|  |  |
| --- | --- |
| **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 №6**

**«Вывод растровых изображений 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 растровыми изображениями средствами OpenGL, а также простейшим преобразованиям цветовых и пространственных характеристик.

**Задачи:**

* понимать принципы вывода растровых изображений,
* знать отличия битового образа от растрового и их характеристики,
* научиться использовать средства OpenGL для вывода растровых изображений,
* знать основные константы OpenGL, используемые при обработке растровых изображений,
* уметь создавать приложения OpenGL с использованием функций для работы с растровыми изображениями.

**Вариант 7**

**Задание:**

1. Для **Листинга 1** создать битовые образы: создать два разных битовых образа (не из примера) размерами 16х20 и 10х10. Отобразить два образа в виде песочных часов. В верхней части окна до центра окна один образ с уменьшением количества элементов в строке, в нижней части – другой. Минимальное количество образов максимальной строке 10. Цвета каждого образа должны быть разными.
2. Для **Листингов 2, 3, 4** осуществить вывод изображения в формате tga: для собственного изображения формата tga, осуществить 4 его вывода каждый раз увеличивая высоту на 20%. Изначально отображается только 20% рисунка.
3. Для **Листинга 5** реализовать преобразование изображения: для собственного изображения формата tga реализовать не менее трех разных переводов в градации серого (использовать меню).

**Задание 1:**

#include "glew.h"

#include "glut.h"

namespace Task\_6\_1

{

GLubyte one[40] =

{

0xFF, 0xFF,

0xFF, 0xFF,

0xFF, 0xFF,

0xFF, 0xFF,

0x03, 0xC0,

0x03, 0xC0,

0x03, 0xC0,

0x03, 0xC0,

0x03, 0xC0,

0x03, 0xC0,

0x03, 0xC0,

0x03, 0xC0,

0x03, 0xC0,

0x03, 0xC0,

0x03, 0xC0,

0x03, 0xC0,

0xFF, 0xFF,

0xFF, 0xFF,

0xFF, 0xFF,

0xFF, 0xFF

};

GLubyte two[20] = {

0xFF, 0xFF,

0xFF, 0xFF,

0x33, 0x00,

0x33, 0x00,

0x33, 0x00,

0x33, 0x00,

0x33, 0x00,

0x33, 0x00,

0xFF, 0xFF,

0xFF, 0xFF

};

bool Positions[10][10]

{

{ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 },

{ 1, 0, 0, 0, 0, 0, 0, 0, 0, 1 },

{ 1, 0, 0, 1, 1, 1, 1, 0, 0, 1 },

{ 0, 1, 0, 1, 1, 1, 1, 0, 1, 0 },

{ 0, 1, 0, 0, 1, 1, 0, 0, 1, 0 },

{ 0, 0, 1, 0, 1, 1, 0, 1, 0, 0 },

{ 0, 0, 1, 0, 0, 0, 0, 1, 0, 0 },

{ 0, 0, 0, 1, 0, 0, 1, 0, 0, 0 },

{ 0, 0, 0, 1, 0, 0, 1, 0, 0, 0 },

{ 0, 0, 0, 0, 1, 1, 0, 0, 0, 0 },

};

void SetupRC()

{

glPixelStorei(GL\_UNPACK\_ALIGNMENT, 1);

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

}

void ChangeSize(int w, int h)

{

if (h == 0)

h = 1;

glViewport(0, 0, w, h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0.0, (GLfloat)w, 0.0f, (GLfloat)h);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

}

void RenderScene(void)

{

glClear(GL\_COLOR\_BUFFER\_BIT);

for (int y = 0; y < 10; y++)

{

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

for (int x = 0; x < 10; x++)

{

if (Positions[9 - y][x])

{

glRasterPos2i(96 + 32 \* x, 56 + 20 \* (y + 10));

glBitmap(16, 20, 0.0, 0.0, 0.0, 0.0, one);

glBitmap(16, 20, -16.0, 0.0, 0.0, 0.0, one);

}

}

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

for (int x = 0; x < 10; x++)

{

if (Positions[y][x])

{

glRasterPos2i(96 + 32 \* x, 56 + 20 \* y);

glBitmap(10, 10, 0.0, 0.0, 0.0, 0.0, two);

glBitmap(10, 10, -11.0, 0.0, 0.0, 0.0, two);

glBitmap(10, 10, -22.0, 0.0, 0.0, 0.0, two);

glBitmap(10, 10, 0.0, -10.0, 0.0, 0.0, two);

glBitmap(10, 10, -11.0, -10.0, 0.0, 0.0, two);

glBitmap(10, 10, -22.0, -10.0, 0.0, 0.0, two);

