Карпенко О. КН20-1

**ЛАБОРАТОРНА РОБОТА 3**

Реалізувати клієнтську частину бібліотеки для управління дисплейним модулем.

Етапи виконання лабораторної роботи:

1. Використовуючи інтерфейс бібліотеки GraphicsLib, реалізувати на мові програмування C++ програму для взаємодії з емулятором дисплейного модуля, який був розроблений в лабораторній роботі №2.

2. Інтерфейс бібліотеки GraphicsLib не змінювати.

Хід Роботи:

Бібліотека:

GraphicsLib.h

#pragma once

#ifndef GraphicsLib\_h

#define GraphicsLib\_h

#include <stdint.h>

#ifndef R\_G\_B

#define R\_G\_B(r,g,b) (((r&0xF8)<<8)|((g&0xFC)<<3)|((b&0xF8)>>3)) //RGB565: 5 red | 6 green | 5 blue

#endif

class GraphicsLib

{

private:

int\_least16\_t width, height; //screen size

public:

GraphicsLib(uint\_least16\_t w, uint\_least16\_t h) : width(w), height(h) {};

GraphicsLib() : width(0), height(0) {};

int\_least16\_t getWidth(void) { return width; };

int\_least16\_t getHeight(void) { return height; };

virtual void fillScreen(uint\_least16\_t color) = 0;

virtual void drawPixel(int\_least16\_t x0, int\_least16\_t y0, uint\_least16\_t color) = 0;

virtual void drawLine(int\_least16\_t x0, int\_least16\_t y0, int\_least16\_t x1, int\_least16\_t y1, uint\_least16\_t color) = 0;

virtual void drawRect(int\_least16\_t x0, int\_least16\_t y0, int\_least16\_t w, int\_least16\_t h, uint\_least16\_t color) = 0;

virtual void fillRect(int\_least16\_t x0, int\_least16\_t y0, int\_least16\_t w, int\_least16\_t h, uint\_least16\_t color) = 0;

virtual void drawEllipse(int\_least16\_t x0, int\_least16\_t y0, int\_least16\_t r\_x, int\_least16\_t r\_y, uint\_least16\_t color) = 0;

virtual void fillEllipse(int\_least16\_t x0, int\_least16\_t y0, int\_least16\_t r\_x, int\_least16\_t r\_y, uint\_least16\_t color) = 0;

virtual int\_least16\_t drawChar(int\_least16\_t x, int\_least16\_t y, char c, uint\_least16\_t color, uint\_least16\_t bg, uint\_least8\_t size) = 0;

virtual int\_least16\_t drawChar(int\_least16\_t x, int\_least16\_t y, unsigned char c, uint\_least16\_t color, uint\_least16\_t bg, uint\_least8\_t size) = 0;

virtual int\_least16\_t drawText(int\_least16\_t x, int\_least16\_t y, const char\* s, uint\_least16\_t color, uint\_least16\_t bg, uint\_least8\_t size) = 0;

virtual void loadSprite(uint\_least8\_t index, int\_least16\_t width, int\_least16\_t height, char\* data) = 0;

virtual void showSprite(uint\_least8\_t index, uint\_least16\_t x, uint\_least16\_t y) = 0;

};

#endif //GraphicsLib\_h

GraphicsLib.cpp

#pragma comment(lib, "Ws2\_32.lib")

#pragma warning( disable : 4996)

#include <iostream>

#include "GraphicsLib.h"

#include <winsock2.h>

#include <string.h>

#include <vector>

#include <WS2tcpip.h>

#include <opencv2/highgui/highgui.hpp>

#include <opencv2/core/core.hpp>

using namespace cv;

using namespace std;

class graphicsLib : GraphicsLib

{

private:

int\_least16\_t width, height; //screen size

string ipaddr;

int port;

SOCKET out;

public:

int ClientStartup()

{

//Create and Open Socket

WSADATA wsaData;

int ws = WSAStartup(MAKEWORD(2, 2), &wsaData);

out = socket(AF\_INET, SOCK\_DGRAM, 0);

if (ws != 0)

{

cout << "Windows Socket Creation failed:" << socket;

return 1;

}

return 0;

}

uint\_least16\_t ColorInput()

{

int\_least16\_t r, g, b;

uint\_least16\_t output;

cout << "Input red value(0-255): \n";

cin >> r;

cout << "Input green value(0-255): \n";

cin >> g;

cout << "Input blue value(0-255): \n";

cin >> b;

output = R\_G\_B(r, g, b);

return output;

}

void ClientShutdown()

{

closesocket(out);

// Close down Winsock

WSACleanup();

}

// Write out to that socket

void SendMessageServer(vector<byte> v, int length)

{

sockaddr\_in serveraddr;

serveraddr.sin\_family = AF\_INET;

serveraddr.sin\_port = htons(port);

inet\_pton(AF\_INET, ipaddr.c\_str(), &serveraddr.sin\_addr);

int sendOk = sendto(out, (char\*)v.data(), length, 0, (sockaddr\*)&serveraddr, sizeof(serveraddr));

if (sendOk == SOCKET\_ERROR)

