|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования Российской Федерации**  Калужский филиал  федерального государственного бюджетного  образовательного учреждения высшего образования  ***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»***  ***(КФ МГТУ им. Н.Э. Баумана)*** |

**ФАКУЛЬТЕТ** ***ИУК «Информатика и управление»***

**КАФЕДРА** \_\_***ИУК4 «Программное обеспечение ЭВМ, информационные технологии»***

**ЛАБОРАТОРНАЯ РАБОТА №1**

**«Введение в OpenGL»**

**ДИСЦИПЛИНА: «Компьютерная графика»**

|  |  |  |
| --- | --- | --- |
| Выполнил: студент гр. ИУК4-42Б | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ( Карельский М.К. )  (Подпись) |
| Проверил: | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ( Глебов С.А. )  (Подпись) |
| Дата сдачи (защиты):  Результаты сдачи (защиты): | | |
|  | - Балльная оценка:  - Оценка: | |

Калуга, 2022

**Цель:** формирование практических навыков по работе с проекционной матрицей средствами OpenGL, а также созданию простейших анимаций графических примитивов и их адаптации в абсолютных и относительных оконных координатах.

**Задачи:**

* сформировать представление о методах и секторе решаемых OpenGL задач,
* изучить основные принципы работы OpenGL,
* представлять и понимать основные реализации OpenGL,
* знать типы данных OpenGL и специфику именования переменных,
* понимать основные принципы трехмерного программирования компьютерной графики,
* иметь представление о проекциях, уметь создавать типовой проект в различных средах разработки (Visual Studio),
* иметь представление о двойной буферизации.

**Вариант №7**

Выполнить пульсирующее масштабирование с вращением квадрата.

**Листинг 2:**

// подключаем заголовочные файлы библиотек

#include "glew.h"

#include "glut.h"

void RenderScene(void)

{

// Окно очищается текущим цветом очистки

glClear(GL\_COLOR\_BUFFER\_BIT);

// В буфер вводятся команды рисования

glFlush();

}

// Устанавливается состояние визуализации

void SetupRC(void)

{

glClearColor(0.721f, 0.972f, 0.709f, 1.0f);

}

// Точка входа основной программы

void main(void)

{

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutCreateWindow("Dimple");

glutDisplayFunc(RenderScene);

SetupRC();

glutMainLoop();

}

**Листинг 3:**

// подключаем заголовочные файлы библиотек

#include "glew.h"

#include "glut.h"

void RenderScene(void) {

// Очищаем окно, используя текущий цвет очистки

glClear(GL\_COLOR\_BUFFER\_BIT);

**//glColor3f(1.0f, 0.0f, 0.0f);**

**glColor3f(0.670f, 0.082f, 0.898f);**

// Рисует прямоугольник, закрашенный текущим цветом

**//glRectf(-25.0f, 25.0f, 25.0f, -25.0f);**

**//glRectf(0.0f, 50.0f, 50.0f, 0.0f);**

**glRectf(0.0f, 75.0f, 75.0f, 0.0f);**

**glRectf(-75.0f, 0.0f, -25.0f, -50.0f);**

// Очищает очередь текущих команд

glFlush();

}

/////////////////// Задает состояние визуализации

void SetupRC(void)

{

**//glClearColor(0.0f, 0.0f, 1.0f, 1.0f);**

**glClearColor(0.721f, 0.972f, 0.709f, 1.0f);**

}

///////Вызывается библиотекой GLUT при изменении размеров окна

void ChangeSize(GLsizei w, GLsizei h)

{

GLfloat aspectRatio;

// Предотвращает деление на нуль

if (h == 0)

h = 1;

// Устанавливает поле просмотра с размерами окна

glViewport(0, 0, w, h);

