SOFT 356 Programming for Entertainment Systems-C2

1. interaction

1.1 Keystroke interaction:

F1: Easy

F2: Normal

F3: Hard

F4: Harder

TAB: Map off/on

W :Forward

A :Move to the left

S : back

D :Move to the right

ESC:Exit

1.2 open EXE：

Double click OpenGL.exe

2.program

2.1 The overall structure of the program:

bool Scene01();

bool Scene02();

bool Scene03();

The main body of the program is composed of Scene01,Scene02,Scene03, and its functions are as follows:

Scene01, rendering game initially enters the interface menu, in which you can select the difficulty of the game and press the key to explain.

Scene02, program theme, render sky box, render maze map, render mini-map;

Scene03, rendering game transition (next level);

2.2 Maze realization：

2.2.1 Draw a maze：

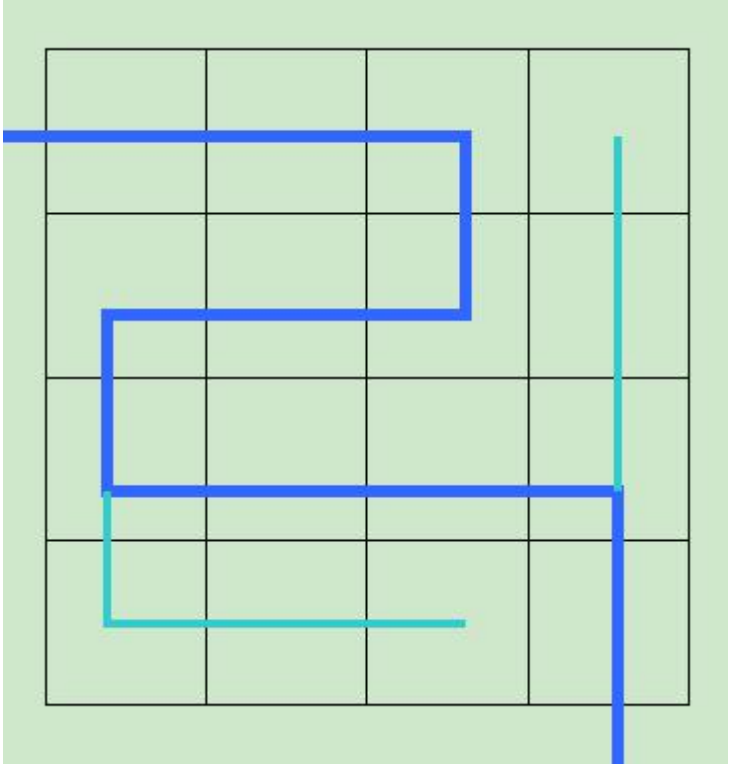
[^001]: txt

The maze map file is stored in txt, 1 for channel, 0 for closed; first, call SetupWorld () to read txt file, if 0, draw wall, 1 path; then call LoadVertices () to load vertices for collision detection; finally, call GenerateList () to generate display list storage, which is called later.

2.2.2Principle explanation：

After drawing the maze diagram, we encode it in the direction of XBI y. 1 is on, 0 is off, and finally forms two matrices of row*column. These two matrices represent the maze.
The number of rows can be arbitrary.
Save the matrix as a maze.txt file and load it in the program.

2.2.3Coding mode(Easy):



[^002]: The top view of the maze with difficulty for easy shows one route out of the palace in blue, and the other pathways in green.

A grid is used to represent the maze, blue lines indicate a certain route out of the palace, and green indicates other paths.
If the grid intersects the line, it is marked as 1, otherwise it is marked as zero.

1 is on, 0 is off, where the passage is 1, the wall is 0.
The x direction after coding is（row-1）*column，the y direction isrow*（column-1），to make the two matrices the same size, add a column 1 as the last column for the matrix in the x direction, and 1 as the last row for the matrix in the y direction

2.3 Draw a mini map

The mini-map is drawn in proportion to the original map, and the previous binary file is also read, but the data in the display list is directly called. The drawing method is as follows: if 0, draw the wall, 1 way.

```
if(Menu.bMazeMap)
    {
        glLoadIdentity();
        glTranslatef(g_cw*3/4+(0.25f-0.23f)*g_cw/2,(0.25f-0.23f)*g_ch/2,0.0f);
        maze.DrawMazeMap();
        Scene02_DrawPoint();
```

```
    }
```

