|  |
| --- |
| **МИНОБРНАУКИ РОССИИ**  федеральное государственное бюджетное образовательное  учреждение высшего образования  **«Национальный исследовательский университет «МЭИ»** |

|  |  |
| --- | --- |
| **Институт** | ИВТИ |
| **Кафедра** | ПМИИ |

|  |  |
| --- | --- |
|  |  |
| **Дисциплина: «Системное программирование»**  **Отчет по лабораторной работе №5**  **«Основы GDI»**  **Выполнил: студент группы А-13а-19**  **Рамазанов Н. М.**  **Преподаватель: Меньшикова К. Г.**  **Москва, 2021** | |

**1. Краткое описание функций, параметров и используемых событий.**

1) Дескриптор контекста устройства и функции для получения и освобождения контекста:

* GetWindowDC

HDC GetWindowDC (HWND hWnd);

Функция GetWindowDC извлекает контекст устройства (DC) для всего окна, включая области заголовка, меню и полосы прокрутки. Контекст устройства окна разрешает красить в любом месте окна, потому что начало координат контекста устройства - левый верхний угол окна вместо рабочей области.

GetWindowDC назначает значения атрибутов контекста устройства окна по умолчанию каждый раз, когда она извлекает этот контекст устройства. Предыдущие атрибуты теряются.

Параметр: hWnd – дескриптор окна с контекстом устройства, который должен извлечься. Если это значение равно NULL, GetWindowDC извлекает контекст устройства для всего экрана.

Если функция завершается успешно, возвращаемое значение – дескриптор контекста устройства заданного окна. Если функция завершается ошибкой, возвращаемое значение - NULL, что служит признаком ошибки или недопустимого параметра hWnd.

* GetDC

HDC GetDC (HWND hWnd);

Функция GetDC извлекает дескриптор дисплейного контекста устройства (DC) для рабочей области заданного окна или для всего экрана.

Параметр: hWnd – дескриптор окна, контекст устройства (DC) которого должен извлечься. Если это значение - ПУСТО (NULL), GetDC извлекает контекст устройства (DC) для всего экрана.

Если функция завершается успешно, возвращаемое значение - дескриптор контекста устройства (DC) для рабочей области заданного окна. Если функция завершается ошибкой, возвращаемое значение - ПУСТО (NULL).

* ReleaseDC

HDC ReleaseDC (HWND hWnd, HDC hDC);

Функция ReleaseDC освобождает контекст устройства (DC) для использования другими приложениями. Действие функции ReleaseDC зависит от типа контекста устройства (DC). Она освобождает только общий и контекст устройства (DC) окна. Она не имеет никакого действия на контексты устройства класса или частный DC.

Параметры: hWnd – дескриптор окна, контекст устройства (DC) которого должен быть освобожден; hDC – дескриптор контекста устройства (DC), который будет освобожден.

Возвращаемое значение указывает, был ли контекст устройства (DC) освобожден. Если контекст устройства был освобожден, возвращаемое значение равно 1. Если контекст устройства (DC) не был освобожден, величина возвращаемого значения - ноль.

2) Выбор и создание инструментов рисования.

* GetStockObject

HGDIOBJ GetStockObject (int fnObject);

Функция GetStockObject извлекает дескриптор одного из предопределенных (стандартных) перьев, кистей, шрифтов или палитр.

Параметр: fnObject - устанавливает тип предопределенного объекта.

Если функция завершается успешно, величина возвращаемого значения - не нуль. Если функция завершается ошибкой, величина возвращаемого значения нулевая.

* SelectObject

HGDIOBJ SelectObject (HDC hDC, HGDIOBJ hGDIobj);

Функция SelectObject выбирает объект в заданный контекст устройства (DC). Новый объект заменяет предыдущий объект того же самого типа.

Параметры: hDC - дескриптор контекста устройства; hGDIobj - дескриптор объекта, который выбирается.

Если функция завершается успешно, величина возвращаемого значения - не нуль. Если функция завершается ошибкой, величина возвращаемого значения нулевая.

* CreateSolidBrush

HBRUSH CreateSolidBrush (COLORREF crColor);

Функция CreateSolidBrush создает логическую кисть, которая имеет заданный сплошной тон.

Параметр: crColor - устанавливает цвет кисти.

Если функция завершается успешно, возвращаемое значение идентифицирует логическую кисть. Если функция завершается ошибкой, величина возвращаемого значения нулевая.

* CreatePen

HPEN CreatePen(int fnPenStyle, int nWidth, COLORREF crColor);

Функция CreatePen создает логическое перо, которое имеет заданные стиль, ширину и цвет.

Параметры: fnPenStyle - определяет стиль пера; nWidth - определяет толщину пера в логических единицах измерения; crColor - определяет опорный уровень цвета для цвета пера.

Если функция завершается успешно, возвращаемое значение – дескриптор, который идентифицирует логическое перо. Если функция завершается ошибкой, величина возвращаемого значения нулевая.

* DeleteObject

HGDIOBJ DeleteObject (HGDIOBJ hGDIobj);

Функция DeleteObject удаляет логическое перо, кисть, шрифт, точечную картинку, регион или палитру, освобождая все системные ресурсы, связанные с объектом. После того, как объект удаляется, его дескриптор более не допустим.

Параметр: hGDIobj - дескриптор логического пера, кисти, шрифта, точечной картинки, региона или палитры.

Если функция завершается успешно, возвращаемое значение - не нуль. Если установленный дескриптор не допустим или является текущим выбранным в DC, возвращаемое значение - нуль.

* SetBkColor

HGDIOBJ SetBkColor (HDC hDC, COLORREF crColor);

Функция SetBkColor устанавливает текущий цвет фона в заданном коде цвета или в самом близком физическом цвете, если устройство не может предоставить указанный код цвета.

Параметры: hDC - дескриптор контекста устройства; crColor - определяет новый цвет фона.

Если функция завершается успешно, возвращаемое значение определяет предыдущий цвет фона как значение COLORREF. Если функция завершается ошибкой, величина возвращаемого значения нулевая.

