Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Московский государственный технический университет
имени Н.Э. Баумана»
(МГТУ им. Н.Э. Баумана)

Факультет: Информатика и системы управления
Кафедра: Теоретическая информатика и компьютерные технологии

Лабораторная работа №3
«Моделирование»
по курсу: «Алгоритмы компьютерной графики»

Выполнил:
Студент группы ИУ9-41Б
Гречко Г.В.

Проверил:
Цалкович П.А.

Москва, 2023

Цели

Получение навыков компьютерного моделирования.

Задачи

Смоделировать объемную фигуру согласно заданному варианту - наклонная призма.

Основная теория

Модель – это абстрактное представление сущности реального мира – математическое моделирование физических, химических процессов и др.

Основные проблемы **Компьютерного моделирования**:

- данные о физических объектах не могут быть целиком введены в компьютер;
- необходимо априори ограничить объем хранимой информации об объекте;

Задача моделирования - найти вид модели, наилучшим образом отвечающий решаемой задаче.

В **компьютерной графике** используется **геометрическое моделирование** - моделирование объектов различной природы с помощью геометрических типов данных;

Типы представлений объектов для последующей визуализации

- Воксельное представление
- Конструктивная геометрия
- Каркасное представление
- Граничное представление
- Параметрическое задание поверхности
- ...

Практическая реализация

main.cpp

```
#include <GL/freeglut.h>
#include <GL/gl.h>
#include <GLFW/glfw3.h>

#include <cmath>
#include <iostream>

float delta_x = 0.5;
float delta_y = 0.5;

float theta = 0.61;
float phi = 0.78;

float theta1 = 0.61;
float phi1 = 0.78;
int VIEW_MODE = 4;

bool fill = false;

int N = 10;

using std::cos, std::sin;

```

```
void key(GLFWwindow *window, int key, int scancode, int action, int mods)
  {
  if (action == GLFW_PRESS || action == GLFW_REPEAT) {
    if (key == GLFW_KEY_ESCAPE) {
      glfwSetWindowShouldClose(window, GL_TRUE);
    } else if (key == GLFW_KEY_UP) {
      theta1 -= 0.1;
    } else if (key == GLFW_KEY_DOWN) {
      theta1 += 0.1;
    } else if (key == GLFW_KEY_LEFT) {
      phi1 += 0.1;
    } else if (key == GLFW_KEY_RIGHT) {
      phi1 -= 0.1;
    } else if (key == GLFW_KEY_Q) {
      fill = false;
    } else if (key == GLFW_KEY_W) {
      fill = true;
    } else if (key == GLFW_KEY_J) {
      N++;
    } else if (key == GLFW_KEY_K) {
      N--;
    }
  }
}

void DrawCube(GLfloat size) {
  float x, y, z;
  float l = 0.5;
  float a = M_PI * 2 / N;
  glBegin(GL_TRIANGLE_FAN);
  glColor3f(1.0f, 0.0f, 1.f);
  for (int i = -1; i < N; i++) {
    x = sin(a * i) * l;
    y = cos(a * i) * l;
    z = -size / 2;
    glColor3f(0.5f, i / 10.0, 0.5f);
    glVertex3f(x, y, z);
  }
  glEnd();

  glBegin(GL_TRIANGLE_FAN);
  for (int i = -1; i < N; i++) {
    x = sin(a * i) * l + delta_x;
    y = cos(a * i) * l + delta_y;
    z = size / 2;
    glColor3f(0.f, i / 10.0, 1.f);
    glVertex3f(x, y, z);
  }
  glEnd();

  glBegin(GL_QUADS);

  for (int i = -1; i < N; i++) {
    float x1 = sin(a * i) * l;
    float y1 = cos(a * i) * l;
    float x2 = sin(a * (i + 1)) * l;
    float y2 = cos(a * (i + 1)) * l;

    glColor3f(0.f, i / 10.0, 1.f);
    glVertex3f(x1, y1, -size / 2);
    glVertex3f(x1 + delta_x, y1 + delta_y, size / 2);
    glVertex3f(x2 + delta_x, y2 + delta_y, size / 2);
```

```
    glVertex3f(x2, y2, -size / 2);
  }

  glEnd();
}

void display(GLFWwindow *window) {
  glEnable(GL_DEPTH_TEST);
  glDepthFunc(GL_LESS);
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
  glMatrixMode(GL_MODELVIEW);
  glClearColor(0.9f, 0.9f, 0.9f, 1.0f);
  glPushMatrix();
  glLoadIdentity();

  glPolygonMode(GL_FRONT_AND_BACK, fill ? GL_FILL : GL_LINE);

  GLfloat m[4][4] = {{0.87, -0.09f, 0.98f, 0.49f},
                     {0.0f, 0.98f, 0.35f, 0.17f},
                     {0.5f, 0.15f, -1.7f, -0.85f},
                     {0.0f, 0.0f, 1.0f, 2.0f}};

  // GLfloat m_perspective[4][4] = {
  //     {1.f, 0.f, 0.f, 0.f},
  //     {0.f, 1.f, 0.f, 0.f},
  //     {0.f, 0.f, 1.f, 0.f},
  //     {-1 / x, -1 / y, -1 / z, 1.f}};

  GLfloat front_view[4][4] = {{1.f, 0.f, 0.f, 0.f},
                              {0.f, 1.f, 0.f, 0.f},
                              {0.f, 0.f, -1.f, 0.f},
                              {0.f, 0.f, 0.f, 1.f}};

  GLfloat side_view[4][4] = {{0.f, 0.f, -1.f, 0.f},
                             {0.f, 1.f, 0.f, 0.f},
                             {-1.f, 0.f, 0.f, 0.f},
                             {0.f, 0.f, 0.f, 1.f}};

  GLfloat top_view[4][4] = {{1.f, 0.f, 0.f, 0.f},
                            {0.f, 0.f, -1.f, 0.f},
                            {0.f, -1.f, 0.f, 0.f},
                            {0.f, 0.f, 0.f, 1.f}};

  GLfloat m_rotate[4][4] = {
      {cos(phi1), sin(theta1) * sin(phi1), sin(phi1) * cos(theta1), 0.f},
      {0.0f, cos(theta1), -sin(theta1), 0.f},
      {sin(phi1), -cos(phi1) * sin(theta1), -cos(phi1) * cos(theta1),
  ↪  0.f},
      {0.0f, 0.0f, 0.0f, 1.0f}};

  glMultMatrixf(&m_rotate[0][0]);

  DrawCube(0.8);

  glPopMatrix();

  glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
}

int main(int argc, char **argv) {
  if (!glfwInit())
    exit(1);
```

```
146
147   GLFWwindow *window = glfwCreateWindow(1280, 1280, "Lab 1", NULL, NULL);
148
149   if (window == NULL) {
150     glfwTerminate();
151     exit(1);
152   }
153
154   glfwMakeContextCurrent(window);
155   glfwSwapInterval(1);
156   glfwSetKeyCallback(window, key);
157
158   while (!glfwWindowShouldClose(window)) {
159     display(window);
160     glfwSwapBuffers(window);
161     glfwPollEvents();
162   }
163
164   glfwDestroyWindow(window);
165   glfwTerminate();
166   return 0;
167 }
```

Заключение

В ходе данной лабораторной работы были получены практические навыки по динамическому моделированию объемных изображений с помощью триангуляции, а так же получены навыки по динамического редактированиб отображаемых объектов.