Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей Кафедра информатики

Дисциплина: Операционные среды и системное программирование

**ОТЧЕТ**

к лабораторной работе № 2

на тему «Расширенное использование оконного интерфейса Win 32 и GDI. Формирование сложных изображений, создание и использование элементов управления, обработка различных сообщений, механизм перехвата сообщений»

Выполнил:

студент гр. 153504

Хрищанович А.К.

Проверил:

Гриценко Н.Ю.

Минск 2023

СОДЕРЖАНИЕ

[1 Постановка задачи 3](#_Toc146631498)

[2 Краткие теоретические сведения 4](#_Toc146631499)

[3 Результаты выполнения лабораторной работы 6](#_Toc146631500)

[Выводы 8](#_Toc146631501)

[Список использованных источников 9](#_Toc146631502)

[Приложение А (обязательное) Листинг кода 10](#_Toc146631503)

## 1 ПОСТАНОВКА ЗАДАЧИ

Целью выполнения данной лабораторной работы является разработка графического приложения с использованием Win32 API и GDI, которое будет обладать расширенным функционалом для работы с оконным интерфейсом. В ходе лабораторной работы будет реализовано графическое приложение для анимации движения объектов с возможностью настройки траектории и скорости.

## 2 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Win32 API (Windows API) представляет собой набор функций и интерфейсов, предоставляемых операционной системой Windows для разработки приложений. Этот мощный набор инструментов обеспечивает доступ к различным функциональным возможностям Windows, включая создание и управление окнами, обработку сообщений, работу с файлами и реестром, а также многие другие операции. Win32 API играет ключевую роль в разработке приложений для Windows и обеспечивает высокую степень контроля над поведением приложений.

GDI (Graphics Device Interface) — это часть Win32 API, отвечающая за графический вывод и взаимодействие с графическим оборудованием. GDI предоставляет функции для рисования графических объектов, создания изображений, управления цветами и шрифтами, а также реализации различных эффектов. Этот компонент позволяет разработчикам создавать интерфейсы пользовательских приложений, включая окна, кнопки, текст и другие графические элементы. GDI обеспечивает визуальное качество и интерактивность приложений, делая их более привлекательными и функциональными для пользователей.

Совместное использование Win32 API и GDI позволяет разработчикам создавать мощные и интерактивные графические приложения под операционной системой Windows, обеспечивая широкие возможности по работе с графикой, окнами и элементами управления.

Для выполнения данной лабораторной работы, были использованы следующие теоретические сведения и концепции:

– Win32 API (Application Programming Interface): это набор функций и структур, предоставляемых операционной системой Windows для разработки приложений под Windows. Для создания приложения было использовано множество функций Win32 API для создания окна, обработки сообщений, управления таймерами и других задач.

– Процедура обработки сообщений (WndProc): это ключевой элемент приложений Windows. WndProc — это функция, которая обрабатывает сообщения, отправляемые операционной системой приложению. В данном приложении WndProc была реализована для обработки различных сообщений, таких как создание окна, команды меню, перерисовка и таймер.

– GDI (Graphics Device Interface): GDI — это подсистема Windows для рисования графики на экране и на печать. В данном приложении функции GDI, такие как Ellipse и Rectangle, использовались для рисования объектов на окне.

– Таймеры: для обновления позиции объекта и перерисовки окна с определенной частотой были использованы таймеры с помощью функций SetTimer и KillTimer. Это обеспечило анимацию в приложении.

– Меню: создано пользовательское меню с использованием функций CreateMenu и AppendMenu. Это меню позволяет пользователю выбирать тип фигуры, направление движения и цвет фигуры, что делает его важным элементом графического интерфейса пользователя (GUI).

– Выбор параметров: пользователь имеет возможность настраивать различные параметры анимации, такие как тип траектории, скорость, тип фигуры и цвет. Это демонстрирует концепцию интерактивности в приложении.

– Управление окнами: приложение управляет окном, его размерами, заголовком и стилями с использованием функций Win32 API, таких как CreateWindow и ShowWindow.

– Обработка событий: код обрабатывает различные события, такие как нажатия кнопок, изменение текста в поле ввода и закрытие окна. Это обеспечивает реакцию приложения на действия пользователя.

## 3 РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

В ходе выполнения лабораторной работы было реализовано графическое приложение для анимации движения объектов с возможностью настройки траектории и скорости. При запуске программы пользователь видит интерфейс графического приложения, с реализацией выбора траектории, формы и цвета фигуры, а также с возможностью изменения скорости в числовом поле ввода. Результат работы программы показан на рисунке 3.1.