3) Функции рисования.

|  |  |
| --- | --- |
| **Функция** | **Смысл** |
| BeginPaint | Подготавливает окно для рисования |
| EndPaint | Помечает конец рисования в окне |
| SetPixel | Устанавливает заданный цвет в точке с указанными координатами |
| LineTo | Рисует линию от текущей позиции до места, указанного в аргументах |
| Arc | Рисует дугу по заданным координатам |
| Rectangle | Рисует прямоугольник по заданным координатам |
| RoundRect | Рисует прямоугольник с скругленными углами |
| CreateSolidBrush | Устанавливает логическую сплошную кисть |
| CreateHatchBrush | Устанавливает логическую кисть с заданным шаблоном штриховки и цветом |
| CreatePen | Устанавливает логическое перо с заданным стилем, шириной и цветом |
| FillRect | Закрашивает прямоугольник |
| FrameRect | Рисует прямоугольную рамку |
| InvertRect | Производит инверсию цветов точек в заданной области |
| Ellipse | Рисует эллипс по заданным координатам |
| Chord | Рисует хорду в эллипсе |
| Pie | Рисует сектор эллипса |
| PolyDraw | Рисует многоугольник по вершинам |
| PolyLine | Рисует многоугольник набором отрезков |

4) Функции заливки.

* ExtFloodFill

BOOL ExtFloodFill (HDC hDC, int nXStart, int nYStart,

COLORREF crColor, UINT fuFillType);

Функция ExtFloodFill закрашивает область поверхности изображения текущей кистью.

Параметры: hDC - дескриптор контекста устройства; nXStart - определяет x-координату, в логических единицах, точки, откуда начинается заливка; nYStart - определяет y-координату, в логических единицах, точки, откуда начинается заливка; crColor - определяет цвет рамки или закрашиваемой области; fuFillType - определяет тип операции заливки, которая будет выполнена: этот параметр должен быть одним из нижеперечисленных значений.

|  |  |
| --- | --- |
| FLOODFILLBORDER | Закрашиваемая область ограничивается цветом, заданным параметром crColor. Этот стиль идентичен заливке, выполненной функцией FloodFill |
| FLOODFILLSURFACE | Закрашенная область определяется цветом, который задается crColor. Заливка происходит снаружи во всех направлениях, пока цвет не натолкнется на контур. |

Если функция завершается успешно, величина возвращаемого значения - не нуль. Если функция завершается ошибкой, величина возвращаемого значения нулевая.

* FillRect

BOOL FillRect (HDC hDC, const RECT \*lpRect, HBRUSH hBrush);

Функция FillRect заполняет прямоугольную область на устройстве, используя указанную кисть. Граница прямоугольной области не рисуется, и основание и правые грани данного прямоугольника не заполняются (они не являются частью внутренней области прямоугольника).

Параметры: hDC - дескриптор контекста устройства; lpRect - координаты прямоугольной области для заполнения; hBrush - описатель кисти для заполнения прямоугольной области.

Если функция завершается успешно, величина возвращаемого значения - не нуль. Если функция завершается ошибкой, величина возвращаемого значения нулевая.

5) Функции для работы с пикселами.

* GetPixel

COLORREF GetPixel (HDC hdc, int nXPos, int nYPos);

Функция GetPixel извлекает значения красного, зеленого, синего (RGB) цвета пикселя в заданных координатах.

Параметры: hDC - дескриптор контекста устройства; nXPos - определяет x-координату проверяемого пикселя, в логических единицах измерения; nYPos - определяет y-координату проверяемого пикселя, в логических единицах измерения.

Возвращаемое значение - значение RGB пикселя. Если пиксель - вне текущего региона отсечения, возвращаемое значение - CLR\_INVALID.

* SetPixel

COLORREF SetPixel (HDC hdc, int X, int Y, COLORREF crColor);

Функция SetPixel устанавливает пиксель в заданных координатах в заданном цвете.

Параметры: hDC - дескриптор контекста устройства; X - определяет x-координату устанавливаемой точки, в логических единицах измерения; Y - определяет y-координату устанавливаемой точки, в логических единицах измерения; crColor - устанавливает цвет, используемый для рисования точки.

Если функция завершается успешно, возвращаемое значение - значение RGB, в которое функция устанавливает пиксель. Это значение может отличаться от цвета, заданного параметром crColor; это происходит тогда, когда точное соответствие для заданного цвета не может быть найдено. Если функция завершается с ошибкой, возвращаемое значение равно -1.

6) Режимы рисования.

int SetROP2 (HDC hdc, int fnDrawMode);

Функция SetROP2 устанавливает текущий высокоприоритетный режим смешивания. GDI использует высокоприоритетный режим смешивания, чтобы объединять перья и внутренние области закрашенных объектов с цветом уже на экране. Высокоприоритетный режим смешивания определяет, как должны комбинироваться цвета кисти или пера и цвета в существующем изображении.

Параметры: hDC - дескриптор контекста устройства; fnDrawMode - устанавливает режим смешивания. Этот параметр может быть одним из нижеперечисленных значений.

|  |  |
| --- | --- |
| Режим смешивания | Описание |
| R2\_BLACK | Пиксель всегда 0. |
| R2\_COPYPEN | Пиксель - цвет пера. |
| R2\_MASKNOTPEN | Пиксель - комбинация цветов, общих и для экрана, и для инверсии пера. |
| R2\_MASKPEN | Пиксель - комбинация цветов, общих и для пера, и для экрана. |
| R2\_MASKPENNOT | Пиксель - комбинация цветов, общих и для пера, и для инверсии экрана. |
| R2\_MERGENOTPEN | Пиксель - комбинация цвета экрана и инверсии цвета пера. |
| R2\_MERGEPEN | Пиксель - комбинация цвета пера и цвета экрана. |
| R2\_MERGEPENNOT | Пиксель - комбинация цвета пера и инверсии цвета экрана. |
| R2\_NOP | Пиксель остается неизменным. |
| R2\_NOT | Пиксель - инверсия цвета экрана. |
| R2\_NOTCOPYPEN | Пиксель - инверсия цвета пера. |
| R2\_NOTMASKPEN | Пиксель - инверсия цвета R2\_MASKPEN. |
| R2\_NOTMERGEPEN | Пиксель - инверсия цвета R2\_MERGEPEN. |
| R2\_NOTXORPEN | Пиксель - инверсия цвета R2\_XORPEN. |
| R2\_WHITE | Пиксель всегда 1. |
| R2\_XORPEN | Пиксель - комбинация цветов в пере и в экране, но не обоих. |

Если функция завершается успешно, возвращаемое значение определяет предыдущий режим смешивания. Если функция завершается ошибкой, возвращаемое значение - нуль.