// Обновляет систему координат

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

// С помощью плоскостей отсечения (левая, правая, нижняя,

// верхняя, ближняя, дальняя) устанавливает объем отсечения

aspectRatio = (GLfloat)w / (GLfloat)h;

if (w <= h)

glOrtho(-100.0, 100.0, -100 / aspectRatio, 100.0 / aspectRatio, 1.0, -1.0);

else

glOrtho(-100.0 \* aspectRatio, 100.0 \* aspectRatio, -100.0, 100.0, 1.0, -1.0);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

}

///////////////////Точка входа основной программы

void main(void) {

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutCreateWindow("GLRect");

glutDisplayFunc(RenderScene);

glutReshapeFunc(ChangeSize);

SetupRC();

glutMainLoop();

}

**Листинг 5:**

// подключаем заголовочные файлы библиотек

#include "glew.h"

#include "glut.h"

// Исходное положение и размер прямоугольника

GLfloat x1 = 0.0f; GLfloat y1 = 0.0f; GLfloat rsize = 25; GLfloat angle = 0.0f;

// Величина шага в направлениях х и у (число пикселей,

// на которые на каждом шаге перемещается прямоугольник)

GLfloat xstep = 1.0f; GLfloat ystep = 1.0f; GLfloat sizeStep = 1.0f; GLfloat angleStep = 5.0f;

// Отслеживание изменений ширины и высоты окна

GLfloat windowWidth; GLfloat windowHeight;

//Вызывается для рисования сцены

void RenderScene(void)

{

// Очищаем окно, используя текущий цвет очистки

glClear(GL\_COLOR\_BUFFER\_BIT);

glLoadIdentity();

glTranslatef(x1, y1, 0.0f);

glRotatef(angle, 0.0f, 0.0f, 1.0f);

glTranslatef(-x1, -y1, 0.0f);

glColor3f(0.670f, 0.082f, 0.898f);

// Рисует прямоугольник, закрашенный текущим цветом

glRectf(x1 - rsize, y1 + rsize, x1 + rsize, y1 - rsize);

// Очищает очередь текущих команд и переключает буферы

glutSwapBuffers();

}

//Вызывается библиотекой GLUT в холостом состоянии (окно не меняет размера и не перемещается)

void TimerFunction(int value)

{

// Меняет направление на противоположное при подходе

// к левому или правому краю

if (x1 > windowWidth - rsize || x1 < -windowWidth + rsize)

xstep = -xstep;

// Меняет направление на противоположное при подходе

// к верхнему или нижнему краю

if (y1 > windowHeight - rsize || y1 < -windowHeight + rsize)

ystep = -ystep;

// Перемещает квадрат

x1 += xstep;

y1 += ystep;

if (rsize > 30 || rsize < 25)

sizeStep = -sizeStep;

rsize += sizeStep;

angle += angleStep;

if (angle > 360.0f)

angle = 0.0f;

// Проверка границ. Если окно меньше прямоугольника,

// который прыгает внутри, и прямоугольник обнаруживает

// себя вне нового объема отсечения

if (x1 > (windowWidth - rsize + xstep))

x1 = windowWidth - rsize - 1;

else if (x1 < -(windowWidth + xstep))

x1 = -windowWidth - 1;

if (y1 > (windowHeight + ystep))

y1 = windowHeight - 1;

else if (y1 < -(windowHeight - rsize + ystep))

y1 = -windowHeight + rsize - 1;

// Перерисовывает сцену с новыми координатами

glutPostRedisplay();

glutTimerFunc(33, TimerFunction, 1);

}

//Задает состояние визуализации

void SetupRC(void)

{

glClearColor(0.721f, 0.972f, 0.709f, 1.0f);

}

//Вызывается библиотекой GLUT при изменении размеров окна

void ChangeSize(GLsizei w, GLsizei h)

{

GLfloat aspectRatio;

// Предотвращает деление на нуль

if (h == 0) h = 1;

// Устанавливает поле просмотра с размерами окна

glViewport(0, 0, w, h);

