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 №3

на тему

Анимация с возможностью восстановления последнего кадра. Сжатие и распаковка данных

Выполнила: студентка группы 153502

Шайкова Вероника Андреевна

Проверил: Гриценко Никита Юрьевич

Минск 2023

СОДЕРЖАНИЕ

[1 Постановка задачи 3](#_Toc147885133)

[2 Георетические сведения 4](#_Toc147885134)

[3 Результаты выполнения лабораторной работы 5](#_Toc147885135)

[Выводы 6](#_Toc147885136)

[Список использованных источников 7](#_Toc147885137)

[ПРИЛОЖЕНИЕ А (обязательное) Листинг кода 8](#_Toc147885138)

## 1 ПОСТАНОВКА ЗАДАЧИ

Целью выполнения лабораторной работы является создание оконного приложения на Win32 API на основе предыдущей лабораторной работы, используя расширенные возможности для работы с памятью и вводом/выводом. Реализовать сохранение последнего кадра путем сохранения в файл на диске необходимых данных в сжатом виде (использование библиотеки <compressapi.h>)

## 2 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Для сжатия данных используется функция

BOOL Compress(

[in] COMPRESSOR\_HANDLE CompressorHandle,

[in] LPCVOID UncompressedData,

[in] SIZE\_T UncompressedDataSize,

[out] PVOID CompressedBuffer,

[in] SIZE\_T CompressedBufferSize,

[out] PSIZE\_T CompressedDataSize

),

которая принимает в качестве параметров дескриптор компрессора, указатель на буфер данных для сжатия, размер этого буфера в байтах, указатель на буфер для сжатых данных, максимальный размер сжатых данных, а также указатель на переменную для записи настоящего размера сжатых данных.

Для использования этой функции необходимо предварительно создать компрессор:

BOOL CreateCompressor(

[in] DWORD Algorithm,

[in, optional] PCOMPRESS\_ALLOCATION\_ROUTINES AllocationRoutines,

[out] PCOMPRESSOR\_HANDLE CompressorHandle

),

а после закрыть его:

BOOL CloseCompressor(

[in] COMPRESSOR\_HANDLE CompressorHandle

);

Кроме того доступны функции QueryCompressorInformation (запрос информации об алгоритме компрессора), ResetCompressor (подготовка к сжатию нового блока), SetCompressorInformation (задает информацию в компрессоре для определенного алгоритма).

Аналогично доступны функции распаковки: CreateDecompressor, Decompress, ResetDecompressor, CloseDecompressor, SetDecompressorInformation, QueryDecompressorInformation.

В WIN32 API доступны следующие алгоритммы сжатия: COMPRESS\_ALGORITHM\_LZMS, COMPRESS\_ALGORITHM\_MAX, COMPRESS\_ALGORITHM\_MSZIP, COMPRESS\_ALGORITHM\_XPRESS, COMPRESS\_ALGORITHM\_XPRESS\_HUFF

## 3 РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

В ходе выполнения лабораторной работы было создано приложение, позволяющее управлять анимированным шаром (рисунок 1), а также сохраняющее его положение в окне, характеристики (радиус, скорость, угол движения) в файл на диске в сжатом виде.

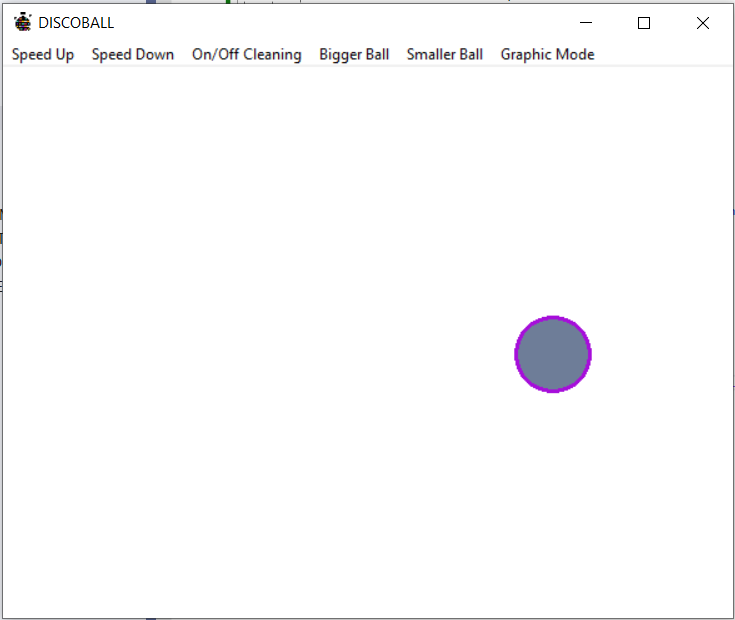


Рисунок 1 ─ Анимированный шар

В момент закрытия окна необходимые данные преобразуются в строку-словарь, после чего происходит сжатие и сохранение данных в файл (рисунок 2):

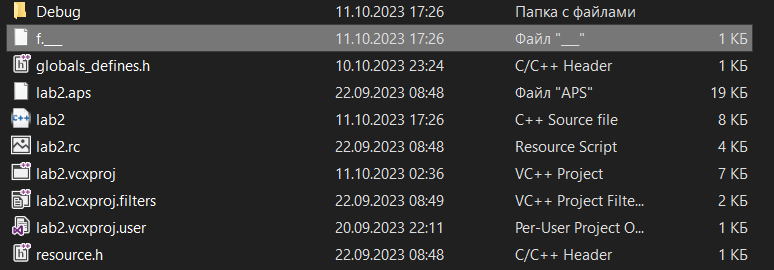
****

Рисунок 2 ─ файл сохранения

При запуске приложения происходит считывание данных из файла, их распаковка и восстановление кадра. В случае, если данные повреждены или файл не обнаружен, используются данные по умолчанию.

