Московский авиационный институт

(Национальный исследовательский университет)

Факультет: «Информационные технологии и прикладная математика»

Кафедра: «Вычислительная математика и программирование»

Дисциплина: «Компьютерная графика»

**Курсовая работа**

Вариант № 1

Тема: Бикубическая поверхность Безье

Студент: Бронников М. А.

Преподаватель: Филиппов Г. С.

Группа: М80-307Б

Дата:

Оценка:

Подпись:

Москва, 2019

Содержание:

1. Решение задачи
2. Листинг программы
3. Выводы

**Постановка задачи**

Составить и отладить программу, обеспечивающую каркасную визуализацию порции поверхности заданного типа. Исходные данные готовятся самостоятельно и вводятся из файла или в панели ввода данных. Должна быть обеспечена возможность тестирования программы на различных наборах исходных данных. Программа должна обеспечивать выполнение аффинных преобразований для заданной порции поверхности, а также возможность управлять количеством изображаемых параметрических линий. Для визуализации параметрических линий поверхности разрешается использовать только функции отрисовки отрезков в экранных координатах.

**Листинг программы**

i#include "BezierSurface.h"

const GLfloat BezierSurface::PI = 3.1415926539;

GLint BezierSurface::clicked = -1;

BasePlain BezierSurface::clickPlain = BasePlain(Vector(100, 100), 700, 700);

BSurface BezierSurface::bezier = BSurface(BezierSurface::ctrlpointsReal);

Matrix BezierSurface::cameraMatrix = Matrix({

{ 1, 0, 0, 0 },

{ 0, 1, 0, 0 },

{ (float)(1.0 / 2.0 \* std::cos(45 \* (BezierSurface::PI / 180.0))), (float)(1.0 / 2.0 \* std::sin(45 \* (BezierSurface::PI / 180.0))), 0, 0 },

{ 0, 0, 0, 1 }

});

Matrix BezierSurface::reverseMatrix = Matrix({

{ 1, 0, 0, 0 },

{ (float)(-std::cos(45 \* (BezierSurface::PI / 180.0)) / std::sin(45 \* (BezierSurface::PI / 180.0))), 0, (float)(2.0/std::sin(45 \* (BezierSurface::PI / 180.0))), 0 },

{ 0, 0, 0, 0 },

{ 0, 0, 0, 1 }

});

std::vector<Vector\*> BezierSurface::ctrlpointsPlain = std::vector<Vector\*>();

std::vector<Vector\*> BezierSurface::ctrlpointsReal = std::vector<Vector\*>();

BezierSurface::~BezierSurface() {

for (Vector\* p : ctrlpointsPlain) {

delete p;

}

for (Vector\* p : ctrlpointsReal) {

delete p;

}

}

void BezierSurface::init() {

glClearColor(1.0, 1.0, 1.0, 0.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glOrtho(0.f, BezierSurface::SCREEN\_WIDTH, 0.f, BezierSurface::SCREEN\_HEIGHT, 0.f, 1.f);

glEnable(GL\_POINT\_SMOOTH);

}

void BezierSurface::renderPoints()

{

glPointSize(5.0);

glColor3f(0, 1, 0);

glBegin(GL\_POINTS);

for (Vector\* plainPoint : BezierSurface::ctrlpointsPlain) {

Vector p = cameraMatrix \* (\*plainPoint);

glVertex2f( p[0], p[1] );

}

glEnd();

glColor3f(1, 0, 0);

glBegin(GL\_POINTS);

for (Vector\* realPoint : BezierSurface::ctrlpointsReal) {

Vector p = cameraMatrix \* (\*realPoint);

glVertex2f( p[0], p[1] );

}

glEnd();

}

void BezierSurface::renderNet()

{

if (ctrlpointsReal.size() < 16) {

return;

}

glColor3f(0, 0, 0);

glBegin(GL\_LINE\_STRIP);

GLint u = 1;

for (GLint i = 0; i < ctrlpointsReal.size(); i+=u) {

Vector p = cameraMatrix \* (\*ctrlpointsReal[i]);

glVertex2f(p[0], p[1]);

if (u > 0 && i % 4 == 3) {

u = -1;

i += 5;

} else if (u < 0 && i % 4 == 0) {

u = 1;

i += 3;

}

}

glEnd();

*/\**

*glBegin(GL\_LINES);*

*for (GLint i = 0; i < ctrlpointsReal.size(); ++i) {*

*if (i + 4 < 16 ) {*

*Vector p = cameraMatrix \* (\*ctrlpointsReal[i]);*

*Vector o = cameraMatrix \* (\*ctrlpointsReal[i+4]);*

*glVertex2f(p[0], p[1]);*

*glVertex2f(o[0], o[1]);*

*}*

*}*

*glEnd(); \*/*

}

void mouse\_button\_callback(GLFWwindow\* *window*, int *button*, int *action*, int *mods*)