7) Структура COLORREF

typedef DWORD COLORREF;

typedef DWORD \*LPCOLORREF;

Значение COLORREF используется, чтобы определить цвет RGB.

8) Методы InvalidateRect и UpdateWindow.

* InvalidateRect

BOOL InvalidateRect(HWND hWnd, CONST RECT\* lpRect, BOOL bErase);

Функция InvalidateRect добавляет прямоугольник к обновляемому региону заданного окна. Обновляемый регион представляет часть рабочей области окна, которая должна быть перерисована.

Параметры: hWnd - дескриптор окна, обновляемый регион которого изменился; lpRect - указатель на структуру RECT, содержащую в себе координаты рабочей области прямоугольника, который будет добавлен к обновляемому региону; bErase - устанавливает, должен ли фон внутри обновляемого региона быть стерт, когда обновляемый регион обрабатывается.

Если функция завершается успешно, величина возвращаемого значения - не нуль. Если функция завершается ошибкой, величина возвращаемого значения нулевая.

* UpdateWindow

BOOL UpdateWindow (HWND hWnd);

Функция UpdateWindow обновляет рабочую область заданного окна, отправляя сообщение WM\_PAINT окну, если регион обновления окна не пуст. Функция отправляет сообщение WM\_PAINT непосредственно оконной процедуре указанного окна, обходя очередь приложения. Если регион обновления пуст, никакое сообщение не отправляется.

Параметр: hWnd - дескриптор обновляемого окна.

Если функция завершается успешно, величина возвращаемого значения - не нуль. Если функция завершается ошибкой, величина возвращаемого значения нулевая.

**2. Выполнение работы.**

Для выполнения работы создал приложение Win32 Project (в Microsoft Visual Studio).

1. Закрасил окно приложения основным (преобладающим) цветом рабочего стола и нарисовать на нем геометрические фигуры и надписи (по событию WM\_PAINT).

Создал соответствующую кнопку (можно было сделать это и без кнопки, просто сразу выполнять команды при создании окна, но тогда оно достаточно долго создается из-за небыстрой работы функции, получающей преобладающий цвет рабочего стола, и впоследствии окно сильно зависает, так как каждое действие с окном, вызывающее WM\_PAINT, заставляет каждый раз находить преобладающий цвет рабочего стола; я же с помощью кнопки предусмотрел отключение закраски окна основным цветом рабочего стола):

ColorBack = CreateWindow(L"BUTTON", L"Фон + фигуры", WS\_VISIBLE | WS\_CHILD | BS\_PUSHBUTTON, 570, 10, 200, 30, hWnd, (HMENU)idColorBack, hInstance, NULL);

Добавил в блок WM\_COMMAND реакцию на нажатие кнопки:

case idColorBack:

back\_coloring = !back\_coloring;

InvalidateRect(hWnd, NULL, TRUE);

UpdateWindow(hWnd);

break;

Добавил в WM\_PAINT инструкции для решения задачи:

case WM\_PAINT:

{

PAINTSTRUCT ps;

HDC hdc = BeginPaint(hWnd, &ps);

if (back\_coloring) {

RECT rWnd;

GetWindowRect(hWnd, &rWnd);

rWnd.bottom = rWnd.bottom - rWnd.top;

rWnd.right = rWnd.right - rWnd.left;

rWnd.top = 0;

rWnd.left = 0;

COLORREF clrDesktop = GetPixel(hdc, 1000, 1000);

SetBkColor(hdc, clrDesktop);

FillRect(hdc, &rWnd, CreateSolidBrush(clrDesktop));

HPEN pen = CreatePen(PS\_DOT, 1, RGB(255, 0, 0));

SelectObject(hdc, pen);

Rectangle(hdc, 100, 400, 200, 450);

SetTextColor(hdc, RGB(255, 0, 0));

TextOut(hdc, 105, 370, TEXT("Прямоугольник"), 14);

HBRUSH brush1 = CreateHatchBrush(HS\_CROSS, RGB(0, 255, 0));

SelectObject(hdc, brush1);

Ellipse(hdc, 300, 400, 400, 450);

TextOut(hdc, 320, 370, TEXT("Эллипс"), 7);