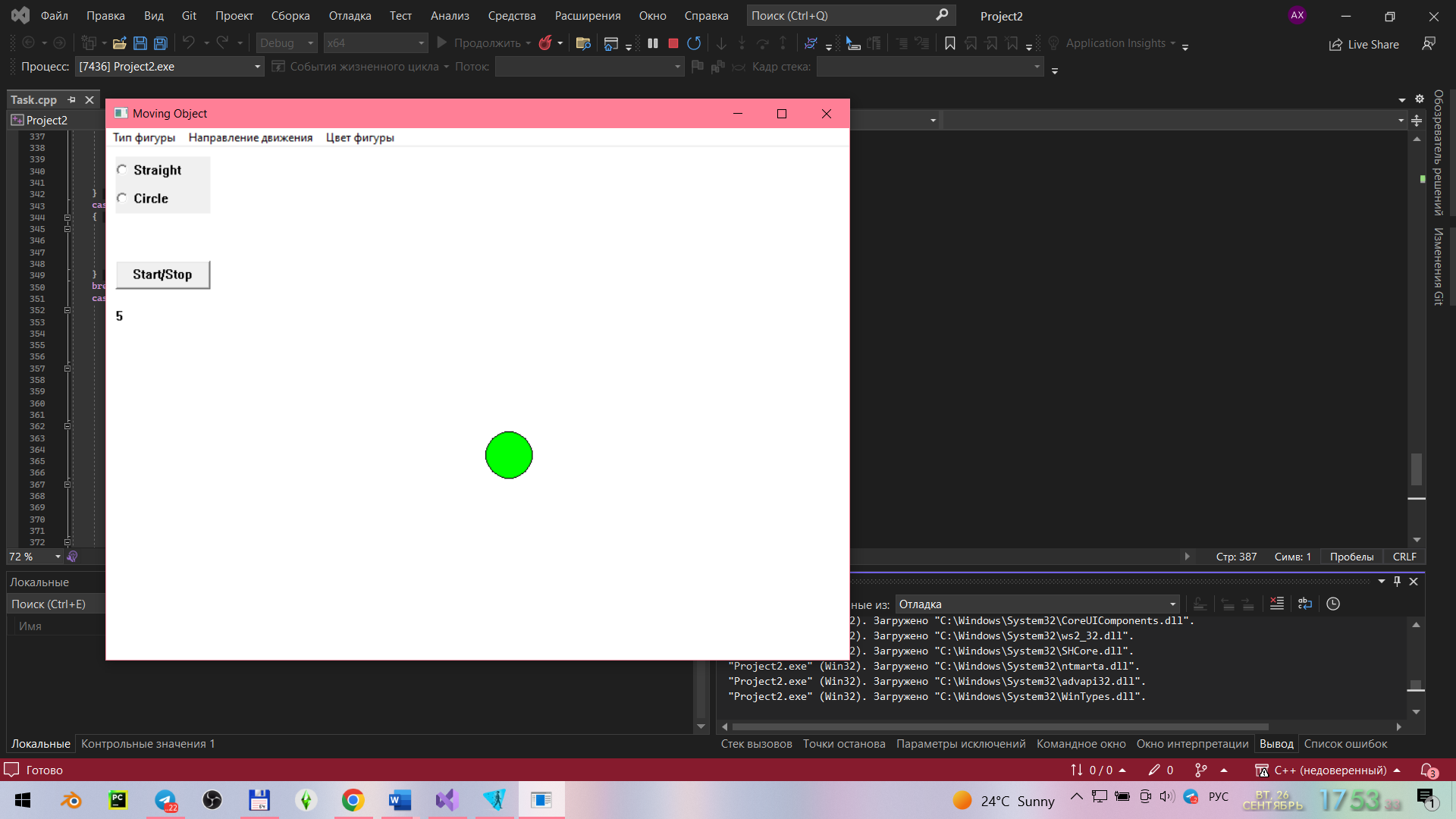


Рисунок 3.1 – Результат работы программы

Для наглядного примера на рисунке 3.2 будет предоставлен результаты работы программы с измененной скоростью, формой фигуры, с измененным цветом фигуры, а также с выбором траектории.