{

cout << "That didn't work! " << WSAGetLastError() << endl;

}

}

void SetScreenSize()

{

byte widthArr[] = { width & 0x00FF, width >> 8 };

byte heightArr[] = { height & 0x00FF, height >> 8 };

vector<byte> v;

v.resize(0);

v.push\_back(254);

v.push\_back(widthArr[0]);

v.push\_back(widthArr[1]);

v.push\_back(heightArr[0]);

v.push\_back(heightArr[1]);

SendMessageServer(v, v.size());

}

char\* ConvertImage(string path, short &width, short &height)

{

string pth(path);

Mat image = imread("D:/Data/Uni/Year3/Part1/CSCS/Lab3/Images/" + pth);

char\* arr = new char[image.rows \* image.cols \* 3];

int k = 0;

Vec3b buf;

width = image.cols;

height = image.rows;

for (int i = 0; i < width; i++)

{

for (int j = 0; j < height; j++)

{

buf = image.at<Vec3b>(j, i);

arr[k] = buf[2];

arr[k + 1] = buf[1];

arr[k + 2] = buf[0];

k += 3;

}

}

return arr;

}

graphicsLib(uint\_least16\_t \_width, uint\_least16\_t \_height) {

width = \_width;

height = \_height;

};

graphicsLib() {

width = 0;

height = 0;

};

int\_least16\_t getWidth(void) { return width; };

int\_least16\_t getHeight(void) { return height; };

string getIP(void) { return ipaddr; };

int getPort(void) { return port; };

void setIP(string \_ipaddr)

{

ipaddr = \_ipaddr;

}

void setPort(int \_port)

{

port = \_port;

}

void fillScreen(uint\_least16\_t color) override { //clear display

vector<byte> v;

v.resize(0);

v.push\_back(1);

uint\_least16\_t r = (color & 0xF800) >> 8;

uint\_least16\_t g = (color & 0x07E0) >> 3;

uint\_least16\_t b = (color & 0x001F) << 3;

v.push\_back(r);

v.push\_back(g);

v.push\_back(b);

SendMessageServer(v, v.size()) ;

}

void drawPixel(int\_least16\_t x0, int\_least16\_t y0, uint\_least16\_t color) override

{

byte x0Arr[] = { x0 & 0x00FF, x0 >> 8 };

byte y0Arr[] = { y0 & 0x00FF, y0 >> 8 };

vector<byte> v;

v.resize(0);

v.push\_back(2);

v.push\_back(x0Arr[0]);

v.push\_back(x0Arr[1]);

v.push\_back(y0Arr[0]);

v.push\_back(y0Arr[1]);

uint\_least16\_t r = (color & 0xF800) >> 8;

uint\_least16\_t g = (color & 0x07E0) >> 3;

uint\_least16\_t b = (color & 0x001F) << 3;

v.push\_back(r);

v.push\_back(g);

v.push\_back(b);

SendMessageServer(v, v.size());

}

void drawLine(int\_least16\_t x0, int\_least16\_t y0, int\_least16\_t x1, int\_least16\_t y1, uint\_least16\_t color) override

{

uint\_least16\_t r = (color & 0xF800) >> 8;

uint\_least16\_t g = (color & 0x07E0) >> 3;

uint\_least16\_t b = (color & 0x001F) << 3;

byte x0Arr[] = { x0 & 0x00FF, x0 >> 8 };

byte y0Arr[] = { y0 & 0x00FF, y0 >> 8 };

byte x1Arr[] = { x1 & 0x00FF, x1 >> 8 };

byte y1Arr[] = { y1 & 0x00FF, y1 >> 8 };

vector<byte> v;

v.resize(0);

v.push\_back(3);

v.push\_back(x0Arr[0]);

v.push\_back(x0Arr[1]);

v.push\_back(y0Arr[0]);

v.push\_back(y0Arr[1]);

v.push\_back(x1Arr[0]);

v.push\_back(x1Arr[1]);

v.push\_back(y1Arr[0]);

v.push\_back(y1Arr[1]);

v.push\_back(r);

v.push\_back(g);

v.push\_back(b);

SendMessageServer(v, v.size());

}

void drawRect(int\_least16\_t x0, int\_least16\_t y0, int\_least16\_t w, int\_least16\_t h, uint\_least16\_t color) override

{

uint\_least16\_t r = (color & 0xF800) >> 8;

uint\_least16\_t g = (color & 0x07E0) >> 3;

uint\_least16\_t b = (color & 0x001F) << 3;

byte x0Arr[] = { x0 & 0x00FF, x0 >> 8 };

byte y0Arr[] = { y0 & 0x00FF, y0 >> 8 };

byte wArr[] = { w & 0x00FF, w >> 8 };

byte hArr[] = { h & 0x00FF, h >> 8 };

vector<byte> v;

v.resize(0);

v.push\_back(4);

v.push\_back(x0Arr[0]);

v.push\_back(x0Arr[1]);

v.push\_back(y0Arr[0]);

v.push\_back(y0Arr[1]);

v.push\_back(wArr[0]);

v.push\_back(wArr[1]);

v.push\_back(hArr[0]);

v.push\_back(hArr[1]);

v.push\_back(r);

v.push\_back(g);

v.push\_back(b);