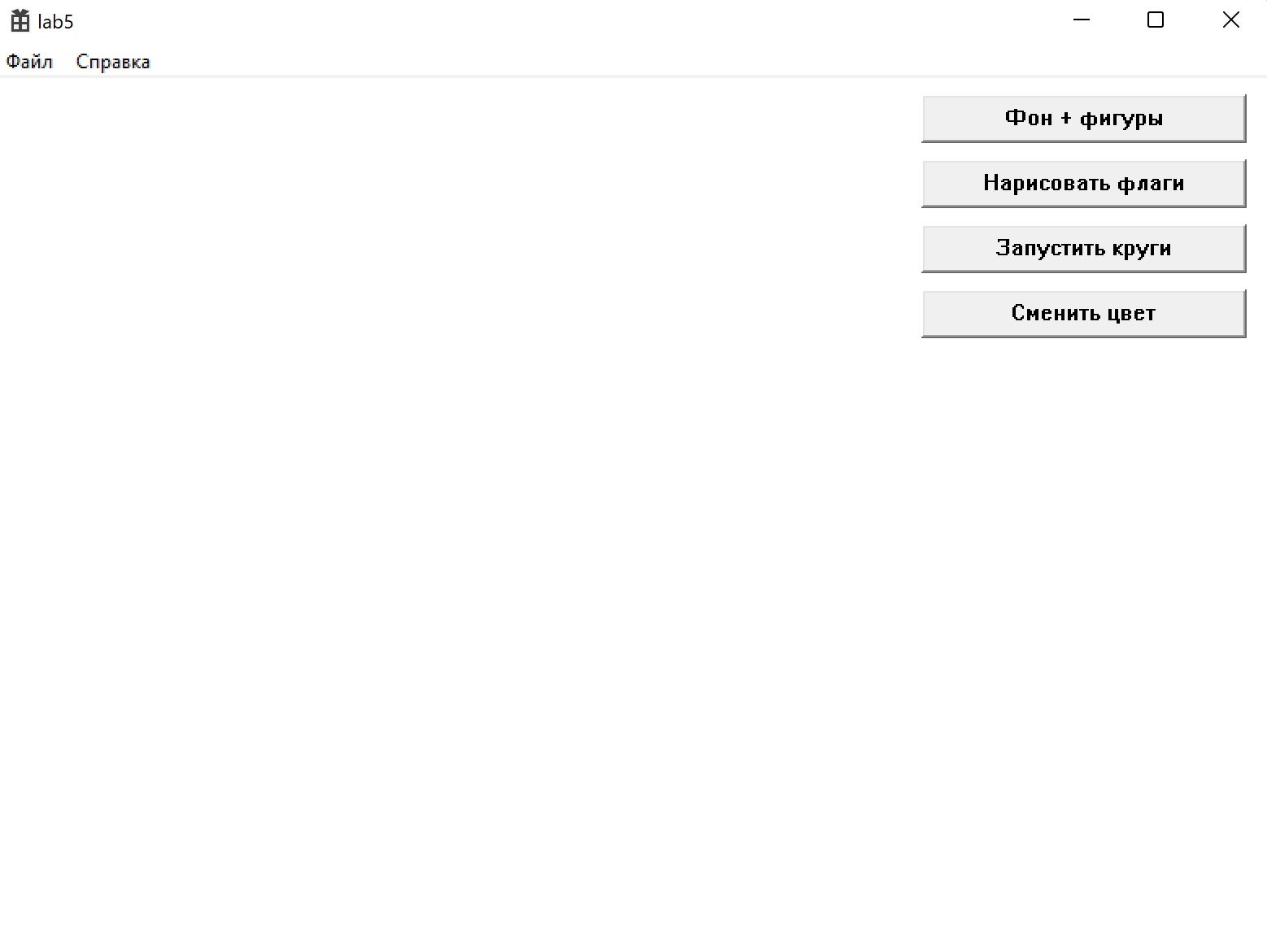
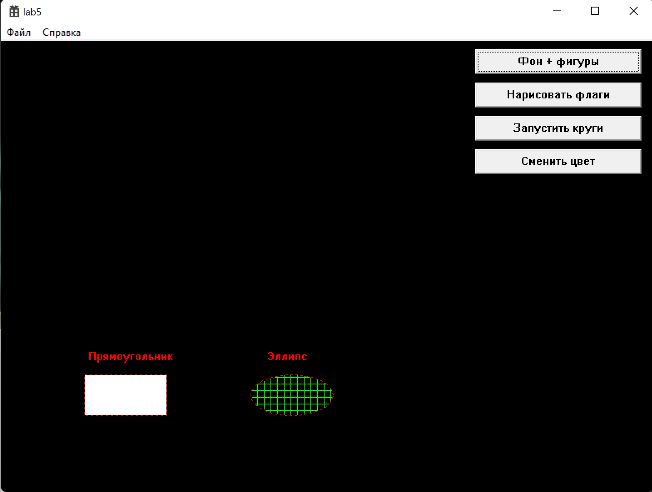
}

EndPaint(hWnd, &ps);

}

break;

До нажатия кнопки: После нажатия кнопки:



1. На рабочем столе, в рабочей области окна приложения и на поверхности окна приложения нарисовал флаг России.

Флаги рисуются по кнопке:

DrawFlags = CreateWindow(L"BUTTON", L"Нарисовать флаги", WS\_VISIBLE | WS\_CHILD | BS\_PUSHBUTTON, 570, 50, 200, 30, hWnd, (HMENU)idDrawFlags, hInstance, NULL);

Добавил в WM\_COMMAND реакцию на ее нажатие:

case idDrawFlags:

is\_draw\_flag = !is\_draw\_flag;

InvalidateRect(hWnd, NULL, TRUE);

UpdateWindow(hWnd);

break;

И в WM\_PAINT вызов функции рисования флагов:

if (is\_draw\_flag) {

PaintFlag(GetDesktopWindow());

PaintFlag(hWnd);

PaintFlag(hWnd, true);

}

Функция рисования флага:

VOID PaintFlag(HWND hWnd, bool is\_client) {

HDC hdc;

RECT rect;

if (is\_client) {

hdc = GetDC(hWnd);

rect.left = 0, rect.top = 0, rect.bottom = 100, rect.right = 200;

FillRect(hdc, &rect, CreateSolidBrush(RGB(255, 255, 255)));

}

else {

hdc = GetWindowDC(hWnd);

rect.left = 300, rect.top = 0, rect.bottom = 100, rect.right = 500;

FillRect(hdc, &rect, CreateSolidBrush(RGB(255, 255, 255)));

}

rect.top = 100, rect.bottom = 200;

FillRect(hdc, &rect, CreateSolidBrush(RGB(0, 0, 255)));

