Лабораторная работа №2 «Построение GLUT-приложения».

Как вы уже знаете, OpenGL является мультиплатформенной библиотекой, т.е. программы написанные с помощью OpenGL можно легко переносить на различные операционные системы, при этом получая один и тот же визуальный результат. Единственное что плохо - это то, что для конкретной операционной системы необходимо по своему производить настройку OpenGL. То есть, допустим, вы написали OpenGL программу под Windows и захотели перенести её в Linux, код OpenGL должен перенестись без проблем, но вот операции с окнами, интерфейс управления, операции с устройствами ввода/вывода нужно заново переписать уже под другую операционную систему - Linux.

К счастью, существует специальная мультиплатформенная библиотека, позволяющая решить вышеописанные проблемы. И называется эта библиотека - GLUT.GLUT-библиотека, в основном, отвечает за системный уровень операций ввода-вывода при работе с операционной системой. Из функций можно привести следующие: создание окна, управление окном, мониторинг за вводом с клавиатуры и событий мыши. Она также включает функции для рисования ряда геометрических примитивов: куб, сфера, чайник. GLUT даже включает возможность создания несложных всплывающих меню. Использование библиотеки GLUT преследует две цели. Во-первых, это создание кроссплатформенного кода. Во-вторых, GLUT позволяет облегчить изучение OpenGL, так как написание аналогичных вещей на API требует существенно большего времени и знания API управления окнами операционной системы.

Основные шаги для построения минимальной программы

Прежде всего необходимо создать окно, в котором можно будет рисовать с помощью OpenGL. Минимальная программа, которая создает окно и рисует в нем какие-либо объекты состоит из следующих шагов:

1. Инициализация GLUT
2. Установка параметров окна.
3. Создание окна.
4. Установка функций, отвечающих за рисование в окне и изменении формы окна.
5. Вход в главный цикл GLUT.

Инициализация GLUT производится командой:

```
void glutInit(int *argcp, char **argv);
```

Первый параметр представляет из себя указатель на количество аргументов в командной строке, а второй - указатель на массив аргументов. Обычно эти значения берутся из главной функции программы: int main(int argc, char *argv[]).

Установка параметров окна содержит в себе несколько этапов. Прежде всего необходимо указать размеры окна:

```
void glutInitWindowSize(int width, int height);
```

Первый параметр width - ширина окна в пикселях, второй height - высота окна в пикселях.

Далее можно задать положение создаваемого окна относительно верхнего левого угла экрана. Делается это командой:

```
void glutInitWindowPosition(int x, int y);
```

Необходимо также установить для окна режим отображения информации. Т.е. установить для окна такие параметры как: используемая цветовая модель, количество различных буферов, и т.д. Для этого в GLUT существует команда:

```
void glutInitDisplayMode(unsigned int mode);
```

У команды имеется единственный параметр, который может быть представлен одной из следующих констант или комбинацией этих констант с помощью побитового ИЛИ.

Константа	Значение
GLUT_RGB	Для отображения графической информации используются 3 компоненты цвета RGB.
GLUT_RGBA	То же что и RGB, но используется также 4 компонента ALPHA (прозрачность).
GLUT_INDEX	Цвет задается не с помощью RGB компонентов, а с помощью палитры. Используется для старых дисплеев, где количество цветов например 256.
GLUT_SINGLE	Вывод в окно осуществляется с использованием 1 буфера. Обычно используется для статического вывода информации.
GLUT_DOUBLE	Вывод в окно осуществляется с использованием 2 буферов. Применяется для анимации, чтобы исключить эффект мерцания.
GLUT_ACCUM	Использовать также буфер накопления (Accumulation Buffer). Этот буфер применяется для создания специальных эффектов, например отражения и тени.
GLUT_ALPHA	Использовать буфер ALPHA. Этот буфер, как уже говорилось используется для задания 4-го компонента цвета - ALPHA. Обычно применяется для таких эффектов как прозрачность объектов и антиалиасинг.
GLUT_DEPTH	Создать буфер глубины. Этот буфер используется для отсечения невидимых линий в 3D пространстве при выводе на плоский экран монитора.
GLUT_STENCIL	Буфер трафарета используется для таких эффектов как вырезание части фигуры, делая этот кусок прозрачным. Например, наложив прямоугольный трафарет на стену дома, вы получите окно, через которое можно увидеть что находится внутри дома.
GLUT_STEREO	Этот флаг используется для создания стереоизображений. Используется редко, так как для просмотра такого изображения нужна специальная аппаратура.