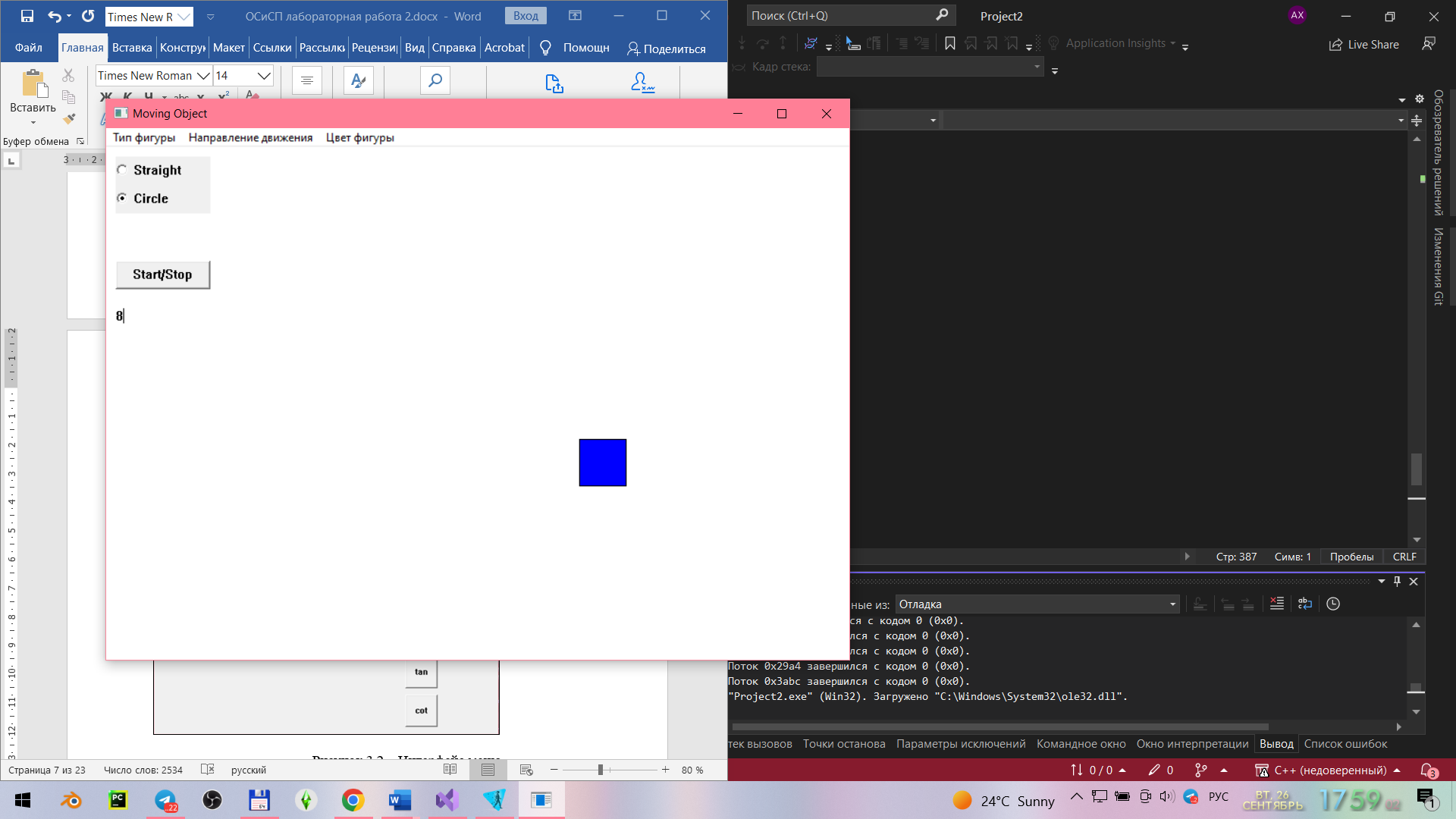


Рисунок 3.2 – Результат работы программы с измененными параметрами

## ВЫВОДЫ

В ходе выполнения данной лабораторной работы был создан простой графический анимационный пример с использованием Win32 API и GDI. Эта работа позволила ознакомиться с основными концепциями программирования под Windows, включая создание окна, обработку сообщений, работу с графической библиотекой GDI, а также использование таймеров и элементов управления.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Щупак Ю. Win32 API. Разработка приложений для Windows. – СПб: Питер, 2008. – 592 с.: ип.