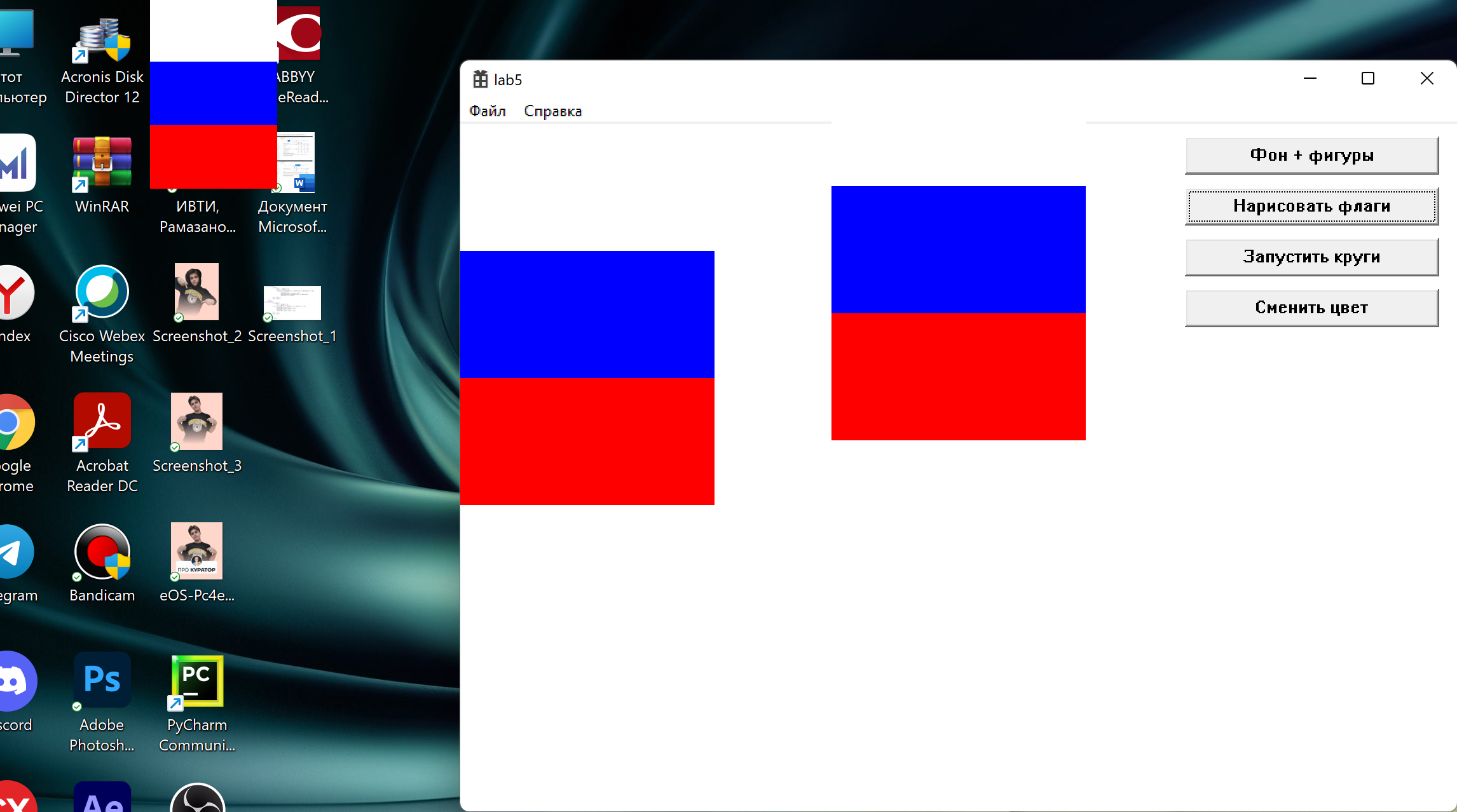
rect.top = 200, rect.bottom = 300;

FillRect(hdc, &rect, CreateSolidBrush(RGB(255, 0, 0)));

ReleaseDC(hWnd, hdc);

}

Демонстрация работоспособности:



1. В своем окне, в окне чужой программы (эту программу создал на С#, разместив в ее окне управляющие компоненты) и на рабочем столе нарисовал круг, движущийся по окну. Сравнил работу программы при рисовании в рабочей области окна, на поверхности окна и на рабочем столе.

Рисуем и запускаем движение кругов по кнопке:

MoveCircles = CreateWindow(L"BUTTON", L"Запустить круги", WS\_VISIBLE | WS\_CHILD | BS\_PUSHBUTTON, 570, 90, 200, 30, hWnd, (HMENU)idMoveCircles, hInstance, NULL);

Движение кругов будет осуществляться по таймеру:

case WM\_TIMER:

{

int wmId = wParam;

switch (wmId)

{

case idMove:

MoveOn(hWnd);

MoveDesktop();

MoveWinF();

break;

default:

break;

}

}

break;

case idMoveCircles:

if (timer) {

timer = false;

KillTimer(hWnd, idMove);

}

else {

timer = true;

MoveOn(hWnd, true);

MoveDesktop(true);

MoveWinF(true);

SetTimer(hWnd, idMove, 100, NULL);

}

break;

Функции перемещения кружков:

Перемещение круга на вектор {x, y}:

VOID MoveCircle(HWND hWnd, RECT& circle, int x, int y) {

HDC hdc = GetDC(hWnd);

InvalidateRect(hWnd, &circle, true);

UpdateWindow(hWnd);

circle.left += x, circle.right += x;

circle.top += y, circle.bottom += y;

SelectObject(hdc, CreateSolidBrush(RGB(200, 100, 0)));

Ellipse(hdc, circle.left, circle.top, circle.right, circle.bottom);

ReleaseDC(hWnd, hdc);

}

Перемещение круга в окне приложения ЛР №5:

VOID MoveOn(HWND hWnd, bool init) {

static RECT circle, rect;

static int x = 5, y = -5;

if (init) {

circle.left = 50, circle.right = 200, circle.top = 50, circle.bottom = 200;

}

GetWindowRect(hWnd, &rect);

if (rect.right - rect.left - circle.right <= 0)

x = -5;

if (circle.left <= 0)

x = 5;

if (circle.top <= 0)

y = 5;

if (rect.bottom - rect.top - circle.bottom <= 0)

y = -5;

MoveCircle(hWnd, circle, x, y);

}

Перемещение круга по рабочему столу:

VOID MoveDesktop(bool init) {

static RECT circle, rect;

static int x = 5, y = -5;

static HWND hDsktp = GetDesktopWindow();

if (init) {

circle.left = 50, circle.right = 400, circle.top = 50, circle.bottom = 400;

GetWindowRect(hDsktp, &rect);

}

if (rect.right - rect.left - circle.right <= 0)

x = -5;

if (circle.left <= 0)

x = 5;

if (circle.top <= 0)

y = 5;

if (rect.bottom - rect.top - circle.bottom <= 0)

y = -5;

MoveCircle(hDsktp, circle, x, y);

}

Перемещение круга в окне приложения WinForms на C#:

VOID MoveWinF(bool init) {

static RECT circle, rect;

static int x = 5, y = -5;

static HWND hWinF = FindWindow(0, L"C#");

if (hWinF == NULL) {

return;

}

if (init) {

circle.left = 50, circle.right = 200, circle.top = 50, circle.bottom = 200;

}

GetWindowRect(hWinF, &rect);

if (rect.right - rect.left - circle.right <= 0)

x = -5;

if (circle.left <= 0)

x = 5;

if (circle.top <= 0)

y = 5;

