ГУАП

КАФЕДРА № 43

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| асс. |  |  |  | И.М. Лозоватский |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №8 |
| ОСВОЕНИЕ РАБОТЫ С ПРОГРАММИРУЕМЫМИ ШЕЙДЕРАМИ |
| по курсу: КОМПЬЮТЕРНАЯ ГРАФИКА |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4831 |  |  |  | К.А. Корнющенков |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2020

**Задание**

Выполнить на выбор один из вариантов:

- Шейдер с сохранением данных в текстуре,

- Процедурный, текстурный шейдер,

- Традиционные шейдер,

- Фрагментный шейдер,

- Вершинный шейдер.

Примеры из пакета с компилятором Cg не допускаются

**Ход Работы**

#include <string.h>

#include <stdlib.h>

#include <stdio.h>

#include <cmath>

#include <GLUT/glut.h>

#include <cmath>

#include <stdlib.h>

#include <stdio.h>

#include <iostream>

#include <math.h>

#include <stdio.h>

#include <stdlib.h>

GLint loc;

GLuint v, f, f2, p;

float a = 0;

void changeSize(int w, int h) {

if (h == 0)

h = 1;

float ratio = 1.0\* w / h;

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glViewport(0, 0, w, h);

gluPerspective(45, ratio, 1, 100);

glMatrixMode(GL\_MODELVIEW);

}

void drawCar() {

glPushMatrix();

glPushMatrix(); // основание

glScalef(2,.5,1);

glutSolidCube(.5);

glPopMatrix();

glTranslatef(0,0,.25);

glPushMatrix();

glTranslatef(-.4,-.2,0);

glutSolidTorus(.05,.1,8,8); // колесо

glTranslatef(.8,0,0);

glutSolidTorus(.05,.1,8,8); // колесо

glPopMatrix();

glTranslatef(0,0,-.5);

glPushMatrix();

glTranslatef(-.4,-.2,0);

glutSolidTorus(.05,.1,8,8); // колесо

glTranslatef(.8,0,0);

glutSolidTorus(.05,.1,8,8); // колесо

glPopMatrix();

glPopMatrix();

}

void renderScene(void) {

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glLoadIdentity();

gluLookAt(0.0, 5.0, 5.0, 0.0, 0.0, 0.0, 0.0f, 1.0f, 0.0f);

glRotatef(a, 0, 1, 0);

drawCar();

a += 0.50;

glutSwapBuffers();

}

int printOglError(char \*file, int line) {

GLenum glErr;

int retCode = 0;

glErr = glGetError();

while (glErr != GL\_NO\_ERROR){

retCode = 1;

glErr = glGetError();

}

return retCode;

}

void setShaders() {

char \*vs = NULL,

\*fs = NULL;

v = glCreateShader(GL\_VERTEX\_SHADER);

f = glCreateShader(GL\_FRAGMENT\_SHADER);

f2 = glCreateShader(GL\_FRAGMENT\_SHADER);

free(vs);

free(fs);

glCompileShader(v);

glCompileShader(f);

glCompileShader(f2);

p = glCreateProgram();

glAttachShader(p, f);

glAttachShader(p, f2);

glAttachShader(p, v);

glLinkProgram(p);

glUseProgram(p);

loc = glGetUniformLocation(p, "time");

}

int main(int argc, char \*\*argv) {

glutInitDisplayMode(GLUT\_DEPTH | GLUT\_DOUBLE | GLUT\_RGBA);

glutInitWindowPosition(100, 100);

glutInitWindowSize(200, 200);

glutInit(&argc, argv);

glutCreateWindow("лаба 7");

glutDisplayFunc(renderScene);

glutIdleFunc(renderScene);

glutReshapeFunc(changeSize);

glEnable(GL\_DEPTH\_TEST);

glClearColor(0.5, 0.2, 0.2, 1.0);

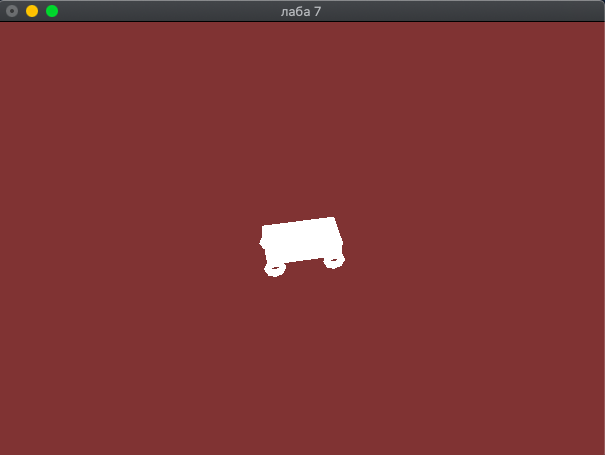
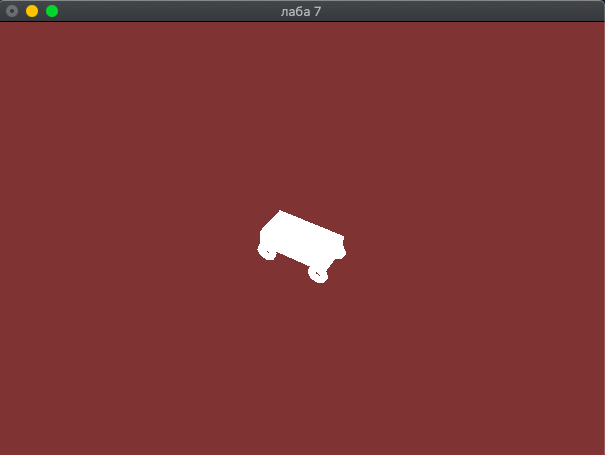
glEnable(GL\_CULL\_FACE);

setShaders();

glutMainLoop();

}

**Результат *работы программы***



**Вывод**

В ходе работы я познакомился с технологией работы с программируемыми шейдерами и применил её на практике.