|  |  |  |
| --- | --- | --- |
| Министерство образования и науки Российской Федерации | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования | | |
| «Новосибирский государственный технический университет» | | |
| path817.png | | |
| Кафедра прикладной математики | | |
|  | | |
| Лабораторная работа № 1 | | |
| по дисциплине «Компьютерная графика» | | |
|  | | |
|  | | |
|  | | |
| эмблема_светлая.png | Факультет: | ПМИ |
| Группа: | ПМИ-62 |
| Бригада: | 4 |
| Студенты: | Мамонова Е. В., Ершов П. К. |
| Преподаватель: | Задорожный А. Г. |
|  | | |
| Новосибирск | | |
| 2019 | | |

1. Цель работы

Ознакомиться с основами использования библиотеки OpenGL и работе с примитивами.

1. **Задание**
   1. Отобразить в окне множество примитивов (вершины которых задаются кликами мыши) в соответствии с вариантом задания.
   2. Для завершения текущего (активного) набора (множества) примитивов и начала нового зарезервировать специальную клавишу (пробел или правый клик).
   3. Для текущего набора примитивов предоставить возможность изменения цвета и координат его вершин.
   4. Текущее множество примитивов выделять среди других, например, изменением размера его вершин командой glPointSize(\*).
   5. Использовать контейнер vector из библиотеки STL для хранения набора примитивов и множества вершин каждого примитива, а для хранения атрибутов рекомендуется использовать стандартный класс struct.
   6. Предусмотреть возможность удаления последнего примитива и последнего набора примитивов.
   7. Продублировать команды в меню, созданном с помощью библиотеки GLUT.
2. **Описание управления**

Меню включает в себя три основных пункта:

1. Тип линии.
2. Толщина линии.
3. Цвет линии.

Управление клавишами.

Перемещение группы:

W/w – вверх;

A/a – влево;

S/s – вниз;

D/d – вправо;

E/e – вверх и вправо;

Q/q – вверх и влево;

Z/z –вниз и влево;

C/c – вниз и вправо;

R – прибавить параметру RED 1;

G – прибавить параметру GREEN 1;

B – прибавить параметру BLUE 1;

R – отнять от параметра RED 1;

G – отнять от параметра GREEN 1;

B – отнять от параметра BLUE 1;

X/x – сохранить группу;

Пробел – создать новую группу;

Backspace – удалить примитив из группы;

U/u – прибавить параметру RED 10;

I/i – прибавить параметру GREEN 10;

O/o – прибавить параметру BLUE 10;

J/j – отнять от параметра RED 10;

K/k – отнять от параметра GREEN 10;

L/l – отнять от параметра BLUE 10;

1. Текст программы

pch.h

#ifndef PCH\_H

#define PCH\_H

#include <windows.h>

#include <gl/glut.h>

#include <vector>

#include <gl\gl.h>

#include <vector>

// константы для подменю тип линии

#define SOLID 1

#define DOT 2

#define DASH 3

#define DOTDASH 4

// константы для подменю толщина линии

#define WIDTH\_0\_5 5

#define WIDTH\_1 6

#define WIDTH\_1\_5 7

#define WIDTH\_2 8

#define WIDTH\_2\_5 9

#define WIDTH\_3 10

#define WIDTH\_3\_5 11

#define WIDTH\_4 12

// константы для подменю цвет линии

#define RED 13

#define BLUE 14

#define GREEN 15

#define ORANGE 16

#define YELLOW 17

#define VIOLET 18

#define BLACK 19

#define WHITE 20

GLint width, height; //ширина и высота окна

int current\_group; //текущий набор примитивов

bool drawing\_mode; //true - режим отрисовки

int max\_group; //максимальный порядковый номер набора (число наборов-1)

bool prev\_space; //true - был добавлен новый набор, но не отрисована ни одна линия

class point

{

public:

GLint x, y; // GLint - signed int

point(GLint p1, GLint p2);

};

struct line

{

public:

std::vector <point> coord;

int group;

};

class colour

{

public:

GLint red, blue, green;

int group;

colour(GLint r, GLint b, GLint g, int gr);

};

class linetype

{

public:

float widht;

unsigned short type;

int group;

linetype(float \_widht, unsigned short \_type, int \_gr);

};

point::point(GLint p1, GLint p2)

{

x = p1;

y = p2;

}

colour::colour(GLint r, GLint b, GLint g, int gr)