2.3.1Marking

Get the coordinates of the characters in the world coordinate system, map them to the mini-map according to a certain proportion, and then call the function to draw.

```
void Scene02_DrawPoint()
{
    CVector3 vTemp = g_Camera.Position();
    CVector3 vNew;

    vNew.x = (vTemp.x * SCREEN_WIDTH/fMaze_Width+SCREEN_WIDTH/2)*0.23f;
    vNew.y = (-vTemp.z * SCREEN_LENGTH/fMaze_Length+SCREEN_LENGTH/2)*0.23f;
    //If the boundary is exceeded, do not render
    if(vNew.x<(-fMaze_Width/2*SCREEN_WIDTH/fMaze_Width+SCREEN_WIDTH/2)*0.23f
      ||vNew.x>(fMaze_Width/2*SCREEN_WIDTH/fMaze_Width+SCREEN_WIDTH/2)*0.23f
      ||vNew.y<(-fMaze_Length/2 * SCREEN_LENGTH/fMaze_Length+SCREEN_LENGTH/2)*0.23f
      ||vNew.y>(fMaze_Length/2 * SCREEN_LENGTH/fMaze_Length+SCREEN_LENGTH/2)*0.23f)
      return;

    //Use 2D projection
    glPushAttrib(GL_LIGHTING_BIT);
    glDisable(GL_LIGHTING);
    glDisable(GL_TEXTURE_2D);
    Start2DMode();
        glColor3f(1.0f,0.0f,0.0f);
        //Set the size of the point according to the complexity of the maze
        if(Menu.nLevel>NORMAL_ID) glPointSize(3);
        else glPointSize(5);
        glBegin(GL_POINTS);
            glVertex2f(vNew.x, vNew.y);
        glEnd();
    End2DMode();
    glEnable(GL_TEXTURE_2D);
    glPopAttrib();
}
```

2.5PositionCamera()

Write the incoming parameters, each row in the form of a vector, and assign values to the camera's position, orientation, direction in the world coordinate system:

```
void CCamera::PositionCamera(float positionX, float positionY, float positionZ,
                             float viewX,     float viewY,     float viewZ,
                             float upVectorX, float upVectorY, float upVectorZ)
{
    CVector3 vPosition  = CVector3(positionX, positionY, positionZ);
    CVector3 vView      = CVector3(viewX, viewY, viewZ);
    CVector3 vUpVector  = CVector3(upVectorX, upVectorY, upVectorZ);

    m_vPosition = vPosition;                    // Assign the position
    m_vView     = vView;                        // Assign the view
    m_vUpVector = vUpVector;                    // Assign the up vector
}
//For the position of the camera in the world coordinate system
float positionX, float positionY, float positionZ,
//For viewport orientation
 float viewX,     float viewY,     float viewZ,
 //Control the up direction of the camera
 float upVectorX, float upVectorY, float upVectorZ
```

2.6SetViewByMouse()

GetCursorPos (& mousePos) captures the XY coordinate information of the mouse and assigns the current coordinate information to the viewport orientation parameter of the camera, so as to achieve the purpose of following the mouse orientation.

```
void CCamera::SetViewByMouse()
{
    POINT mousePos;                                 // Window structure containing X
and Y
    float angleY = 0.0f;                            // The direction of looking up or down.
    float angleZ = 0.0f;                            // The required values for rotation around
the Y axis (left and right)

    // Get the mouse's current X,Y position
    GetCursorPos(&mousePos);

    // If the cursor is still in the middle, do not update the screen
    if( (mousePos.x == m_ScreenMiddleX) && (mousePos.y == m_ScreenMiddleY) ) return;

    // Set the mouse position to the middle of our window
    SetCursorPos(m_ScreenMiddleX, m_ScreenMiddleY);
```

```
    // Get the direction of mouse movement
    angleY = (float)( (m_ScreenMiddleX - mousePos.x) ) / 500.0f;
    angleZ = (float)( (m_ScreenMiddleY - mousePos.y) ) / 500.0f;
    m_currentRotX -= angleZ;

    // If the current rotation (in radians) is greater than 1.0, we want to cap it.
    if(m_currentRotX > 1.0f)
        m_currentRotX = 1.0f;
    // Check if the rotation is below -1.0, if so we want to make sure it doesn't continue
    else if(m_currentRotX < -1.0f)
        m_currentRotX = -1.0f;
    // Rotate the view around the position
    else
    {
        // To find the axis we need to rotate around for up and down
        // movements, we need to get a perpendicular vector from the
        // camera's view vector and up vector.    This will be the axis.
        CVector3 vAxis = Cross(m_vView - m_vPosition, m_vUpVector);
        vAxis = Normalize(vAxis);

        // Rotate around our perpendicular axis and along the y-axis
        RotateView(angleZ, vAxis.x, vAxis.y, vAxis.z);
        RotateView(angleY, 0, 1, 0);
    }
}
```