Вот пример использования этой команды:

```
void glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE);
```

3. Создание окна. После того как окно установлено необходимо его создать.

```
int glutCreateWindow(const char *title);
```

Эта команда создаёт окно с заголовком, который вы укажете в качестве параметра и возвращает HANDLER окна в виде числа int. Этот HANDLER обычно используется для последующих операций над этим окном, таких как изменение параметров окна и закрытие окна.

4. Установка функций, отвечающих за рисование в окне и изменении формы окна.

После того, как окно, в которое будет выводится графическая информация, подготовлено и создано, необходимо связать с ним процедуры, которые будут отвечать за вывод графической информации, следить за размерами окна, следить за нажатиями на клавиши и т.д. Самая первая и самая необходимая функция которую мы рассмотрим, отвечает за рисование. Именно она всегда будет вызываться операционной системой, чтобы нарисовать (перерисовать) содержимое окна. Итак, задаётся эта функция командой:

```
void glutDisplayFunc(void (*func)(void));
```

Единственный параметр этой функции - это указатель на функцию, которая будет отвечать за рисование в окне. Например чтобы функция void Draw(void), определенная в вашей программе отвечала за рисование в окне, надо присоединить ее к GLUT следующим образом: `glutDisplayFunc(Draw);`

И ещё одна важная функция - это функция, которая отслеживает изменения окна. Как только у окна изменились размеры, необходимо перестроить вывод графической информации уже в новое окно с другими размерами. Если этого не сделать, то например увеличив размеры окна, вывод информации будет производиться в старую область окна, с меньшими размерами. Определить функцию, отвечающую за изменение размеров окна нужно следующей командой:

```
void glutReshapeFunc(void (*func)(int width, int height));
```

Единственный параметр - это указатель на функцию, отвечающую за изменение размеров окна, которая как видно должна принимать два параметра width и height, соответственно ширина и высота нового (измененного) окна.

5. Вход в главный цикл GLUT.

Ну и последнее, что необходимо сделать, чтобы запустить программу - это войти в так называемый главный цикл GLUT. Этот цикл запускает на выполнение так называемое сердце GLUT, которое обеспечивает взаимосвязь между операционной системой и теми функциями, которые отвечают за окно, получают информацию от устройств ввода/вывода. Для того, чтобы перейти в главный цикл GLUT, надо выполнить единственную команду:

```
void glutMainLoop(void);
```

Связь между GLUT и OpenGL программой

Самой первой по значимости (без которой не обойтись ни в одной программе) идет функция отвечающая за перерисовку содержимого окна:

```
void glutDisplayFunc(void (*func)(void));
```

Эта функция устанавливает функцию в вашей программе, которая будет отвечать за перерисовку окна. Это нужно для того, что когда системе потребуется перерисовать ваше окно (например когда вы из свернутого состояния развернули окно), то она вызовет соответствующую функцию в вашей программе, которая ответственна за перерисовку содержимого окна. Также функция перерисовки окна вызывается при самом первом запуске вашего приложения. Отметим, что если вы создаете окно в GLUT, то определять функцию перерисовки окна нужно обязательно. С другой стороны, вы сами можете заставить систему перерисовать окно, это нужно например тогда, когда вы изменили состояния объектов или изменили их внешность, то вам понадобится перерисовать содержимое окна, чтобы отобразить

сделанные изменения. Поэтому в GLUT существует специальная функция glutPostRedisplay(), которая заставляет систему перерисовать текущее окно. Если вдруг вам захочется перерисовать не текущее окно, а какое-то другое, то предусмотрена следующая функция: glutPostWindowRedisplay(int win), которая перерисовывает содержимое win окна.

```
void glutReshapeFunc(void (*func)(int width, int height));
```

Как уже говорилось, это функция устанавливает другую функцию в вашей программе, которая будет ответственна за изменение размеров окна. Как только у окна изменились размеры, необходимо перестроить вывод графической информации уже в новое окно с другими размерами. Для этого в вызываемую функцию (т.е. функцию в вашей программе) передаются 2 параметра width и height, т.е. ширина и высота измененного окна. Обычно в самом начале функции, вызывается функция glViewport, например так: glViewport(0, 0, width, height), чтобы перестроить вывод изображения уже в окно с другими размерами.

