計算機圖學單元介紹

一、英文主題：

Chapter 6: Shading

二、中文主題：

單元 06： 打光與製造陰影

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、作業分工：

Ch6.1-Ch6.2：王紹丞

Ch6.3-Ch6.4：何妍霖

Ch6.5-Ch6.6：陳昱慈

Ch6.7-Ch6.8：沈沛錡

Ch6.9：黃聖文

Ch6.10：葉季儒

Ch6.11：黃聖文

展示程式：葉季儒

報告 PPT：全員共同製作

報告 WORD：葉季儒

六、功能簡述：

調整圖形的繪圖模式、法向量，與設定光照強度、光照係數，達成不同光照效果。

七、主要程式碼：

為完整解釋程式流程，以下為全部原始碼，相關檔案：Ch_06_tm6_src1.cpp

```
/* Recursive subdivision of cube (Chapter 6). Three display
modes: wire frame, constant, and interpolative shading */
/*Program also illustrates defining materials and light sources
in myiit() */

/* mode 0 = wire frame, mode 1 = constant shading,
mode 3 = interpolative shading */

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

typedef float point[4];

/* initial tetrahedron */

point v[]={{0.0, 0.0, 1.0}, {0.0, 0.942809, -0.33333},
        {-0.816497, -0.471405, -0.333333}, {0.816497, -0.471405, -
0.333333}};
```

```
int n;
int mode;

void triangle( point a, point b, point c)

/* display one triangle using a line loop for wire frame, a single
normal for constant shading, or three normals for interpolative shading */
{
    if (mode==0) glBegin(GL_LINE_LOOP);
    else glBegin(GL_POLYGON);
        if(mode==1) glNormal3fv(a);
        if(mode==2) glNormal3fv(a);
        glVertex3fv(a);
        if(mode==2) glNormal3fv(b);
        glVertex3fv(b);
        if(mode==2) glNormal3fv(c);
        glVertex3fv(c);
    glEnd();
}

void normal(point p)
{

/* normalize a vector */
```

```
        double sqrt();
        float d =0.0;
        int i;
        for(i=0; i<3; i++) d+=p[i]*p[i];
        d=sqrt(d);
        if(d>0.0) for(i=0; i<3; i++) p[i]/=d;
}

void divide_triangle(point a, point b, point c, int m){
/* triangle subdivision using vertex numbers
righthand rule applied to create outward pointing faces */
        point v1, v2, v3;
        int j;
        if(m>0)
        {
                for(j=0; j<3; j++) v1[j]=a[j]+b[j];
                normal(v1);
                for(j=0; j<3; j++) v2[j]=a[j]+c[j];
                normal(v2);
                for(j=0; j<3; j++) v3[j]=b[j]+c[j];
                  normal(v3);
                divide_triangle(a, v1, v2, m-1);
                divide_triangle(c, v2, v3, m-1);
```