[2] [Электронный ресурс]. – Режим доступа: [https://learn.microsoft.com/ru-ru/windows/win32/api – Дата доступа 20.09.2023](https://learn.microsoft.com/ru-ru/windows/win32/api%20–%20Дата%20доступа%2020.09.2023)

[3] [Электронный ресурс]. – Режим доступа: [https://stackoverflow.com/questions/17187265/how-to-group-radio-box-buttons-using-win32-api – Дата доступа 23.09.2023](https://stackoverflow.com/questions/17187265/how-to-group-radio-box-buttons-using-win32-api%20–%20Дата%20доступа%2023.09.2023)

[4] [Электронный ресурс]. – Режим доступа: <https://prog-cpp.ru/winelements/> – Дата доступа 23.09.2023

[5] [Электронный ресурс]. – Режим доступа: [https://learn.microsoft.com/en-us/windows/win32/gdi/windows-gdi – Дата доступа 24.09.2023](https://learn.microsoft.com/en-us/windows/win32/gdi/windows-gdi%20–%20Дата%20доступа%2024.09.2023)

[6] [Электронный ресурс]. – Режим доступа: <https://learn.microsoft.com/en-us/windows/win32/controls/button-messages> – Дата доступа 24.09.2023

## ПРИЛОЖЕНИЕ А

## (обязательное)

## Листинг кода

**Task.cpp**

#include <Windows.h>

#include <cmath>

void DrawObject(HDC hdc);

void UpdateObjectPosition();

void OnPaint(HDC hdc);

void OnTimer(HWND hWnd);

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam);

HWND g\_hWnd; // Хранение окна

HBRUSH g\_hBrush = NULL; // Хранение

HMENU hMenu = CreateMenu();

HMENU hSubMenuShape = CreatePopupMenu();

HMENU hSubMenuMove = CreatePopupMenu();

HMENU hSubMenuColor = CreatePopupMenu();

COLORREF g\_shapeColor = RGB(0, 255, 0);

int g\_startX = 400; // Начальные координаты X объекта

int g\_startY = 300; // Начальные координаты Y объекта

int g\_angle = 45; // Угол для траектории окружности

int g\_speed = 5; // Скорость движения объекта

int g\_x = g\_startX; // Текущие координаты объекта

int g\_y = g\_startY;

bool g\_isMoving = false;

int g\_direction = 0; // Направление движения (0 - вправо, 1 - вниз, 2 - влево, 3 - вверх)

int g\_selectedTrajectory = 0; // 0 - Прямая, 1 - Окружность

int g\_selectedShape = 0; // 0 - Круг, 1 - Квадрат

bool g\_clockwise = true; // true - по часовой стрелке, false - против часовой стрелки

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)

{

// 1. Заполняем структуру WNDCLASSEX

WNDCLASSEX wcex;

wcex.cbSize = sizeof(WNDCLASSEX);

wcex.style = CS\_HREDRAW | CS\_VREDRAW;

wcex.lpfnWndProc = WndProc;

wcex.cbClsExtra = 0;

wcex.cbWndExtra = 0;

wcex.hInstance = hInstance;

wcex.hIcon = LoadIcon(hInstance, IDI\_APPLICATION);

wcex.hCursor = LoadCursor(NULL, IDC\_ARROW);

wcex.hbrBackground = (HBRUSH)(COLOR\_WINDOW + 1);

wcex.lpszMenuName = NULL;

wcex.lpszClassName = L"Animation";

wcex.hIconSm = LoadIcon(wcex.hInstance, IDI\_APPLICATION);

// 2. Регистрируем класс окна

if (!RegisterClassEx(&wcex))

{

MessageBox(NULL, L"Window Registration Failed!", L"Error!", MB\_ICONEXCLAMATION | MB\_OK);

return 0;

}

// 3. Создаем окно

g\_hWnd = CreateWindow(L"Animation", L"Moving Object", WS\_OVERLAPPEDWINDOW,

CW\_USEDEFAULT, CW\_USEDEFAULT, 800, 600, NULL, NULL, hInstance, NULL);

if (g\_hWnd == NULL)

{

MessageBox(NULL, L"Window Creation Failed!", L"Error!", MB\_ICONEXCLAMATION | MB\_OK);

return 0;

}

// 4. Отображаем окно

ShowWindow(g\_hWnd, nCmdShow);

UpdateWindow(g\_hWnd);

// 5. Запускаем основной цикл сообщений

MSG msg;

while (GetMessage(&msg, NULL, 0, 0))

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

return msg.wParam;

}

void DrawObject(HDC hdc)

{

HBRUSH hBrush = CreateSolidBrush(g\_shapeColor);

