Министерство образования Республики Беларусь
Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Операционные Системы и Системное Программирование (ОСиСП)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовой работе
на тему:

«Видеозахват экрана»

БГУИР КП 1-40 01 01 605 ПЗ

Студент: гр. 851006 Верещагин Н. В.

Руководитель: Жиденко А. Л.

Минск 2020

Учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ
Заведующий кафедрой ПОИТ

_____
(подпись)
Лапицкая Н.В. 2020 г.

ЗАДАНИЕ
по курсовому проектированию

Студенту Верещагину Николаю Владимировичу

1. Тема работы «Видеозахват экрана»
2. Срок сдачи студентом законченной работы 01.12.2020
3. Исходные данные к работе Документация по WinApi
4. Содержание расчётно-пояснительной записки (перечень вопросов, которые подлежат разработке)

Введение.

1.Анализ прототипов, литературных источников и формирование требований к проектироемому ПС;

2.Разработка алгоритма;

3. Разработка программного средства;

4.Тестирование, экспериментальные исследования и анализ полученных результатов;

5. Руководство пользователя программы;

Заключение, список литературы, ведомость, приложения.

5. Перечень графического материала (с точным обозначением обязательных чертежей и графиков)

1. Схема программы на А1

6. Консультант по курсовому проекту Жиденко А. Л.

7. Дата выдачи задания 05.09.2020 г.

8. Календарный график работы над проектом на весь период проектирования (с обозначением сроков выполнения и процентом от общего объёма работы):

раздел 1 к 20.09.2020 – 15 % готовности работы;

разделы 2, 3 к 13.10.2020 – 30 % готовности работы;

раздел 4 к 02.11.2020 – 60 % готовности работы;

раздел 5 к 26.11.2020 – 90 % готовности работы;

оформление пояснительной записки и графического материала к 01.12.2020 – 100 % готовности работы. Защита курсового проекта с 01.12 по 22.12 2020 г.

РУКОВОДИТЕЛЬ_____ Жиденко А. Л.

(подпись)

Задание принял к исполнению Верещагин Н.В_____ 05.09.2020 г.

(дата и подпись студента)

СОДЕРЖАНИЕ

Введение 5
1 Анализ предметной области 6
1.1 Обзор аналогов 6
1.2 Постановка задачи 9
2 Разработка программного средства 10
2.1 Структура программы 10
2.2 Интерфейс программного средства 10
2.3 Выделение области для съемки 11
2.4 Обработка нажатий 13
2.5 Запись видео 14
2.6 Оптимизация размера GIF-файла 15
2.7 Сжатие кадров алгоритмом LZW 16
2.8 Меню настроек программы 18
2.9 Автозапуск программы 18
3 Тестирование программного средства 19
4 Руководство пользователя 21
4.1 Руководство по установке и запуску 21
4.2 Установка автозапуска программы 21
4.3 Руководство по использованию 23
Заключение 25
Список использованных источников 26
Приложение 1 27

ВВЕДЕНИЕ

О возможности осуществления видеозахвата экрана известно давно. В последнее время чаще используется термин скринскастинг, то есть цифровая аудио и видеозапись, которая производится непосредственно с монитора компьютера.

Как правило, этот инструмент используется для более наглядного и динамичного представления того или иного цифрового приложения. Однако происходящее на экране может касаться и совершенно других вещей. И очень хорошо, если учитель для записи своих видеоуроков пользуется таким инструментом.

Специалисты подчёркивают, что особенностью обучения с видео является возможность задействовать сразу несколько «каналов восприятия информации»: зрительный, моторный и слуховой.

Видеозахват делят на ряд жанров, большинство которых можно использовать в образовании, прежде всего, в дистанционном обучении:

– демонстрация работы компьютерной программы, интернет-сервиса, обучающей программы. Задача такого видеоролика — показать привлекательность, эффективность, полезной такой программы.

– демонстрация последовательности действий с программой, приложением. Своеобразная видео инструкция о алгоритме действий.

– обзор программного обеспечения, электронного учебного пособия, любого цифрового образовательного ресурса. В данном случае используется, как скриншот, сопровождаемый текстом, так и скринкаст, который делает обзор более наглядным и динамичным. Иногда эти два приёма сочетаются в одном ролике.

Целью данного курсового проекта является разработка видеозахватывающей программы для демонстрации работы компьютерных программ.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Обзор аналогов

Существует ряд программ для видеозахвата экрана, каждая из них имеет свои преимущества и недостатки. На рисунке 1.1 представлена одна из наиболее удобных для освоения программа Recordit.