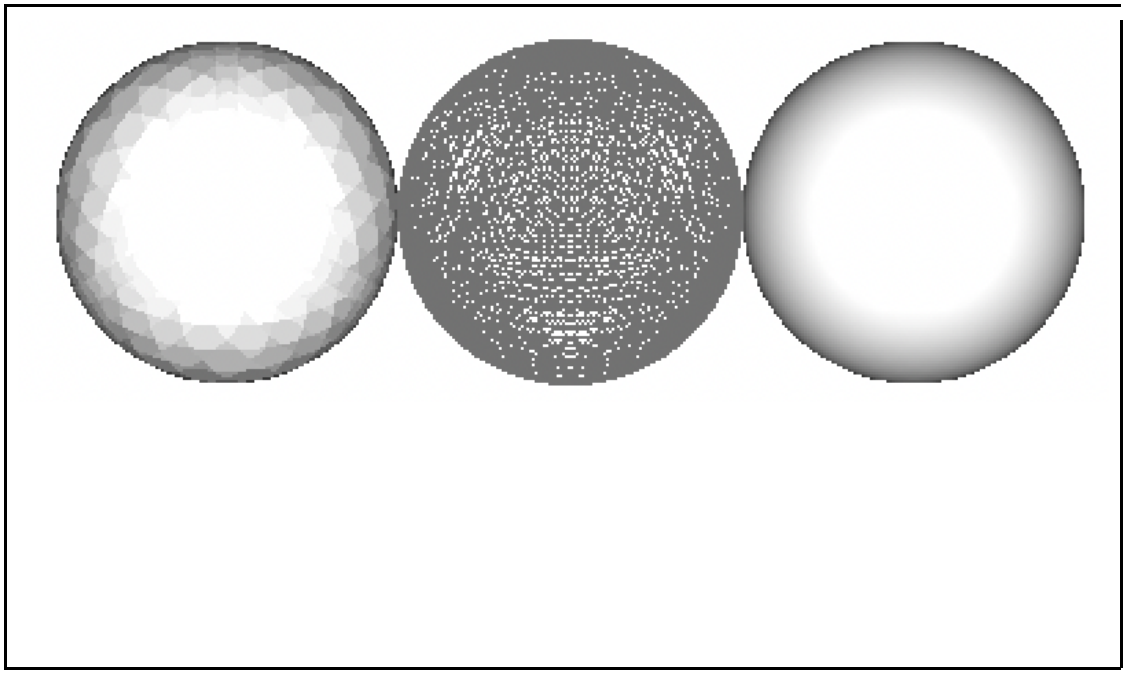
```
            divide_triangle(b, v3, v1, m-1);
            divide_triangle(v1, v3, v2, m-1);
        }
        else(triangle(a,b,c)); /* draw triangle at end of recursion */
    }

    void tetrahedron(int m){
    /* Apply triangle subdivision to faces of tetrahedron */

        divide_triangle(v[0], v[1], v[2], m);
        divide_triangle(v[3], v[2], v[1], m);
        divide_triangle(v[0], v[3], v[1], m);
        divide_triangle(v[0], v[2], v[3], m);
    }

    void display(void)
    {

    /* Displays all three modes, side by side */

        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
           glLoadIdentity();
        mode=0;
        tetrahedron(n);
```

```
        mode=1;
        glTranslatef(-2.0, 0.0,0.0);
        tetrahedron(n);
        mode=2;
        glTranslatef( 4.0, 0.0,0.0);
        tetrahedron(n);


        glFlush();
}


void myReshape(int w, int h)
{
        glViewport(0, 0, w, h);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        if (w <= h)
                glOrtho(-4.0, 4.0, -4.0 * (GLfloat) h / (GLfloat) w,
                        4.0 * (GLfloat) h / (GLfloat) w, -10.0, 10.0);
        else
                glOrtho(-4.0 * (GLfloat) w / (GLfloat) h,
                        4.0 * (GLfloat) w / (GLfloat) h, -4.0, 4.0, -10.0, 10.0);
        glMatrixMode(GL_MODELVIEW);
        display();
}
```

```
void myinit()
{
	GLfloat mat_specular[]={1.0, 1.0, 1.0, 1.0};
	GLfloat mat_diffuse[]={1.0, 1.0, 1.0, 1.0};
	GLfloat mat_ambient[]={1.0, 1.0, 1.0, 1.0};
	GLfloat mat_shininess={100.0};
	GLfloat light_ambient[]={0.0, 0.0, 0.0, 1.0};
	GLfloat light_diffuse[]={1.0, 1.0, 1.0, 1.0};
	GLfloat light_specular[]={1.0, 1.0, 1.0, 1.0};

/* set up ambient, diffuse, and specular components for light 0 */

	glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);
	glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);
	glLightfv(GL_LIGHT0, GL_SPECULAR, light_specular);

/* define material proerties for front face of all polygons */

	glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
	glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
	glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
	glMaterialf(GL_FRONT, GL_SHININESS, mat_shininess);
```

```
    glShadeModel(GL_SMOOTH); /*enable smooth shading */
    glEnable(GL_LIGHTING); /* enable lighting */
    glEnable(GL_LIGHT0);    /* enable light 0 */
    glEnable(GL_DEPTH_TEST); /* enable z buffer */

    glClearColor (1.0, 1.0, 1.0, 1.0);
    glColor3f (0.0, 0.0, 0.0);
}

void main(int argc, char **argv){
    n=5;     //default=5
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowSize(500, 500);
    glutCreateWindow("sphere");
    myinit();
    glutReshapeFunc(myReshape);
    glutDisplayFunc(display);
    glutMainLoop();
}
```

