**Министерство науки и высшего образования Российской Федерации**

**Пензенский государственный университет**

**Кафедра “Вычислительная техника”**

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовой работе

по курсу «Компьютерная графика и 3D моделирование»

на тему «Разработка программы построения и визуализации 3D объекта»

Выполнили студенты группы 23ВВВ4:

Зарубин Я.Д.\_\_\_\_

Герасимов К.А.\_\_\_\_

Принял:

к.т.н., доцент Финогеев А. А.\_\_\_\_

Пенза 2025

**Оглавление**

[1 Введение 4](#__RefHeading___401)

[2 Создание модели 5](#__RefHeading___402)

[4 Реализация видовых преобразований 9](#__RefHeading___403)

[7. Результаты работы 12](#__RefHeading___404)

[8. Вывод 16](#__RefHeading___405)

[9 Список использованной литературы 17](#__RefHeading___406)

[Приложение А 18](#__RefHeading___407)

[Приложение Б 20](#__RefHeading___408)

# **1 Введение**

В современном мире трехмерная графика играет важную роль в различных сферах, включая игровую индустрию, проектирование и научные исследования. Основной задачей при работе с 3D-объектами является их корректное отображение на экране. В рамках данной курсовой работы рассматривается подход к визуализации трехмерных моделей, представленных в виде вершин и полигонов, на основе данных из файлов формата OBJ. Этот формат широко применяется для хранения геометрической информации и поддерживается многими графическими приложениями. В исследовании анализируются теоретические аспекты 3D-графики, реализуются алгоритмы загрузки и обработки OBJ-файлов, а также рассматриваются методы отображения трехмерных объектов на экране.

# **2 Создание модели**

Для создания 3D-модели человека был использован графический редактор Blender.