// Обновляет систему координат

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

// С помощью плоскостей отсечения (левая, правая, нижняя,

// верхняя, ближняя, дальняя) устанавливает объем отсечения

aspectRatio = (GLfloat)w / (GLfloat)h;

if (w <= h)

{

windowWidth = 100;

windowHeight = 100 / aspectRatio;

glOrtho(-100.0, 100.0, - windowHeight, windowHeight, 1.0, -1.0);

}

else

{

windowWidth = 100 \* aspectRatio;

windowHeight = 100;

glOrtho(-windowWidth, windowWidth, - 100.0, 100.0, 1.0, -1.0);

}

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

}

//Точка входа основной программы

void main(void)

{

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);

glutCreateWindow("Bounce");

glutDisplayFunc(RenderScene);

glutReshapeFunc(ChangeSize);

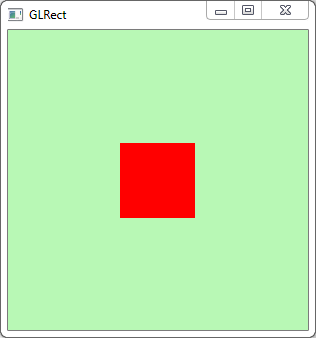
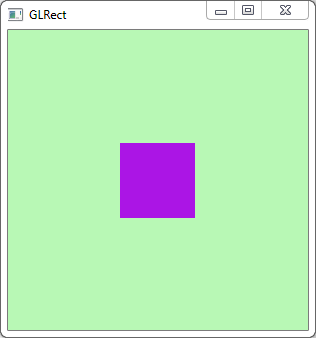
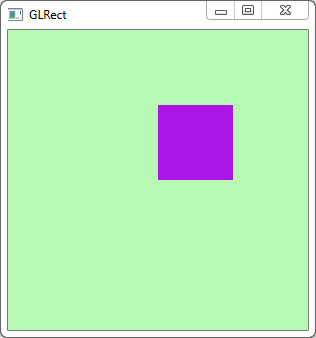
glutTimerFunc(33, TimerFunction, 1);

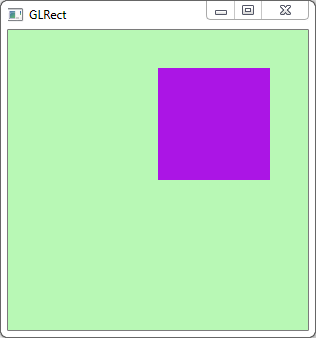
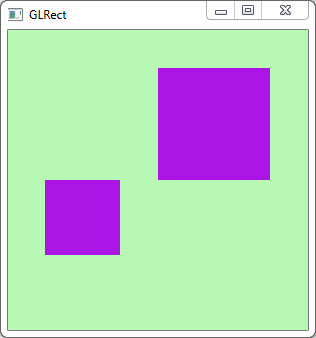
SetupRC();

glutMainLoop();

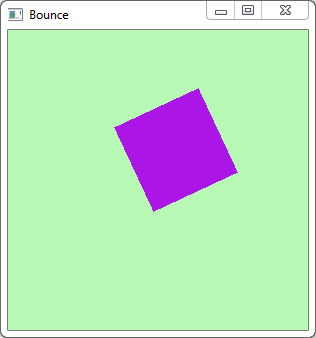
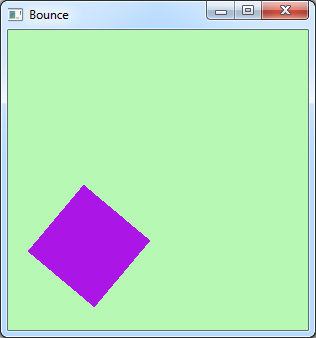
}

**Демонстрация:**

**Рисунок 1.** Варианты листинга 3

**Рисунок 2.** Листинг 5

**Вывод:** в ходе выполнения лабораторной работы были получены практические навыки работы с OpenGL, создания окон, графических примитивов, изменения их размеров, цвета, положения, угла наклона, реализации их анимации.