}

}

}

glutSwapBuffers();

}

int main(int argc, char\* argv[])

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_RGB | GLUT\_DOUBLE);

glutInitWindowSize(512, 512);

glutCreateWindow("OpenGL Bitmaps");

glutReshapeFunc(ChangeSize);

glutDisplayFunc(RenderScene);

SetupRC();

glutMainLoop();

return 0;

}

}

**Результат задания 1:**



**Рисунок 1.** Задание 1

**Задание 2:**

#include "glew.h"

#include "glut.h"

#include <cstddef>

#include <stdlib.h>

#include "misc.h"

namespace Task\_6\_2

{

const int WINDOW\_HEIGHT = 512;

const int WINDOW\_WIDTH = 512;

GLint renderMode = 1;

void SetupRC()

{

glPixelStorei(GL\_UNPACK\_ALIGNMENT, 1);

glClearColor(0.0f, 0.0f, 0.0f, 0.0f);

}

void ChangeSize(int w, int h)

{

if (h == 0)

h = 1;

glViewport(0, 0, w, h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0.0, (GLfloat)w, 0.0f, (GLfloat)h);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

}

void RenderScene(void)

{

GLubyte\* pImage = NULL;

GLint iWidth, iHeight, iComponents;

GLenum eFormat;

glClear(GL\_COLOR\_BUFFER\_BIT);

glPixelStorei(GL\_UNPACK\_ALIGNMENT, 1);

pImage = (GLubyte\*)ConvertTgaToBitmap("Ruins.tga",

&iWidth, &iHeight, &iComponents, &eFormat);

glRasterPos2i(0, 0);

if (pImage != NULL)

{

glDrawPixels(iWidth, iHeight \* renderMode / 5, eFormat,

GL\_UNSIGNED\_BYTE, pImage);

}

free(pImage);

glutSwapBuffers();

}

void ProcessMenu(int value)

{

if (value == 0)

WriteTga("result.tga");

else

renderMode = value;

glutPostRedisplay();

}

int main(int argc, char\* argv[])

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_RGB | GLUT\_DOUBLE);

glutInitWindowSize(WINDOW\_WIDTH, WINDOW\_HEIGHT);

glutCreateWindow("OpenGL Bitmaps");

glutReshapeFunc(ChangeSize);

glutDisplayFunc(RenderScene);

glutCreateMenu(ProcessMenu);

glutAddMenuEntry("Save", 0);

glutAddMenuEntry("100%", 5);

glutAddMenuEntry("80%", 4);

glutAddMenuEntry("60%", 3);

glutAddMenuEntry("40%", 2);

glutAddMenuEntry("20%", 1);

glutAttachMenu(GLUT\_RIGHT\_BUTTON);

SetupRC();

