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

ОТЧЁТ

к лабораторной работе №2

на тему

Расширенное использование оконного интерфейса Win32 и GDI

Выполнила: студентка группы 153502

Шайкова Вероника Андреевна

Проверил: Гриценко Никита Юрьевич

Минск 2023

СОДЕРЖАНИЕ

[1 Постановка задачи 3](#_Toc145670720)

[2 Результаты выполнения лабораторной работы 4](#_Toc145670721)

[Выводы](#_Toc145670722) 6

[Список использованных источников 7](#_Toc145670723)

[Приложение А (обязательное) Листинг кода 8](#_Toc145670724)

## 1 ПОСТАНОВКА ЗАДАЧИ

Целью выполнения лабораторной работы является создание оконного приложения на Win32 API с использованием интерфейса GDI. Для этого создать окно, которое отображает анимированный мяч и позволяет менять его скорость, размер, траекторию, изменять режим прокрашивания, и позволяющее отследить траекторию при необходимости. Дополнительно создать собственную иконку приложения.

## 2 РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

В ходе выполнения лабораторной работы было создано приложение, позволяющее управлять анимированным шаром (Рисунок 1).

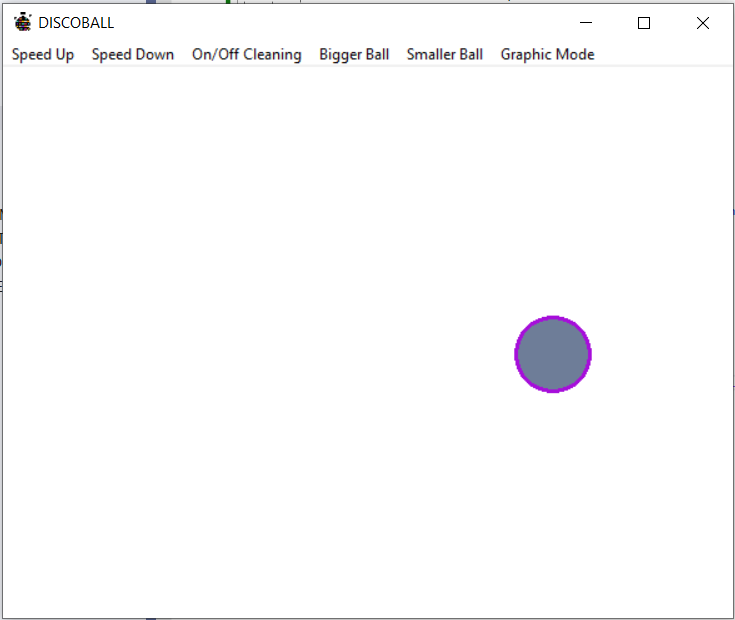


Рисунок 1 ─ Анимированный шар

Для управлением движения шара пользователь может использовать 6 пунктов оконного меню (Рисунок 2), с помощью которого возможны изменения размера, скорости шара, а также возможность отслеживания траектории (Рисунок 3) или переход в черно-белый режим (Рисунок 4).

