КАЗАНСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМ. А. Н. ТУПОЛЕВА – КАИ

Институт компьютерных технологий и защиты информации

Кафедра: Автоматизированных систем обработки информации и управления

Г.М. НАБЕРЕЖНОВ, Н.Н. МАКСИМОВ

ЭЛЕМЕНТАРНОЕ ПРОГРАММИРОВАНИЕ ГРАФИКИ В OpenGL

**Методическое пособие
к лабораторным работам по курсу
«Компьютерная геометрия и графика»**

Студент, гр. 4210

Гауиш М.Г

Преподаватель

Гаптуллазянова Гульшат Ильдусовна

Казань – 2023

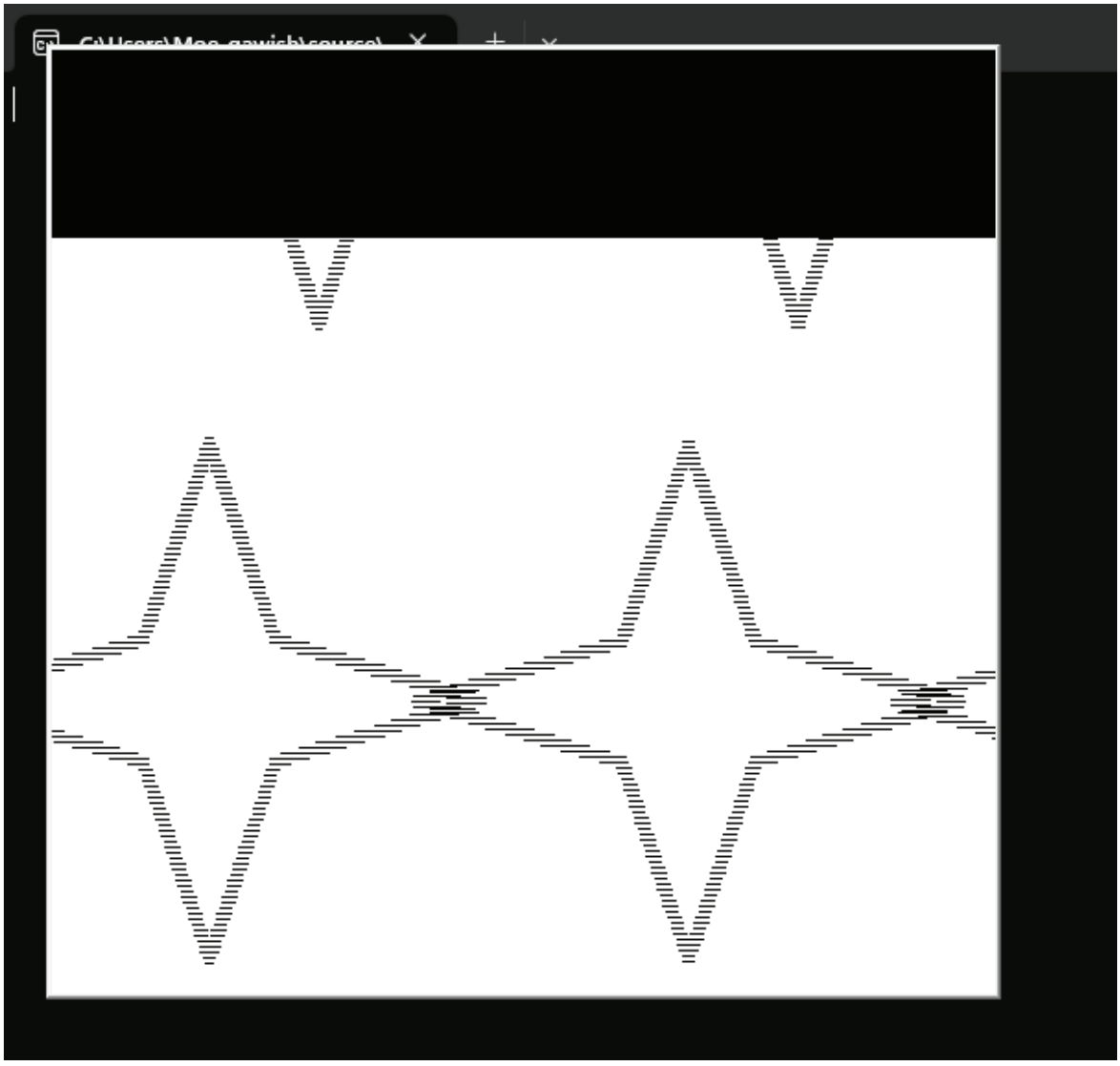
Лабораторная работа №5 (Отчет)

