# 國立臺南大學資訊工程學系

# Computer Graphics

## 第四次作業

## 題目 : 3D Graphics Input and Interaction Transformation Shading

班級 ： 資工三

姓名 ： 卓楚庭

學號 ： S11159020

# 目錄

# 作業說明

## 1. 題目:

實作 Sierpinski Gasket in a Tetrahedron，添加旋轉相機與物體角度之功能，並增加至少兩個光源呈現出陰影效果。

## 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 THREE pop menus in main menu and exit the program in the main menu.
  - One is to select and change the subdivision level of the displaying tetrahedron.
  - Another 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.
  - The other is lighting control. It can turn on/off the lights separately in the submenu. The initialization of lighting control is "On".
- 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,

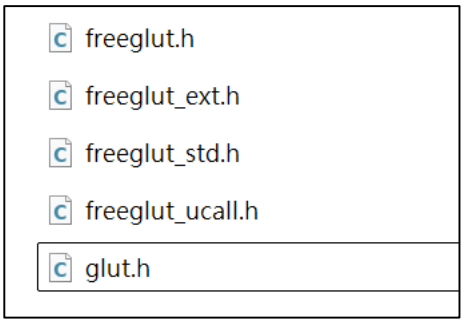
SSD: PCIe 512GB

## 4. 編譯器

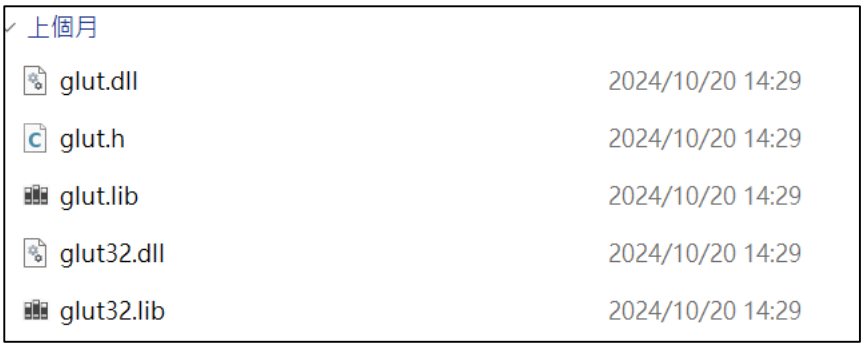
```
g++.exe (x86_64-win32-seh-rev0, Built by MinGW-W64 project) 8.1.0
Copyright (C) 2018 Free Software Foundation, Inc.
```

## 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 // // creating the normal vector//
    GLfloat normal[3];
    calculateNormal(a, b, c, normal);
    glNormal3fv(normal);
    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);
}
```

計算出兩邊向量，以兩條邊向量利用外積求出法向量，最後將法向量正規化，長度為 1。

```
void calculateNormal(GLfloat* a, GLfloat* b, GLfloat* c, GLfloat* normal) {
    // Calculate two edge vectors
    GLfloat u[3] = { b[0] - a[0], b[1] - a[1], b[2] - a[2] };
    GLfloat v[3] = { c[0] - a[0], c[1] - a[1], c[2] - a[2] };

    // Compute the cross product of u and v
    normal[0] = u[1] * v[2] - u[2] * v[1]; // x component
    normal[1] = u[2] * v[0] - u[0] * v[2]; // y component
    normal[2] = u[0] * v[1] - u[1] * v[0]; // z component

    // Normalize the resulting normal vector
    GLfloat length = sqrt(normal[0] * normal[0] + normal[1] * normal[1] + normal[2] * normal[2]);
    if (length != 0) {
        normal[0] /= length;
        normal[1] /= length;
        normal[2] /= length;
    }
}
```

## 2. Rotate the Camera

將相機設置於球體座標( $\gamma$ , $\theta$ , $\varphi$ )中，將( $\gamma$ , $\theta$ , $\varphi$ )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 $\theta$ 和 $\varphi$，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 $\gamma$，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。

```
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 light0 = glutCreateMenu(l0);
    glutAddMenuEntry("On", 1);
    glutAddMenuEntry("Off", 2);
```

