Computer Graphics 372

Assignment Number 8

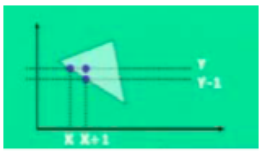
Zbuffer:Calculating Zdepth

NAVEEN
1601CS28

Z-Buffer: Calculating Z-depth algorithm)

0.1

Z-Buffer: Calculating Z-Depth



Let us assume that we know the depth or z-value for position (x,y). Updating z-value for every other positions using the z-buffer algorithm (already discussed in class), requires extensive computation. An alternate way to estimate z-values for every position in a polygon is to take help of the polygon filling algorithm.

From plane equation Ax + By + Cz + D = 0, depth at position (x, y) can be computed as:

$$z = (-Ax - By - D) / C$$

Incrementally across scan line (x+1, y), the z-value will be:

$$\begin{aligned} z' &= (-A(x+1) - By - D) / C \\ &= (-Ax - By - D) / C - A/C \\ &= z - A/C \end{aligned}$$

Incrementally between scan lines (x', y-1)

$$\begin{aligned} z' &= (-A(x') - B(y-1) - D) / C \\ \text{Using } x' &= x - 1/m \text{ (Refer Polygon filling algorithm)} \\ &= (-A(x - 1/m) - B(y-1) - D) / C \\ &= z + (A/m + B) / C \end{aligned}$$

And if the polygon has a vertical edge, then m = infinity, and the above equation reduces to

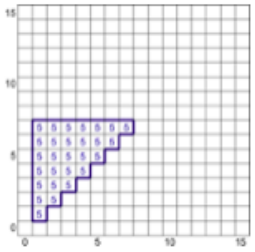
$$z' = z + B/C$$

So, by remembering these 3 terms -A/C, (A/m + B)/C and B/C we can compute the z-values of the corresponding pixels in a given scan line and for the next scan line, without going for extensive computations.

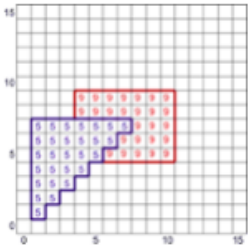
0.2

Depth (Z) based sorting of polygons in a Z-buffer.

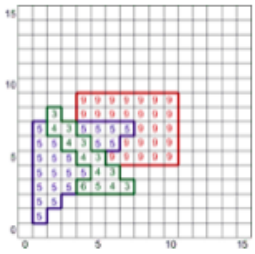
Here every polygon is given by its list of vertices (x,y,z), the z-value corresponds the depth of the polygon. We assume +ve z-direction as the viewing direction.



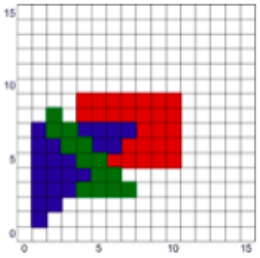
First polygon: (1, 1, 5), (7, 7, 5), (1, 7, 5)
9), (3, 9, 9)



Second polygon: (3, 5, 9), (10, 5, 9), (10, 9,



Third polygon: (2, 6, 3), (2, 3, 8), (7, 3, 3)