## ВЫВОДЫ

В ходе выполнения лабораторной работы было создано приложение с собственной иконкой, позволяющее управлять простой анимацией эллипса. И выполнять сохранение последнего кадра. 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.

Сохранение происходит в сжатом виде в файл на диске по алгоритму COMPRESS\_ALGORITHM\_MSZIP.

## 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 <vector>

#include <iostream>

#include <sstream>

#include <windowsx.h>

#include <wingdi.h>

#include <math.h>

#include <compressapi.h>

#include <fileapi.h>

#include <string>

#include "resource.h"

#include "globals\_defines.h"

void compresss\_and\_save\_to\_file()

{

// compress

std::string to\_compress\_data = "";

to\_compress\_data += "v:\t" + std::to\_string(v) + "\n";

to\_compress\_data += "x0:\t" + std::to\_string(x\_0) + "\n";

to\_compress\_data += "y0:\t" + std::to\_string(y\_0) + "\n";

to\_compress\_data += "sin(a):\t" + std::to\_string(sin\_a) + "\n";

to\_compress\_data += "cos(a):\t" + std::to\_string(cos\_a) + "\n";

to\_compress\_data += "radius:\t" + std::to\_string(rad) + "\n";

to\_compress\_data += "this is just spam \n\n to make\n compression reasonable\ndkfjnbgisledfvbiernkv jarlg kberjh grdjbg jdfb gjhdbgj rjjerbg bshg sdb bjd jbv cv jd df dh iudfoudsf uosdf df hfs ";

PCOMPRESSOR\_HANDLE a = new COMPRESSOR\_HANDLE();

char compressed\_data[600];

PSIZE\_T compressed\_data\_size = new SIZE\_T();

CreateCompressor(COMPRESS\_ALGORITHM\_MSZIP, NULL, a);

Compress(\*a, to\_compress\_data.c\_str(), to\_compress\_data.size(), compressed\_data, sizeof(compressed\_data), compressed\_data\_size);

CloseCompressor(\*a);

// save compressed

HANDLE wfile = CreateFileA(

"f.\_\_\_", GENERIC\_WRITE, 0, NULL, CREATE\_ALWAYS, FILE\_ATTRIBUTE\_COMPRESSED, NULL);

DWORD res;

WriteFile(wfile, compressed\_data, \*compressed\_data\_size, &res, NULL);

CloseHandle(wfile);

}

void load\_from\_file\_and\_decompress() {

// load data

HANDLE rfile = CreateFileA(

"f.\_\_\_", GENERIC\_READ, 0, NULL, OPEN\_EXISTING, FILE\_ATTRIBUTE\_COMPRESSED, NULL);

DWORD deres;

char fromfile[600];

bool success = ReadFile(rfile, fromfile, sizeof(fromfile), &deres, NULL);

CloseHandle(rfile);

if (!success) return;

// decompress

PDECOMPRESSOR\_HANDLE b = new DECOMPRESSOR\_HANDLE();

char decompressed\_data[600];

PSIZE\_T decompressed\_size = new SIZE\_T();

CreateDecompressor(COMPRESS\_ALGORITHM\_MSZIP, NULL, b);

Decompress(\*b, fromfile, deres, decompressed\_data, sizeof(decompressed\_data), decompressed\_size);

CloseDecompressor(\*b);

// initialize variables with loaded data

std::string initialize\_data(decompressed\_data);

std::istringstream iss(initialize\_data);

std::string temp;

std::vector<std::string> strings;

while (std::getline(iss, temp, '\n')) {

strings.push\_back(temp);

}

for (int i = 0; i < strings.size(); i++) {

if (strings[i].starts\_with("v:\t")) {

v = std::stod(strings[i].substr(3));

}

else if (strings[i].starts\_with("x0:\t")) {

x\_0 = std::stod(strings[i].substr(4));

}

else if (strings[i].starts\_with("y0:\t")) {

y\_0 = std::stod(strings[i].substr(4));

}

else if (strings[i].starts\_with("sin(a):\t")) {

sin\_a = std::stod(strings[i].substr(8));

}

else if (strings[i].starts\_with("cos(a):\t")) {

cos\_a = std::stod(strings[i].substr(8));

}

else if (strings[i].starts\_with("radius:\t")) {

rad = std::stod(strings[i].substr(8));

}

}

}

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:

{

load\_from\_file\_and\_decompress();

SetTimer(window, 1, 6, NULL);

CreateMenus(window);

break;

}

case WM\_DESTROY:

{

compresss\_and\_save\_to\_file();

// 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 + 2.0 \* rad, y\_0 + 2.0 \* rad);

SelectObject(hdc, oldPen);

SelectObject(hdc, oldBrush);

DeleteObject(newPen);

DeleteObject(newBrush);

}

else Ellipse(hdc, x\_0, y\_0, x\_0 + 2.0 \* rad, y\_0 + 2.0 \* 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:

{

int b = 6;

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(hInst, MAKEINTRESOURCE(IDI\_ICON1)), L"MainWindow", createNewProcedure);

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

return -1;

}

MSG main\_window\_message = { 0 };

CreateWindow(L"MainWindow", L"DISCOBALL", 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

#pragma comment ( lib , "Cabinet.lib" )

#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