{

red = r;

blue = b;

green = g;

group = gr;

}

linetype::linetype(float \_widht, unsigned short \_type, int \_gr)

{

widht = \_widht;

type = \_type;

group = \_gr;

}

std::vector <line> v; //вектор ломаных

line l; //отрисовываемая ломаная

std::vector <colour> colours;

std::vector <linetype> lineTypeVec;

void Up\_Grup();//переход к следующему набору точек

void Down\_Grup();//переход к предыдущему набору точек

void New\_Grup();//новый набор точек

void Plus\_Red();//увеличить значение RED на 10

void Plus\_Green();//увеличить значение GREEN на 10

void Plus\_Blue();//увеличить значение BLUE на 10

void Minus\_Red();//уменьшить значение RED на 10

void Minus\_Green();//уменьшить значение GREEN на 10

void Minus\_Blue();//уменьшить значение BLUE на 10

void Del\_Group();//удалить группу

void Save\_Prim();//сохранить примитив

void Delete\_Last\_Prim();//удалить последний примитив

void MoveUp();//движение вверх

void MoveDown();//движение вниз

void MoveLeft();//движение влево

void MoveRight();//движение вправо

void MoveUpR();//движение в верхний правый угол

void MoveUpL();//движение в верхний левый угол

void MoveDownR();//движение в нижний правый угол

void MoveDownL();//движение в нижний левый угол

void param(int opt);//обраьботка всех параметров

void Color(unsigned char key);//обработка изменения цветов

void Menu(int opt);//меню

void Mouse(int button, int state, int x, int y);//обработка мыши

void Disp(void);//подключение дисплея

void ASCII\_Keys(unsigned char key, int x, int y);//обрабока ASCII исмволов

void NASCII\_Keys(int key, int x, int y);//обрабока спец исмволов

void Init();//начальные параметры

void Resh(GLint w, GLint h);//изменение параметров окна

#endif

pch.cpp

#include "pch.h"

void Up\_Grup()

{

char str[50];

//перейти к следующему набору

current\_group++;

//перейти к нулевому, если был достигнут последний добавленный набор

current\_group = current\_group % (max\_group + 1);

sprintf(str, "Current group: %i", current\_group + 1);

glutSetWindowTitle(str);

}

void Down\_Grup()

{

char str[50];

//перейти к предыдущему набору

current\_group--;

//перейти к последнему добавленному набору, если был пройден нулевой набор

if (current\_group < 0) current\_group = max\_group;

sprintf(str, "Current group: %i", current\_group + 1);

glutSetWindowTitle(str);

}

void New\_Grup()

{

char str[50];

if (!prev\_space) //исключает возможность добавления "пустого" набора

{

prev\_space = true; //был нажат пробел

max\_group++; //увеличить число наборов

current\_group = max\_group; //перейти к новому набору

colours.push\_back(colour(0, 0, 0, current\_group)); //установить изначальный (чёрный) цвет

lineTypeVec.push\_back(linetype(1.0, 0xFFFF, current\_group));

sprintf(str, "%i groups", max\_group + 1);

glutSetWindowTitle(str);

}

}

void Plus\_Red()

{

if (colours[current\_group].red + 10 < 255) colours[current\_group].red += 10;

}

void Plus\_Green()

{

if (colours[current\_group].green + 10 < 255) colours[current\_group].green += 10;

}

void Plus\_Blue()

{

if (colours[current\_group].blue + 10 < 255) colours[current\_group].blue += 10;

}

void Minus\_Red()

{

if (colours[current\_group].red - 10 >= 0) colours[current\_group].red -= 10;

}

void Minus\_Green()

{

if (colours[current\_group].green - 10 >= 0) colours[current\_group].green -= 10;

}

void Minus\_Blue()

{

if (colours[current\_group].blue - 10 >= 0) colours[current\_group].blue -= 10;

}

void Del\_Group()

{

char str[50];

int j, k;

k = v.size();

for (j = 0; j < k; j++)

{

//удалить все ломаные, принадлежащие текущему набору

if (v[j].group == current\_group)

{

v.erase(v.begin() + j);

k = v.size();

j--;

}

}

for (j = 0; j < v.size(); j++)

{

//уменьшить на 1 те номера наборов, которые были добавлены позже удаляемого

if (v[j].group > current\_group)

{

v[j].group--;

}

}

//удалить информацию о цвете текущего набора

colours.erase(colours.begin() + current\_group);

for (j = 0; j < colours.size(); j++)