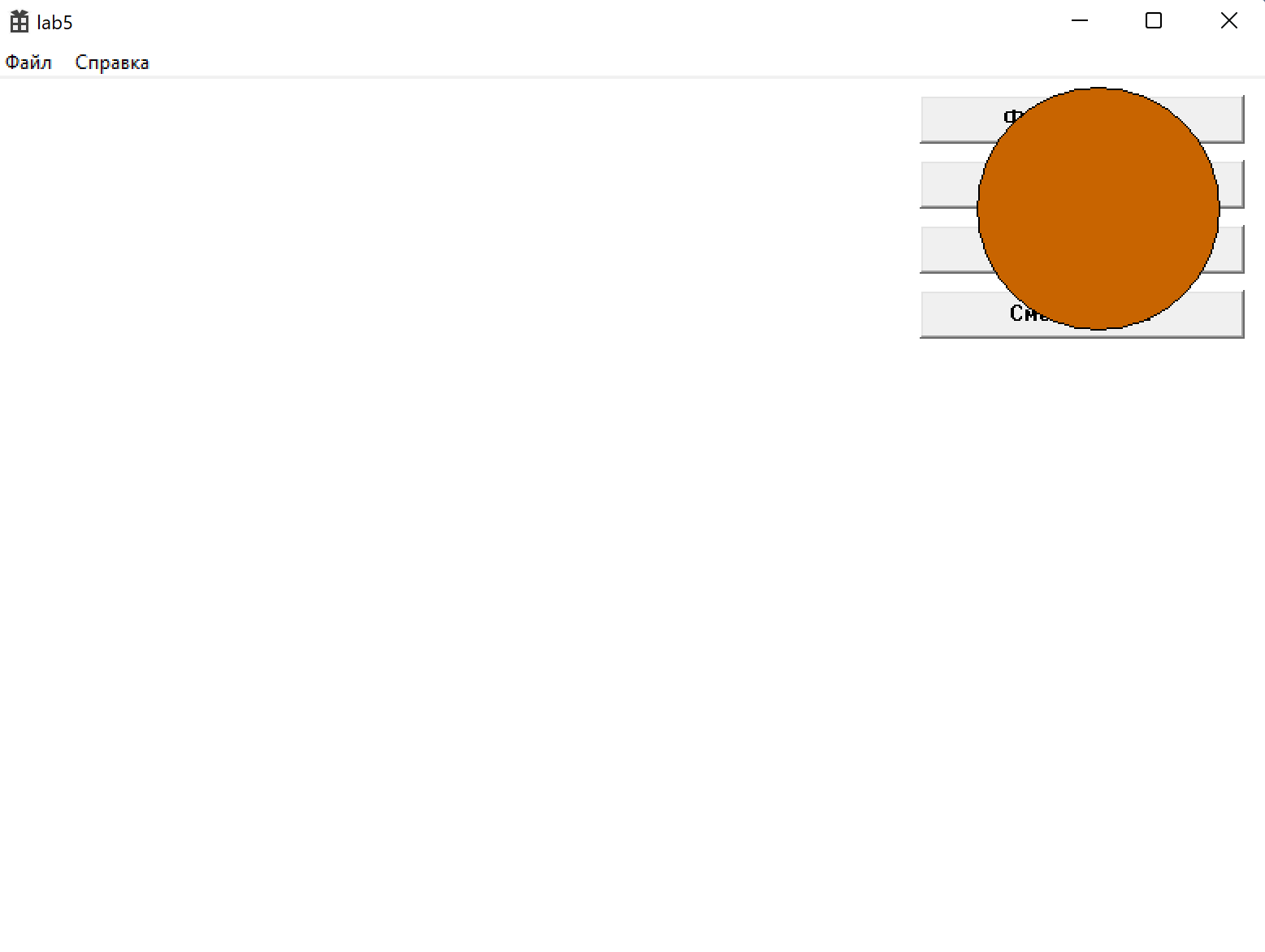
if (rect.bottom - rect.top - circle.bottom <= 0)

y = -5;

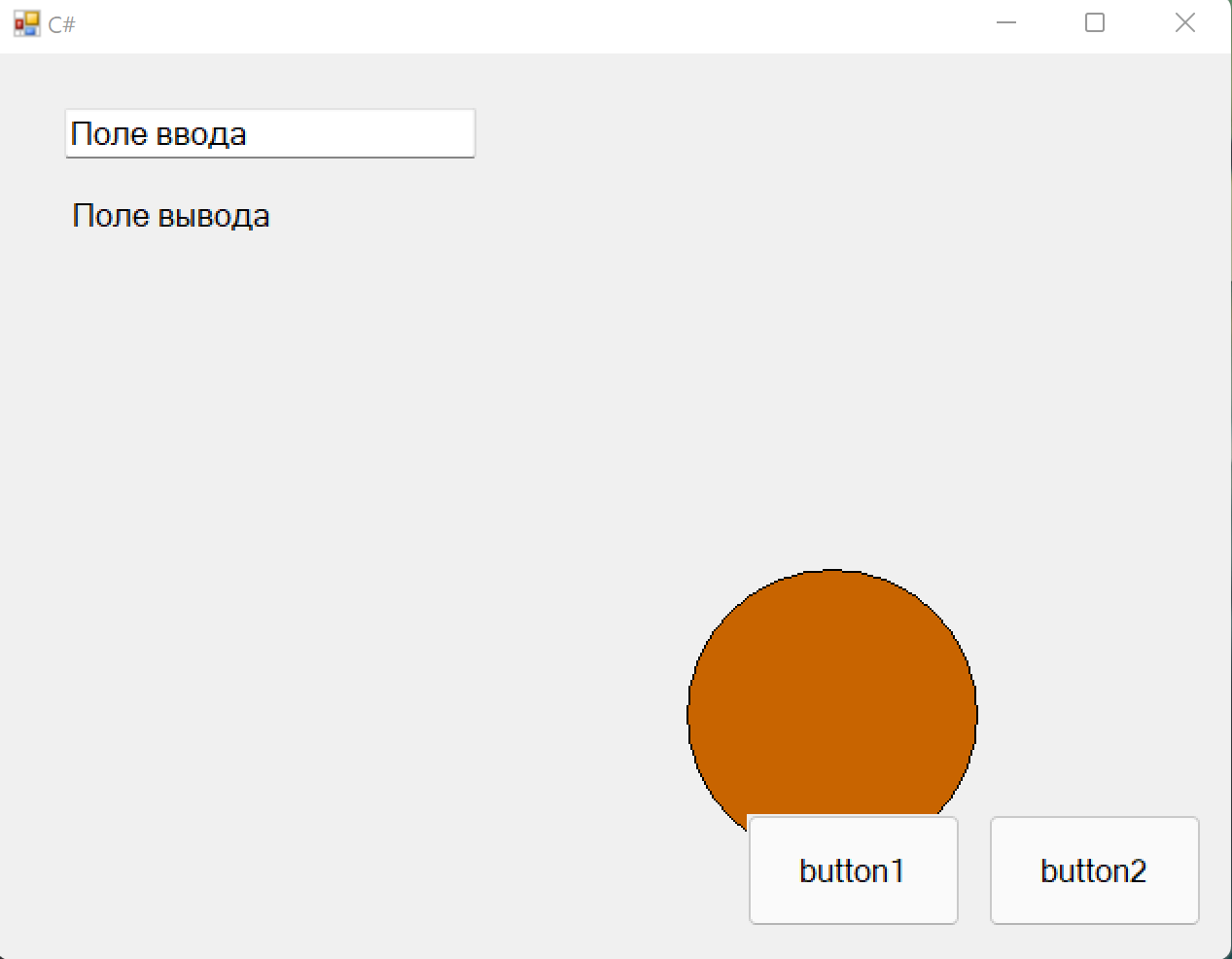
MoveCircle(hWinF, circle, x, y);