SendMessageServer(v, v.size());

}

void fillRect(int\_least16\_t x0, int\_least16\_t y0, int\_least16\_t w, int\_least16\_t h, uint\_least16\_t color) override

{

uint\_least16\_t r = (color & 0xF800) >> 8;

uint\_least16\_t g = (color & 0x07E0) >> 3;

uint\_least16\_t b = (color & 0x001F) << 3;

byte x0Arr[] = { x0 & 0x00FF, x0 >> 8 };

byte y0Arr[] = { y0 & 0x00FF, y0 >> 8 };

byte wArr[] = { w & 0x00FF, w >> 8 };

byte hArr[] = { h & 0x00FF, h >> 8 };

vector<byte> v;

v.resize(0);

v.push\_back(5);

v.push\_back(x0Arr[0]);

v.push\_back(x0Arr[1]);

v.push\_back(y0Arr[0]);

v.push\_back(y0Arr[1]);

v.push\_back(wArr[0]);

v.push\_back(wArr[1]);

v.push\_back(hArr[0]);

v.push\_back(hArr[1]);

v.push\_back(r);

v.push\_back(g);

v.push\_back(b);

SendMessageServer(v, v.size());

}

void drawEllipse(int\_least16\_t x0, int\_least16\_t y0, int\_least16\_t r\_x, int\_least16\_t r\_y, uint\_least16\_t color) override

{

uint\_least16\_t r = (color & 0xF800) >> 8;

uint\_least16\_t g = (color & 0x07E0) >> 3;

uint\_least16\_t b = (color & 0x001F) << 3;

byte x0Arr[] = { x0 & 0x00FF, x0 >> 8 };

byte y0Arr[] = { y0 & 0x00FF, y0 >> 8 };

byte r\_xArr[] = { r\_x & 0x00FF, r\_x >> 8 };

byte r\_yArr[] = { r\_y & 0x00FF, r\_y >> 8 };

vector<byte> v;

v.resize(0);

v.push\_back(6);

v.push\_back(x0Arr[0]);

v.push\_back(x0Arr[1]);

v.push\_back(y0Arr[0]);

v.push\_back(y0Arr[1]);

v.push\_back(r\_xArr[0]);

v.push\_back(r\_xArr[1]);

v.push\_back(r\_yArr[0]);

v.push\_back(r\_yArr[1]);

v.push\_back(r);

v.push\_back(g);

v.push\_back(b);

SendMessageServer(v, v.size());

}

void fillEllipse(int\_least16\_t x0, int\_least16\_t y0, int\_least16\_t r\_x, int\_least16\_t r\_y, uint\_least16\_t color) override

{

uint\_least16\_t r = (color & 0xF800) >> 8;

uint\_least16\_t g = (color & 0x07E0) >> 3;

uint\_least16\_t b = (color & 0x001F) << 3;

byte x0Arr[] = { x0 & 0x00FF, x0 >> 8 };

byte y0Arr[] = { y0 & 0x00FF, y0 >> 8 };

byte r\_xArr[] = { r\_x & 0x00FF, r\_x >> 8 };

byte r\_yArr[] = { r\_y & 0x00FF, r\_y >> 8 };

vector<byte> v;

v.resize(0);

v.push\_back(7);

v.push\_back(x0Arr[0]);

v.push\_back(x0Arr[1]);

v.push\_back(y0Arr[0]);

v.push\_back(y0Arr[1]);

v.push\_back(r\_xArr[0]);

v.push\_back(r\_xArr[1]);

v.push\_back(r\_yArr[0]);

v.push\_back(r\_yArr[1]);

v.push\_back(r);

v.push\_back(g);

v.push\_back(b);

SendMessageServer(v, v.size());

}

int\_least16\_t drawChar(int\_least16\_t x, int\_least16\_t y, char c, uint\_least16\_t color, uint\_least16\_t bg, uint\_least8\_t size) override

{

return x;

}

int\_least16\_t drawChar(int\_least16\_t x, int\_least16\_t y, unsigned char c, uint\_least16\_t color, uint\_least16\_t bg, uint\_least8\_t size) override

{

return x;

}

int\_least16\_t drawText(int\_least16\_t x, int\_least16\_t y, const char\* s, uint\_least16\_t color, uint\_least16\_t bg, uint\_least8\_t size) override

{

uint\_least16\_t r = (color & 0xF800) >> 8;

uint\_least16\_t g = (color & 0x07E0) >> 3;

uint\_least16\_t b = (color & 0x001F) << 3;

byte x0Arr[] = { x & 0x00FF, x >> 8 };

byte y0Arr[] = { y & 0x00FF, y >> 8 };

byte length[] = { bg & 0x00FF, bg >> 8 };

byte font[] = { size & 0x00FF, size >> 8 };

vector<byte> v;

v.resize(0);

v.push\_back(12);

v.push\_back(x0Arr[0]);

v.push\_back(x0Arr[1]);

v.push\_back(y0Arr[0]);

v.push\_back(y0Arr[1]);

v.push\_back(r);

v.push\_back(g);

v.push\_back(b);

v.push\_back(font[0]);

v.push\_back(font[1]);

v.push\_back(length[0]);

v.push\_back(length[1]);

for (int i = 0; i < bg; i++)