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОГО ВЫПОЛНЕНИ

Задания для самостоятельного выполнения

1. **Осуществите рисование объекта, созданного в предыдущей лабораторной работе, при помощи дисплейного списка.**
2. **Наложите на объект три различные текстуры на три различные грани. (Для этого необходимо предварительно создать текстурные изображения в формате *.bmp.)**
3. **Осуществите изменение параметров отображения текстуры объекта таким образом, чтобы при нажатии на левую кнопку мыши объект излучал свет согласно заданной текстуре, при нажатии на правую кнопку мыши объект закрашивался текстурой, и при этом моделировались коэффициенты отражения.**

Скриншот работы программы:



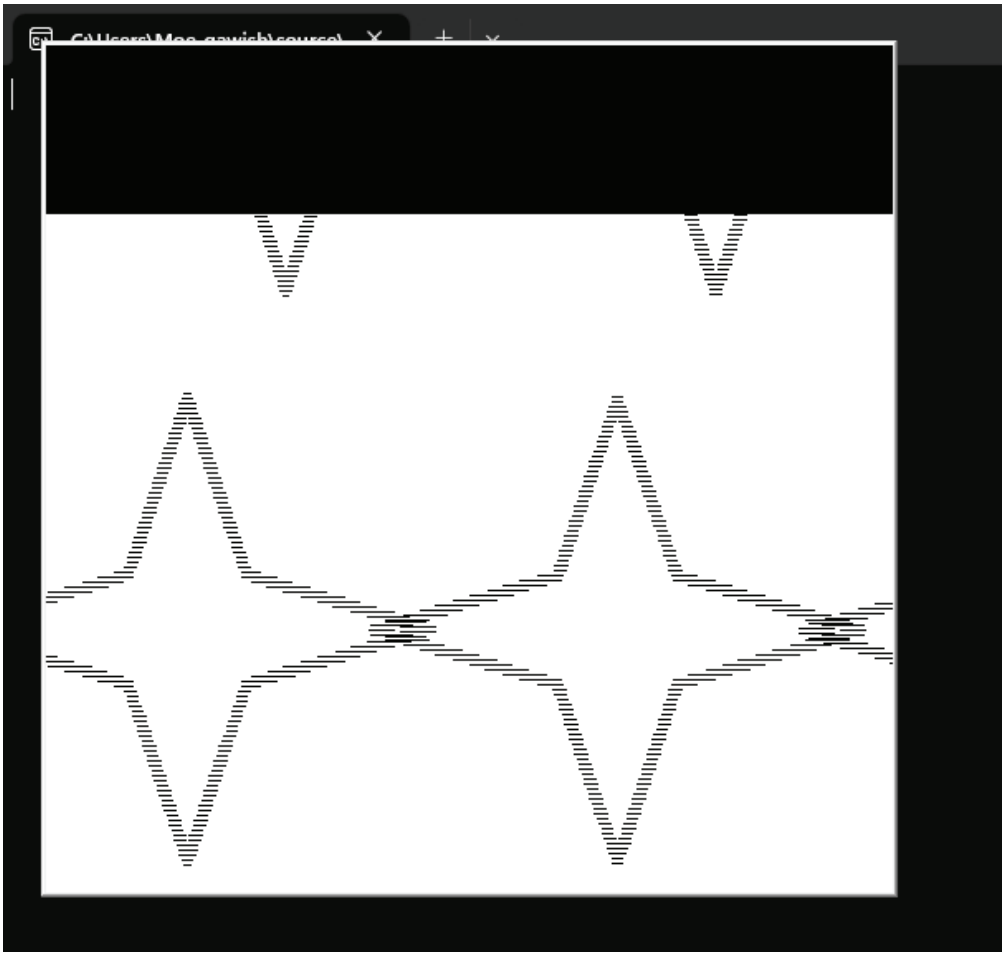
Код:

