МИНОБРНАУКИ РОССИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«Череповецкий государственный университет»

**Лабораторная работа № 3**

**«Текстурирование трехмерных изображений с использованием библиотеки OpenGL»**

**Выполнил:**

студент гр. 1ИВТпб-01-31оп

Климов А.Г.

**Проверил:** преподаватель

Юдина О.В., доцент  
Отметка о зачете:

Череповец

2017 год

**Цель работы**

Цель работы: изучить основные процедуры и функции при текстурировании изображений, приобрести практические навыки наложения текстуры на объемную фигуру.

**Задание**

Написать программу вращения куба, каждая грань которого текстурирована, на фоне фотографического изображения. Направления вращения задается пользователем. При этом каждая грань должна быть текстурирована разными изображениями. Требуется продемонстрировать текстурирование с повторением и без него. Для текстурирования выбрать растровое изображение.

**Текст программы**

#include <GL/glut.h>

#include <GL/GLAux.h>

unsigned int textures[7];

float angle\_x = 0;

float angle\_y = 0;

void LoadTextures()

{

AUX\_RGBImageRec \*texture1=auxDIBImageLoadA("1.bmp"),

\*texture2 = auxDIBImageLoadA("2.bmp"),

\*texture3 = auxDIBImageLoadA("3.bmp"),

\*texture4 = auxDIBImageLoadA("4.bmp"),

\*texture5 = auxDIBImageLoadA("5.bmp"),

\*texture6 = auxDIBImageLoadA("6.bmp"),

\*texture7 = auxDIBImageLoadA("7.bmp");

glGenTextures(7, &textures[0]);

glBindTexture(GL\_TEXTURE\_2D, textures[0]);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_S, GL\_REPEAT);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_T, GL\_REPEAT);

glTexImage2D(GL\_TEXTURE\_2D, 0, 3, texture1->sizeX, texture1->sizeY, 0,

GL\_RGB, GL\_UNSIGNED\_BYTE, texture1->data);

glBindTexture(GL\_TEXTURE\_2D, textures[1]);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR);

glTexImage2D(GL\_TEXTURE\_2D, 0, 3, texture2->sizeX, texture2->sizeY, 0,

GL\_RGB, GL\_UNSIGNED\_BYTE, texture2->data);

glBindTexture(GL\_TEXTURE\_2D, textures[2]);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR);

glTexImage2D(GL\_TEXTURE\_2D, 0, 3, texture3->sizeX, texture3->sizeY, 0,

GL\_RGB, GL\_UNSIGNED\_BYTE, texture3->data);

glBindTexture(GL\_TEXTURE\_2D, textures[3]);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR);

glTexImage2D(GL\_TEXTURE\_2D, 0, 3, texture4->sizeX, texture4->sizeY, 0,

GL\_RGB, GL\_UNSIGNED\_BYTE, texture4->data);

glBindTexture(GL\_TEXTURE\_2D, textures[4]);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR);

glTexImage2D(GL\_TEXTURE\_2D, 0, 3, texture5->sizeX, texture5->sizeY, 0,

GL\_RGB, GL\_UNSIGNED\_BYTE, texture5->data);

glBindTexture(GL\_TEXTURE\_2D, textures[5]);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR);

glTexImage2D(GL\_TEXTURE\_2D, 0, 3, texture6->sizeX, texture6->sizeY, 0,

GL\_RGB, GL\_UNSIGNED\_BYTE, texture6->data);

glBindTexture(GL\_TEXTURE\_2D, textures[6]);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR);

glTexImage2D(GL\_TEXTURE\_2D, 0, 3, texture7->sizeX, texture7->sizeY, 0,

GL\_RGB, GL\_UNSIGNED\_BYTE, texture7->data);

}

void Draw()

{

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

glLoadIdentity();

glBindTexture(GL\_TEXTURE\_2D, textures[0]);

glBegin(GL\_QUADS);

// Фон

glTexCoord2d(5, 5); glVertex3d(0, -5, -1);

glTexCoord2d(5, 10); glVertex3d(0, 5, -1);

glTexCoord2d(10, 10); glVertex3d(5, 5, -1);

glTexCoord2d(10, 5); glVertex3d(5, -5, -1);

glEnd();

glBindTexture(GL\_TEXTURE\_2D, textures[0]);

glBegin(GL\_QUADS);

// Фон

glTexCoord2d(5, 5); glVertex3d(-5, -5, -1);

glTexCoord2d(5, 10); glVertex3d(-5, 5, -1);

glTexCoord2d(10, 10); glVertex3d(0, 5, -1);

glTexCoord2d(10, 5); glVertex3d(0, -5, -1);

glEnd();

glRotatef(angle\_x, 0.0, 1.0, 0.0);

glRotatef(angle\_y, 1.0, 0.0, 0.0);