{

std::cout << "cl: " << BezierSurface::clicked << std::endl;

BezierSurface::clicked = -1;

if (button == GLFW\_MOUSE\_BUTTON\_LEFT && action == GLFW\_PRESS) {

double xpos, ypos;

glfwGetCursorPos(window, &xpos, &ypos);

ypos = std::fabs(BezierSurface::SCREEN\_HEIGHT - ypos);

std::cout << xpos << ", " << ypos << std::endl;

Vector projPoint = Vector(xpos, ypos);

for (GLint i = 0; i < BezierSurface::ctrlpointsReal.size(); ++i) {

Vector cp = BezierSurface::cameraMatrix \* \*BezierSurface::ctrlpointsReal.at(i);

if (std::fabs(cp[0] - projPoint[0]) < 5 && std::fabs(cp[1] - projPoint[1]) < 5) {

BezierSurface::clicked = 16 + i;

std::cout << "real cl: " << BezierSurface::clicked << std::endl;

return;

}

}

for (GLint i = 0; i < BezierSurface::ctrlpointsPlain.size(); ++i) {

Vector cp = BezierSurface::cameraMatrix \* \*BezierSurface::ctrlpointsPlain.at(i);

if (std::fabs(cp[0] - projPoint[0]) < 5 && std::fabs(cp[1] - projPoint[1]) < 5) {

BezierSurface::clicked = i;

std::cout << "plain cl: " << BezierSurface::clicked << std::endl;

return;

}

}

if (BezierSurface::ctrlpointsPlain.size() < 16) {

Vector p = BezierSurface::reverseMatrix \* Vector(xpos, ypos);

std::cout << p << std::endl;

BezierSurface::ctrlpointsPlain.push\_back(new Vector(p));

BezierSurface::ctrlpointsReal.push\_back(new Vector(p));

if (BezierSurface::ctrlpointsPlain.size() == 16) {

*//BezierSurface::bezier = BSurface();*

BezierSurface::bezier.setCtrls(BezierSurface::ctrlpointsReal);

BezierSurface::bezier.enable();

}

}

}

}

void cursor\_pos\_callback(GLFWwindow\* *window*, double *xpos*, double *ypos*)

{

if (BezierSurface::clicked != -1) {

ypos = std::fabs(BezierSurface::SCREEN\_HEIGHT - ypos);

if (BezierSurface::clicked / 16 == 0) {

Vector p = BezierSurface::reverseMatrix \* Vector(xpos, ypos);

(\*BezierSurface::ctrlpointsPlain[BezierSurface::clicked % 16])[0] = p[0];

(\*BezierSurface::ctrlpointsPlain[BezierSurface::clicked % 16])[2] = p[2];

(\*BezierSurface::ctrlpointsReal[BezierSurface::clicked % 16])[0] = p[0];

(\*BezierSurface::ctrlpointsReal[BezierSurface::clicked % 16])[2] = p[2];

}

else {

(\*BezierSurface::ctrlpointsReal[BezierSurface::clicked % 16])[1] = (ypos - 1.0 / 2.0 \* std::sin(45 \* (BezierSurface::PI / 180.0)) \* (\*BezierSurface::ctrlpointsPlain[BezierSurface::clicked % 16])[2]);

}

}

}

int main() {

GLFWwindow\* window;

if (!glfwInit())

return -1;

window = glfwCreateWindow(BezierSurface::SCREEN\_WIDTH, BezierSurface::SCREEN\_HEIGHT, "BezierSurface", NULL, NULL);

if (!window)

{

glfwTerminate();

return -1;

}

glfwMakeContextCurrent(window);

glfwSetMouseButtonCallback(window, mouse\_button\_callback);

glfwSetCursorPosCallback(window, cursor\_pos\_callback);

*/\* -- Init -- \*/*

BezierSurface::init();

while (!glfwWindowShouldClose(window))

{

glClear(GL\_COLOR\_BUFFER\_BIT);

*/\* -- Render -- \*/*

BezierSurface::clickPlain.draw( BezierSurface::cameraMatrix);

BezierSurface::bezier.draw(BezierSurface::cameraMatrix);

BezierSurface::renderNet();

BezierSurface::renderPoints();

glfwSwapBuffers(window);

glfwPollEvents();

}

glfwTerminate();

*/\* -- Terminate -- \*/*

return 0;

}

**Выводы**

Выполнив курсовой проект, я научился реализовывать поверхность Безье. Выполнить данную работу было довольно интересно и в меру сложно.

**Список литературы**

1. <http://scask.ru/a_book_mm3d.php?id=105>
2. <https://ru.wikipedia.org/wiki/Билинейная_интерполяция>