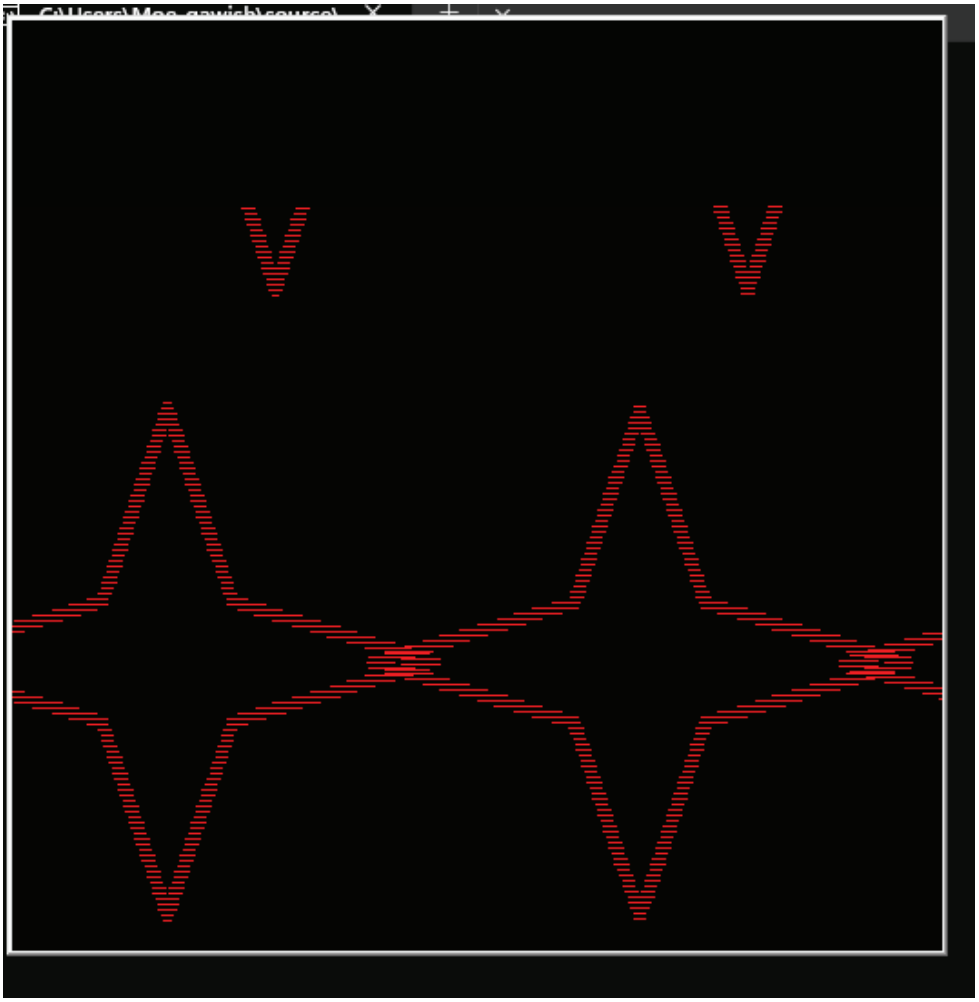
```
<include <gl\glut.h#
<include <math.h#
<Semester\Computer Graphic\Комп.графика\glaux.h 4\course 2\include <E:\study#
<include <stdio.h#
Semester\\Computer Graphic\\ 4\\course 2\\lib, "E:\\study)pragma comment#
("Комп.графика\\GLAUX.lib
pragma comment(lib, "legacy_stdio_definitions.lib")#
/*Структура для хранения заголовка файла изображения*/
struct Zagolovok
}
/*GLint shirina; /*Ширина
/*GLint vysota; /*Высота
/*GLenum formatCveta; /*Формат представления цвета
/*GLenum formatKomponenty; /*Формат данных компоненты цвета
/*int kol_voKomponent; /*Количество компонент цвета
;{
/*Структура для хранения изображения*/
struct Izobrajenie
}
;unsigned char* pikseli
;Zagolovok zagolovok
;{
/*Функция чтения изображения из фалйа .kai*/
Izobrajenie* ChtenieIzobrajeniyaIzFaila(const char* szFileName)
}
FILE* pFile; // Указатель файла
Izobrajenie* iz = (Izobrajenie*)malloc(sizeof(Izobrajenie)); //Создание структуры для
хранения изображения
Открытие файла //
;fopen_s(&pFile, "szFileName", "rb")
;if (pFile == NULL) return NULL
Считываем заголовок //
;(pFile ,1 ,sizeof(Zagolovok) ,(iz->zagolovok)&)fread
Создание массива для битов//
iz->pikseli = (unsigned char*)malloc(iz->zagolovok.shirina * iz->zagolovok.vysota *
;iz->zagolovok.kol_voKomponent)
Считывание битов //
iz->pikseli, iz->zagolovok.shirina * iz->zagolovok.vysota *)fread
;(pFile ,1 ,iz->zagolovok.kol_voKomponent
Работа с файлом завершается //
;fclose(pFile)
Возвращает указатель на данные изображения //
;return iz
{
/*Функция записи изображения в файл*/
GLint SohranenieIzobrajeniyaVfail(const char* szFileName)
}
FILE* pFile; // Указатель файла
Создание структуры для хранения изображения//
Izobrajenie* iz = (Izobrajenie*)malloc(sizeof(Izobrajenie)); GLint
Массив для хранения размеров порта просмотра// ;[4]iViewport
Получение размеров порта просмотра //
;[4]GLint iViewport//
;glGetIntegerv(GL_VIEWPORT, iViewport)
```