Еще одна интересная функция, которая устанавливает функцию в вашей программе, которая будет вызываться системой всякий раз, когда окно становиться видимым или наоборот становиться невидимым.

```
void glutVisibilityFunc(void (*func)(int state));
```

Единственный параметр, который передается в вызываемую системой функцию: state. Сравнивая этот параметр с константами GLUT_NOT_VISIBLE и GLUT_VISIBLE можно определить, что произошло с окном, либо оно стало невидимым или наоборот - видимым.

```
void glutIdleFunc(void (*func)(void));
```

Функция, которая устанавливает функцию в вашей программе, которая будет вызываться системой всякий раз, когда ваше приложение простаивает. Часто эту функцию используют для анимации.

Ну и последняя функция, которую мы рассмотрим:

```
void glutTimerFunc(unsigned int millis, void (*func)(int value),
int value);
```

Эта функция устанавливает таймер. Первый параметр millis задает время в миллисекундах, по истечении которых вызывается функция, которая указана как второй параметр. Третий параметр value указывает идентификатор таймера, т.е. таймеров может быть одновременно запущено несколько. То же самое значение value есть и в функции, которая вызывается таймером. Это нужно для того, чтобы определить, какой таймер сработал, если ваша функция обрабатывает несколько таймеров.

Пример программы, которая при нажатии на клавиши ‘r’ ,‘g’, ‘b’ заливает все окно программы соответственно красным, голубым и зеленым цветом.

```
// Lab_02.cpp: определяет точку входа для консольного приложения.
//
#include "stdafx.h"
#include <windows.h>
#include <GL/gl.h>
#include <GL/glu.h>
#include <GL/glut.h>
```

```
#pragma comment (lib,"opengl32.lib")
#pragma comment (lib,"glu32.lib")

void reshape(int width,int height)
{
glViewport(0,0,(GLsizei) width, (GLsizei) height);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluOrtho2D(0.0,400.0,0.0,150.0);
}

void display(void)
{
       glClearColor(1.0,1.0,1.0,1.0);
       glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
 glBegin(GL_QUADS);
 glVertex2f(0,0);
 glVertex2f(0,400);
 glVertex2f(600,400);
 glVertex2f(600,0);
 glEnd();
glFinish();
}

void Keyboard(unsigned char key, int x, int y)
{
       switch (key)
       {case 'r':
              glColor3f(1,0,0);
              break;
       case 'g':
              glColor3f(0,1,0);
              break;
       case 'b':
              glColor3f(0,0,1);
              break;
              }
       glutPostRedisplay();
}

void main(int argc, char *argv[])
{
glutInit(&argc,argv);
glutInitDisplayMode(GLUT_RGB);
glutInitWindowSize(600,400);
glutInitWindowPosition(100,200);
glutCreateWindow("RGB");
glutDisplayFunc(display);
glClearColor(1.0,1.0,1.0,0.0);
glShadeModel(GL_FLAT);
glutDisplayFunc(display);
glutReshapeFunc(reshape);
glutKeyboardFunc(Keyboard);
glutMainLoop();
}
```

Задание:

1. Составить программу, генерирующую случайным образом заданное количество определенных объектов произвольного цвета. Тип и количество объектов определяется номером варианта.
2. Реализовать в программе возможность произвольного изменение цвета и положения всех объектов при нажатии на левую кнопку мыши.

Таблица 1. Изменения объектов к лабораторной работе № 2

№ вар	Задание	№ вар	Задание
1	Прямоугольники, горизонтальные и вертикальные размеры которых более 40, но менее 100 пикселей Количество -10	6	Параллелограммы, высота которых не более 35 пикселей и ширина не менее 30 и не более 50. Количество -8
2	Ромбы, со стороной от 20 до 80 пикселей Количество -12	7	Круги с радиусом от 30 до 55 пикселей Количество -10
3	Равнобедренные треугольники, высота которых не более 40 пикселей и ширина не менее 20. Количество -15	8	Правильные шестиугольники со стороной не более 55 пикселей. Количество -10
4	Прямоугольные треугольники с гипотенузой от 25 до 100 пикселей Количество -12	9	Равнобедренные трапеции, с основаниями не более 100 пикселей и боковыми сторонами не более 70 Количество -11
5	Правильные пятиугольники со стороной от 30 до 90 пикселей. Количество -8	10	Ромбы, со стороной от 20 до 60 пикселей, и меньшим углом от 30 до 90 градусов. Количество -10

Замечание: Для рисования отдельной фигуры необходимо создать новую функцию.