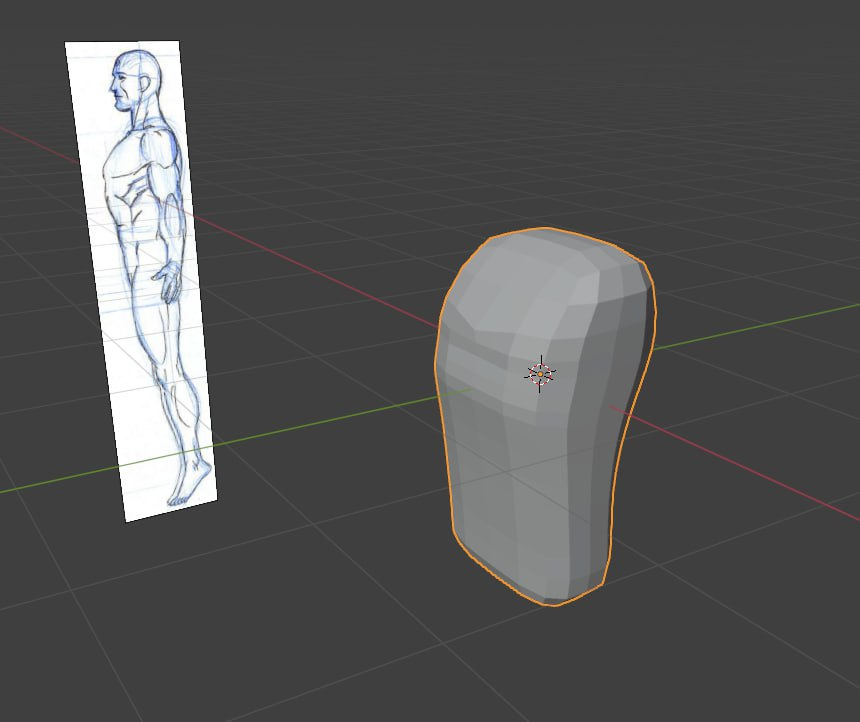


Рисунок 1 – Первый этап создания 3D модели человека в программе Blender

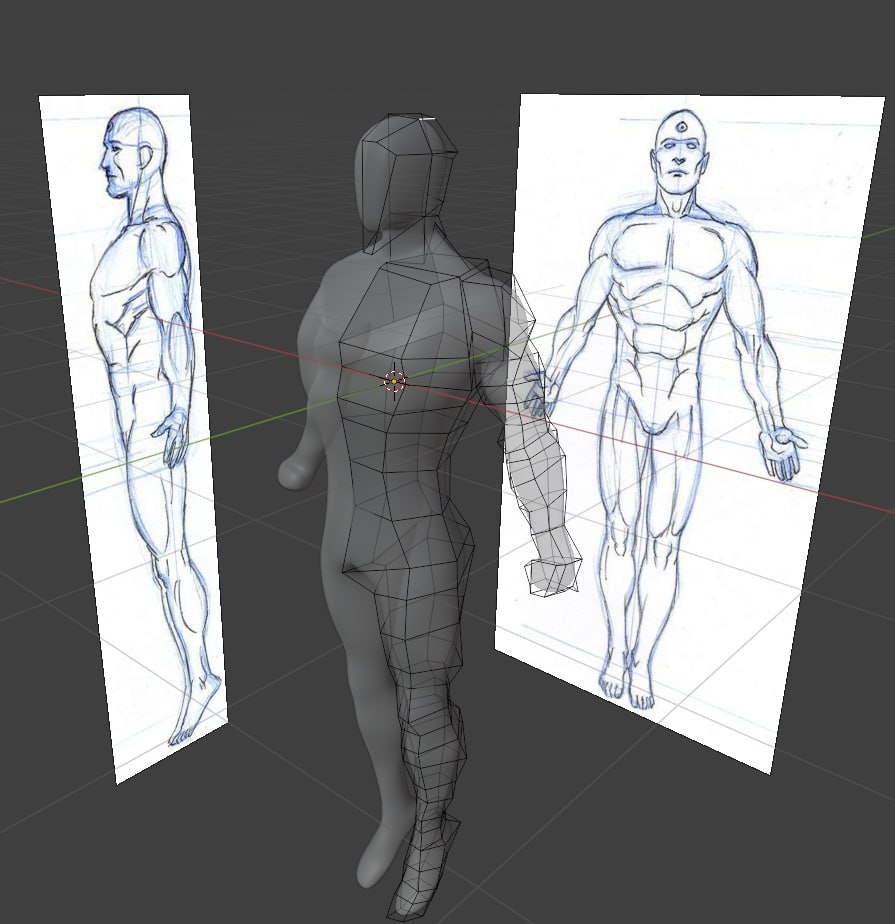


Рисунок 2 – Второй этап создания 3D модели человека в программе Blender с полигонами