Рисунок 1.1 – Программа Recordit

Данная программа позволяет захватить определенную область экрана и начать запись в GIF формате. Преимуществом данной программы является то, что она бесплатна и мало весит. Recordit не имеет хорошей документации и не работает без интернета, также имеет низкое количество кадров в секунду. Не

имеет хорошего интерфейса, а также можно нарушить ход работы программы нечаянно закрыв ее при смене окон, рисунок 1.2 [1].

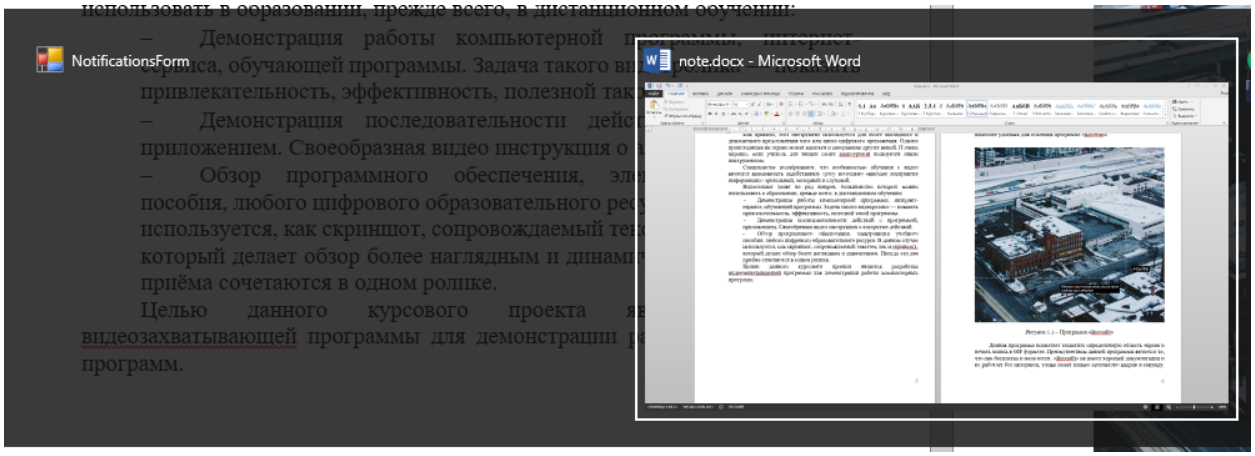

Рисунок 1.2 – Проблема при смене окон Recordit

Хорошим выбором может стать Bandicam. Главное меню приложения представлено на рисунке 1.3.