{

//уменьшить на 1 те номера наборов, которые были добавлены позже удаляемого

if (colours[j].group > current\_group)

{

colours[j].group--;

}

}

//уменьшить максимальный номер наборов

//максимальный номер наборов не может быть меньше 0

if (max\_group) max\_group--;

//сменить текущий набор на следующий

current\_group = current\_group % (max\_group + 1);

//если были удалены все наборы, добавить цвет для нулевого набора

if (colours.size() == 0)

{

colours.push\_back(colour(0, 0, 0, current\_group));

lineTypeVec.push\_back(linetype(1.0, 0xFFFF, current\_group));

}

sprintf(str, "%i groups", max\_group + 1);

glutSetWindowTitle(str);

}

void Save\_Prim()

{

int k, j;

//добавить ломаную в вектор ломаных

v.push\_back(l);

//"выйти" из режима рисования

drawing\_mode = false;

//удалить данные о текущей ломаной

k = l.coord.size();

for (j = 0; j < k; j++)

{

l.coord.pop\_back();

}

}

void Delete\_Last\_Prim()

{

int k;

bool found, one\_in\_group;

k = v.size();

k--;

found = false;

one\_in\_group = true;

//просмотреть вектор ломаных с конца

for (k; k >= 0 && one\_in\_group; k--)

{

if (v[k].group == current\_group) //если ломаная принадлежит текущей группе

{

if (!found)

{

//удалить ломаную

v.erase(v.begin() + k);

//зафиксировать факт удаления ломаной

found = true;

}

else

{

//если в наборе еще есть ломаные, закончить цикл

one\_in\_group = false;

}

}

}

//если в наборе не осталось ломаных, удалить набор

if (one\_in\_group) Del\_Group();

}

void MoveUp()

{

int j, k;

int jj, l;

k = v.size();

for (j = 0; j < k; j++)

{

if (v[j].group == current\_group)

{

l = v[j].coord.size();

for (jj = 0; jj < l; jj++)

v[j].coord[jj].y += 3;

}

}

}

void MoveUpR()

{

int j, k;

int jj, l;

k = v.size();

for (j = 0; j < k; j++)

{

if (v[j].group == current\_group)

{

l = v[j].coord.size();

for (jj = 0; jj < l; jj++)

{

v[j].coord[jj].y += 3;

v[j].coord[jj].x += 3;

}

}

}

}

void MoveUpL()

{

int j, k;

int jj, l;

k = v.size();

for (j = 0; j < k; j++)

{

if (v[j].group == current\_group)

{

l = v[j].coord.size();

for (jj = 0; jj < l; jj++)

{

v[j].coord[jj].y += 3;

v[j].coord[jj].x -= 3;

}

}

}

}

void MoveDown()

{

int j, k;

int jj, l;

k = v.size();

for (j = 0; j < k; j++)

{

if (v[j].group == current\_group)

{

l = v[j].coord.size();

for (jj = 0; jj < l; jj++)

v[j].coord[jj].y -= 3;

}

}

}

void MoveDownR()

{

int j, k;

int jj, l;

k = v.size();

for (j = 0; j < k; j++)

{

if (v[j].group == current\_group)

{

l = v[j].coord.size();

for (jj = 0; jj < l; jj++)

{

v[j].coord[jj].y -= 3;

v[j].coord[jj].x += 3;

}

}

}

}

void MoveDownL()

{

int j, k;

int jj, l;

k = v.size();

for (j = 0; j < k; j++)

{

if (v[j].group == current\_group)

{

l = v[j].coord.size();

for (jj = 0; jj < l; jj++)

{

v[j].coord[jj].y -= 3;

v[j].coord[jj].x -= 3;

}

}

}

}

void MoveLeft()

{

int j, k;

int jj, l;

k = v.size();

for (j = 0; j < k; j++)

{

if (v[j].group == current\_group)

{

l = v[j].coord.size();

for (jj = 0; jj < l; jj++)

v[j].coord[jj].x -= 3;

}

}

}

void MoveRight()

{

int j, k;

int jj, l;

k = v.size();

for (j = 0; j < k; j++)

{

if (v[j].group == current\_group)

{

l = v[j].coord.size();

for (jj = 0; jj < l; jj++)

v[j].coord[jj].x += 3;

}

}

}

void Color(unsigned char key)//обработчик подменю цветов

{

if (key == 'R')