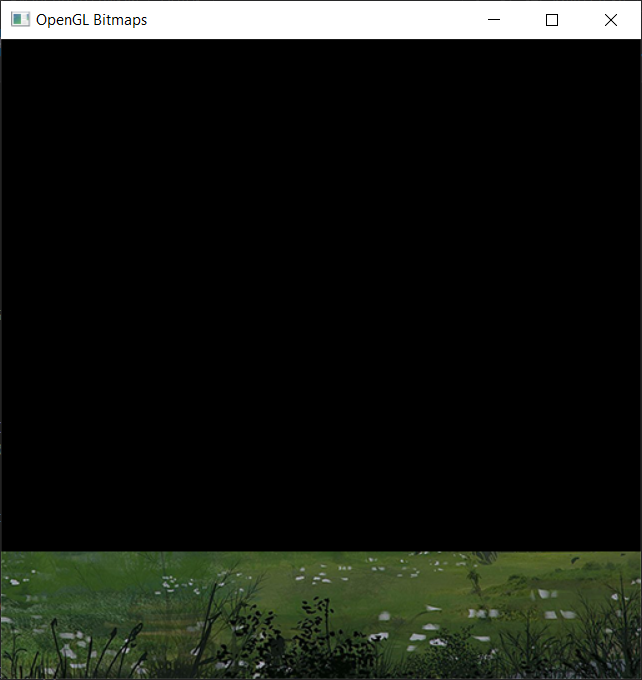
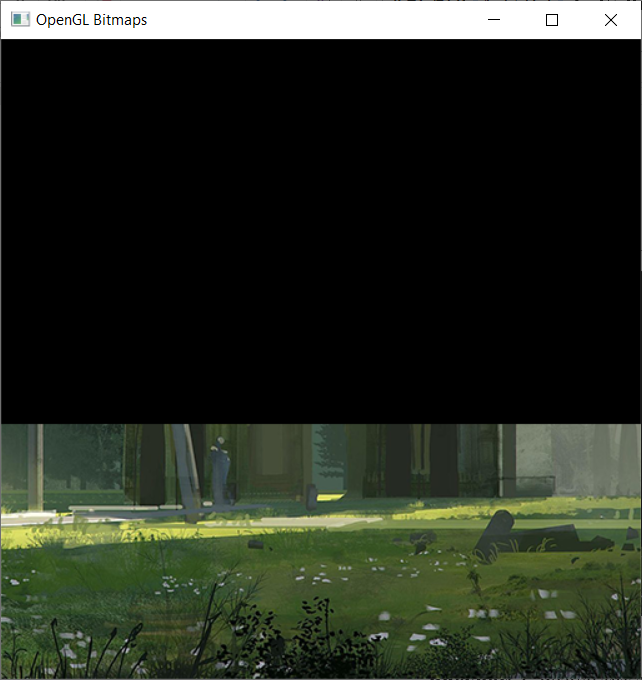
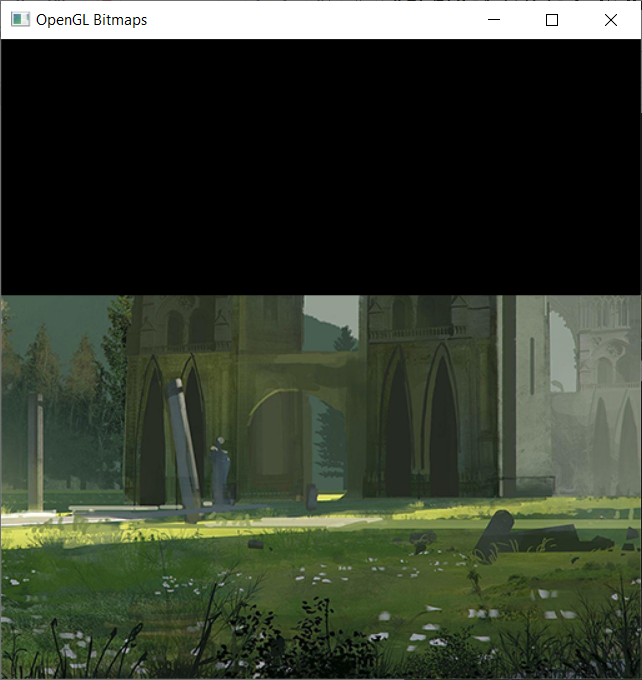
glutMainLoop();

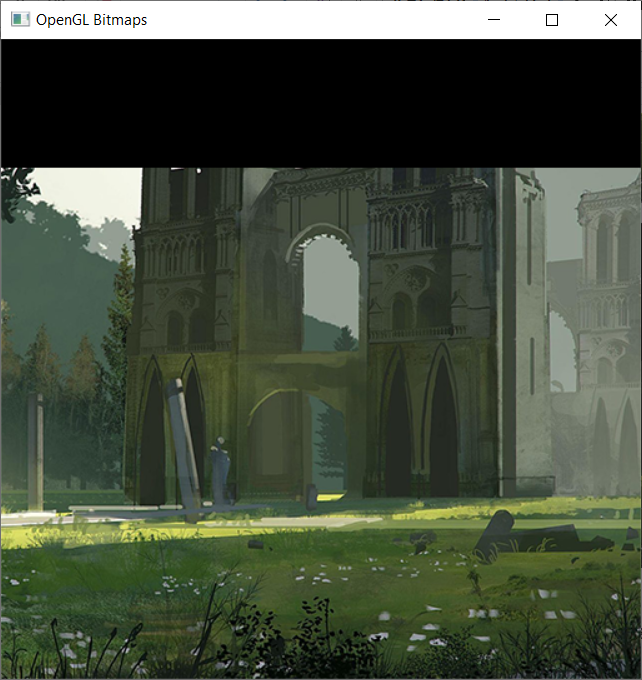
return 0;

}

}

**Результат задания 2:**

**Рисунок 2.** Результат задания 2

**Задание 3:**

#include "glew.h"

#include "glut.h"

#include <cstddef>

#include <stdlib.h>

#include "misc.h"

namespace Task\_6\_3

{

const int WINDOW\_HEIGHT = 512;

const int WINDOW\_WIDTH = 512;

GLint renderMode = 1;

void SetupRC()