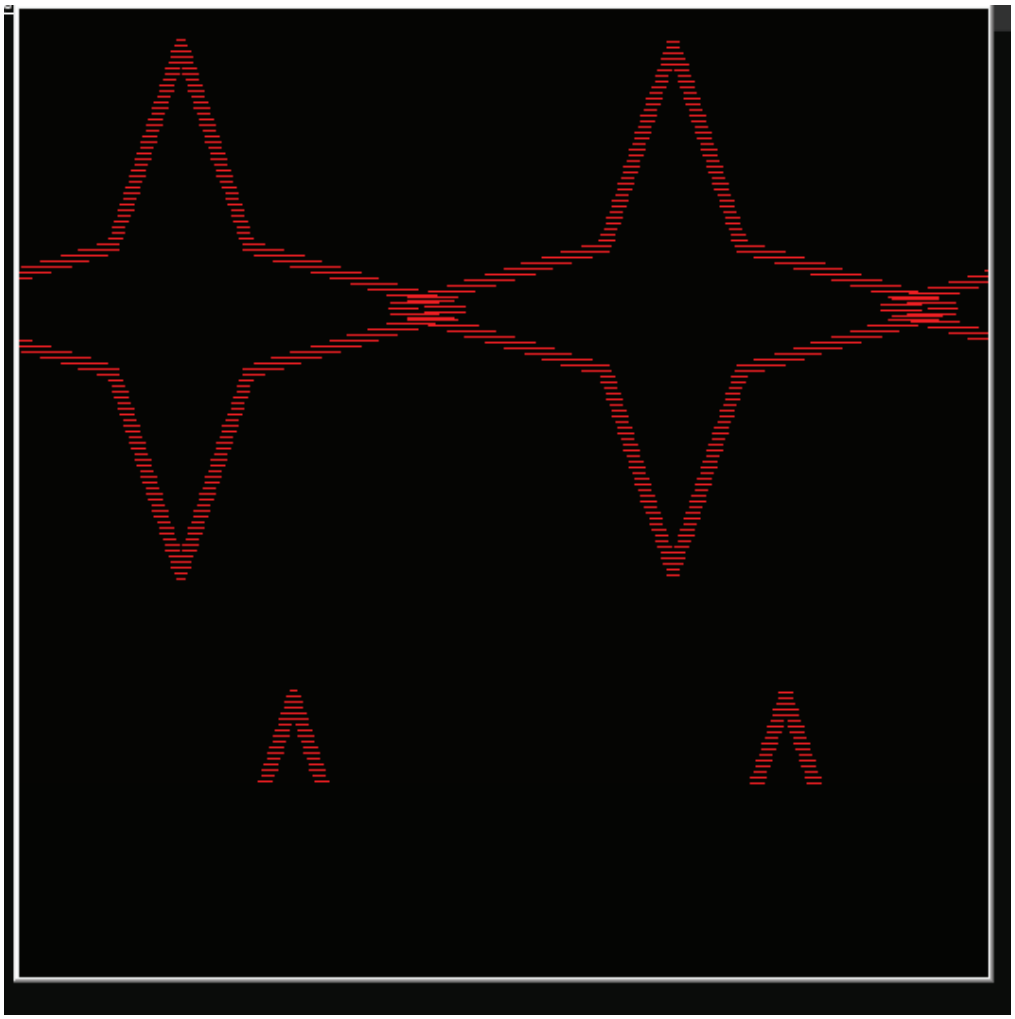
Считывание битов из буфера цвета //
;(2 ,GL_PACK_ALIGNMENT)glPixelStorei
;(0 ,GL_PACK_ROW_LENGTH)glPixelStorei
;(0 ,GL_PACK_SKIP_ROWS)glPixelStorei
;(0 ,GL_PACK_SKIP_PIXELS)glPixelStorei
Переключение на передний буфер //
;glReadBuffer(GL_FRONT)
Установка параметров изображения//
;[2]iz->zagolovok.shirina = iViewport
;[3]iz->zagolovok.vysota = iViewport
;iz->zagolovok.formatCveta = GL_RGB
;iz->zagolovok.formatKomponenty = GL_UNSIGNED_BYTE
;3 = iz->zagolovok.kol_voKomponent
Выделение памяти для хранения битов//
iz->pikseli = (unsigned char*)malloc(iz->zagolovok.shirina * iz->zagolovok.vysota *
;iz->zagolovok.kol_voKomponent)
Чтение битов//
iz->zagolovok.shirina, iz->zagolovok.vysota, ,0 ,0)glReadPixels
;(iz->zagolovok.formatCveta, iz->zagolovok.formatKomponenty, iz->pikseli
Откpытие файла //
;fopen_s(&pFile, "szFileName", "wb")
Запись заголовка //
;(pFile ,1 ,sizeof(Zagolovok) ,(iz->zagolovok)&)fwrite
Запись данных об изображении //
iz->pikseli, iz->zagolovok.shirina * iz->zagolovok.vysota *)fwrite