Final Z-Buffer

Listing 1: C++ code using listings

```
#include <iostream>
#include <cstring>
#include <cstdio>
#include <fstream>
#include <cstdlib>
#include <cmath>
#include <vector>
#include <list>
#include <algorithm>
#include <functional>
#include <GL/glut.h>
#include <GL/glu.h>
#include <GL/gl.h>

using namespace std;

void init()
{
    glClearColor(0.0, 0.0, 0.0, 0.0);
    glColor3f(1.0f,0.0f,0.0f);
    glPointSize(1.0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0.0, 500.0, 0.0, 500.0);
    glMatrixMode(GL_MODELVIEW);
}

GLfloat xRot = 0.0;
GLfloat yRot = 0.0;

void DrawCube()
{
    glClearColor(0.0f, 0.0f, 0.0f, 1.0f );
    glColor3f(0.0f, 1.0f, 0.0f);
    glShadeModel(GL_FLAT);
    glFrontFace(GL_CW);
```

```
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
glEnable(GL_DEPTH_TEST);
glPushMatri);

glRotatef(xRot, 1.0f, 0.0f, 0.0f);
glRotatef(yRot, 0.0f, 1.0f, 0.0f);

glBegin(GL_QUADS);

glColor3f(0.0,1.0,0.0);
glVertex3f( 50.0f, 50.0f,50.0f);
glVertex3f(-50.0f, 50.0f,50.0f);
glVertex3f(-50.0f,-50.0f,50.0f);
glVertex3f( 50.0f,-50.0f,50.0f);

glColor3f(0.0,0.0,1.0);
glVertex3f(50.0, 50.0,-50.0);
glVertex3f(50.0, 50.0, 50.0);
glVertex3f(50.0,-50.0, 50.0);
glVertex3f(50.0,-50.0,-50.0);

glColor3f(1.0,0.0,0.0);
glVertex3f( 50.0,-50.0,-50.0);
glVertex3f(-50.0,-50.0,-50.0);
glVertex3f(-50.0, 50.0,-50.0);
glVertex3f( 50.0, 50.0,-50.0);

glColor3f(0.0,1.0,1.0);
glVertex3f(-50.0, 50.0, 50.0);
glVertex3f(-50.0, 50.0,-50.0);
glVertex3f(-50.0,-50.0,-50.0);
glVertex3f(-50.0,-50.0, 50.0);

glColor3f(1.0,1.0,0.0);
glVertex3f( 50.0,50.0,-50.0);
glVertex3f(-50.0,50.0,-50.0);
glVertex3f(-50.0,50.0, 50.0);
glVertex3f( 50.0,50.0, 50.0);
```

```
    glColor3f(1.0,1.0,1.0);
    glVertex3f( 50.0,-50.0, 50.0);
    glVertex3f(-50.0,-50.0, 50.0);
    glVertex3f(-50.0,-50.0,-50.0);
    glVertex3f( 50.0,-50.0,-50.0);

    glEnd();
    glPopMatri);
    glutSwapBuffers();
}

void SpecialKeys(int key, int x, int y)
{
        if(key == GLUT_KEY_UP)
                xRot-= 5.0f;
        else if(key == GLUT_KEY_DOWN)
                xRot += 5.0f;
        else if(key == GLUT_KEY_RIGHT)
                yRot -= 5.0f;
        else if(key == GLUT_KEY_LEFT)
                yRot += 5.0f;
        else if(key > 356.0f)
                xRot = 0.0f;
        else if(key < -1.0f)
                xRot = 355.0f;
        else if(key > 356.0f)
                yRot = 0.0f;
        else if(key < -1.0f)
                yRot = 355.0f;
        else
                exit(0);
        glutPostRedisplay();
}

void KeyboardAction(unsigned char key, int x,int y)
{
        exit(0);
```

```
}

void ChangeSize(int w, int h)
{
        GLfloat nRange = 100.0f;
        if(h == 0)
        {
                h = 1;
        }
        glViewport(0, 0, w, h);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();

        if (w <= h)
        {
                glOrtho(-nRange, nRange, -nRange*h/w, nRange*h/w, -nRan
        }
        else
        {
                glOrtho(-nRange*w/h, nRange*w/h, -nRange, nRange, -nRan
        }
        glMatrixMode(GL_MODELVIEW);
        glLoadIdentity();
}


int main(int argc, char **argv)
{
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
        glutInitWindowSize(500, 500);
        glutInitWindowPosition(100, 100);
        glutCreateWindow("Z-Buffering");
    glViewport(0, 0, 500, 500);

        glutReshapeFunc(ChangeSize);
        glutKeyboardFunc(KeyboardAction);
        glutSpecialFunc(SpecialKeys);
```

```
        glutDisplayFunc(DrawCube);

        init();

        glutMainLoop();

        return 0;
}
```