0;
    glutPostRedisplay();
}

void mouse(int btn, int state, int x, int y)
{
    if(btn==GLUT_LEFT_BUTTON && state == GLUT_DOWN) axis = 0;
    if(btn==GLUT_MIDDLE_BUTTON && state == GLUT_DOWN) axis = 1;
    if(btn==GLUT_RIGHT_BUTTON && state == GLUT_DOWN) axis = 2;
}

void myReshape(int w, int h)
```

```
{
    glViewport(0, 0, w, h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    if (w <= h)
        glOrtho(-2.0, 2.0, -2.0 * (GLfloat) h / (GLfloat) w,
            2.0 * (GLfloat) h / (GLfloat) w, -10.0, 10.0);
    else
        glOrtho(-2.0 * (GLfloat) w / (GLfloat) h,
            2.0 * (GLfloat) w / (GLfloat) h, -2.0, 2.0, -10.0, 10.0);
    glMatrixMode(GL_MODELVIEW);
}

void key(unsigned char k, int x, int y)
{
    if(k == '1') glutIdleFunc(spinCube);
    if(k == '2') glutIdleFunc(NULL);
    if(k == 'q') exit(0);
}

int main(int argc, char **argv)
{
   GLubyte image[64][64][3];
//Texture(紋理)，一二維是texture的長寬，第三位存放RGB的數值
```

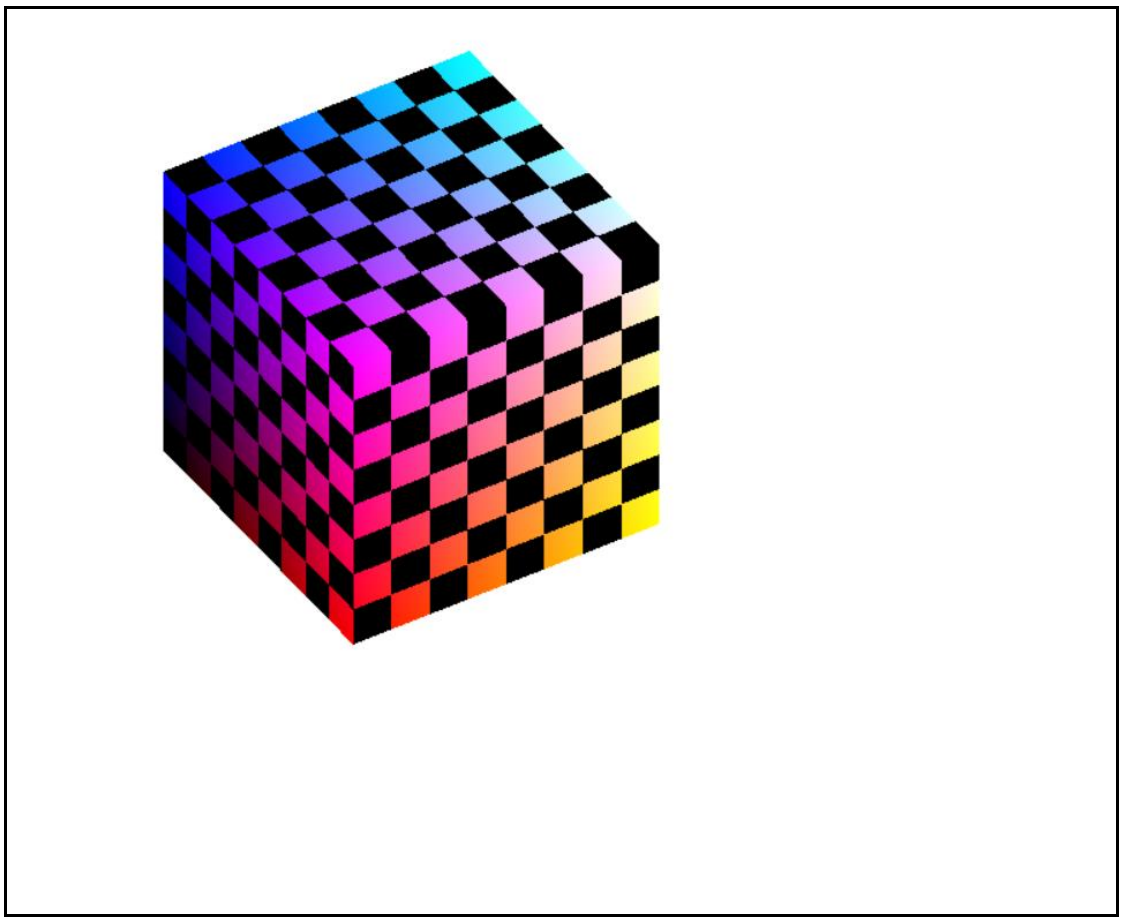
```
    int i, j,  c;
    for(i=0;i<64;i++)//texture
    {
     for(j=0;j<64;j++)
     {
       c = ((((i&0x4)==0)^((j&0x4))==0))*255;
       //每個面每4個長度單位會從深色texture變成透明的，透明變深色
       image[i][j][0]= (GLubyte) c;
       image[i][j][1]= (GLubyte) c-30;
       image[i][j][2]= (GLubyte) c-10;
     }
    }
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowSize(500, 500);
    glutCreateWindow("colorcube");
    glutReshapeFunc(myReshape);
    glutDisplayFunc(display);
    glutIdleFunc(spinCube);
    glutMouseFunc(mouse);
    glEnable(GL_DEPTH_TEST);
    glEnable(GL_TEXTURE_2D);
    glTexImage2D(GL_TEXTURE_2D,0,3,64,64,0,GL_RGB,GL_UNSIGNED_BYTE, ima
ge);
    glTexParameterf(GL_TEXTURE_2D,GL_TEXTURE_WRAP_S,GL_REPEAT);
```