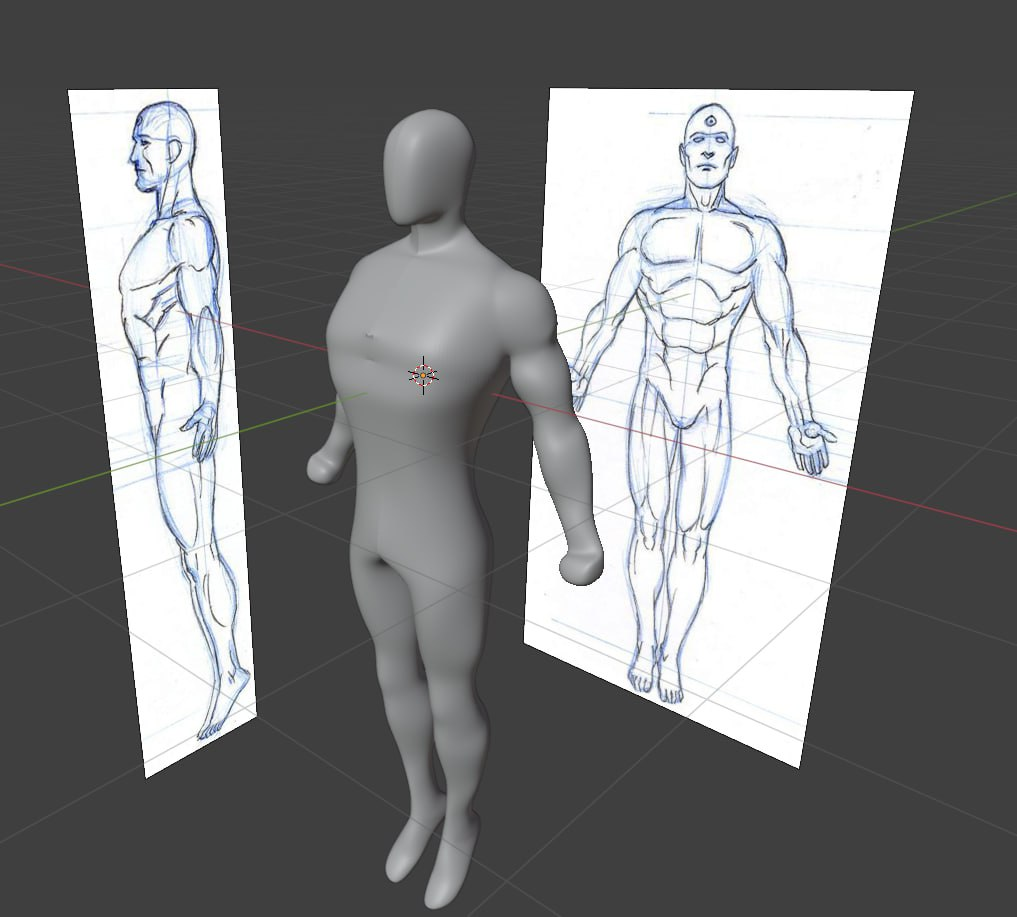


Рисунок 3 – Итоговый этап создания 3D модели человека в программе Blender

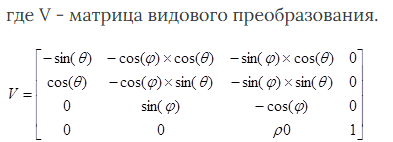
**3. Формат OBJ для представления трехмерных моделей**

Wavefront OBJ представляет собой текстовый стандарт хранения данных о трехмерных объектах. В его структуре используются следующие ключевые элементы описания геометрии: геометрические вершины (v) - задают координаты точек в трехмерном пространстве, текстурные координаты (vt) - определяют параметры наложения текстуры, нормали вершин (vn) - содержат информацию о направлении нормалей, полигональные грани (f) - описывают поверхности объектов через ссылки на вершины, текстурные координаты и нормали. Дополнительно файл может включать данные о группах объектов и материалах, однако минимально необходимый набор для визуализации модели составляют именно вершины и полигоны. Такой подход к организации данных позволяет эффективно хранить и передавать информацию о трехмерных объектах между различными графическими приложениями.

# **4 Реализация видовых преобразований**

Вид 3D-объекта на экране зависит от положения глаз наблюдающего. Большое значение имеет расстояние от глаз до объекта. Этому расстоянию будет обратно пропорционален «эффект перспективы», то есть на большом расстоянии, относительно размера объекта, параллельные линии объекта будут оставаться параллельными, но при приближении изображение начнёт искажаться, продолжения линий начнут пересекаться. Для получения перспективной проекции, нам необходимо исходное множество точек P(x, у, z) объекта (точки объекта задаются в мировых координатах) преобразовать в координаты точек Р'(Х, У) изображения. Предположим, что экран находится между объектом и глазом Е. Для каждой точки Р нашего объекта прямая линия РЕ пересекает экран в точке Р'. Для данного отображения необходимы 2 этапа: видовые и перспективные преобразования. Где видовые преобразования переносит систему мировых координат в систему видовых координат, а перспективные преобразования преобразуют точку Р в точку Р', переходя из трёхмерной системы видовых координат в двухмерную систему экранных координат. Для преобразований зададим точку наблюдения в сферических координатах ρ, θ, ϕ. Видовое преобразование может быть представлено в форме:

[Xe Ye Ze 1] = [Xw Yw Zw 1] x V,



**5. Реализация перспективных преобразований**

В процессе выполнения перспективных преобразований была определена точка наблюдения Q с видовыми координатами (0, 0, d), где d - положительное число. Данная конфигурация предполагает расположение плоскости проекции (экрана) на расстоянии d от начала координат, перпендикулярно оси z. В этой системе каждая точка P' проекции представляет собой результат пересечения линии зрения (соединяющей точку P трехмерного объекта и начало координат) с плоскостью экрана. Такая модель преобразования позволяет корректно отображать трехмерные объекты на двумерной плоскости с учетом перспективных искажений, что соответствует естественному восприятию пространства человеческим зрением.