{

v.push\_back(s[i]);

}

SendMessageServer(v, v.size());

return x;

}

void loadSprite(uint\_least8\_t index, int\_least16\_t width, int\_least16\_t height, char\* data) override

{

int k = 0;

byte widthArr[] = { width & 0x00FF, width >> 8 };

byte heightArr[] = { height & 0x00FF, height >> 8 };

vector<byte> v;

v.resize(0);

v.push\_back(17);

v.push\_back(index);

v.push\_back(widthArr[0]);

v.push\_back(widthArr[1]);

v.push\_back(heightArr[0]);

v.push\_back(heightArr[1]);

for (int i = 0; i < width\*height\*3; i++)

{

v.push\_back(data[i]);

}

SendMessageServer(v, v.size());

}

void showSprite(uint\_least8\_t index, uint\_least16\_t x, uint\_least16\_t y) override

{

vector<byte> v;

v.resize(0);

v.push\_back(18);

byte indexArr[] = { index & 0x00FF, index >> 8 };

byte xArr[] = { x & 0x00FF, x >> 8 };

byte yArr[] = { y & 0x00FF, y >> 8 };

v.push\_back(indexArr[0]);

v.push\_back(indexArr[1]);

v.push\_back(xArr[0]);

v.push\_back(xArr[1]);

v.push\_back(yArr[0]);

v.push\_back(yArr[1]);

SendMessageServer(v, v.size());

}

};

C++ Client:

#pragma comment(lib, "Ws2\_32.lib")

#pragma warning( disable : 4996)

#include <iostream>

#include <winsock2.h>

#include <cstring>

#include <WS2tcpip.h>

#include "GraphicsLib.cpp"

#define LINEMAX = 1024;

using namespace std;

int main()

{

char\* data;

int x1, x2, y1, y2;

uint\_least16\_t color;

int\_least16\_t widthImage, heightImage;

const int maxnum = 32767;

string ipaddr = "127.0.0.1", messagetext;

int port = 1984;

short width = 0, height = 0;

cout << "Input Ip Address: \n";

cin >> ipaddr;

cout << "Input port:\n";

cin >> port;

cout << "Input the size of window, max number: " << maxnum << ", min number: -" << maxnum << endl;;

while (width == 0)

{

cout << "Input width:\n";

cin >> width;

if (width > maxnum || width < -maxnum)

{

cout << "Error, number too big/small!";

width = 0;

}

}

while (height == 0)

{

cout << "Input height: \n";

cin >> height;

if (height > maxnum || height < -maxnum)

{

cout << "Error, number too big/small!";

height = 0;

}

}

graphicsLib lib = graphicsLib(width, height);

lib.setIP(ipaddr);

lib.setPort(port);

int client = lib.ClientStartup();

lib.SetScreenSize();

int command;

while (true)

{

cout << "Input Command, for available commands enter 69, to exit enter 0;";

cin >> command;

switch (command)

{

case 69:

cout << "Available commands: \n1.fillScreen\n2.drawPixel\n3.drawLine\n4.drawRect\n5.fillRect\n6.drawEllipse\n7.fillEllipse\n8.drawText\n9.loadSprite\n10.showSprite";

break;

case 1:

lib.fillScreen(lib.ColorInput());

break;

case 2:

cout << "Input x1:";

cin >> x1;

cout << "Input y1:";

cin >> y1;

lib.drawPixel (x1,y1,lib.ColorInput());

break;

case 3:

cout << "Input x1:";

cin >> x1;

cout << "Input y1:";

cin >> y1;

cout << "Input x2:";

cin >> x2;

cout << "Input y2:";

cin >> y2;

lib.drawLine(x1, y1,x2,y2, lib.ColorInput());

break;

case 4:

cout << "Input x1:";

cin >> x1;

cout << "Input y1:";

cin >> y1;

cout << "Input width:";

cin >> x2;

cout << "Input height:";

cin >> y2;

lib.drawRect(x1, y1, x2, y2, lib.ColorInput());

break;

case 5:

cout << "Input x1:";

cin >> x1;

cout << "Input y1:";

cin >> y1;

cout << "Input width:";

cin >> x2;

cout << "Input height:";

cin >> y2;

lib.fillRect(x1, y1, x2, y2, lib.ColorInput());

break;

case 6:

cout << "Input x1:";

cin >> x1;

cout << "Input y1:";

cin >> y1;

cout << "Input horizontal radius:";

cin >> x2;

cout << "Input vertical radius:";

cin >> y2;

lib.drawEllipse(x1, y1, x2, y2, lib.ColorInput());

break;

case 7:

cout << "Input x1:";

cin >> x1;

cout << "Input y1:";

cin >> y1;

cout << "Input horizontal radius:";

cin >> x2;

cout << "Input vertical radius:";

cin >> y2;

lib.fillEllipse(x1, y1, x2, y2, lib.ColorInput());

break;

case 8:

cout << "Input x1:";

cin >> x1;