}

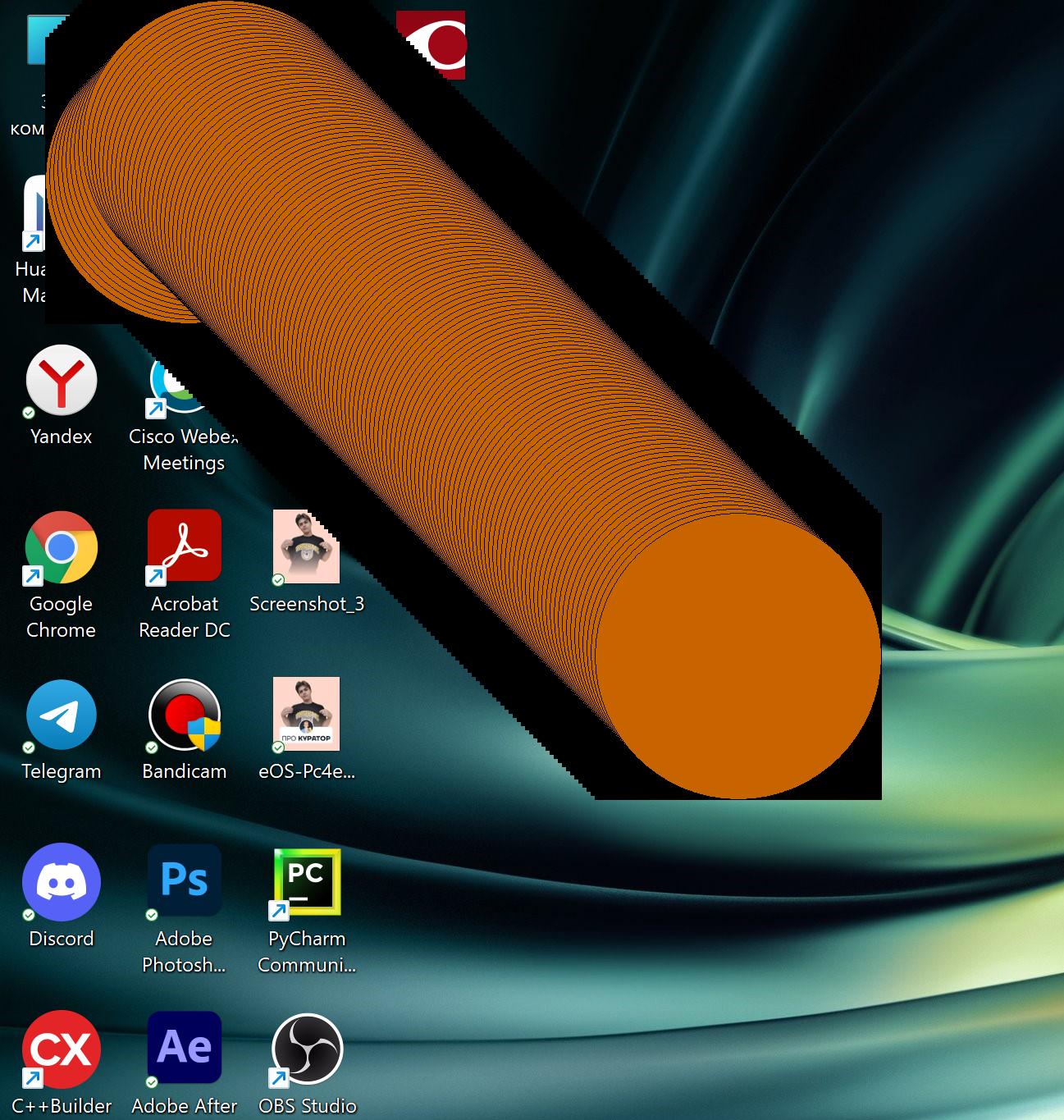
Демонстрация работоспособности:



В окне приложения ЛР №5 все работает корректно, кружок перемещается, причем он движется поверх управляющих элементов.



В окне WinForms на C# аналогично, но кружок движется под управляющими элементами (кнопками) и полями.



На рабочем столе остается след от предыдущих положений кружка, причем фон затирается и появляется изображение, которое было там до этого – в данном случае просто черный фон, поскольку тест выполнялся после перезагрузки системы, соответственно кроме фона рабочего стола на этом месте ничего не было. Вероятно, это связано с особенностями WM\_PAINT’а рабочего стола. Однако, я не нашел в интернете никакой информации по этому поводу и, к сожалению, разобраться в причинах этого явления полностью не сумел.