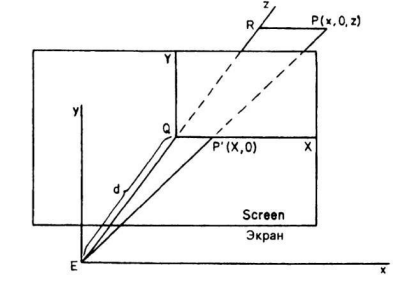
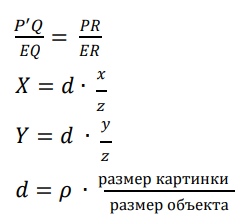


Рисунок 4 - Видовые координаты



**6. Реализация управления с использованием горячих клавиш**

Управление осуществляется с помощью следующих клавиш:

W — перемещает камеру вверх.

S — перемещает камеру вниз.

D — вращает объект по оси Z по часовой стрелке.

A — вращает объект по оси

Z против часовой стрелки.

E — приближает объект к наблюдателю.

Q — отдаляет объект от наблюдателя.

G — переключает анимацию.

### 7. Результаты работы

В ходе выполнения курсовой работы была создана программа на языке C++, отображающая на экране разработанную 3D-модель (см. рис. 1-3). Визуальные результаты работы программы представлены на рисунках 5–.

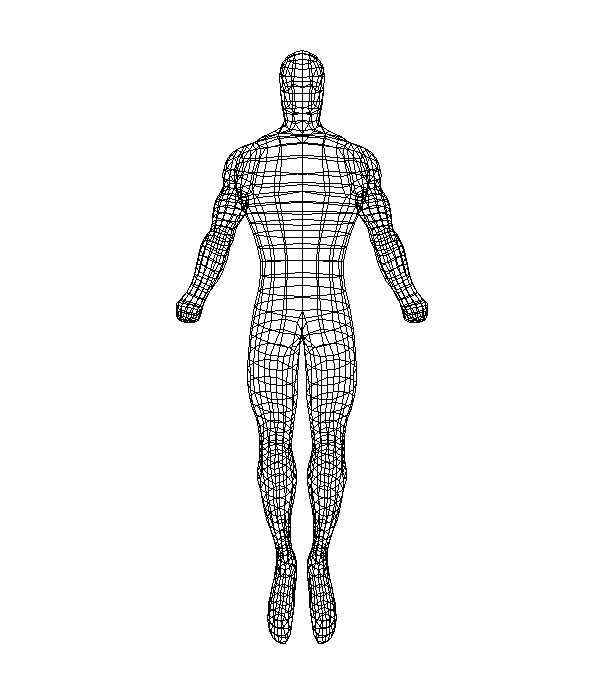


Рисунок 3 – Обычный вид (вид спереди)