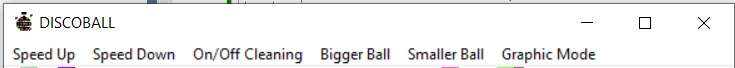


Рисунок 2 ─ Меню



Рисунок 3 ─ Отслеживание траектории

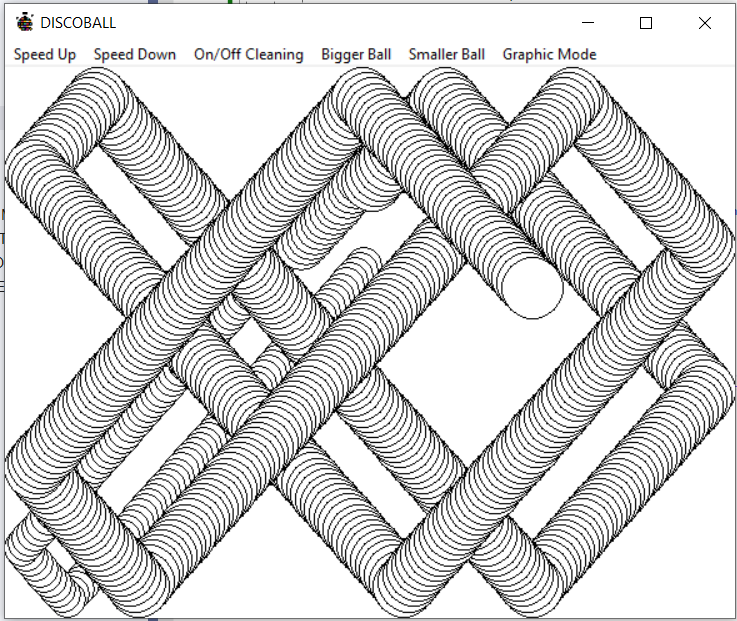


Рисунок 4 ─ Переход в ч/б режим

Изменения траектории движения происходят самостоятельно при достижении края клиентской области окна, а также по клику левой кнопкой мыши (сообщение WM\_LBUTTONDOWN) по клиентской области. Во втором случае меняется угол наклона к горизонтальной оси (по умолчанию ─ 0). Анимация осуществляется путем вызова функции RedrawWindow при срабатывании таймера каждые 5 мс.

Дополнительно было реализовано добавление собственной иконки приложения (Рисунок 5).



Рисунок 5 ─ Иконка

## ВЫВОДЫ

В ходе выполнения лабораторной работы было создано приложение с собственной иконкой, позволяющее управлять простой анимацией эллипса. Анимация происходит путем постоянной частой перерисовки окна с изменением позиций отрисовки объекта на каждом шагу. При этом вся область окна признается недействительной (т.е. нуждающейся в перерисовке), а новые «кадры» наслаиваются на старые. Это позволяет отследить траекторию анимации, но требует дополнительного закрашивания всего окна на каждом шаге для анимации без побочных эффектов.

Поскольку рисование производится в условиях обработки события WM\_PAINT, изменение размеров окна также может повлиять на скорость или плавность анимации.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

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

[2] Создание классических приложений для Windows с использованием API Win32 [Электронный ресурс]. ─ Режим доступа: https://learn.microsoft.com/ru-ru/windows/win32/api ─ Дата доступа 20.09.2023

## ПРИЛОЖЕНИЕ А

## (обязательное)

## Листинг кода

**Lab2.cpp**

#include <windows.h>

#include <windowsx.h>

#include <wingdi.h>

#include <math.h>

#include "resource.h"

#include "globals\_defines.h"

WNDCLASS createWindowClass(HBRUSH background\_color, HCURSOR cursor\_type, HINSTANCE hInstance, HICON Icon, LPCWSTR window\_name, WNDPROC procedure) {

// создание пустого класса

WNDCLASS new\_window = { 0 };

// наполняем пустой класс параметрами, требуемыми для конкретного окна

new\_window.hbrBackground = background\_color;

new\_window.hCursor = cursor\_type;

new\_window.hInstance = hInstance;

new\_window.hIcon = Icon;

new\_window.lpszClassName = window\_name;

new\_window.lpfnWndProc = procedure;

return new\_window;

}

void CreateMenus(HWND wnd) {

HMENU root = CreateMenu();

AppendMenu(root, MF\_STRING, M\_SPEED\_UP, L"Speed Up");

AppendMenu(root, MF\_STRING, M\_SPEED\_DOWN, L"Speed Down");

AppendMenu(root, MF\_STRING, M\_CLEANING, L"On/Off Cleaning");

AppendMenu(root, MF\_STRING, M\_RAD\_UP, L"Bigger Ball");

AppendMenu(root, MF\_STRING, M\_RAD\_DOWN, L"Smaller Ball");

AppendMenu(root, MF\_STRING, M\_GRAPHIC, L"Graphic Mode");

SetMenu(wnd, root);

}

LRESULT CALLBACK createNewProcedure(HWND window, UINT message, WPARAM wp, LPARAM lp) {

switch (message) {

case WM\_CREATE:

{

SetTimer(window, 1, 6, NULL);

CreateMenus(window);

break;

}

case WM\_DESTROY:

{

// выход из программы по закрытию окна

PostQuitMessage(0);

break;

}

case WM\_PAINT:

{

GetClientRect(window, &clientRect);

PAINTSTRUCT ps;