```
    int light1 = glutCreateMenu(l1);
    glutAddMenuEntry("On", 1);
    glutAddMenuEntry("Off", 2);

    int light = glutCreateMenu(NULL);
    glutAddSubMenu("Light-0", light0);
    glutAddSubMenu("Light-1", light1);

    int menu = glutCreateMenu(main_menu);
    glutAddSubMenu("Subdivision Level", level);
    glutAddSubMenu("Rotation", rotate);
    glutAddSubMenu("Lighting Control", light);
    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。

```
66  void main_menu(int id) {
67      if (id == 1) {
68          exit(0);
69      }
70  }
```

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)

```
72  void sub(int id) {
73      switch (id) {
74      case 1:
75          n = 0;
76          break;
77      case 2:
78          n = 1;
79          break;
80      case 3:
81          n = 2;
82          break;
83      case 4:
84          n = 3;
85          break;
86      default:
87          n = 0;
88          break;
89      }
90
```

```
91      // display()
92      glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
93      glutSwapBuffers();
94      glBegin(GL_TRIANGLES);
95
96      divide_tetra(v[0], v[1], v[2], v[3], n);
97
98      glEnd();
99      glFlush();
100 }
```

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
}
```

第三個 Submenu 再另外延伸兩個 submenus，決定開啟或關閉光源。

a. 以回傳值決定 LIGHT_0 新的狀態是開啟或關閉

b. 以回傳值決定 LIGHT_1 新的狀態是開啟或關閉

```
void l0(int id){
    switch (id) {
    case 1: // Turn Light On
        Light0 = true;
        glEnable(GL_LIGHT0);
        glEnable(GL_LIGHTING); // Re-enable lighting
        break;
    case 2: // Turn Light Off
        Light0 = false;
        glDisable(GL_LIGHT0);
        break;
    default:
        Light0 = true;
        glEnable(GL_LIGHT0);
        glEnable(GL_LIGHTING);
        break;
    }
    glutPostRedisplay();

};
```

```
void l1(int id){
    switch (id) {
    case 1: // Turn Light On
        Light1 = true;
        glEnable(GL_LIGHT1);
        glEnable(GL_LIGHTING); // Re-enable lighting
        break;
    case 2: // Turn Light Off
        Light1 = false;
        glDisable(GL_LIGHT1);
        break;
    default:
        Light1 = true;
        glEnable(GL_LIGHT1);
        glEnable(GL_LIGHTING);
        break;
    }
    glutPostRedisplay();
};
```

# 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. Display Function

a. 更新每次拖動滑鼠後新的滑鼠位置

b. 設置兩個光源的位置後，以 glLightfv()動態調整光源與相機的相對位置，不論如何改變相機角度，讓光線照射到物體的位置固定。

c. glScalef()調整每次滾動滑鼠滾輪，需要放大或縮小多少倍率

d. glRotatef()讓錐體基於當前固定軸做出自轉

```
void display() {
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity(); // Reset before applying transformations

    // Update the camera position
    updateCamera();

    // Dynamically update light positions relative to the camera
    GLfloat light0_pos[] = { -5.0f, 5.0f, 0.0f, 1.0f };
    GLfloat light1_pos[] = { 5.0f, 5.0f, 0.0f, 1.0f };
    glLightfv(GL_LIGHT0, GL_POSITION, light0_pos);
    glLightfv(GL_LIGHT1, GL_POSITION, light1_pos);

    // Apply zoomFactor to scale the entire scene
    glScalef(zoomFactor, zoomFactor, zoomFactor);

    glPushMatrix();
    // Apply cumulative rotation based on the fixed axis
    glRotatef(cumulativeRotationX, 1.0f, 0.0f, 0.0f); // X-axis rotation
    glRotatef(cumulativeRotationY, 0.0f, 1.0f, 0.0f); // Y-axis rotation
    glRotatef(cumulativeRotationZ, 0.0f, 0.0f, 1.0f); // Z-axis rotation


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

    glPopMatrix();
    glutSwapBuffers();
}
```

## 10. Initial Function

光源屬性的參數設置(位置、環境光、漫射光、鏡面反射光)與材質屬性的參數設置(環境光、漫射光、鏡面反射光、高光強度)

```
void init() {
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glMatrixMode(GL_MODELVIEW);
    glClearColor(0.0f, 0.0f, 0.0f, 1.0f); // set the color of background in white
    glEnable(GL_DEPTH_TEST); // Enable depth testing
    glEnable(GL_LIGHTING);  //Enable lighting