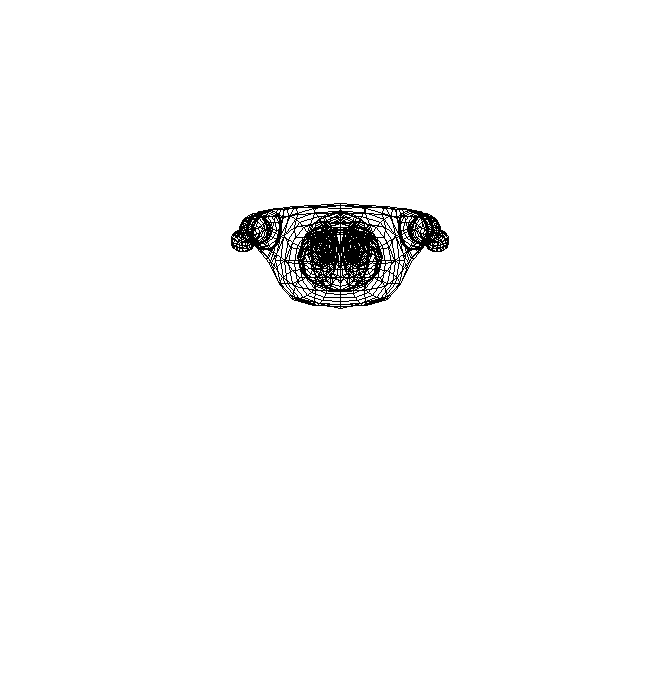


Рисунок 4 – вид машины при вращении вверх

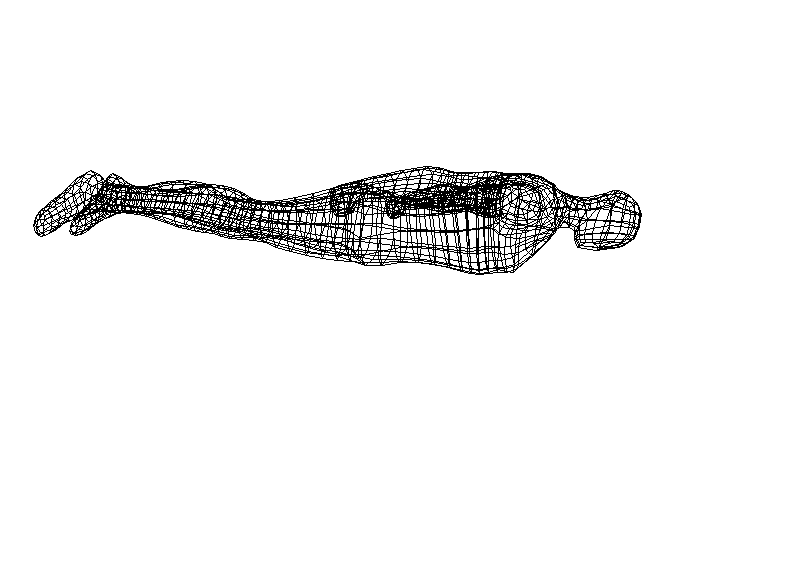


Рисунок 5 – вид машины при вращении в сторону

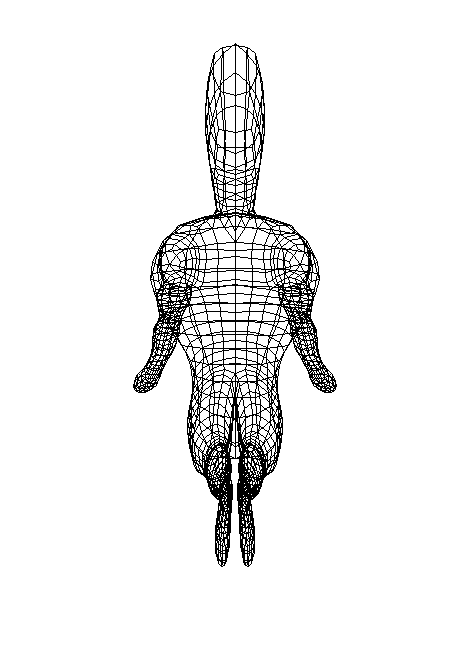


Рисунок 6 – уменьшение расстояния до машины

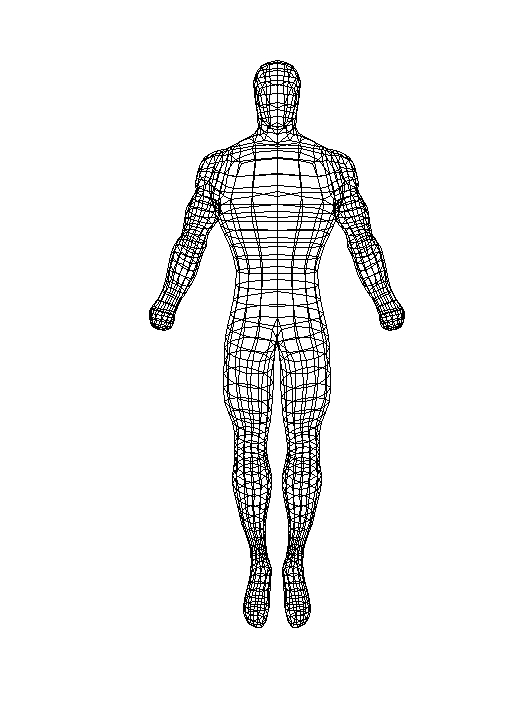


Рисунок 7 – увеличение расстояния до машины

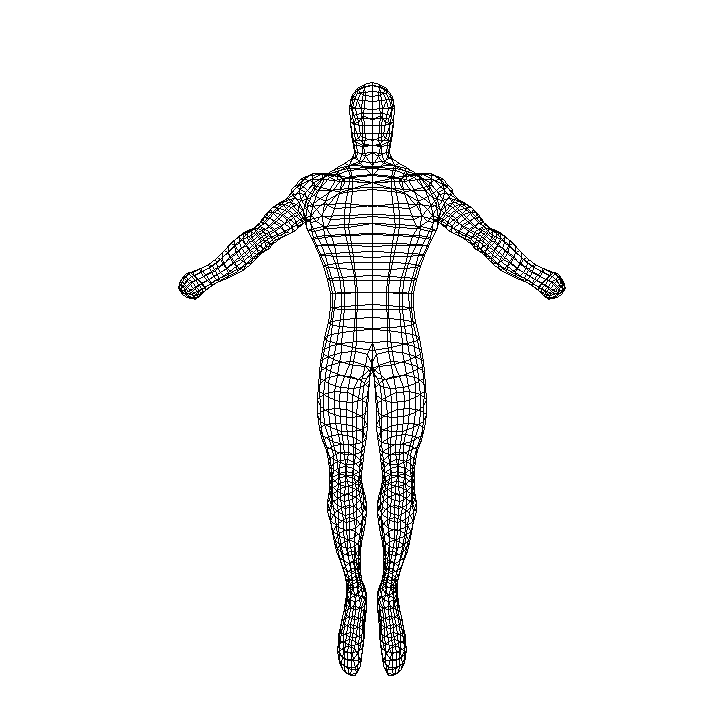


Рисунок 8 – изменение анимации человека

### 8. Вывод

В ходе выполнения данной курсовой работы мы приобрели практические навыки в области 3D-моделирования, а также углублённо изучили математические основы видовых и перспективных преобразований. Получили общее понимание принципов работы с 3D-объектами в контексте компьютерной графики на языке C++. Итогом проекта стала программа, отображающая 3D-модель, загружаемую из файла формата OBJ.

# **9 Список использованной литературы**

1. Аммерал Л. Машинная графика на языке Си. В 4-х кн. Кн.1. Принципы программирования в машинной графике. Пер. с англ. - М: Сол Систем, 1992.
2. Карпов Е.В. Геометрическое моделирование и машинная графика в САПР: Методические указания к курсовому проектированию. - Пенза: ПГТУ, 1994. - 41 с.