HBRUSH hOldBrush = (HBRUSH)SelectObject(hdc, hBrush);

if (g\_selectedShape == 0) // Круг

{

Ellipse(hdc, g\_x, g\_y, g\_x + 50, g\_y + 50);

}

else // Квадрат

{

Rectangle(hdc, g\_x, g\_y, g\_x + 50, g\_y + 50);

}

SelectObject(hdc, hOldBrush);

DeleteObject(hBrush);

}

void UpdateObjectPosition()

{

if (g\_selectedTrajectory == 0) // Прямая

{

if (g\_clockwise)

{

// В зависимости от текущего направления движения изменяем координаты объекта

if (g\_direction == 0) // Вправо

{

g\_x += g\_speed;

if (g\_x >= 600) // Достигли правого края

{

g\_x = 600;

g\_direction = 1; // Меняем направление на вниз

}

}

else if (g\_direction == 1) // Вниз

{

g\_y += g\_speed;

if (g\_y >= 450) // Достигли нижнего края

{

g\_y = 450;

g\_direction = 2; // Меняем направление на влево

}

}

else if (g\_direction == 2) // Влево

{

g\_x -= g\_speed;

if (g\_x <= 100) // Достигли левого края

{

g\_x = 100;

g\_direction = 3; // Меняем направление на вверх

}

}

else if (g\_direction == 3) // Вверх

{

g\_y -= g\_speed;

if (g\_y <= 50) // Достигли верхнего края

{

g\_y = 50;

g\_direction = 0; // Меняем направление на вправо

}

}

}

else // Против часовой стрелки

{

// В зависимости от текущего направления движения изменяем координаты объекта

if (g\_direction == 0) // Вправо

{

g\_x -= g\_speed;

if (g\_x <= 100) // Достигли левого края

{

g\_x = 100;

g\_direction = 3; // Меняем направление на вверх

}

}

else if (g\_direction == 1) // Вниз

{

g\_y -= g\_speed;

if (g\_y <= 50) // Достигли верхнего края

{

g\_y = 50;

g\_direction = 0; // Меняем направление на вправо

}

}

else if (g\_direction == 2) // Влево

{

g\_x += g\_speed;

if (g\_x >= 600) // Достигли правого края

{

g\_x = 600;

g\_direction = 1; // Меняем направление на вниз

}

}

else if (g\_direction == 3) // Вверх

{

g\_y += g\_speed;

if (g\_y >= 450) // Достигли нижнего края

{

g\_y = 450;

g\_direction = 2; // Меняем направление на влево

}

}

}

}

else if (g\_selectedTrajectory == 1) // Окружность

{

// Изменяем угол движения, чтобы объект двигался по окружности

if (g\_clockwise)

{

g\_angle += g\_speed;

if (g\_angle >= 360)

{

g\_angle -= 360;

}

}

else

{

g\_angle -= g\_speed;

if (g\_angle < 0)

{

g\_angle += 360;

}

}

double radians = g\_angle \* 3.14159265358979323846 / 180.0;

// Вычисляем новые координаты на окружности

g\_x = g\_startX - static\_cast<int>(100 \* cos(radians)); // Изменить число, чтобы увеличить диаметр

g\_y = g\_startY - static\_cast<int>(100 \* sin(radians));

}

// Проверка на выход за границы окна

if (g\_x < 100) g\_x = 100;

if (g\_x > 600) g\_x = 600;

if (g\_y < 50) g\_y = 50;

if (g\_y > 450) g\_y = 450;

}

void OnPaint(HDC hdc)

{

if (g\_isMoving)

{

UpdateObjectPosition();

}

SelectObject(hdc, g\_hBrush); // Выбираем кисть для заливки

DrawObject(hdc);

}

void OnTimer(HWND hWnd)

{

if (g\_isMoving)

{

UpdateObjectPosition();

// Обновляем окно

InvalidateRect(hWnd, NULL, TRUE); // Перерисовка

}

}

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)

{

switch (message)

{

case WM\_CREATE:

// Создаем элементы управления для выбора траектории и скорости

CreateWindowA("button", "Straight", WS\_CHILD | WS\_VISIBLE | BS\_AUTORADIOBUTTON, 10, 10, 100, 30, hWnd, (HMENU)101, NULL, NULL);

CreateWindowA("button", "Circle", WS\_CHILD | WS\_VISIBLE | BS\_AUTORADIOBUTTON, 10, 40, 100, 30, hWnd, (HMENU)102, NULL, NULL);