//Лицевая грань

glBindTexture(GL\_TEXTURE\_2D, textures[1]);

glBegin(GL\_QUADS);

glTexCoord2f(0.0f, 0.0f); glVertex3f(-0.5f, -0.5f, 0.5f);

glTexCoord2f(1.0f, 0.0f); glVertex3f(0.5f, -0.5f, 0.5f);

glTexCoord2f(1.0f, 1.0f); glVertex3f(0.5f, 0.5f, 0.5f);

glTexCoord2f(0.0f, 1.0f); glVertex3f(-0.5f, 0.5f, 0.5f);

glEnd();

// Задняя грань

glBindTexture(GL\_TEXTURE\_2D, textures[2]);

glBegin(GL\_QUADS);

glTexCoord2f(1.0f, 0.0f); glVertex3f(-0.5f, -0.5f, -0.5f);

glTexCoord2f(1.0f, 1.0f); glVertex3f(-0.5f, 0.5f, -0.5f);

glTexCoord2f(0.0f, 1.0f); glVertex3f(0.5f, 0.5f, -0.5f);

glTexCoord2f(0.0f, 0.0f); glVertex3f(0.5f, -0.5f, -0.5f);

glEnd();

// Верхняя грань

glBindTexture(GL\_TEXTURE\_2D, textures[3]);

glBegin(GL\_QUADS);

glTexCoord2f(0.0f, 1.0f); glVertex3f(-0.5f, 0.5f, -0.5f);

glTexCoord2f(0.0f, 0.0f); glVertex3f(-0.5f, 0.5f, 0.5f);

glTexCoord2f(1.0f, 0.0f); glVertex3f(0.5f, 0.5f, 0.5f);

glTexCoord2f(1.0f, 1.0f); glVertex3f(0.5f, 0.5f, -0.5f);

glEnd();

// Нижняя грань

glBindTexture(GL\_TEXTURE\_2D, textures[4]);

glBegin(GL\_QUADS);

glTexCoord2f(1.0f, 1.0f); glVertex3f(-0.5f, -0.5f, -0.5f);

glTexCoord2f(0.0f, 1.0f); glVertex3f(0.5f, -0.5f, -0.5f);

glTexCoord2f(0.0f, 0.0f); glVertex3f(0.5f, -0.5f, 0.5f);

glTexCoord2f(1.0f, 0.0f); glVertex3f(-0.5f, -0.5f, 0.5f);

glEnd();

// Правая грань

glBindTexture(GL\_TEXTURE\_2D, textures[5]);

glBegin(GL\_QUADS);

glTexCoord2f(1.0f, 0.0f); glVertex3f(0.5f, -0.5f, -0.5f);

glTexCoord2f(1.0f, 1.0f); glVertex3f(0.5f, 0.5f, -0.5f);

glTexCoord2f(0.0f, 1.0f); glVertex3f(0.5f, 0.5f, 0.5f);

glTexCoord2f(0.0f, 0.0f); glVertex3f(0.5f, -0.5f, 0.5f);

glEnd();

// Левая грань

glBindTexture(GL\_TEXTURE\_2D, textures[6]);

glBegin(GL\_QUADS);

glTexCoord2f(0.0f, 0.0f); glVertex3f(-0.5f, -0.5f, -0.5f);

glTexCoord2f(1.0f, 0.0f); glVertex3f(-0.5f, -0.5f, 0.5f);

glTexCoord2f(1.0f, 1.0f); glVertex3f(-0.5f, 0.5f, 0.5f);

glTexCoord2f(0.0f, 1.0f); glVertex3f(-0.5f, 0.5f, -0.5f);

glEnd();

glutSwapBuffers();

}

void Initialize()

{

LoadTextures();

glEnable(GL\_TEXTURE\_2D);

glViewport(0, 0, 400, 400);

glClearColor(1.0, 1.0, 1.0, 1.0);

glClearDepth(1.0);

glDepthFunc(GL\_LESS);

glEnable(GL\_DEPTH\_TEST);

glShadeModel(GL\_SMOOTH);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(60.0f, 400 / 400, 1, 1000000.0);

gluLookAt(0, 0, 3, 0, 0, 0, 0, 1, 0);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

}

void Keyboard(int key, int x, int y)

{

switch (key)

{

case GLUT\_KEY\_UP: angle\_y -= 5;

break;

case GLUT\_KEY\_DOWN: angle\_y += 5;

break;

case GLUT\_KEY\_RIGHT: angle\_x -= 5;

break;

case GLUT\_KEY\_LEFT: angle\_x += 5;

break;

}

// Запрос обновления экрана

glutPostRedisplay();

}

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

{

//Инициализация

glutInit(&argc, argv);

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

glutInitWindowSize(400, 400);

glutInitWindowPosition(100, 200);

glutCreateWindow("Лаборатоная работа 3");

glEnable(GL\_DEPTH\_TEST);

//Регистрация

glutDisplayFunc(Draw); // Рисование

glutSpecialFunc(Keyboard);

Initialize();

glutMainLoop();

return 0;

}

**Контрольные вопросы**

1. *Дайте определение текстуры.*

Текстура - это одномерное или двумерное изображение поверхности.