cout << "Input y1:";

cin >> y1;

cout << "Font Size:";

cin >> x2;

cout << "Input color:";

color = lib.ColorInput();

cout << "Input text:";

fflush(stdin);

cin.ignore();

getline(cin, messagetext);

lib.drawText(x1, y1, messagetext.c\_str(), color, messagetext.length(), x2);

break;

case 9:

cout << "Input Index:";

cin >> x1;

cout << "Input Image Name with Extension!!!:";

fflush(stdin);

cin.ignore();

getline(cin, messagetext);

data = lib.ConvertImage(messagetext, widthImage, heightImage);

lib.loadSprite(x1, widthImage, heightImage, data);

break;

case 10:

cout << "Input Index:";

cin >> x2;

cout << "Input x:";

cin >> x1;

cout << "Input y:";

cin >> y1;

lib.showSprite(x2, x1, y1);

break;

case 0:

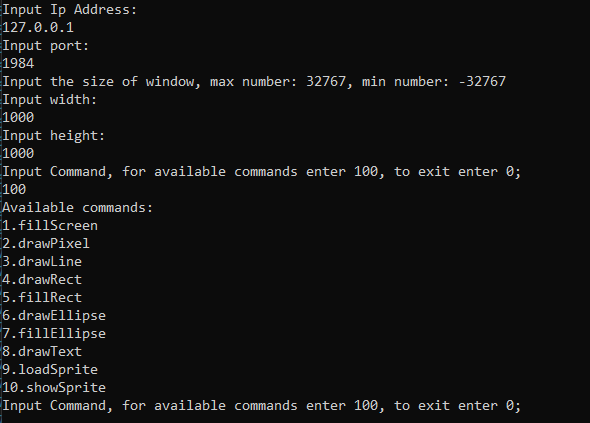
return 0;

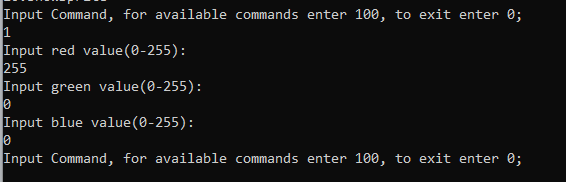
break;

}

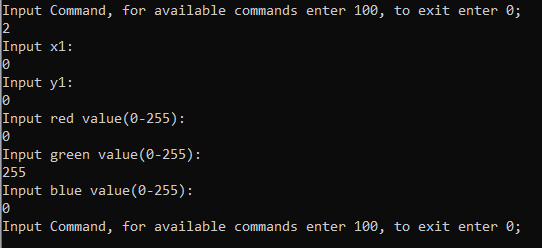
}

};