# **Приложение А**

Блок-схема алгоритма

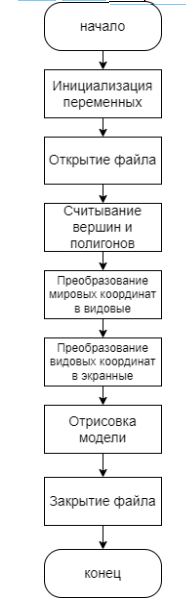


Рисунок 8 – Блок-схема алгоритма программы

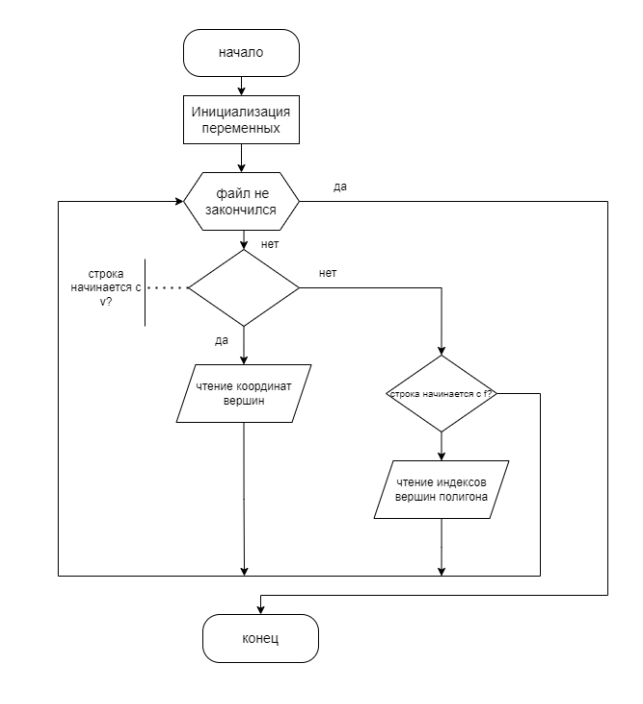


Рисунок 9 – Блок-схема алгоритма функции чтения файла obj

# **Приложение Б**

Листинг программы

**Файл kurs3d.cpp**

// kurs3d.cpp : Определяет точку входа для приложения.

//

#include "framework.h"

#include "kurs3d.h"

#include <cmath>

#include <sstream>

#include <vector>

#include <string>

#include <fstream>

using namespace std;

#define MAX\_LOADSTRING 100

// Глобальные переменные:

HINSTANCE hInst;

WCHAR szTitle[MAX\_LOADSTRING];

WCHAR szWindowClass[MAX\_LOADSTRING];

bool useAltFile = false;

const double PI = 3.14159265358979323846;

double r = 500, horangle = 90, verangle = 0;

double h = horangle \* PI / 180;

double v = verangle \* PI / 180;

int picsize = 1, objsize = 1;

ifstream f;

// Массивы для хранения координат и полигонов

vector<vector<float>> world;

vector<vector<float>> vid;

vector<vector<float>> ekran;

vector<vector<int>> poligons;

// Загрузка данных из .obj файла

void worldpre() {

ifstream f(useAltFile ? "silverepinich2.obj" : "silverepinich1.obj");

if (!f) return;

string line;

while (getline(f, line)) {

istringstream iss(line);

string token;

iss >> token;

if (token == "v") {

vector<float> vertex(4, 1.0f);

for (int i = 0; i < 3; ++i) {

float val;

iss >> val;

vertex[i] = val \* 100.0f;

}

world.push\_back(vertex);

}

else if (token == "f") {

vector<int> polygon;

string part;

while (iss >> part) {

size\_t pos = part.find('/');

int index = stoi(pos != string::npos ? part.substr(0, pos) : part);

polygon.push\_back(index - 1);

}

poligons.push\_back(polygon);

}

}

f.close();

}

// Преобразование в видовые координаты

void viewtrans() {

vid.clear();

for (const auto& vtx : world) {

vector<float> vcoord(4, 1.0f);

vcoord[0] = vtx[0] \* -sin(h) + vtx[1] \* cos(h);

vcoord[1] = vtx[0] \* -cos(v) \* cos(h) + vtx[1] \* -cos(v) \* sin(h) + vtx[2] \* sin(v);

vcoord[2] = vtx[0] \* -sin(v) \* cos(h) + vtx[1] \* -sin(v) \* sin(h) + vtx[2] \* -cos(v) + r;

vid.push\_back(vcoord);

}

}

// Преобразование в экранные координаты

void screentrans() {

ekran.clear();

float d = r \* picsize / objsize;

for (const auto& vtx : vid) {

vector<float> scr(3, 1.0f);

scr[0] = d \* vtx[0] / vtx[2] + 600;

scr[1] = d \* vtx[1] / vtx[2] + 350;

ekran.push\_back(scr);

}

}

// Отрисовка полигонов

void paint\_screen(HDC hdc) {

for (const auto& poly : poligons) {

if (poly.empty()) continue;

MoveToEx(hdc, ekran[poly[0]][0], ekran[poly[0]][1], NULL);

for (size\_t i = 1; i < poly.size(); ++i) {

LineTo(hdc, ekran[poly[i]][0], ekran[poly[i]][1]);

}

LineTo(hdc, ekran[poly[0]][0], ekran[poly[0]][1]);

}

}

ATOM MyRegisterClass(HINSTANCE hInstance);

BOOL InitInstance(HINSTANCE, int);

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

INT\_PTR CALLBACK About(HWND, UINT, WPARAM, LPARAM);

int APIENTRY wWinMain(\_In\_ HINSTANCE hInstance, \_In\_opt\_ HINSTANCE hPrevInstance, \_In\_ LPWSTR lpCmdLine, \_In\_ int nCmdShow) {

