國立臺南大學資訊工程學系

Computer Graphics

第三次作業

題目 : 3D Graphics
Input and Interaction
Transformation

班級 ： 資工三

姓名 ： 卓楚庭

學號 ： S11159020

目錄

- 作業說明----------P.3
- 程式設計環境架構----------P.4
- Function 展示----------P.6
- 執行成果展示----------P.12
- 心得----------P.13

作業說明

1. 題目:

實作 Sierpinski Gasket in a Tetrahedron，並添加旋轉相機與物體角度之功能

2. Requirements :

- 3D Gasket – Regular tetrahedron with volume subdivision
- Input Devices – Mouse
  - Trigger the menu by pressing the right mouse button
  - Press the left mouse button and move the mouse to change θ and φ according to changes in x and y.
  - Press both the left and right mouse buttons and move the mouse to change r according to changes in x and/or y
- Input Devices – Keyboard
  - Press the ‘q’ or ‘Q’ to quit the program
- Pop Menu – Make two pop menus in main menu and exit the program in the main menu.
  - One of the pop menus in main menu is to select and change the subdivision level of the displaying tetrahedron.
  - Another one is to select which direction the tetrahedron rotate in. It can rotate automatically according to the x, y and z axes in clockwise or counter-clockwise. The automatically rotation can be stopped by press the “stop” button in this submenu as well.
- Initial Subdivision Level = 0
- The window title is your Student ID
- Locate the camera position in spherical coordinates and orient the camera to the sphere.

程式設計環境架構

1. 程式語言

C++ in MS Windows 11

2. 程式開發工具

Microsoft Visual Studio

3. 電腦硬體

CPU: Intel i5-1135G7 ,

Main Memory: 16GB LPDDR4X,

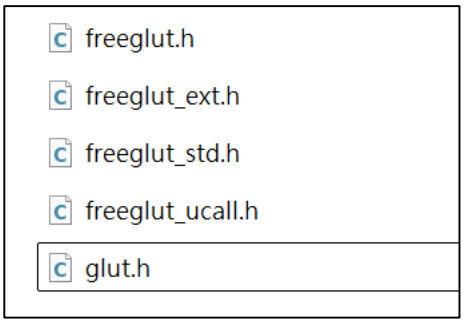
SSD: PCIe 512GB

4. 編譯器

```
g++.exe (x86_64-win32-seh-rev0, Built by MinGW-W64 project) 8.1.0
Copyright (C) 2018 Free Software Foundation, Inc.
```

5. GLUT 函式庫名稱與版本

Freegult 3.6.0



OpenGL 3.2

上個月	
glut.dll	2024/10/20 14:29
glut.h	2024/10/20 14:29
glut.lib	2024/10/20 14:29
glut32.dll	2024/10/20 14:29
glut32.lib	2024/10/20 14:29

Function 展示

1. Draw a tetrahedron

三角錐中共有 4 個點形成 6 條邊，利用二維陣列存取 6 條邊個別的中點座標位置(x, y, z)。以找出每條邊之中點的方式切割圖形，將切割出的中點座標位置存進 mid 陣列之中，以 Recursion 的方式切割直到 Base case (n = 0)，當 n = 0 時則直接顯示未切割過的 tetrahedron。

```
// Recursive function to subdivide the tetrahedron
void divide_tetra(GLfloat* a, GLfloat* b, GLfloat* c, GLfloat* d, int m) {
    // triangle subdividion using vertex numbers //
    GLfloat mid[6][3]; // Array to store midpoints of edges
    int j;

    if (m > 0) {
        for (j = 0; j < 3; j++) {
            mid[0][j] = (a[j] + b[j]) / 2;
            mid[1][j] = (a[j] + c[j]) / 2;
            mid[2][j] = (a[j] + d[j]) / 2;
            mid[3][j] = (b[j] + c[j]) / 2;
            mid[4][j] = (c[j] + d[j]) / 2;
            mid[5][j] = (b[j] + d[j]) / 2;
        }

        divide_tetra(a, mid[0], mid[1], mid[2], m - 1);
        divide_tetra(mid[0], b, mid[3], mid[5], m - 1);
        divide_tetra(mid[1], mid[3], c, mid[4], m - 1);
        divide_tetra(mid[2], mid[5], mid[4], d, m - 1);
    }
    else {
        tetra(a, b, c, d);
        // draw triangle at end of recursion //
    }
}
```

```
void triangle(GLfloat* a, GLfloat* b, GLfloat* c) {
    // display one triangle //
    glVertex3fv(a);
    glVertex3fv(b);
    glVertex3fv(c);
}

// Function to draw a tetrahedron using four triangles with a different color
void tetra(GLfloat* a, GLfloat* b, GLfloat* c, GLfloat* d) {
    glColor3f(1.0, 0.753, 0.796); //Pink
    triangle(a, b, c);
    glColor3f(0.486, 0.988, 0.0); //grass green
    triangle(a, c, d);
    glColor3f(0.678, 0.847, 0.902); //light blue
    triangle(a, d, b);
    glColor3f(1.0, 0.647, 0.0); //Orange
    triangle(b, d, c);
}
```