;(pFile ,1 ,iz->zagolovok.kol_voKomponent
Закрытие файла//
;fclose(pFile)
Успех //
;1 return
{
Izobrajenie* izobr; //Текущее изображение
Izobrajenie* izobr_bmp; //Изображение из файла .bmp
Izobrajenie* izobr_kai; //Изображение из файла .kai
Izobrajenie* izobr_ch_b; //Черно-белое изображение
AUX_RGBImageRec* pImage = NULL; //Изображение AUX_RGB
Переменная для хранения режима рисования//
;1 = static GLint rejim
Должным образом обновляем флаги в ответ на выбор позиции из меню //
void ObrabotkaMenu(int punktMenu)
}
,меняем индекс режима визуализации на индекс //
соответствующий позиции меню //
;rejim = punktMenu
{ ;()Активизируем перерисовывание изображения glutPostRisplay //
;()glutPostRedisplay
{
;0 = int k; int iz
void Pererisovka(void)
}
;[4]GLint iViewport
;GLbyte* pModifiedBytes = NULL
;[256]GLfloat invertMap
;GLint i
Очищаем окно текущим цветом очистки //
;glClear(GL_COLOR_BUFFER_BIT)
Текущее растровое положение всегда соответствует левому нижнему углу окна //
glRasterPos2i