{

if (colours[current\_group].red < 256)

colours[current\_group].red += 1;

else

colours[current\_group].red = 0;

}

if (key == 'G')

{

if (colours[current\_group].green < 256)

colours[current\_group].green += 1;

else

colours[current\_group].green = 0;

}

if (key == 'B')

{

if (colours[current\_group].blue < 256)

colours[current\_group].blue += 1;

else

colours[current\_group].blue = 0;

}

if (key == 'r')

{

if (colours[current\_group].red > 0)

colours[current\_group].red -= 1;

else

colours[current\_group].red = 255;

}

if (key == 'g')

{

if (colours[current\_group].green > 0)

colours[current\_group].green -= 1;

else

colours[current\_group].green = 255;

}

if (key == 'b')

{

if (colours[current\_group].blue > 0)

colours[current\_group].blue -= 1;

else

colours[current\_group].blue = 255;

}

}

void param(int opt)

{

//обработка типа линий

if (opt == SOLID)

lineTypeVec[current\_group].type = 0xFFFF;

if (opt == DOT)

lineTypeVec[current\_group].type = 0x0101;

if (opt == DASH)

lineTypeVec[current\_group].type = 0x00F0;

if (opt == DOTDASH)

lineTypeVec[current\_group].type = 0x1C47;

//обработка толщины линий

if (opt == WIDTH\_0\_5)

lineTypeVec[current\_group].widht = 0.5;

if (opt == WIDTH\_1)

lineTypeVec[current\_group].widht = 1.0;

if (opt == WIDTH\_1\_5)

lineTypeVec[current\_group].widht = 1.5;

if (opt == WIDTH\_2)

lineTypeVec[current\_group].widht = 2.0;

if (opt == WIDTH\_2\_5)

lineTypeVec[current\_group].widht = 2.5;

if (opt == WIDTH\_3)

lineTypeVec[current\_group].widht = 3.0;

if (opt == WIDTH\_3\_5)

lineTypeVec[current\_group].widht = 3.5;

if (opt == WIDTH\_4)

lineTypeVec[current\_group].widht = 4.0;

if (opt == RED)

{

colours[current\_group].red = 252;

colours[current\_group].green = 48;

colours[current\_group].blue = 58;

int k, j;

//добавить ломаную в вектор ломаных

v.push\_back(l);

//"выйти" из режима рисования

drawing\_mode = false;

//удалить данные о текущей ломаной

k = l.coord.size();

for (j = 0; j < k; j++)

{

l.coord.pop\_back();

}

}

if (opt == GREEN)

{

colours[current\_group].red = 105;

colours[current\_group].green = 197;

colours[current\_group].blue = 22;

int k, j;

//добавить ломаную в вектор ломаных

v.push\_back(l);

//"выйти" из режима рисования

drawing\_mode = false;

//удалить данные о текущей ломаной

k = l.coord.size();

for (j = 0; j < k; j++)

{

l.coord.pop\_back();

}

}

if (opt == BLUE)

{

colours[current\_group].red = 42;

colours[current\_group].green = 128;

colours[current\_group].blue = 245;

int k, j;

//добавить ломаную в вектор ломаных

v.push\_back(l);

//"выйти" из режима рисования

drawing\_mode = false;

//удалить данные о текущей ломаной

k = l.coord.size();

for (j = 0; j < k; j++)

{

l.coord.pop\_back();

}

}

if (opt == ORANGE)

{

colours[current\_group].red = 246;

colours[current\_group].green = 138;

colours[current\_group].blue = 42;

int k, j;

//добавить ломаную в вектор ломаных

v.push\_back(l);

//"выйти" из режима рисования

drawing\_mode = false;

//удалить данные о текущей ломаной

k = l.coord.size();

for (j = 0; j < k; j++)

{

l.coord.pop\_back();

}

}

if (opt == YELLOW)

{

colours[current\_group].red = 246;

colours[current\_group].green = 224;

colours[current\_group].blue = 42;

int k, j;

//добавить ломаную в вектор ломаных

v.push\_back(l);

//"выйти" из режима рисования

drawing\_mode = false;

//удалить данные о текущей ломаной

k = l.coord.size();

for (j = 0; j < k; j++)

{

l.coord.pop\_back();

}

}

if (opt == VIOLET)

{

colours[current\_group].red = 182;

colours[current\_group].green = 97;

colours[current\_group].blue = 183;

int k, j;