1. Приложение WinAPI реализует функции простейшего графического редактора, в котором пользователь может рисовать (линиями двух заданных цветов) в пределах заданного прямоугольного поля (поле закрашивается заданным цветом). Рисунок создается с помощью линий, левая кнопка мыши определяет начало линии, при перемещении мыши линия тоже перемещается, так что пользователь может оценить правильность ее расположения, при отпускании мыши линия фиксируется.

Напишем реакции на события нажатия и отжатия ЛКМ, а также перемещения мыши (при зажатой ЛКМ):

case WM\_LBUTTONDOWN:

{

is\_drawing = true;

hPen = CreatePen(PS\_SOLID, 4, RGB(50, 20, 200));

pt.x = LOWORD(lParam);

pt.y = HIWORD(lParam);

old.x = -1; old.y = -1;

}

break;

case WM\_LBUTTONUP:

is\_drawing = false;

break;

case WM\_MOUSEMOVE:

{

if (is\_drawing) {

cur.x = LOWORD(lParam);

cur.y = HIWORD(lParam);

HDC hdc = GetDC(hWnd);

SelectObject(hdc, hPen);

MoveToEx(hdc, pt.x, pt.y, NULL);

if (is\_change\_color)

SetROP2(hdc, R2\_XORPEN);

else

SetROP2(hdc, R2\_NOTXORPEN);

if (old.x >= 0)

LineTo(hdc, old.x, old.y);

MoveToEx(hdc, pt.x, pt.y, NULL);

LineTo(hdc, cur.x, cur.y);

ReleaseDC(hWnd, hdc);

old.x = cur.x;

old.y = cur.y;

}

}

break;

Также реализуем кнопку смены цвета:

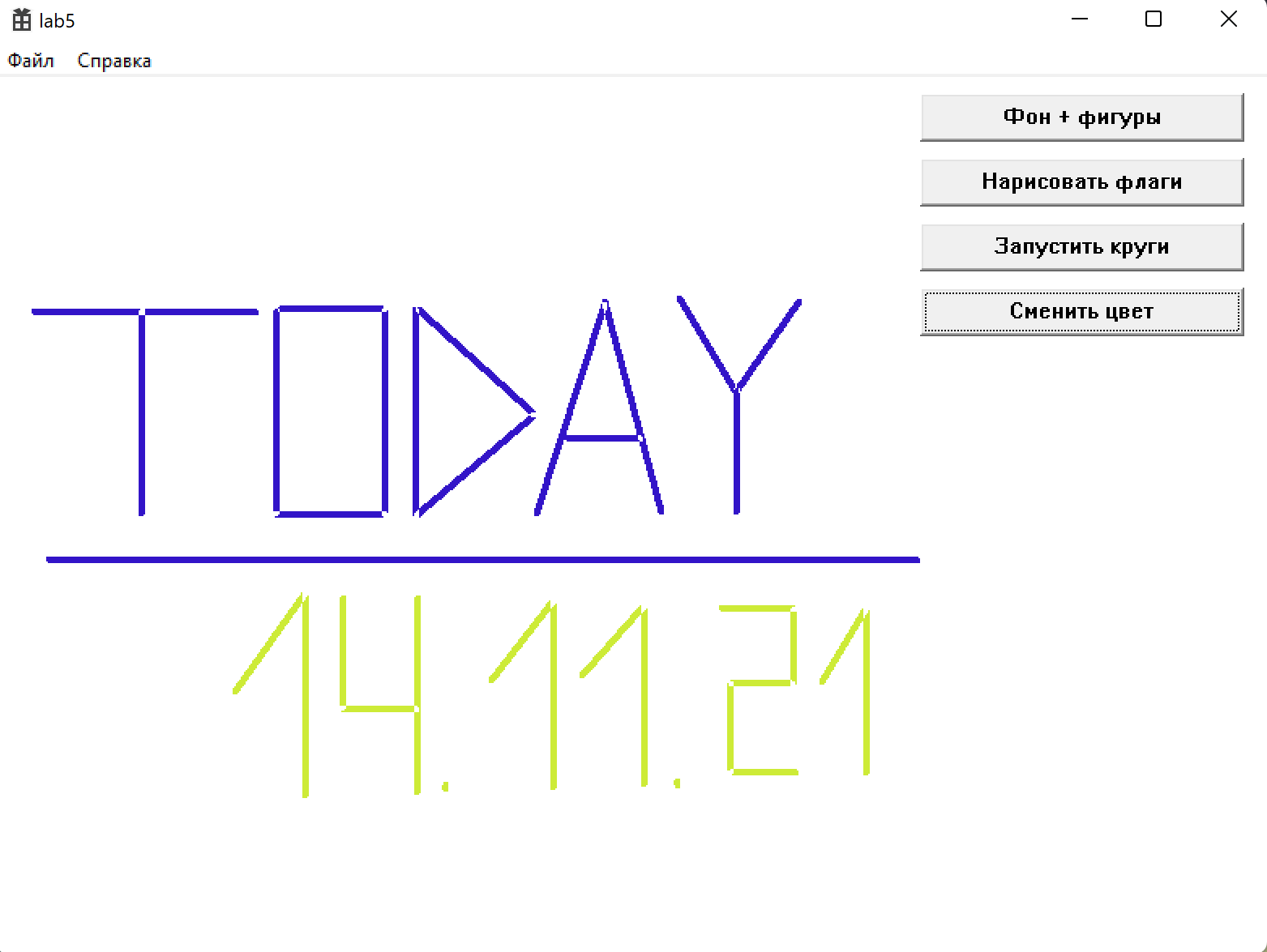
ChangeColor = CreateWindow(L"BUTTON", L"Сменить цвет", WS\_VISIBLE | WS\_CHILD | BS\_PUSHBUTTON, 570, 130, 200, 30, hWnd, (HMENU)idChangeColor, hInstance, NULL);

Добавим в WM\_COMMAND реакцию на ее нажатие:

case idChangeColor:

is\_change\_color = !is\_change\_color;

break;



Полный код программы доступен по ссылке: <https://github.com/programzan/SystemProgramming/blob/lab5/lab5.cpp>

Здесь же размещены все сопутствующие файлы: <https://github.com/programzan/SystemProgramming/tree/lab5>