執行結果：



八、程式說明：

此程序執行時會打開一個窗口，將背景設為黑色，並使用不同作法/光照，繪製三個不同的球型。學習更加細緻的光照方法。

(1)開頭引用 <GL/freeglut.h>函式庫，可以執行 OpenGL 的相關函數，其他函式庫用於數學計算與其他用途。

(2) glutInit(&argc, argv)；初始化 GLUT 函式庫，並且引入程序參數。

(3) glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGBA)；設定 GLUT_DOUBLE 是雙緩衝，而 GLUT_RGB 是指色彩模式是 RGB。

(4) glutInitWindowSize(600, 600)；設定視窗大小為 600*600 的窗口。

(5) void WINAPI glOrtho(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top, GLdouble zNear, GLdouble zFar); left 左方垂直裁剪平面的座標。 right Theright 垂直裁剪平面的座標。bottom 底部水準裁剪平面的座標。 top 上方水準裁剪計畫的座標。 zNear 靠近深度裁剪平面的距離。 如果平面位於檢視器後方，此距離就會是負數。 zFar 距離深度裁剪平面的距離。 如果平面位於檢視器後方，此距離就會是負數。

通過設定此函數，可以將顯示畫面擷取，選擇需要的部分顯示(或拉伸)。

(6) gluLookAt()；指定攝影機位置，與朝向其上方的矩陣。城市會自動移動場景，達成指定的觀測效果。

(7) **glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient)；** 設定光源強度，GL_LIGHT0 指光源 0，OPENGL 共有 8 個光源可用，GL_AMBIENT 指環境光。**light_ambient** 為強度向量。