//добавить ломаную в вектор ломаных

v.push\_back(l);

//"выйти" из режима рисования

drawing\_mode = false;

//удалить данные о текущей ломаной

k = l.coord.size();

for (j = 0; j < k; j++)

{

l.coord.pop\_back();

}

}

if (opt == BLACK)

{

colours[current\_group].red = 0;

colours[current\_group].green = 0;

colours[current\_group].blue = 0;

int k, j;

//добавить ломаную в вектор ломаных

v.push\_back(l);

//"выйти" из режима рисования

drawing\_mode = false;

//удалить данные о текущей ломаной

k = l.coord.size();

for (j = 0; j < k; j++)

{

l.coord.pop\_back();

}

}

if (opt == WHITE)

{

colours[current\_group].red = 255;

colours[current\_group].green = 255;

colours[current\_group].blue = 255;

int k, j;

//добавить ломаную в вектор ломаных

v.push\_back(l);

//"выйти" из режима рисования

drawing\_mode = false;

//удалить данные о текущей ломаной

k = l.coord.size();

for (j = 0; j < k; j++)

{

l.coord.pop\_back();

}

}

glutPostRedisplay();

}

void Menu(int opt)

{

int main\_menu;

int width\_menu;

int color\_menu;

int type\_menu;

switch (opt)

{

case SOLID:

param(SOLID); break;

case DOT:

param(DOT); break;

case DASH:

param(DASH); break;

case DOTDASH:

param(DOTDASH); break;

case WIDTH\_0\_5:

param(WIDTH\_0\_5); break;

case WIDTH\_1:

param(WIDTH\_1); break;

case WIDTH\_1\_5:

param(WIDTH\_1\_5); break;

case WIDTH\_2:

param(WIDTH\_2); break;

case WIDTH\_2\_5:

param(WIDTH\_2\_5); break;

case WIDTH\_3:

param(WIDTH\_3); break;

case WIDTH\_3\_5:

param(WIDTH\_3\_5); break;

case WIDTH\_4:

param(WIDTH\_4); break;

case RED:

param(RED); break;

case GREEN:

param(GREEN); break;

case BLUE:

param(BLUE); break;

case ORANGE:

param(ORANGE); break;

case YELLOW:

param(YELLOW); break;

case VIOLET:

param(VIOLET); break;

case BLACK:

param(BLACK); break;

case WHITE:

param(WHITE); break;

default:

//подменю тип линии

type\_menu = glutCreateMenu(Menu);

glutAddMenuEntry("Solid", SOLID);//добавить пункты подменю

glutAddMenuEntry("Dot", DOT);

glutAddMenuEntry("Dash", DASH);

glutAddMenuEntry("Dotdash", DOTDASH);

//подменю толщина линии

width\_menu = glutCreateMenu(Menu);

glutAddMenuEntry("0.5", WIDTH\_0\_5);//добавить пункты подменю

glutAddMenuEntry("1.0", WIDTH\_1);

glutAddMenuEntry("1.5", WIDTH\_1\_5);

glutAddMenuEntry("2.0", WIDTH\_2);

glutAddMenuEntry("2.5", WIDTH\_2\_5);

glutAddMenuEntry("3.0", WIDTH\_3);

glutAddMenuEntry("3.5", WIDTH\_3\_5);

glutAddMenuEntry("4.0", WIDTH\_4);

//подменю цвет

color\_menu = glutCreateMenu(Menu);

glutAddMenuEntry("Red", RED);//добавить пункты подменю

glutAddMenuEntry("Blue", BLUE);

glutAddMenuEntry("Green", GREEN);

glutAddMenuEntry("Orange", ORANGE);

glutAddMenuEntry("Yellow", YELLOW);

glutAddMenuEntry("Violet", VIOLET);

glutAddMenuEntry("Black", BLACK);

glutAddMenuEntry("White", WHITE);

//главное меню

main\_menu = glutCreateMenu(Menu);

glutAddSubMenu("Line type", type\_menu);//добавить подменю

glutAddSubMenu("Line thickness", width\_menu);

glutAddSubMenu("Line color", color\_menu);

glutAttachMenu(GLUT\_RIGHT\_BUTTON);//прикрепить меню к правой кнопке мыши

}

}

void Mouse(int button, int state, int x, int y)

{

char str[50];

if (state != GLUT\_DOWN) return; //если кнопка мыши не была отпущена

if (button == GLUT\_LEFT\_BUTTON)