2.7RotateView()：

How to make the viewing angle rotate smoothly, the axis-angle rotation mode is used here:

```
void CCamera::RotateView(float angle, float x, float y, float z)
{
    CVector3 vNewView;

    // Get the view vector (facing direction)
    CVector3 vView = m_vView - m_vPosition;

    // Calculate the sine and cosine of the angle once
    float cosTheta = (float)cos(angle);
    float sinTheta = (float)sin(angle);

    // Find the new x position for the new rotated point
    vNewView.x    = (cosTheta + (1 - cosTheta) * x * x)              * vView.x;
```

```
	vNewView.x += ((1 - cosTheta) * x * y - z * sinTheta)	* vView.y;
	vNewView.x += ((1 - cosTheta) * x * z + y * sinTheta)	* vView.z;

	// Find the new y position for the new rotated point
	vNewView.y	= ((1 - cosTheta) * x * y + z * sinTheta)	* vView.x;
	vNewView.y += (cosTheta + (1 - cosTheta) * y * y)		* vView.y;
	vNewView.y += ((1 - cosTheta) * y * z - x * sinTheta)	* vView.z;

	// Find the new z position for the new rotated point
	vNewView.z	= ((1 - cosTheta) * x * z - y * sinTheta)	* vView.x;
	vNewView.z += ((1 - cosTheta) * y * z + x * sinTheta)	* vView.y;
	vNewView.z += (cosTheta + (1 - cosTheta) * z * z)		* vView.z;

	// Add the newly rotated vector to the position you want to set
	// The new rotated view of the camera.
	m_vView = m_vPosition + vNewView;
}
```

2.8 CheckCameraCollision()

The purpose of this function is to iterate through each triangle in the list and check whether the camera's sphere collides with it.
If so, we will not stop here.
We may have multiple conflicts, so it is important to check all conflicts.
Once the collision is found, we calculate the offset to move the sphere out of the collision plane.

```
void CCamera::CheckCameraCollision(CVector3 *pVertices, int numOfVerts)
{
	for(int i = 0; i < numOfVerts; i += 4)
	{
		CVector3 vTemp[4] = { pVertices[i], pVertices[i+1], pVertices[i+2],pVertices[i+3] };
		CVector3 vTriangle[3];
		for(int j=0; j<2; j++)
		{
			if(j==0)
			{ vTriangle[0] = vTemp[0]; vTriangle[1] = vTemp[1]; vTriangle[2] = vTemp[2]; };
			if(j==1)
			{ vTriangle[0] = vTemp[2]; vTriangle[1] = vTemp[3]; vTriangle[2] = vTemp[0]; };

			CVector3 vNormal = Normal(vTriangle);
			float distance = 0.0f;
			int classification = ClassifySphere(m_vPosition, vNormal, vTriangle[0], m_radius,
distance);
```

```
                if(classification == INTERSECTS)
                {
                    CVector3 vOffset = vNormal * distance;
                    CVector3 vIntersection = m_vPosition - vOffset;

                    if(InsidePolygon(vIntersection, vTriangle, 3) ||
                        EdgeSphereCollision(m_vPosition, vTriangle, 3, m_radius / 2))
                    {
                        vOffset = GetCollisionOffset(vNormal, m_radius, distance);
                        m_vPosition = m_vPosition + vOffset;
                        m_vView = m_vView + vOffset;
                    }
                }
            }
        }
}
```

3. Compared with other similar transactions, this program is too simple, with only four different difficulties, each with only one map.

I got the idea by playing a game like pushing boxes.

At first I wanted to do a 3D game of pushing boxes, and then I felt that I didn't want the boxes, which made me think of the maze.

I make myself different from the place where there is a beautiful sky box.

I started with a given project.

4. Video Report

https://youtu.be/nDHiudO0Zic