(8) glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular) : 設定物體材質的光反射強度。

(9) glEnable(GL_LIGHT0)： 開啟 Light0 光源。

(10) glNormal3fv(a)：設定頂點法向量，會引響光照運算。

(11) glVertex3fv(a)：設定頂點座標，畫圖使用。

(12) glutDisplayFunc(display)； 將 display 函數指標傳入，讓 GLUT 庫將渲染內容交由使用者指定。

(13)glutMainLoop()； 控制全送還給 GLUT 函式庫，進行視窗生成，與場景繪製。

九、延伸應用程式碼： Ch_06_tm6_src2.cpp

```
#include<Windows.h>
#include<GLWglut.h>
#include<cstdlib>
#include<ctime>
//太陽、地球和月亮
//假設每個月都是 30 天
//一年 12 個月，共是 360 天
static int day = 200; // day 的變化:從 0 到 359
float xRotated = 90.0, yRotated = 0.0, zRotated = 0.0;
float distinceX = 0.0,distinceY=0.0,distinceZ=0.0;
float light_position[] = { -20,  20,  0};  //光源的位置
int trigger=0;
int old_rot_x = 0;                          //剛按下滑鼠時的視窗座標
int old_rot_y = 0;

int rot_x = 0;                              //拖曳後的相對座標，用這決定
要旋轉幾度
int rot_y = 0;

int record_x = 0;                           //紀錄上一次旋轉的角度
int record_y = 0;
//void glTranslatef(GLfloat x,GLfloat y,GLfloat z);
```

```
//函數功能：沿 X 軸正方向平移 x 個單位(x 是有符號數)

//  沿 Y 軸正方向平移 y 個單位(y 是有符號數)

//  沿 Z 軸正方向平移 z 個單位(z 是有符號數)

//void glRotatef(GLfloat angle,GLfloat x,GLfloat y,GLfloat z);

//先解釋一下旋轉方向，做(0,0,0)到(x,y,z)的向量，用右手握住這條向量，大拇指指向向量的正方向，四指環繞的方向就是旋轉的方向；

//函數功能：以點(0,0,0)到點(x,y,z)為軸，旋轉 angle 角度；


void star(int i) {

   int var=0;

   var=rand()%199;

   glTranslatef( -var-2*i, var-i, var/2);

   glColor3f(1.0f, 1.0f, 1.0f);

   glutSolidSphere (1, 20, 50);

   glTranslatef( var-8*var/12, var-100, var);

   glutSolidSphere (1, 20, 50);

}


void block() {

//  glTranslatef( 0+distinceX, 0+distinceY, distinceZ);

   int point=100;

   glRotatef( (float)rot_y + (float)record_y, 1.0, 0.0, 0.0);    //以 x 軸當旋轉軸

    glRotatef( (float)rot_x + (float)record_x, 0.0, 1.0, 0.0);    //以 y 軸當旋轉軸
```

```
glPushMatrix ();
glBegin(GL_QUADS);
//正面        //固定 z 軸 x 前後 y 前後四點成一個面
glNormal3f(0,0,1);          //設定法向量
glVertex3f(-point, point, point);
glVertex3f(-point,-point, point);
glVertex3f( point,-point, point);
glVertex3f( point, point, point);
//背面
glNormal3f(0,0,-1);
glVertex3f(-point, point,-point);
glVertex3f( point, point,-point);
glVertex3f( point,-point,-point);
glVertex3f(-point,-point,-point);
//右側面
glNormal3f(1,0,0);
glVertex3f( point,  point, point);
glVertex3f( point, -point, point);
glVertex3f( point, -point,-point);
glVertex3f( point,  point,-point);
//左側面
glNormal3f(-1,0,0);
glVertex3f(-point, point,-point);
glVertex3f(-point,-point,-point);
```

```
    glVertex3f(-point,-point, point);
    glVertex3f(-point, point, point);
    //上面
    glNormal3f(0,1,0);
    glVertex3f(-point, point,-point);
    glVertex3f(-point, point, point);
    glVertex3f( point, point, point);
    glVertex3f( point, point,-point);
    //下面
    glNormal3f(0,-1,0);
    glVertex3f(-point, -point, point);
    glVertex3f(-point, -point,-point);
    glVertex3f( point, -point,-point);
    glVertex3f( point, -point, point);
    glEnd();
    glPopMatrix ();
}

void sun() {
    glColor3f(1.0f, 0.0f, 0.0f);
    glPushMatrix ();
    glTranslatef( 0+distinceX, 0+distinceY, distinceZ);   //太陽起始位置
//  glRotatef       (60.0, 1,0,0);
```

```
    glRotatef       (zRotated*2.0, 0,0,1);    // 自轉.
    glutSolidSphere (69.6, 20 , 50);
    glPopMatrix ();
}

void earth() {
    glColor3f(0.0f, 0.0f, 1.0f);
    glRotatef(day / 360.0*360.0, 0.0f, 0.0f, -1.0f);//地球公轉
    glTranslatef(150, 0.0f, 0.0f); //地球初始位置
    glPushMatrix ();
    glTranslatef( 0+distinceX, 0+distinceY, distinceZ);
    glRotatef       (60.0, 1,0,0);
    glRotatef       (zRotated*2.0, 0,0,1);    // 自轉.
    glutSolidSphere (16, 20, 50);
    glPopMatrix ();
}

void moon() {
    glColor3f(1.0f, 1.0f, 0.0f);
    glRotatef(day / 30.0*360.0 - day / 360.0*360.0, 0.0f, 0.0f, -
1.0f);//月球公轉再多除以 360 因為一天
//  glRotatef(day / 30.0*360.0, 0.0f, 0.0f, -1.0f);
    glTranslatef(38, 0.0f, 0.0f);
    glPushMatrix ();
```

```
    glTranslatef( 0+distinceX, 0+distinceY, distinceZ);
    glRotatef       (60.0, 1,0,0);
    glRotatef       (zRotated, 0,0,1);
    glutSolidSphere (5.0, 20, 50);
    glPopMatrix ();
    glColor3f(1, 0, 0);
}

void reshapeFunc (int x, int y) { //改變視窗重新繪製
    //glEnable(GL_DEPTH_TEST); //3d 模式要開啟
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(75, 1, 1, 2000);
//  glTranslatef    (0.0, 0.0, -15.0); //turn to 3d
    //沿著 z 軸平移
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(0, -200, 200, 0, 0, 0, 0, 0, 1);

//繪製紅色的“太陽”
    sun();
//繪製藍色的“地球”
    earth();
```

```
    //繪製黃色的“月亮”
        moon();


        glFlush();
        glutSwapBuffers();
    }
    void Keyboard(unsigned char key, int x, int y) {
        switch (key) {
            case 'w':
                distinceZ+=1;
                break;
            case 's':
                distinceZ-=1;
                break;
            case 'a':
                distinceX-=1;
                break;
            case 'd':
                distinceX+=1;
                break;
            case 'o':
                distinceY+=1;
                break;
            case 'p':
```

```
            distinceY-=1;
            break;
        case 27:
            glDisable(GL_LIGHT0);
            glDisable(GL_LIGHTING);
            glDisable(GL_DEPTH_TEST);
//          glutDestroyWindow(WinNumber);
            exit(0);
            break;
    }
    glutPostRedisplay();              //令視窗重繪
}

void myDisplay(void) {
    glEnable(GL_DEPTH_TEST); //3d 模式要開啟
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glMatrixMode(GL_PROJECTION); //投影模式
    glLoadIdentity();
    gluPerspective(75, 1, 1, 2000);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(0, -200, 200, 0, 0, 0, 0, 0, 1);

//繪製紅色的“太陽”
```

```
    sun();
    block();
//繪製藍色的“地球”
    earth();
//繪製黃色的“月亮”
    moon();

    if(trigger%2==0) {    //每 2tick 變化一次
        for(int i=0; i<30/4; i++) {
            glPushMatrix ();
            star(i);
            glPopMatrix ();
        }
    }
    trigger++;

    glFlush();
    glutSwapBuffers();
}

void myIdle(void) {
    Sleep(50);
    ++day;
    if (day >= 360)
```