{

glPixelStorei(GL\_UNPACK\_ALIGNMENT, 1);

glClearColor(0.0f, 0.0f, 0.0f, 0.0f);

}

void ChangeSize(int w, int h)

{

if (h == 0)

h = 1;

glViewport(0, 0, w, h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0.0, (GLfloat)w, 0.0f, (GLfloat)h);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

}

void RenderScene(void)

{

GLubyte\* pImage = NULL;

GLint iWidth, iHeight, iComponents;

GLenum eFormat;

glClear(GL\_COLOR\_BUFFER\_BIT);

glPixelStorei(GL\_UNPACK\_ALIGNMENT, 1);

pImage = (GLubyte\*)ConvertTgaToBitmap("Park.tga",

&iWidth, &iHeight, &iComponents, &eFormat);

glRasterPos2i(0, 0);

glPixelTransferf(GL\_RED\_SCALE, 1.0f);

glPixelTransferf(GL\_GREEN\_SCALE, 1.0f);

glPixelTransferf(GL\_BLUE\_SCALE, 1.0f);

if (pImage != NULL)

{

GLbyte\* pModifiedBytes;

glDrawPixels(iWidth, iHeight, eFormat,

GL\_UNSIGNED\_BYTE, pImage);

switch (renderMode)

{

case 2:

pModifiedBytes = (GLbyte\*)malloc(iWidth \*

iHeight);

glPixelTransferf(GL\_RED\_SCALE, 0.3f);

glPixelTransferf(GL\_GREEN\_SCALE, 0.59f);

glPixelTransferf(GL\_BLUE\_SCALE, 0.11f);

glReadPixels(0, 0, iWidth, iHeight, GL\_LUMINANCE,

GL\_UNSIGNED\_BYTE, pModifiedBytes);

glPixelTransferf(GL\_RED\_SCALE, 1.0f);

glPixelTransferf(GL\_GREEN\_SCALE, 1.0f);

glPixelTransferf(GL\_BLUE\_SCALE, 1.0f);

glDrawPixels(iWidth, iHeight, GL\_LUMINANCE,

GL\_UNSIGNED\_BYTE, pModifiedBytes);

break;

case 3:

pModifiedBytes = (GLbyte\*)malloc(iWidth \*

iHeight);

glPixelTransferf(GL\_RED\_SCALE, 0.33f);

glPixelTransferf(GL\_GREEN\_SCALE, 0.34f);

glPixelTransferf(GL\_BLUE\_SCALE, 0.33f);

glReadPixels(0, 0, iWidth, iHeight, GL\_LUMINANCE,

GL\_UNSIGNED\_BYTE, pModifiedBytes);

glPixelTransferf(GL\_RED\_SCALE, 1.0f);

glPixelTransferf(GL\_GREEN\_SCALE, 1.0f);

glPixelTransferf(GL\_BLUE\_SCALE, 1.0f);

glDrawPixels(iWidth, iHeight, GL\_LUMINANCE,

GL\_UNSIGNED\_BYTE, pModifiedBytes);

break;

case 4:

pModifiedBytes = (GLbyte\*)malloc(iWidth \*

iHeight);

glReadPixels(0, 0, iWidth, iHeight, GL\_LUMINANCE,

GL\_UNSIGNED\_BYTE, pModifiedBytes);

glDrawPixels(iWidth, iHeight, GL\_LUMINANCE,

GL\_UNSIGNED\_BYTE, pModifiedBytes);

break;

}

free(pImage);

}

glutSwapBuffers();

}

void ProcessMenu(int value)

{

if (value == 0)

WriteTga("result.tga");

else

renderMode = value;

glutPostRedisplay();

}

int main(int argc, char\* argv[])

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_RGB | GLUT\_DOUBLE);

glutInitWindowSize(WINDOW\_WIDTH, WINDOW\_HEIGHT);

glutCreateWindow("Black and white");

glutReshapeFunc(ChangeSize);

glutDisplayFunc(RenderScene);

glutCreateMenu(ProcessMenu);

glutAddMenuEntry("Save", 0);

glutAddMenuEntry("Full color", 1);

glutAddMenuEntry("NTSC", 2);

glutAddMenuEntry("Thirds", 3);

glutAddMenuEntry("Without pixel transfer", 4);

glutAttachMenu(GLUT\_RIGHT\_BUTTON);

SetupRC();