{

if (!drawing\_mode)

{

//включить режим отрисовки

drawing\_mode = true;

//новая ломаная принадлежит текущему набору

l.group = current\_group;

sprintf(str, "%i groups", max\_group + 1);

glutSetWindowTitle(str);

}

//получить координаты точки

point p = point((GLint)x, height - (GLint)y);

//добавить их в ломаную

l.coord.push\_back(p);

//указать, что набор не пуст

prev\_space = false;

}

glutPostRedisplay();

}

void Disp(void)

{

int i, j;

glClear(GL\_COLOR\_BUFFER\_BIT); //установить цветовой режим буфера

glEnable(GL\_LINE\_SMOOTH); //установить режим сглаживания

glEnable(GL\_LINE\_STIPPLE); //наложение маски для создания штриховых линий

for (i = 0; i < v.size(); i++)

{

if (v[i].group == current\_group)//выделить ломаные текущего набора большей толщиной

{

glLineWidth(5.0);

}

else

{

glLineWidth(lineTypeVec[v[i].group].widht);

}

glLineStipple(3, lineTypeVec[v[i].group].type); //задать тип маски

glBegin(GL\_LINE\_LOOP);//тип примитива - замкнутая ломаная

for (j = 0; j < v[i].coord.size(); j++)

{

//установить цвет набора

glColor3ub(colours[v[i].group].red, colours[v[i].group].green, colours[v[i].group].blue);

//отрисовать ломаную

glVertex2i(v[i].coord[j].x, v[i].coord[j].y);

}

glEnd();

}

glDisable(GL\_LINE\_STIPPLE); //отключить маску

//если отрисовывается текущая ломаная

if (drawing\_mode)

{

//выделить ее большей толщиной

glLineWidth(5.0);

//установить цвет набора и отрисовать ломаную

glColor3ub(colours[l.group].red, colours[l.group].green, colours[l.group].blue);

glBegin(GL\_LINE\_LOOP);

for (j = 0; j < l.coord.size(); j++)

{

glVertex2i(l.coord[j].x, l.coord[j].y);

}

glEnd();

}

glutSwapBuffers();

}

void ASCII\_Keys(unsigned char key, int x, int y)//обработка клавиш

{

if (!drawing\_mode)

{

if (key == 32) //пробел

{

New\_Grup();

}

if (key == 127) //delete

{

Del\_Group();

}

if (key == 8) //backspace

{

Delete\_Last\_Prim();

}

//обработка движения

if (key == 'w' || key == 'W')

{

MoveUp();

}

if (key == 's' || key == 'S')

{

MoveDown();

}

if (key == 'a' || key == 'A')

{

MoveLeft();

}

if (key == 'd' || key == 'D')

{

MoveRight();

}

if (key == 'e' || key == 'E')

{

MoveUpR();

}

if (key == 'q' || key == 'Q')

{

MoveUpL();

}

if (key == 'c' || key == 'C')

{

MoveDownR();

}

if (key == 'z' || key == 'Z')

{

MoveDownL();

}

//обработка изменения цветов (+/-1)

if (key == 'r')

{

Color('r');

}

if (key == 'g')

{

Color('g');

}

if (key == 'b')

{

Color('b');

}

if (key == 'R')

{

Color('R');

}

if (key == 'G')

{

Color('G');

}

if (key == 'B')

{

Color('B');

}

}

//обработка изменения цветов (+/-10)

if (key == 'j' || key == 'J')

{

Minus\_Red();

}

if (key == 'k' || key == 'K')

{

Minus\_Green();

}

if (key == 'l' || key == 'L')

{

Minus\_Blue();

}

if (key == 'u' || key == 'U')

{

Plus\_Red();

}

if (key == 'i' || key == 'I')

{

Plus\_Green();

}

if (key == 'o' || key == 'O')

{

Plus\_Blue();

}

if (key == 'x' || key == 'X')

{

Save\_Prim();

}

char v[50]; sprintf(v, "Текущий цвет всех точек: R=%.3d G=%.3d B=%.3d", colours[current\_group].red, colours[current\_group].green, colours[current\_group].blue);

glutSetWindowTitle(v);

glutPostRedisplay();

}

void NASCII\_Keys(int key, int x, int y)

{

if (!drawing\_mode)

{

if (key == GLUT\_KEY\_UP)

{

Up\_Grup();

}

if (key == GLUT\_KEY\_DOWN)