2. Rotate the Camera

將相機設置於球體座標(γ, θ, φ)中，將(γ, θ, φ)轉換成 x-y 座標的(x, y, z)，基於新計算出的座標值更新相機位置，以 glutLookAt()設置參數調整視角。

```
// Function to update camera position in spherical coordinates
void updateCamera() {
    // Convert spherical to Cartesian coordinates
    GLfloat eyeX = r * sin(theta) * cos(phi);
    GLfloat eyeY = r * sin(theta) * sin(phi);
    GLfloat eyeZ = r * cos(theta);

    // Set the camera view
    gluLookAt(eyeX, eyeY, eyeZ, // Camera position
        0.0f, 0.0f, 0.0f, // Look-at point (origin)
        0.0f, 1.0f, 0.0f); // Up vector
}
```

3. Rotate the Tetrahedron

若當時錐體不為停止旋轉的狀態，則判斷當前的固定軸，以每次 0.05f 的速度逐次增加或減少個別的旋轉角度，最後以 glutPostRedisplay()重新繪製，達到持續轉動的效果。

```
void rotateTetrahedron() {
    if (rotationDirection != 0) {
        // Increment the cumulative rotation for the fixed axis
        switch (fixedAxis) {
        case 'x':
            cumulativeRotationX += rotationDirection * 0.05f;
            break;
        case 'y':
            cumulativeRotationY += rotationDirection * 0.05f;
            break;
        case 'z':
            cumulativeRotationZ += rotationDirection * 0.05f;
            break;
        }
        glutPostRedisplay();
    }
}
```

4. Trigger by mouse button

判斷滑鼠左鍵是否為按下的狀態，若是則以(x, y)記錄當前滑鼠的位置於變數中。

```
// Mouse motion function
void mouse(int button, int state, int x, int y) {
    if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN) {
        leftButtonPressed = true;
    }

    lastMousePosX = x;
    lastMousePosY = y;
}
```

5. Drag the Mouse

按住左鍵並拖移滑鼠去改變相機視角，以當前滑鼠位置(x, y)與按下左鍵時所記錄的位置相減，計算出 x 和 y 的變化量，再以 x 和 y 之變化量個別在垂直移動和水平移動中調整 θ 和 φ，最後以當下滑鼠的位置(x, y)更新 lastMousePosition 並以 glutPostRedisplay()重新繪製。

```
// Function to handle mouse movement while dragging
void motion(int x, int y) {
    int deltaX = x - lastMousePosX;
    int deltaY = y - lastMousePosY;

    if (leftButtonPressed) {
        // Adjust theta and phi based on mouse movement
        theta -= deltaY * 0.005f; // Vertical movement adjusts theta
        phi -= deltaX * 0.005f;   // Horizontal movement adjusts phi

        // Clamp theta to avoid flipping
        if (theta < 0.01f) theta = 0.01f;
        if (theta > M_PI - 0.01f) theta = M_PI - 0.01f;
    }

    lastMousePosX = x;
    lastMousePosY = y;

    glutPostRedisplay(); // Redraw the scene
}
```

6. Zoom by Mouse wheel

以滑鼠滾輪調整相機位置的 γ，實現放大與縮小視角。若滾輪向上滾動即增加 ZoomFactor；反之，滾輪向下滾動則減少 ZoomFactor，設定縮放的上下限，最後基於新的 ZoomFactor 以 glutPostRedisplay()重新繪製。

```
void mouseWheel(int wheel, int direction, int x, int y) {
    // Adjust zoom factor based on wheel scroll direction
    if (direction > 0) {
        zoomFactor += 0.05f; // Scroll up: Zoom in
    }
    else if (direction < 0) {
        zoomFactor -= 0.05f; // Scroll down: Zoom out
    }

    // Clamp zoom factor to avoid extreme zooming
    if (zoomFactor < 0.1f) zoomFactor = 0.1f; // Minimum zoom
    if (zoomFactor > 10.0f) zoomFactor = 10.0f; // Maximum zoom

    glutPostRedisplay(); // Redraw the scene
}
```