UNREFERENCED\_PARAMETER(hPrevInstance);

UNREFERENCED\_PARAMETER(lpCmdLine);

LoadStringW(hInstance, IDS\_APP\_TITLE, szTitle, MAX\_LOADSTRING);

LoadStringW(hInstance, IDC\_KURS3D, szWindowClass, MAX\_LOADSTRING);

MyRegisterClass(hInstance);

if (!InitInstance(hInstance, nCmdShow)) return FALSE;

HACCEL hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC\_KURS3D));

MSG msg;

while (GetMessage(&msg, nullptr, 0, 0)) {

if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg)) {

TranslateMessage(&msg);

DispatchMessage(&msg);

}

}

return (int)msg.wParam;

}

ATOM MyRegisterClass(HINSTANCE hInstance) {

WNDCLASSEXW wcex = {};

wcex.cbSize = sizeof(WNDCLASSEX);

wcex.style = CS\_HREDRAW | CS\_VREDRAW;

wcex.lpfnWndProc = WndProc;

wcex.hInstance = hInstance;

wcex.hIcon = LoadIcon(hInstance, MAKEINTRESOURCE(IDI\_KURS3D));

wcex.hCursor = LoadCursor(nullptr, IDC\_ARROW);

wcex.hbrBackground = (HBRUSH)(COLOR\_WINDOW + 1);

wcex.lpszMenuName = MAKEINTRESOURCEW(IDC\_KURS3D);

wcex.lpszClassName = szWindowClass;

wcex.hIconSm = LoadIcon(wcex.hInstance, MAKEINTRESOURCE(IDI\_SMALL));

return RegisterClassExW(&wcex);

}

BOOL InitInstance(HINSTANCE hInstance, int nCmdShow) {

hInst = hInstance;

HWND hWnd = CreateWindowW(szWindowClass, szTitle, WS\_OVERLAPPEDWINDOW,

CW\_USEDEFAULT, 0, CW\_USEDEFAULT, 0,

nullptr, nullptr, hInstance, nullptr);

if (!hWnd) return FALSE;

ShowWindow(hWnd, nCmdShow);

UpdateWindow(hWnd);

return TRUE;

}

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam) {

switch (message) {

case WM\_KEYDOWN:

switch (wParam) {

case 'E': r -= 100; break;

case 'Q': r += 100; break;

case 'W': verangle += 10; break;

case 'S': verangle -= 10; break;

case 'A': horangle += 10; break;

case 'D': horangle -= 10; break;

case 'G':

useAltFile = !useAltFile; // переключить файл

InvalidateRect(hWnd, NULL, TRUE);

UpdateWindow(hWnd);

break;

}

InvalidateRect(hWnd, NULL, TRUE);

UpdateWindow(hWnd);

break;

case WM\_COMMAND:

switch (LOWORD(wParam)) {

case IDM\_ABOUT:

DialogBox(hInst, MAKEINTRESOURCE(IDD\_ABOUTBOX), hWnd, About);

break;

case IDM\_EXIT:

DestroyWindow(hWnd);

break;

default:

return DefWindowProc(hWnd, message, wParam, lParam);

}

break;

case WM\_PAINT: {

PAINTSTRUCT ps;

HDC hdc = BeginPaint(hWnd, &ps);

h = horangle \* PI / 180;

v = verangle \* PI / 180;

world.clear();

vid.clear();

ekran.clear();

poligons.clear();

worldpre();

viewtrans();

screentrans();

paint\_screen(hdc);

EndPaint(hWnd, &ps);

break;

}

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

return DefWindowProc(hWnd, message, wParam, lParam);

}

return 0;

}

INT\_PTR CALLBACK About(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam) {

UNREFERENCED\_PARAMETER(lParam);

switch (message) {

case WM\_INITDIALOG:

return (INT\_PTR)TRUE;

case WM\_COMMAND:

if (LOWORD(wParam) == IDOK || LOWORD(wParam) == IDCANCEL) {

EndDialog(hDlg, LOWORD(wParam));

return (INT\_PTR)TRUE;

}

break;

}

return (INT\_PTR)FALSE;

}