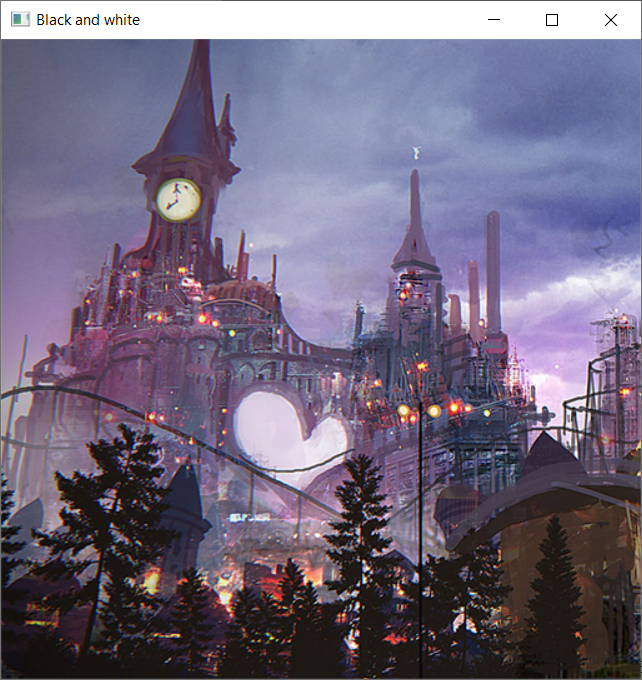
glutMainLoop();

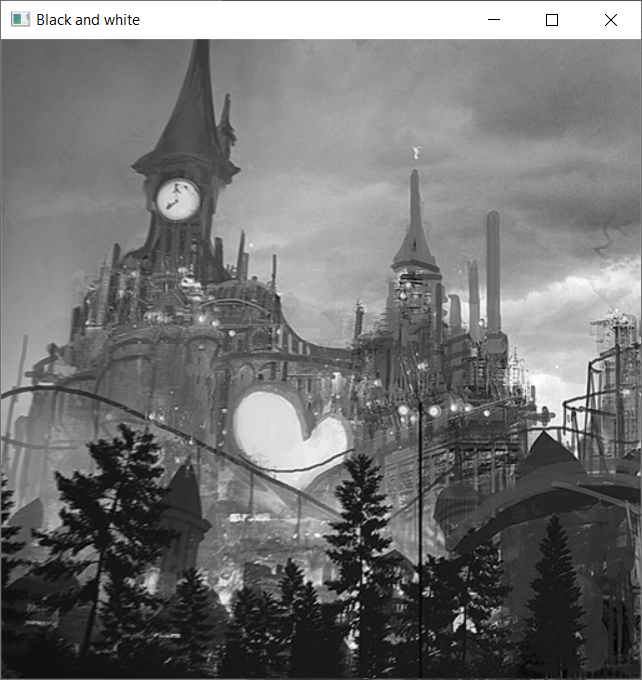
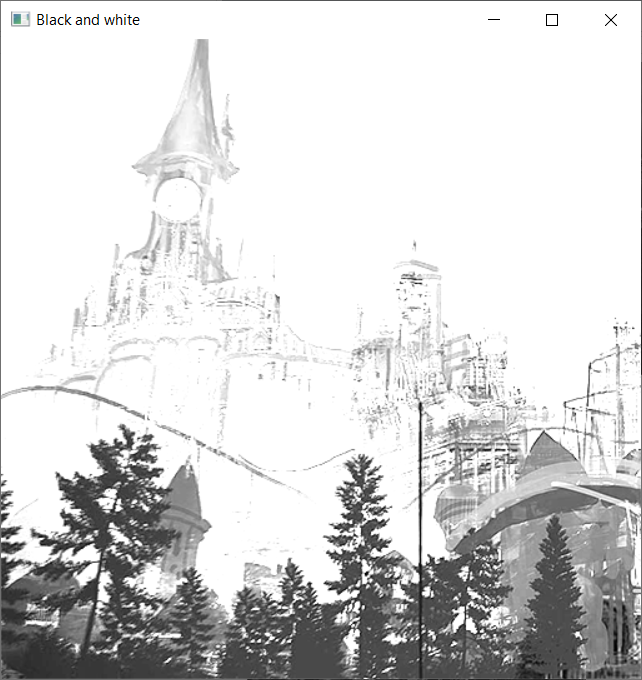
return 0;

}

}

**Результат задания 3:**

**Рисунок 3.** Результат задания 3 (полный цвет, NTSC, деление цветовых каналов на 3, без pixel transfer)

**Вывод:** в ходе выполнения лабораторной работы были получены навыки работы с битовыми и пиксельными образами, загрузки и сохранения TGA-изображений, их обрезки и изменения их цветов.