```
    glTexParameterf(GL_TEXTURE_2D,GL_TEXTURE_WRAP_T,GL_REPEAT);
    glTexParameterf(GL_TEXTURE_2D,GL_TEXTURE_MAG_FILTER,GL_NEAREST);
    glTexParameterf(GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,GL_NEAREST);
    glutKeyboardFunc(key);
    glClearColor(1.0,1.0,1.0,1.0);
    glutMainLoop();
}
```

延伸結果:



原結果:



九、應用說明：

A.應用內容

了解texture原理，改變其黑色區塊顏色，以及方格切分的數量。

B.主程式

(1)Texture生成

```
//Texture(紋理)，一二維是texture的長寬，第三位存放RGB的數值
  int i, j,  c;
  for(i=0;i<64;i++)//texture
  {
    for(j=0;j<64;j++)
    {
```

```
        c = ((((i&0x4)==0)^((j&0x4))==0))*255;
        //每個面每4個長度單位會從深色texture變成較淺的，較淺變深色
        image[i][j][0]= (GLubyte) c;
        image[i][j][1]= (GLubyte) c-30;
        image[i][j][2]= (GLubyte) c-10;
      }
    }
```

(2)初始設定

```
  glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowSize(500, 500);
    glutCreateWindow("colorcube");
```

(3)畫面框度變化的處理

```
 glutReshapeFunc(myReshape);
```

(4)畫面呈現

```
  glutDisplayFunc(display);
```

(5)沒有其他動作時不斷執行spinCube

```
    glutIdleFunc(spinCube);
```

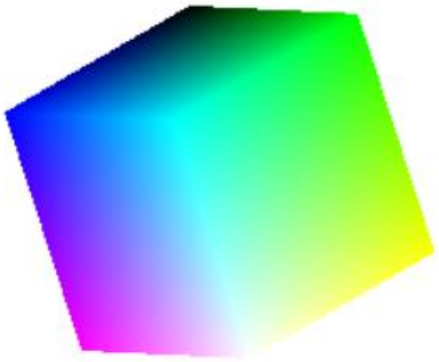
(6)加入texture

```
glTexImage2D(GL_TEXTURE_2D,0,3,64,64,0,GL_RGB,GL_UNSIGNED_BYTE, image);
```

C.Cube相關函式

甲、繪製立方體

```
void colorcube()//繪製cube總共六個面
{
/* map vertices to faces */
    polygon(0,3,2,1);
    polygon(2,3,7,6);
    polygon(0,4,7,3);
    polygon(1,2,6,5);
    polygon(4,5,6,7);
    polygon(0,1,5,4);}
```



乙、畫面呈現

```
void display()
{
   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
   glLoadIdentity();
   glRotatef(theta[0], 1.0, 0.0, 0.0);
   glRotatef(theta[1], 0.0, 1.0, 0.0);
   glRotatef(theta[2], 0.0, 0.0, 1.0);
   colorcube();
   glutSwapBuffers();
}
```

丙、立方體的旋轉

```
void spinCube() //旋轉cube
{
```

```
    theta[axis] += 0.1;//原本為2.0，將此數值調小，可減緩旋轉的速度
    if( theta[axis] > 360.0 ) theta[axis] -= 360.0;
    glutPostRedisplay();
}
```

D.Texture相關函式

甲、glTexImage2D(GL_TEXTURE_2D,0,3,64,64,0,GL_RGB,GL_UNSIGNED_BYTE, image);

指定參數與格式到GL_TEXTURE_2D物件

乙、 glTexParameterf(GL_TEXTURE_2D,GL_TEXTURE_WRAP_S,GL_REPEAT);

指定水平方向(GL_TEXTURE_WRAP_S)的應用面超出texture大小的處理方式 (GL_REPEAT)

丙、glTexParameterf(GL_TEXTURE_2D,GL_TEXTURE_WRAP_T,GL_REPEAT);

指定垂直方向(GL_TEXTURE_WRAP_T)的應用面超出texture大小的處理方式(GL_REPEAT)

丁、glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);

指定材質影像需要放大(GL_TEXTURE_MAG_FILTER)時的處理方式(GL_NEAREST)

戊、glTexParameterf(GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,GL_NEAREST);

指定材質影像需要縮小(GL_TEXTURE_MIN_FILTER)時的處理方式(GL_NEAREST)

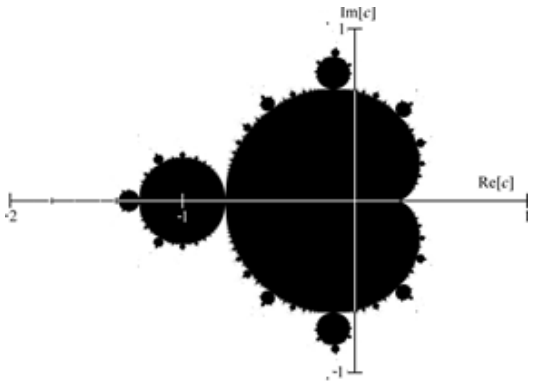
十、相關數學公式：

1.

曼德博集合會利用疊代進行二次多項式的計算:

C為複數參數，在這邊我們先假設x+iy

從z為(0,0)開始，預計會呈現右圖的形狀:



$Z_0 = 0$

$Z_1 = Z_0^2 + c = c$

$Z_2 = Z_1^2 + c = c*c+c$

Im為虛部、Re為實部

十一、參考資料：

(1) 可參考的公開文件

曼德博集合https://zh.wikipedia.org/wiki/曼德博集合

(2) 可參考的公開網址

Silicon software API :https://www.khronos.org/