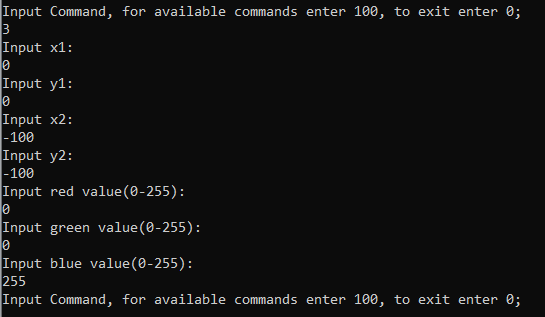




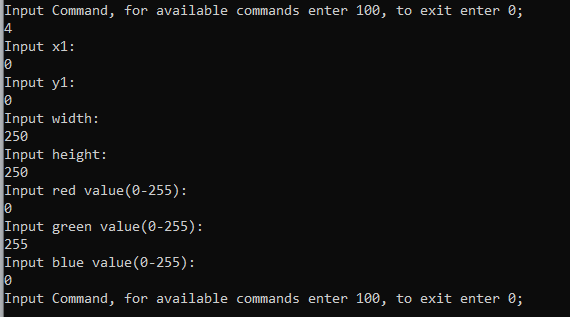




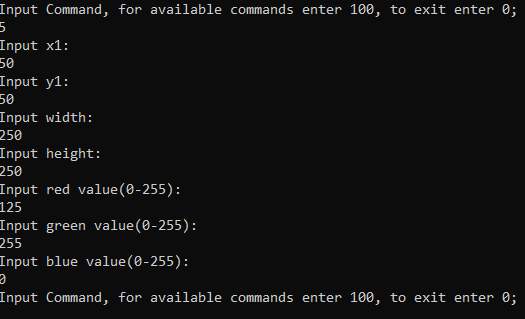
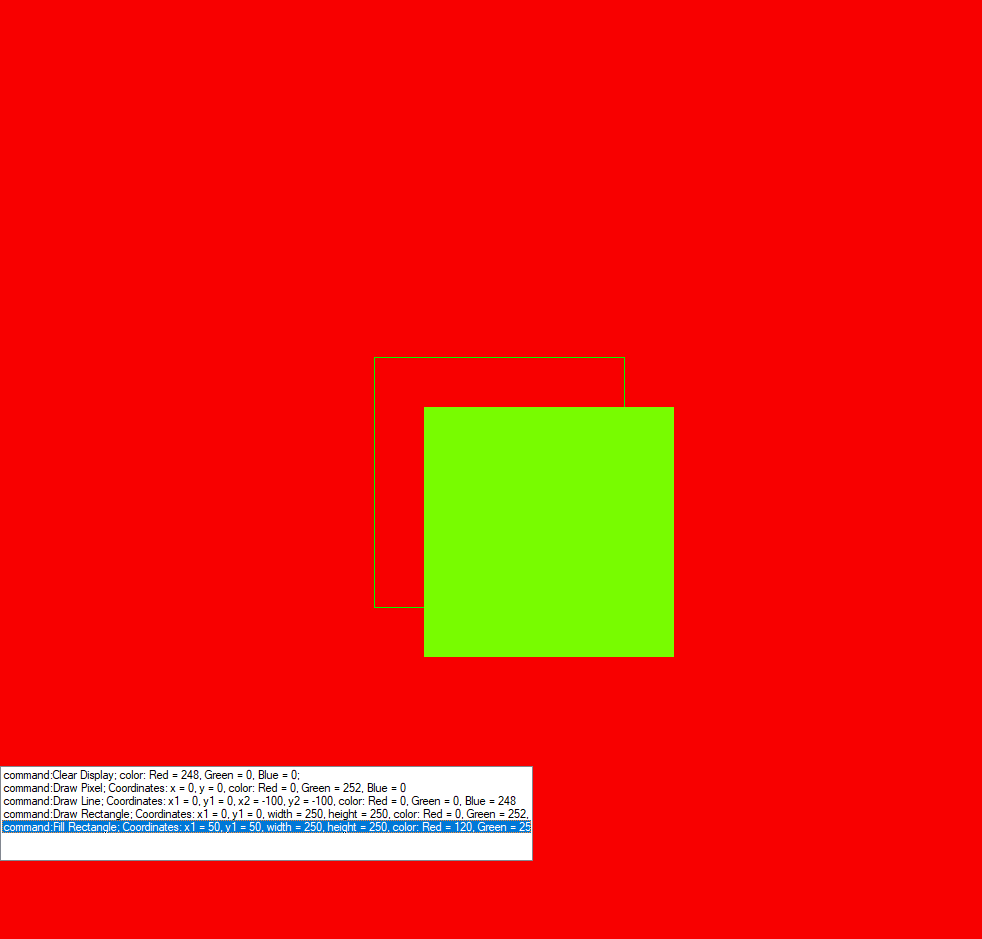


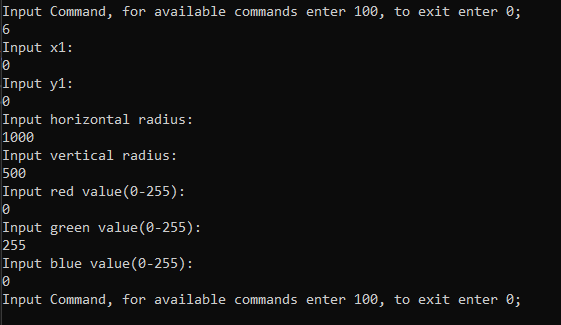
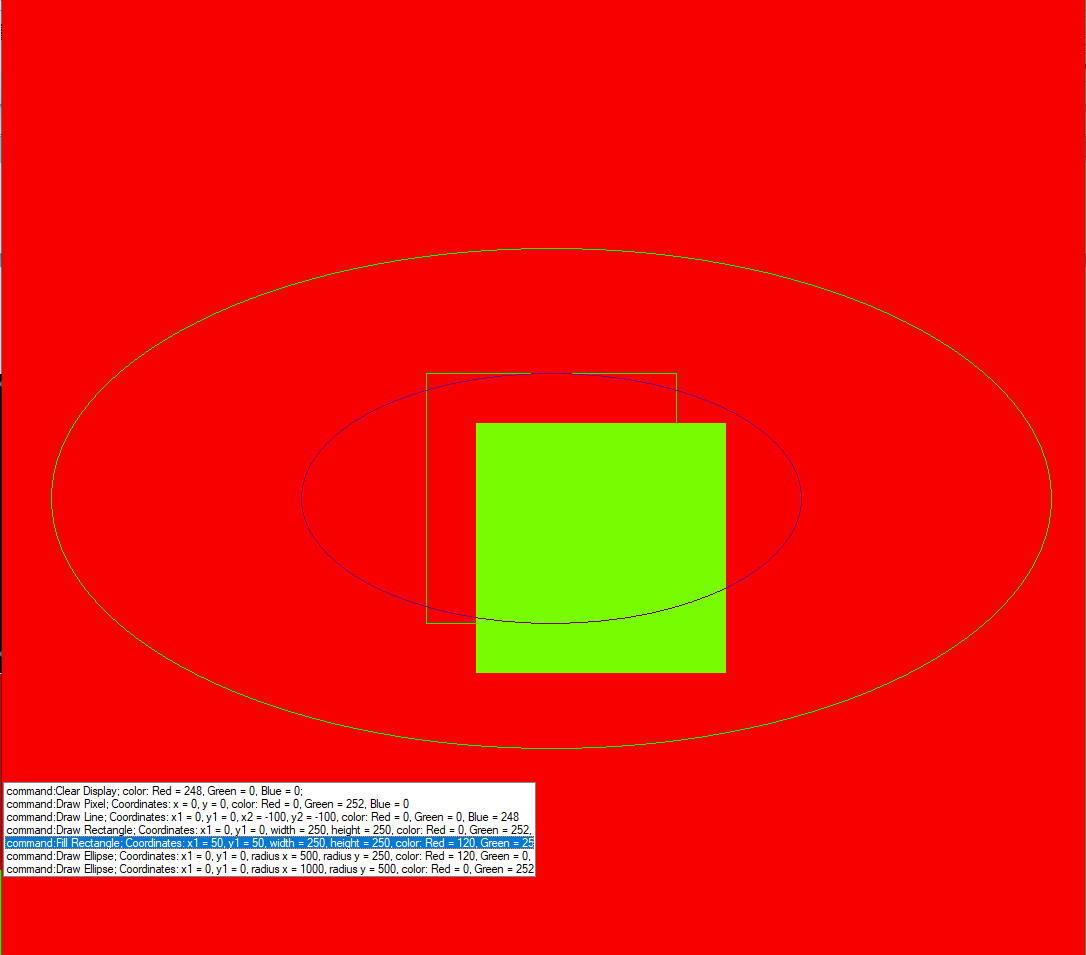


