Computer graphic report 0613457 陳韋霖

第一部分：整體架構

```
void display();
void reshape(int _width, int _height);
void keyboard(unsigned char key, int x, int y);
void idle();
void drawSphere(double r, int lats, int longs,float R,float G,float B);//draw sphere and set color
void lighting();
void setcolor(float R,float G,float B);//setcolor function
void setRevolution(float r, float degree);//set revolution matrix

int width = 400, height = 400;
float X = 1; //default degree value, you can adjust it
GLfloat Y = 0.5; //default radius value, you can adjust it

int degree=0;//increase in idle(), from 0~359
bool oflag=true, pflag=true;//flag for rotating or stop, default set to true
```

函式

除了 main() 和預設就有的函式外，另外新增了三個函式

`drawSphere()`：用來畫地球和八面體

`setcolor()`：因為 `lighting()`會影響上色，所以另外設定顏色的函式

`setRevolution()` ：利用 `glTranslatef()`搭配平面圓座標的數學式計算來實現公轉

另外，也在 `keyboard()`中實作了按 O 鍵和 P 鍵改變狀態，並在 `idle()`中實作角度 X 的增加，用來實現自轉和公轉。

變數

關於變數的部分，詳見上面截圖註解。

第二部分：實現自轉和公轉

設置完基本 MVP transform 後，首先在畫面中央畫太陽，接著

`glPushMatrix()`

畫地球（或八面體），設定自轉函式，做 23.5 度旋轉，接著設定公轉。

`glPopMatrix()`

`glPushMatrix()`

畫月球，設定月球自轉，平移出去到繞地球公轉的半徑，繞地球公轉的半徑旋轉，最後設定公轉（讓月球跟著地球對太陽公轉）

`glPopMatrix()`

`glPushMatrix()`

畫地球和八面體的自轉軸（cylinder），因為 `gluCylinder()` 畫完後位置是錯的，所以先把軸移到畫面中間，將軸轉到 y 軸上，設定軸的傾斜角度 23.5，最後跟著地球對太陽公轉。

`glPopMatrix()`

第三部分： `drawSphere()` 詳細解釋

利用半徑和角度去計算球座標，得出頂點，然後使用這些頂點搭配 `glBegin(GL_QUAD_STRIP)` 及 `glNormal3f` 和 `glVertex3f` 函式去實際畫出立體球體。詳細角度計算和球座標如下：

角度表示：

Θ = 2 * pi * (slice_step / slice)

```
double theta=2 * M_PI * (double)(j - 1)/slice;
```

Φ = (pi / 2) – pi * (stack_step / stack)

```
double phy1= M_PI * ( 0.5 + (double)i/stack );
```

其中 slice_step 和 stack_step 是指 for 迴圈的 index

球座標：

$$
\begin{aligned}
x &= (r \cdot cos\phi) \cdot cos\theta \\
y &= (r \cdot cos\phi) \cdot sin\theta \\
z &= r \cdot sin\phi
\end{aligned}
$$