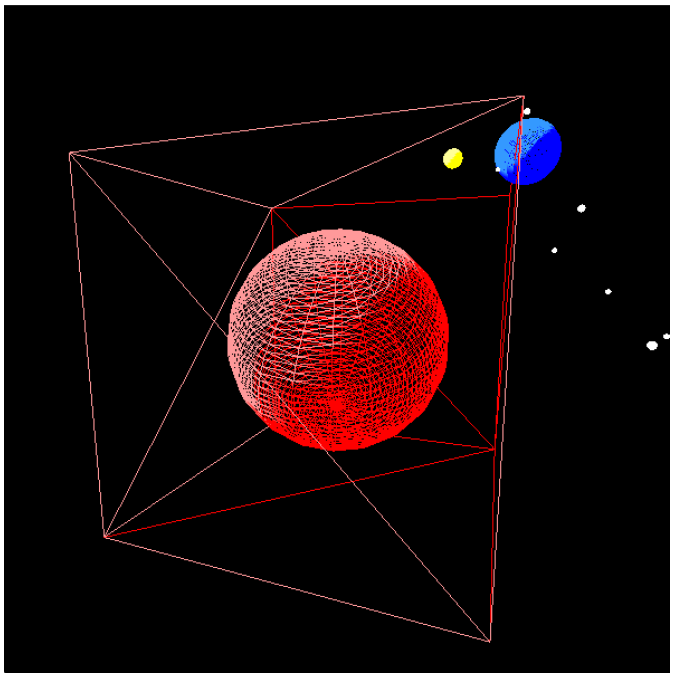
```
        day = 0;
    zRotated += 0.3;
    glutPostRedisplay();
    myDisplay();
}
void SetLightSource() {
    float light_ambient[]  = { 39,39,39,1};
    float light_diffuse[]  = { 50.0, 2.0, 1.0, 1.0};
    float light_specular[] = { 1,3,3,1};

    glEnable(GL_LIGHTING);                                  //開燈

    // 設定發光體的光源的特性
    glLightfv( GL_LIGHT0, GL_AMBIENT, light_ambient);       //環境光(Ambient Light)
    glLightfv( GL_LIGHT0, GL_DIFFUSE, light_diffuse);       //散射光(Diffuse Light)
    glLightfv( GL_LIGHT0, GL_SPECULAR,light_specular);      //反射光(Specular Light)

    glLightfv( GL_LIGHT0, GL_POSITION,light_position);      //光的座標

    glEnable(GL_LIGHT0);
    glEnable(GL_COLOR_MATERIAL);
```

```
    //glEnable(GL_DEPTH_TEST);                                    //啟動
深度測試
}

void SetMaterial() {
    float material_ambient[]  = { 0.2, 0.2, 0.2, 1.0};
    float material_diffuse[]  = { 0.3, 0.3, 0.3, 1.0};
    float material_specular[] = { 0.2, 0.2, 0.2, 1.0};

    glMaterialfv( GL_FRONT, GL_AMBIENT,  material_ambient);
    glMaterialfv( GL_FRONT, GL_DIFFUSE,  material_diffuse);
    glMaterialfv( GL_FRONT, GL_SPECULAR, material_specular);
}
void Mouse(int button, int state, int x, int y){
   if(state){
      record_x += x - old_rot_x;
      record_y += y - old_rot_y;

      rot_x = 0;   //沒有歸零會有不理想的結果
      rot_y = 0;
   }else{
      old_rot_x = x;
      old_rot_y = y;
   }
```

```
}
void MotionMouse(int x, int y){
   rot_x = x - old_rot_x;
   rot_y = y - old_rot_y;
   glutPostRedisplay();
}

int main(int argc, char *argv[]) {
   srand(time(0));

   glutInit(&argc, argv);
   glutInitWindowPosition(200, 100);
   glutInitWindowSize(600, 600);
   glutCreateWindow("Ch06.Example");
   glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE);
   glPolygonMode(GL_FRONT_AND_BACK, GL_LINE); //change view
in order to 3d performance
   glutDisplayFunc(&myDisplay);

   SetLightSource();
   SetMaterial();

   glutKeyboardFunc( Keyboard);
   glutMouseFunc    ( Mouse );
```

```
    glutMotionFunc  ( MotionMouse );
    glutReshapeFunc (reshapeFunc);
    glutIdleFunc(&myIdle);


//glutIdleFunc(&idleFunc);
    glutMainLoop();
    return 0;
}
```