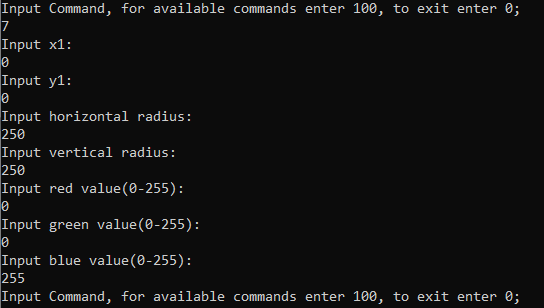
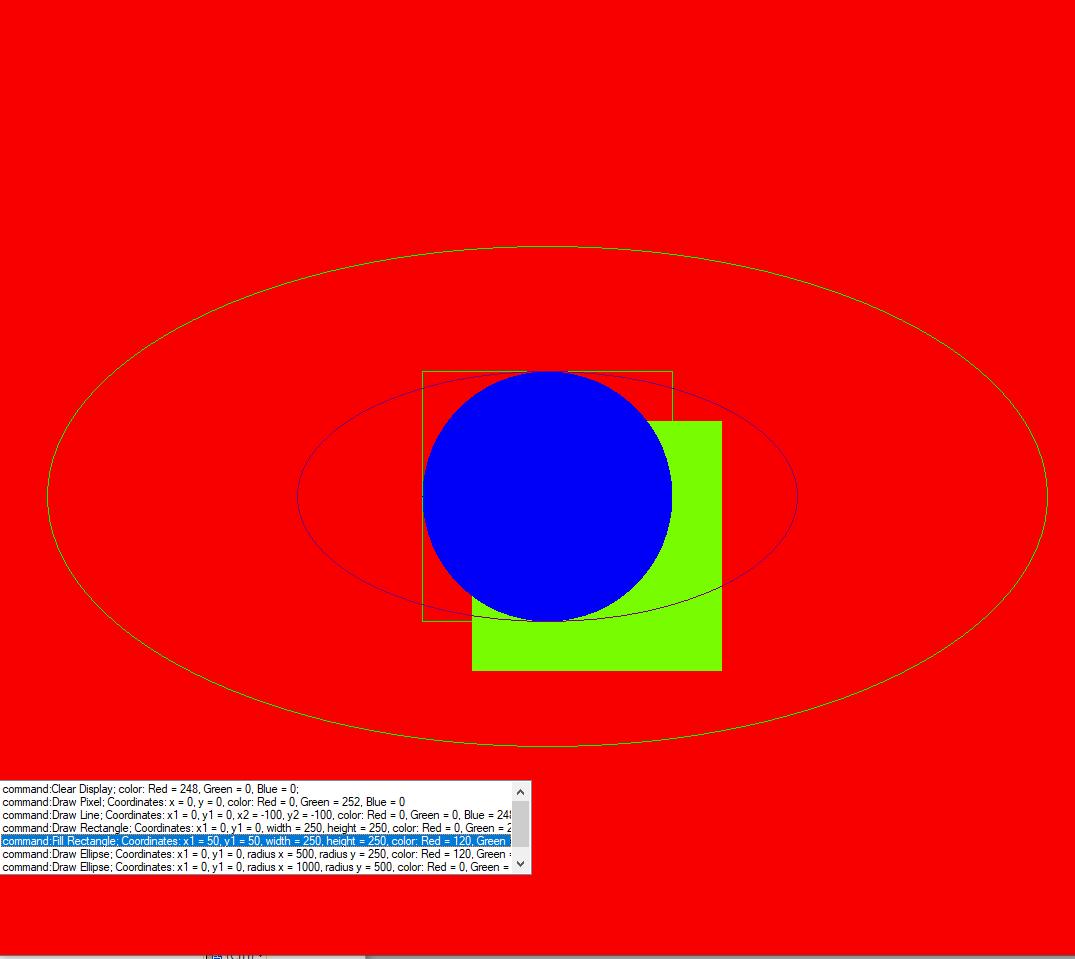