HDC hdc = BeginPaint(window, &ps);

if (need\_cleaning) FillRect(hdc, &clientRect, WHITE\_BRUSH);

if (cos\_a > 0 && x\_0 > clientRect.right - 2 \* rad || cos\_a < 0 && x\_0 < 1) cos\_a \*= -1;

if (sin\_a < 0 && y\_0 < 1 || sin\_a > 0 && y\_0 > clientRect.bottom - 2 \* rad) sin\_a \*= -1;

x\_0 = x\_0 + v \* cos\_a;

y\_0 = y\_0 + v \* sin\_a;

if (need\_colour) {

HPEN newPen = CreatePen(PS\_ENDCAP\_SQUARE, 3, RGB(rand() % 255, rand() % 255, rand() % 255));

HBRUSH newBrush = CreateSolidBrush(RGB(rand() % 255, rand() % 255, rand() % 255));

HGDIOBJ oldPen = SelectObject(hdc, newPen);

HGDIOBJ oldBrush = SelectObject(hdc, newBrush);

Ellipse(hdc, x\_0, y\_0, x\_0 + 2 \* rad, y\_0 + 2 \* rad);

SelectObject(hdc, oldPen);

SelectObject(hdc, oldBrush);

DeleteObject(newPen);

DeleteObject(newBrush);

}

else Ellipse(hdc, x\_0, y\_0, x\_0 + 2 \* rad, y\_0 + 2 \* rad);

EndPaint(window, &ps);

break;

}

case WM\_TIMER:{

RedrawWindow(window, NULL, NULL, RDW\_INVALIDATE);

break;

}

case WM\_LBUTTONDOWN:

{

double x\_1 = GET\_X\_LPARAM(lp), y\_1 = GET\_Y\_LPARAM(lp);

double dx = x\_1 - x\_0, dy = y\_1 - y\_0;

sin\_a = dy / sqrt(dy \* dy + dx \* dx);

cos\_a = dx / sqrt(dx \* dx + dy \* dy);

break;

}

case WM\_COMMAND:

switch (wp)

{

case M\_SPEED\_UP:

if (v < 15) v += 1;

break;

case M\_SPEED\_DOWN:

if (v >= 2) v -= 1;

break;

case M\_CLEANING:

need\_cleaning = !need\_cleaning;

break;

case M\_RAD\_UP:

if (rad < 90.0) rad += 5.0;

break;

case M\_RAD\_DOWN:

if (rad > 15.0) rad -= 5.0;

break;

case M\_GRAPHIC:

need\_colour = !need\_colour;

}

break;

default: return DefWindowProc(window, message, wp, lp);

}

}

int WINAPI WinMain(HINSTANCE hInst, HINSTANCE hPrevInst, LPSTR args, int ncmdshow) {

WNDCLASS main\_window = createWindowClass(CreateSolidBrush(RGB(255,255,255)), LoadCursor(NULL, IDC\_CROSS), hInst, LoadIcon(NULL, IDI\_WINLOGO), L"MainWindow", createNewProcedure);

if (!RegisterClassW(&main\_window)) {

return -1;

}

MSG main\_window\_message = { 0 };

CreateWindow(L"MainWindow", L"Kind of headline", WS\_OVERLAPPEDWINDOW | WS\_VISIBLE, 200, 200, 600, 500, NULL, NULL, NULL, NULL);

while (GetMessage(&main\_window\_message, NULL, NULL, NULL)) {

TranslateMessage(&main\_window\_message);

DispatchMessage(&main\_window\_message);

}

return 0;

}

**globals\_defines.h**

#pragma once

#define M\_SPEED\_UP 2

#define M\_SPEED\_DOWN 3

#define M\_CLEANING 4

#define M\_RAD\_UP 5

#define M\_RAD\_DOWN 6

#define M\_GRAPHIC 7

double v = 2.5;

double x\_0 = 100.0, y\_0 = 200.0;

double sin\_a = 0.0, cos\_a = 1.0;

double rad = 30.0;

BOOL need\_cleaning = true;

BOOL need\_colour = true;

RECT clientRect;

**resource.h**

//{{NO\_DEPENDENCIES}}

// Включаемый файл, созданный в Microsoft Visual C++.

// Используется lab2.rc

//

#define IDI\_ICON1 101

// Next default values for new objects

//

#ifdef APSTUDIO\_INVOKED

#ifndef APSTUDIO\_READONLY\_SYMBOLS

#define \_APS\_NEXT\_RESOURCE\_VALUE 102

#define \_APS\_NEXT\_COMMAND\_VALUE 40001

#define \_APS\_NEXT\_CONTROL\_VALUE 1001

#define \_APS\_NEXT\_SYMED\_VALUE 101

#endif

#endif