執行結果：



十、應用說明：

使用到重要函數：

(1) glutMouseFunc(mouse);設定滑鼠動作的處理函式，本程式中通過按左鍵右鍵滾輪來進行不同軸的旋轉。

(2) glutKeyboardFunc(keys);設定鍵盤動作的處理函式，本程式中可以按 WASDOP 來移動星球位置(相對攝像機位置移動)。

(3) MotionMouse：計算滑鼠移動量，用來記算旋轉畫面的角度。

(4) glutSolidSphere (1, 20, 50)：GLUT 提供的基礎函數，用於繪製圓球。預設都繪製到(0,0,0)，再自行移動。參數 2、3 為經線與緯線的數量(越多越密集，越像實體球)。

(5) glTranslatef(150, 0.0f, 0.0f)：移動繪製位置。

(6) glRotatef(zRotated*2.0, 0,0,1)：以旋轉軸為基準，旋轉目標物體。

主要變動有以下幾項：

1. 新增多顆星球，可以觀察不同位置、大小的星球，受到光照的位置不盡相同。
2. 由於星球大小不同，使用相同數量的線條繪製。可以明顯看到小顆星球較為密實，受光照效果較好。大顆星球則較為粗糙。

十一、參考資料：

均為公開資料，詳細請參考報告投影片。