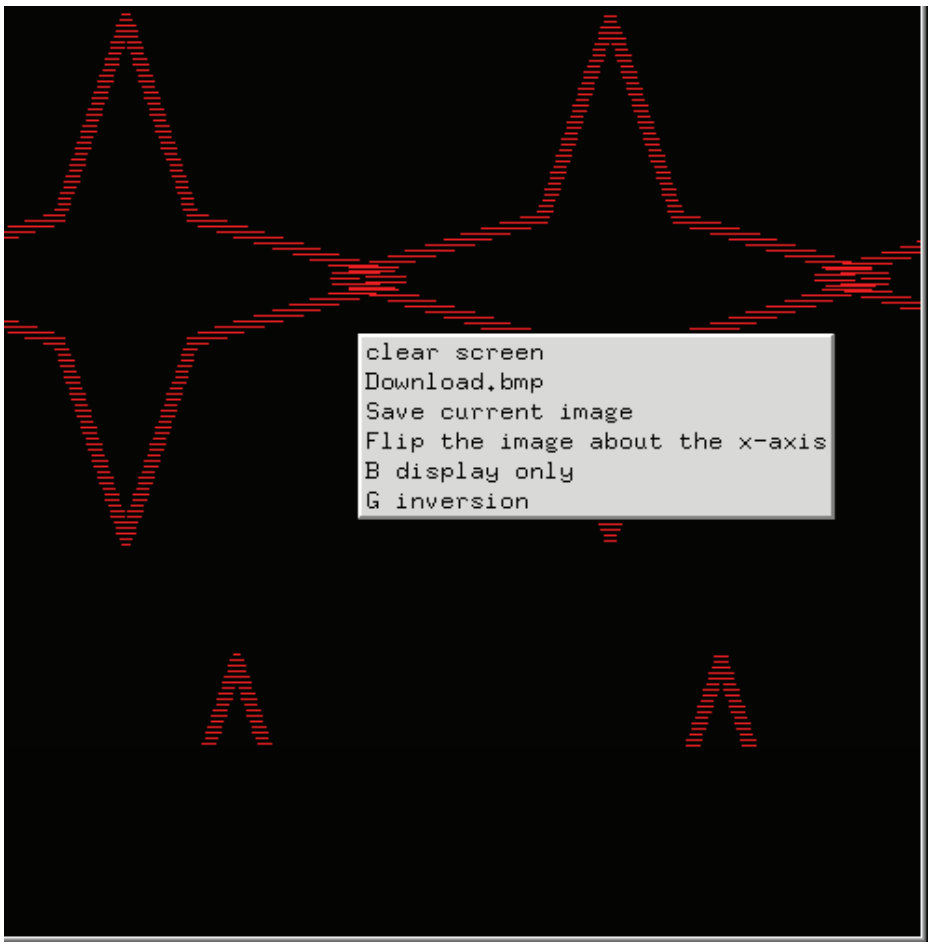
```
;(0 ,0)
В зависимости от индекса режима визуализации выполняются необходимые //
операции с изображением
switch (rejim)
}
/*Очистка экрана*/ :0 case
;(0.0f ,0.0f ,0.0f ,0.0f)glClearColor
;break
/*Загрузка изображения из файла *.bmp*/ :1 case
Semester/Computer 4/course 2/E:/study")pImage = auxDIBImageLoadA
;("bmp.4_Graphic/Комп.графика/texture
;izobr_bmp = (Izobrajenie*)malloc(sizeof(Izobrajenie))
;izobr_bmp->zagolovok.shirina = pImage->sizeX
;izobr_bmp->zagolovok.vysota = pImage->sizeY
;izobr_bmp->pikseli = pImage->data
;izobr_bmp->zagolovok.formatCveta = GL_LUMINANCE_ALPHA
;izobr_bmp->zagolovok.formatKomponenty = GL_UNSIGNED_BYTE
;izobr = izobr_bmp ;3 = izobr_bmp->zagolovok.kol_voKomponent
;break
:2 case
;(1 ,GL_PACK_ALIGNMENT)glPixelStorei
;SohranenieIzobrajeniyaVfail("C:/dorasave.bmp")
;break
:3 case
задает коэффициенты растяжения и сжатия для// ;(1- ,1)glPixelZoom
операций записи пикселей
задает текущую позицию растра// ;(512 ,0)glRasterPos2d
;break
:4 case
;(1.0f ,GL_GREEN_SCALE)glPixelTransferf
;(1.0f ,GL_BLUE_SCALE)glPixelTransferf
;break
/*Инверсия цветов*/ :5 case
;1.0f = [0]invertMap
(++i ;256 > i ;1 = i) for
;(255.0f * (GLfloat)i / 1.0f) - 1.0f = invertMap[i]
загружает// ;(invertMap ,255 ,GL_PIXEL_MAP_R_TO_R)glPixelMapfv
пиксельную карту (модифицикация с помощью таблиц подстановки)
;(invertMap ,0 ,GL_PIXEL_MAP_G_TO_G)glPixelMapfv
;(invertMap ,0 ,GL_PIXEL_MAP_B_TO_B)glPixelMapfv
;glPixelTransferi(GL_MAP_COLOR, GL_TRUE)
;break