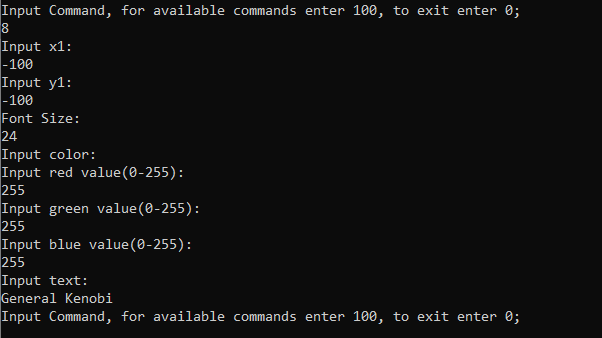


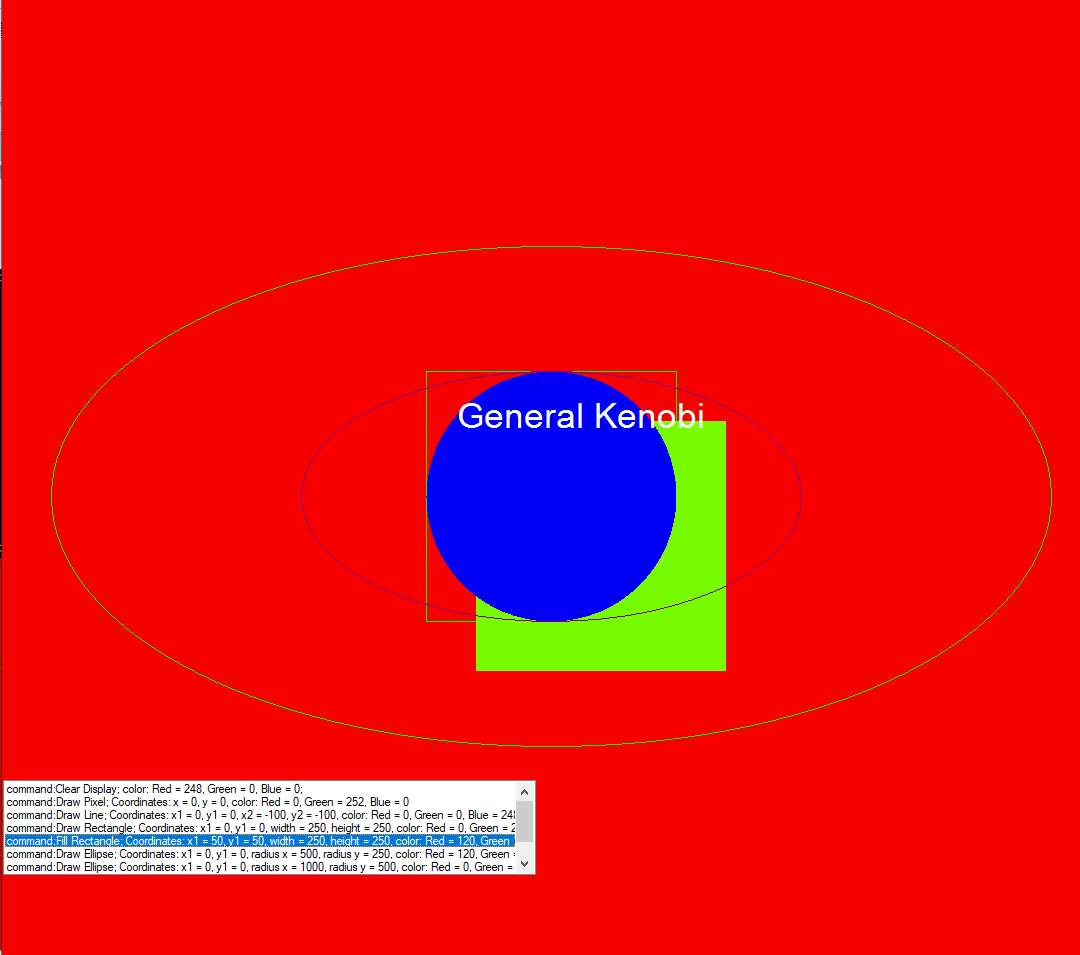


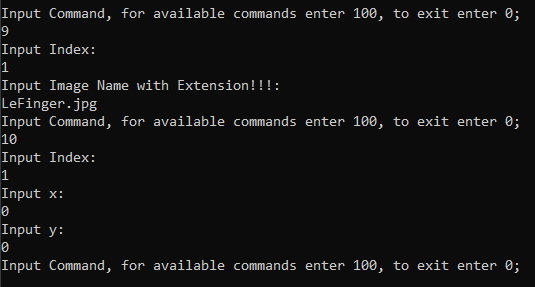
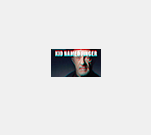
 





Висновок: Реалізував клієнтську частину бібліотеки для управління дисплейним модулем.