7. Create a Menu

Menu 是利用回傳 id 的方式判斷點選該欄位後需要執行的動作。先 Create 出各 Entry 的 Name 和預設回傳的 id 值，利用點選滑鼠右鍵呼叫出 menu，點選滑鼠左鍵則取消顯示 menu。

```c
void createMenu() {
    int level = glutCreateMenu(sub);
    glutAddMenuEntry("0", 1);
    glutAddMenuEntry("1", 2);
    glutAddMenuEntry("2", 3);
    glutAddMenuEntry("3", 4);

    int axis = glutCreateMenu(ax);
    glutAddMenuEntry("X", 1);
    glutAddMenuEntry("Y", 2);
    glutAddMenuEntry("Z", 3);

    int dir = glutCreateMenu(direct);
    glutAddMenuEntry("Stop", 1);
    glutAddMenuEntry("Clockwise", 2);
    glutAddMenuEntry("Counter-Clockwise", 3);

    int rotate = glutCreateMenu(NULL);
    glutAddSubMenu("Axis", axis);
    glutAddSubMenu("Direction", dir);

    int menu = glutCreateMenu(main_menu);
    glutAddSubMenu("Subdivision Level", level);
    glutAddSubMenu("Rotation", rotate);
    glutAddMenuEntry("Exit", 1);

    glutDetachMenu(GLUT_LEFT_BUTTON);
    glutAttachMenu(GLUT_RIGHT_BUTTON);
}
```

Main menu 中若回傳的 id 為 1，則結束程式。

```c
void main_menu(int id) {
    if (id == 1) {
        exit(0);
    }
}
```

第一個 Submenu(sub(id))以回傳的 id 值決定新的 n 值(tetrahedron 要切割幾層)。n 值被更新後再重新畫出新的三角錐。(這裡是將原本 display function 中的內容複製過來，重新繪製圖形)

```c
void sub(int id) {
    switch (id) {
    case 1:
        n = 0;
        break;
    case 2:
        n = 1;
        break;
    case 3:
        n = 2;
        break;
    case 4:
        n = 3;
        break;
    default:
        n = 0;
        break;
    }

```

```c
    // display()
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glutSwapBuffers();
    glBegin(GL_TRIANGLES);

    divide_tetra(v[0], v[1], v[2], v[3], n);

    glEnd();
    glFlush();
}
```

第二個 Submenu 再另外延伸兩個 submenus，決定錐體旋轉的方向。

a. 以回傳值決定新的旋轉方向為(-1, 1, 0)，決定重新繪製的錐體該由逆時針方向、順時針方向旋轉或停止旋轉

b. 以回傳值決定新的固定軸(x, y, z)，更改當前錐體應該以固定哪一個軸去做自轉。

```
void direct(int id) {
    switch (id) {
    case 1: // Stop
        rotationDirection = 0; // Stop automatic rotation
        glutIdleFunc(NULL); // Disable idle function
        break;
    case 2: // Clockwise
        rotationDirection = 1; // Enable clockwise rotation
        glutIdleFunc(rotateTetrahedron); // Enable idle function
        break;
    case 3: // Counter-Clockwise
        rotationDirection = -1; // Enable counter-clockwise rotation
        glutIdleFunc(rotateTetrahedron); // Enable idle function
        break;
    default:
        rotationDirection = 0; // Stop automatic rotation
        glutIdleFunc(NULL); // Disable idle function
        break;
    }
}
```

```
// Submenu for rotation axis
void ax(int id) {
    switch (id) {
    case 1: // Fix rotation on X-axis
        fixedAxis = 'x';
        break;
    case 2: // Fix rotation on Y-axis
        fixedAxis = 'y';
        break;
    case 3: // Fix rotation on Z-axis
        fixedAxis = 'z';
        break;
    }
    glutIdleFunc(NULL); // Disable automatic rotation
    glutPostRedisplay(); // Redraw the scene
}
```

8. Trigger by Keyboard

判斷鍵盤回傳的值，若按下‘Q‘ 或‘q‘ 會結束程式，按下其他按鍵則不做任何動作。

```
void key(unsigned char k, int x, int y) { // keyboard trigger
    switch (k) {
    case 'Q':
        exit(0);
        break;
    case 'q':
        exit(0);
        break;
    default:
        break;
    }
}
```

9. Main Function

```
int main(int argc, char** argv) {

    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowSize(500, 500);
    glutInitWindowPosition(0, 0);
    glutCreateWindow("S11159020");

    createMenu();

    glutKeyboardFunc(key);
    glutReshapeFunc(Reshape);
    glutDisplayFunc(display);
    glutMouseFunc(mouse);
    glutMotionFunc(motion);   // Handle mouse motion while dragging
    glutMouseWheelFunc(mouseWheel);

    init();

    glutMainLoop();
    return 0;
}
```

執行成果展示

https://www.youtube.com/watch?v=1qmahte6PvU

心得

延續上一份作業，這次額外新增了相機視角與轉變視角的功能，自己對球體座標不熟悉導致在轉換到卡式座標時遇到了一點困難，難以單純用想像的方式去模擬相機與物體之間的相對關係，以及應該會在轉動後呈現何種效果，也因為沒有弄懂作業說明的細項，造成實作出的結果與助教預期檢查到的成果有落差，驗收當天聽完解釋後才趕緊把缺失的改變相機視角之功能補上。這次作業所探討的主體很有趣卻也覺得很抽象。