CreateWindowA("button", "Start/Stop", WS\_VISIBLE | WS\_CHILD, 10, 120, 100, 30, hWnd, (HMENU)104, NULL, NULL);

CreateWindowA("edit", "5", WS\_CHILD | WS\_VISIBLE | ES\_NUMBER, 10, 170, 50, 20, hWnd, (HMENU)105, NULL, NULL);

// Создаем меню для выбора типа фигуры, направления движения, изменения цвета

AppendMenu(hSubMenuShape, MF\_STRING, 201, L"Круг");

AppendMenu(hSubMenuShape, MF\_STRING, 202, L"Квадрат");

AppendMenu(hMenu, MF\_STRING | MF\_POPUP, (UINT)hSubMenuShape, L"Тип фигуры");

AppendMenu(hSubMenuMove, MF\_STRING, 203, L"По часовой стрелке");

AppendMenu(hSubMenuMove, MF\_STRING, 204, L"Против часовой стрелки");

AppendMenu(hMenu, MF\_STRING | MF\_POPUP, (UINT)hSubMenuMove, L"Направление движения");

AppendMenu(hSubMenuColor, MF\_STRING, 301, L"Красный");

AppendMenu(hSubMenuColor, MF\_STRING, 302, L"Зеленый");

AppendMenu(hSubMenuColor, MF\_STRING, 303, L"Синий");

AppendMenu(hMenu, MF\_STRING | MF\_POPUP, (UINT)hSubMenuColor, L"Цвет фигуры");

SetMenu(hWnd, hMenu);

break;

case WM\_COMMAND:

switch (LOWORD(wParam))

{

case 101: // Выбрана Прямая траектория

if (HIWORD(wParam) == BN\_CLICKED)

{

g\_selectedTrajectory = 0;

}

break;

case 102: // Выбрана Окружность траектория

if (HIWORD(wParam) == BN\_CLICKED)

{

g\_selectedTrajectory = 1;

}

break;

case 104: // Кнопка Start/Stop

{

if (HIWORD(wParam) == BN\_CLICKED)

{

g\_isMoving = !g\_isMoving; // Переключаем анимацию

if (g\_isMoving)

{

SetTimer(hWnd, 1, 1000 / 60, NULL); // Устанавливаем таймер для анимации (60 FPS)

}

else

{

KillTimer(hWnd, 1); // Останавливаем таймер

// Обновляем окно

InvalidateRect(hWnd, NULL, TRUE);

}

}

}

break;

case 105: // Поле ввода скорости

if (HIWORD(wParam) == EN\_CHANGE)

{

// Обработка изменения значения скорости

WCHAR buffer[10];

GetWindowText(GetDlgItem(hWnd, 105), buffer, 10);

g\_speed = \_wtoi(buffer);

}

break;

case 201: // Круг

g\_selectedShape = 0;

break;

case 202: // Квадрат

g\_selectedShape = 1;

break;

case 203: // По часовой стрелке

g\_clockwise = true;

break;

case 204: // Против часовой стрелки

g\_clockwise = false;

break;

case 301:

g\_shapeColor = RGB(255, 0, 0); // Красный цвет

break;

case 302:

g\_shapeColor = RGB(0, 255, 0); // Зеленый цвет

break;

case 303:

g\_shapeColor = RGB(0, 0, 255); // Синий цвет

break;

}

break;

case WM\_PAINT:

{

PAINTSTRUCT ps;

HDC hdc = BeginPaint(hWnd, &ps);

OnPaint(hdc);

EndPaint(hWnd, &ps);

break;

}

case WM\_TIMER:

{

if (wParam == 1)

{

OnTimer(hWnd);

}

}

break;

case WM\_DESTROY:

if (hMenu != NULL)

{

DestroyMenu(hMenu);

hMenu = NULL;

}

if (hSubMenuShape != NULL)

{

DestroyMenu(hSubMenuShape);

hSubMenuShape = NULL;

}

if (hSubMenuMove != NULL)

{

DestroyMenu(hSubMenuMove);

hSubMenuMove = NULL;

}

if (hSubMenuColor != NULL)

{

DestroyMenu(hSubMenuColor);

hSubMenuColor = NULL;

}

if (g\_hBrush != NULL)

{

DeleteObject(g\_hBrush);

g\_hBrush = NULL;

}

PostQuitMessage(0);

break;

case WM\_CLOSE:

PostQuitMessage(0);

break;

default:

return DefWindowProc(hWnd, message, wParam, lParam);

}

return 0;

}