:default
;break
{
(0 =! rejim) if
Рисуются пиксели //
}
;(1 ,GL_UNPACK_ALIGNMENT)glPixelStorei
glDrawPixels(izobr->zagolovok.shirina, izobr->zagolovok.vysota,
;izobr->zagolovok.formatCveta, izobr->zagolovok.formatKomponenty, izobr->pikseli)
{
Переключает буферы//
;()glutSwapBuffers
{
void IzmenenieRazmera(int w, int h)
```

```
}
Предотвращает деление на ноль, когда окно слишком маленькое//
(0 == h) if
;1 = h
;(w, h ,0 ,0)glViewport
Система координат обновляется перед модификацией//
;glMatrixMode(GL_PROJECTION)
;()glLoadIdentity
{
Точка входа основной программы//
int main(int argc, char* argv[])
}
;glutInit(&argc, argv)
;glutInitDisplayMode(GLUT_RGB | GL_DOUBLE)
;(512 ,512)glutInitWindowSize
;glutCreateWindow("Операции над пикселями")
;glutReshapeFunc(IzmenenieRazmera)
;glutDisplayFunc(Pererisovka)
Создается меню и добавляются опции выбора//
;(0 ,"clear screen")glutCreateMenu(ObrabotkaMenu); glutAddMenuEntry
;(1 ,"Download.bmp")glutAddMenuEntry
;(2 ,"Save current image")glutAddMenuEntry
;(3 ,"Flip the image about the x-axis")glutAddMenuEntry
;(4 ,"B display only")glutAddMenuEntry
;(5 ,"G inversion")glutAddMenuEntry
;glutAttachMenu(GLUT_RIGHT_BUTTON)
;(512 ,0 ,512 ,0)gluOrtho2D
;(0.0f ,0.0f ,0.0f ,3.0f)glClearColor
;0 = rejim
;()glutMainLoop
Освобождаем исходные данные изображений //
;free(izobr)
;free(izobr_ch_b)
;free(izobr_kai)
;free(izobr_bmp)
;free(pImage)
;0 return
{
```








clear screen
Download.bmp
Save current image
Flip the image about the x-axis
B display only
G inversion