    // Enable color tracking for material properties
    glEnable(GL_COLOR_MATERIAL);
    glColorMaterial(GL_FRONT, GL_DIFFUSE); // Use vertex colors for diffuse material

    glShadeModel(GL_SMOOTH);

    // Light 1 properties
    GLfloat light1_pos[] = { -5.0f, 5.0f, 0.0f, 1.0f }; // Positioned at upper left
    GLfloat light1_ambient[] = { 0.1f, 0.1f, 0.1f, 1.0f };
    GLfloat light1_diffuse[] = { 0.8f, 0.8f, 0.8f, 1.0f };
    GLfloat light1_specular[] = { 1.0f, 1.0f, 1.0f, 1.0f };

    GLfloat mat_ambient[] = { 0.7f, 0.7f, 0.7f, 1.0f };
    GLfloat mat_diffuse[] = { 0.8f, 0.8f, 0.8f, 1.0f };
    GLfloat mat_specular[] = { 1.0f, 1.0f, 1.0f, 1.0f };
    GLfloat high_shininess[] = { 100.0f };

    glEnable(GL_LIGHT0); // Enable Light 0
    glLightfv(GL_LIGHT0, GL_POSITION, light1_pos);
    glLightfv(GL_LIGHT0, GL_AMBIENT, light1_ambient);
    glLightfv(GL_LIGHT0, GL_DIFFUSE, light1_diffuse);
    glLightfv(GL_LIGHT0, GL_SPECULAR, light1_specular);
```

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

    glColor3f(0.40, 0.61, 0.94);

    // Light 2 properties
    GLfloat light2_pos[] = { 5.0f, 5.0f, 0.0f, 1.0f }; // Positioned at (-2, 2, -2)
    GLfloat light2_ambient[] = { 0.1f, 0.1f, 0.3f, 1.0f };
    GLfloat light2_diffuse[] = { 0.5f, 0.5f, 1.0f, 1.0f };
    GLfloat light2_specular[] = { 1.0f, 1.0f, 1.0f, 1.0f };

    glEnable(GL_LIGHT1); // Enable Light 1
    glLightfv(GL_LIGHT1, GL_POSITION, light2_pos);
    glLightfv(GL_LIGHT1, GL_AMBIENT, light2_ambient);
    glLightfv(GL_LIGHT1, GL_DIFFUSE, light2_diffuse);
    glLightfv(GL_LIGHT1, GL_SPECULAR, light2_specular);
}
```

## 11. Main Function

```
318  int main(int argc, char** argv) {
319
320      glutInit(&argc, argv);
321      glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
322      glutInitWindowSize(500, 500);
323      glutInitWindowPosition(0, 0);
324      glutCreateWindow("S11159020");
325
326      createMenu();
327
328      glutKeyboardFunc(key);
329      glutReshapeFunc(Reshape);
330      glutDisplayFunc(display);
331      glutMouseFunc(mouse);
332      glutMotionFunc(motion);    // Handle mouse motion while dragging
333      glutMouseWheelFunc(mouseWheel);
334
335      init();
336
337      glutMainLoop();
338      return 0;
339  }
```

**執行成果展示**

**https://youtu.be/lCf-cB_EW9g**

# 心得

經過這幾次的作業，讓三角錐在 3D 的呈現有更多不同的變化，也讓我學習許多改善期末專案的技術，個人覺得最難的是前一次作業的球體座標轉換，這次的光影效果相對之下好操作許多，不過還是要考慮到相機、物體與光源三者之間的相對關係。圖學實作中要考慮到的細節太多了，常常是和同學討論或者經由助教提點，才得以真正理解題目想要的成果與自己所實作的結果兩者的差異之處，但也是因為藉由實作才能將課堂中所學到的技術做出實際運用。

# 參考文獻


[1] https://courses.minia.edu.eg/Attach/IT354Lec2.pdf
[2] https://hackmd.io/@jsyeh/2021graphics
[3] https://www.youtube.com/watch?v=ZbszezwNSZU&t=265s
[4] https://blog.csdn.net/fengyuzaitu_126_com/article/details/119459867