{

Down\_Grup();

}

}

glutPostRedisplay();

}

void Init()

{

//установить белый цвет экрана

glClearColor(1.0, 1.0, 1.0, 1);

//установить проекцию

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glViewport(0, 0, width, height);

gluOrtho2D(0.0, width, 0.0, height);

//установить начальные значения

drawing\_mode = false;

max\_group = 0;

current\_group = 0;

prev\_space = false;

//добавить "цвет по умолчанию" для нулевого набора

colours.push\_back(colour(0, 0, 0, current\_group));

//добавить "тип линии и толщину по умолчанию" для нулевого набора

lineTypeVec.push\_back(linetype(1.0, 0xFFFF, current\_group));

}

void Resh(GLint w, GLint h)

{

//изменить размеры окна

width = w;

height = h;

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glViewport(0, 0, width, height);

gluOrtho2D(0.0, width, 0.0, height);

}

Main.cpp

#include "pch.h"

#include <iostream>

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

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB); //один буфер отрисовки окна, использование всех компонент RGB

glutInitWindowSize(1300, 800);//уставить начальные размеры окна

glutCreateWindow("CG\_L1. GL\_LINE\_LOOP");

Init(); //начальные установки

Menu(0); //создание меню

glutKeyboardFunc(ASCII\_Keys); //обработка клавиш с кодами ascii

glutSpecialFunc(NASCII\_Keys); //обработка не-ascii клавиш

glutReshapeFunc(Resh);

glutDisplayFunc(Disp);

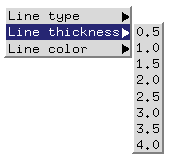
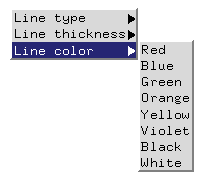
glutMouseFunc(Mouse);

glutMainLoop();

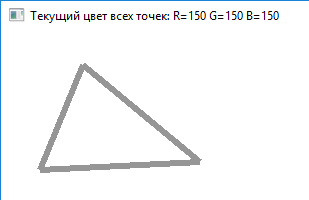
}

1. Тесты

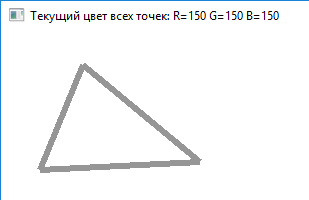
Тест 1. Пример меню.

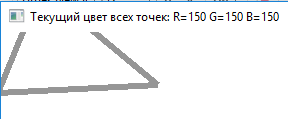


Тест 2. Пример группы.

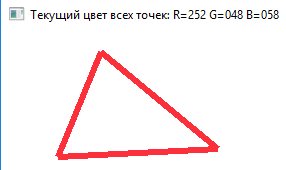


Тест 3. Движение группы.

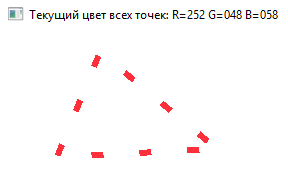
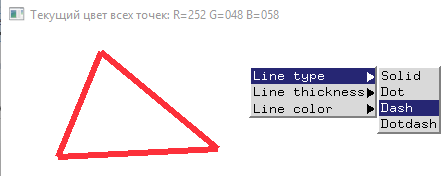




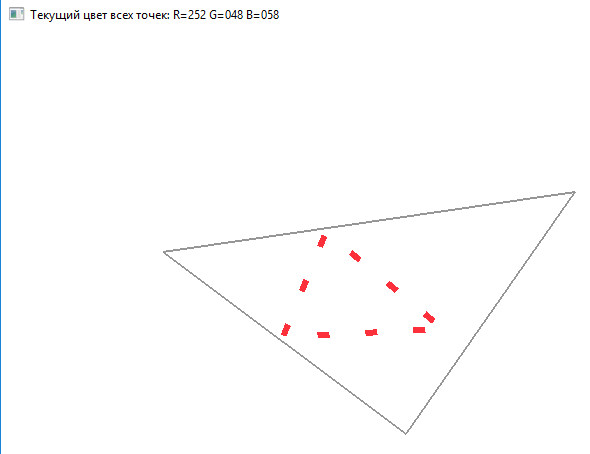
Тест 4. Изменение цвета.



Тест 5. Изменение типа линии.



Тест 6. Добавление нескольких групп.



Тест 7. Удаление примитива из группы.