1. *В каком формате хранятся изображения в OpenGL?*

В библиотеке OpenGL имеется ряд функций для работы с текстурами. Поддержку текстурирования включают командой glEnable( GL\_TEXTURE\_2D). Так как библиотека рассчитана на работу под различными операционными системами, то она имеет собственный формат хранения изображений. Каждая точка храниться как 24 битное число. При этом первый байт - это красная составляющая, второй – зеленая составляющая, третий - синяя составляющая цвета. Текстуры могут иметь размер только кратный степени 2.

1. *Для чего необходимо заполнять структуру BITMAPINFO?*

Для преобразования изображения в системе Windows в формат, совместимый с библиотекой OpenGL, требуется выполнить следующие действия:

1. Загрузить изображение LoadImage
2. Подготовить структуру BITMAPINFO. Очистить ее нулем. В поле bmiHeader.biSize занести размер поля bmiHeader.
3. Получить параметры загруженного изображение (GetDIBits при задании количества линий 0).
4. Заполнить структуру BITMAPINFO:

bmiHeader.biPlanes=1; // 1 цветовая плоскость;

bmiHeader.biBitCount=24; // количество бит на цветов;

biTexture.bmiHeader.biCompression=BI\_RGB; // без компрессии;

biTexture.bmiHeader.biHeight=dwHeight; // установим требуемую высоту;

bmiHeader.biWidth=dwWidth; // установим требуемую ширину;

выделить память под данные. Размер выделяемой памяти можно вычислить как ширина\*высоту\* количество\_байт\_на\_цвет.

1. Получить данные изображения в требуемом формате (GetDIBits с передачей структуры BITMAPINFO)
2. Для каждого пиксела в полученных данных требуется поменять местами красную и синюю составляющую.
3. *Можно ли в изображении использовать несколько текстур?*

Библиотека OpenGL может работать в один момент времени только с одной текстурой, называемой активной. Для работы с несколькими текстурами необходимо задавать имя для текстуры. Затем по имени выбирается, какая из текстур будет использоваться.

1. Включить поддержку текстурирование glEnable( GL\_TEXTURE\_2D).
2. Сгенерировать имена для текстур glGenTextures.
3. Установить текущую текстуру (glBindTexture).
4. Загрузить изображение. Преобразовать в формат OpenGL.
5. Загрузить текстуру glTexImage2D.
6. Если есть еще текстуры, то перейти на п3.
7. Установить текущую текстуру (glBindTexture)
8. Отрисовать объект. Перед каждой вершиной задать координаты в системе координат текстуры. glTexCoord2d(0,0);glVertex3f(1,0,0);.

При текстурировании возможна ситуация, когда один пиксел объекта накрывает несколько пикселов текстуры или наоборот. В этом случае требуется определиться с методом аппроксимации. Для этого служит функция glTexParameteri. Существует 2 вида аппроксимаций: возможность выбора ближайшего пиксела (GL\_NEAREST) и линейная аппроксимация (GL\_LINEAR).

Иногда возникает потребность размножить текстуру по поверхности, например для реализации упорядоченных структур. Для этого создается текстура маленького размера, а затем устанавливаются следующие параметры:

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_S, GL\_REPEAT);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_T, GL\_REPEAT);

которые задают повторение по координатным осям текстуры S и T. При привязке координат текстуры значения координат S и T устанавливаются больше чем 1.

1. *Как реализуется прозрачность в OpenGL?*

В OpenGL прозрачность реализуется с помощью специального режима смешения цветов. Алгоритм смешения комбинирует цвета входящих пикселей (RGBA) с цветами соответствующих пикселей, уже хранящихся в буфере кадра. Режим включается с помощью команды glEnable(GL\_BLEND). Определить параметры смешения можно с помощью команды: glBlendFunc.

Коэффициент прозрачности задается alpha-компонентой цвета. Alpha=1 – непрозрачный объект; alpha=0 – невидимый. Для реализации вывода прозрачных объектов служит следующий код:

glEnable(GL\_BLEND);

glBlendFunc(GL\_SRC\_ALPHA,GL\_SRC\_ONE\_MINUS\_ALPHA);