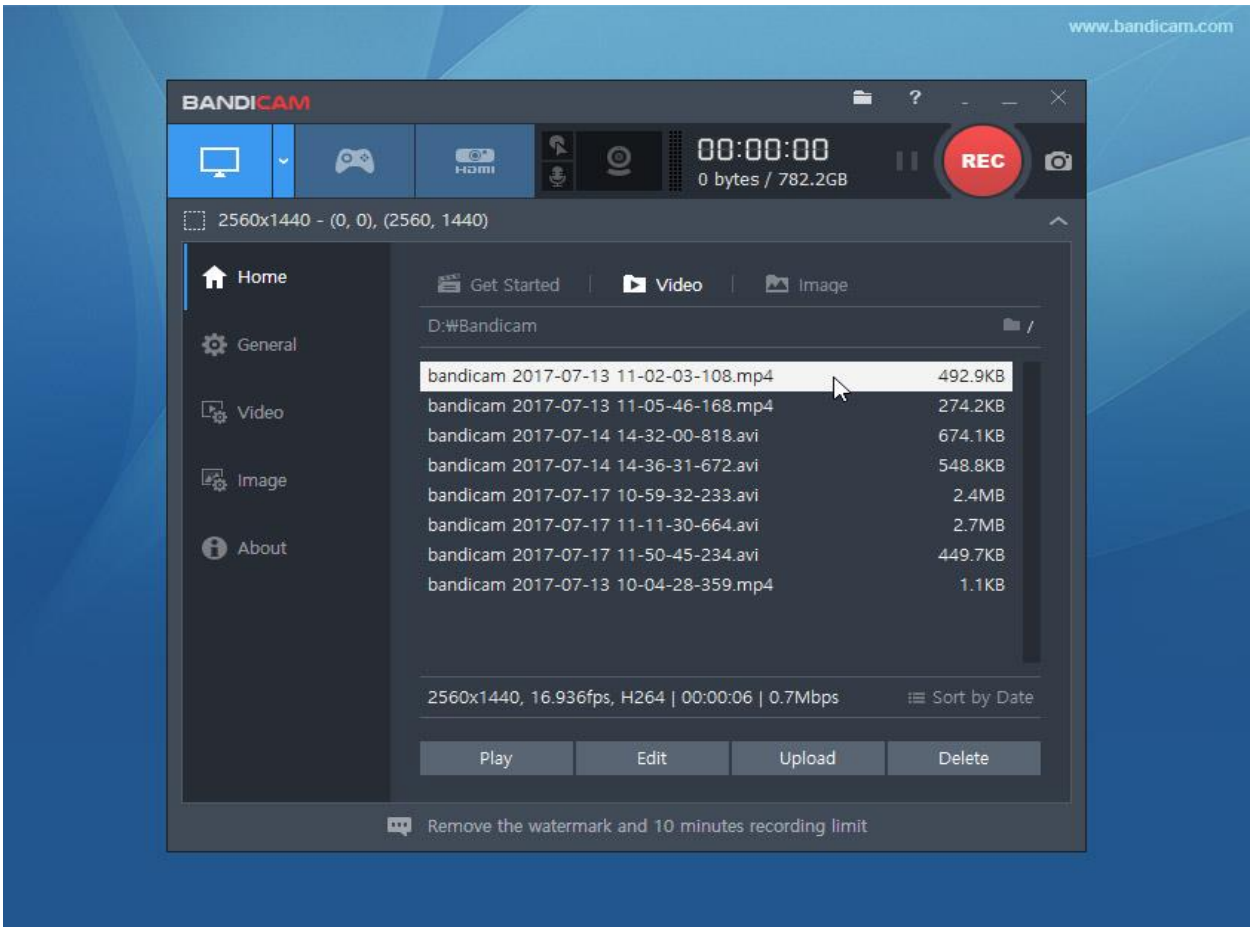

Рисунок 1.3 –Главное меню программы Bandicam

Bandicam — программа для создания скриншотов и захвата видео с экрана монитора. Программа имеет два режима. Один режим «Rectangle on a screen» позволяет захватывать скриншот или видео с экрана в определённом месте и размере. Является платной программой, то есть его можно использовать бесплатно в течение ограниченного срока [2]. Во время бесплатного срока Bandicam помещает своё имя в виде водяного знака в верхней части каждого скриншота или видео, а длина видеозаписи ограничена до 10 минут.

Программа Fraps. Главное меню приложения представлено на рисунке 1.4. Это компьютерная утилита, предназначенная для подсчёта количества FPS (кадров в секунду) в играх, работающих в режимах OpenGL и DirectX для операционных систем Windows. Программа также предназначена для создания скриншотов и записи видеороликов из полноэкранных 3D приложений [3]. Название программы происходит от аббревиатуры FPS и расшифровывается как Frames per second.

Рисунок 1.4 – Главное меню программы Fraps

Программа является платной, поэтому время записи ограничено и имеются водные знаки. Конечные файлы занимают много места на диске. Также программа затрудняет работу компьютера, из-за чего программы и игры хуже работают, чем на самом деле.

1.2 Постановка задачи

В рамках данного курсового проекта планируется разработка программного средства «Видеозахват экрана».

Будут разработаны алгоритмы захвата, записи, кодирования в GIF формат данных и интерактивное меню.

В программном средстве планируется реализовать следующие функции и алгоритмы:

– интуитивно-понятный интерфейс;

– алгоритм захвата экрана;

– алгоритм записи;

– алгоритм перехвата нажатий клавиш;

– конвертация в GIF формат;

– сжатие конечных файлов;

– меню настроек программы;

– алгоритм сжатия разрешения качества;

– алгоритм настройки задержки между кадрами.

Главной задачей является создать удобную и интуитивно-понятную программу для записи работы других программ, с целью наглядной демонстрации возможностей и функций других программ.

Для разработки программного средства будет использоваться язык программирования C++, библиотека Magick++ и операционная система Windows 10.

2 РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА

2.1 Структура программы

Требуется использовать три структурных блока:
– options – модуль, отвечающий за настройку опций программы;
– Video-recording – модуль, отвечающий за захват и запись видео;
– windowFunctions – модуль, отвечающий за основные функции окон.

2.2 Интерфейс программного средства

Внешний вид и удобность в использовании являются одними из главных критериев качества программного средства. Поэтому взаимодействие приложения с пользователем необходимо организовать максимально интуитивно и просто.

Интерфейс получился компактным и позволяет работать с любым приложением во время записи или во время ожидания программы. Как выглядит выделенная область для записи, показано на рисунке 2.1.

Рисунок 2.1 – Выделенная область для записи

Во время ожидания программа свернута и не мешает работе. Меню настроек программы представлено на рисунке 2.2.

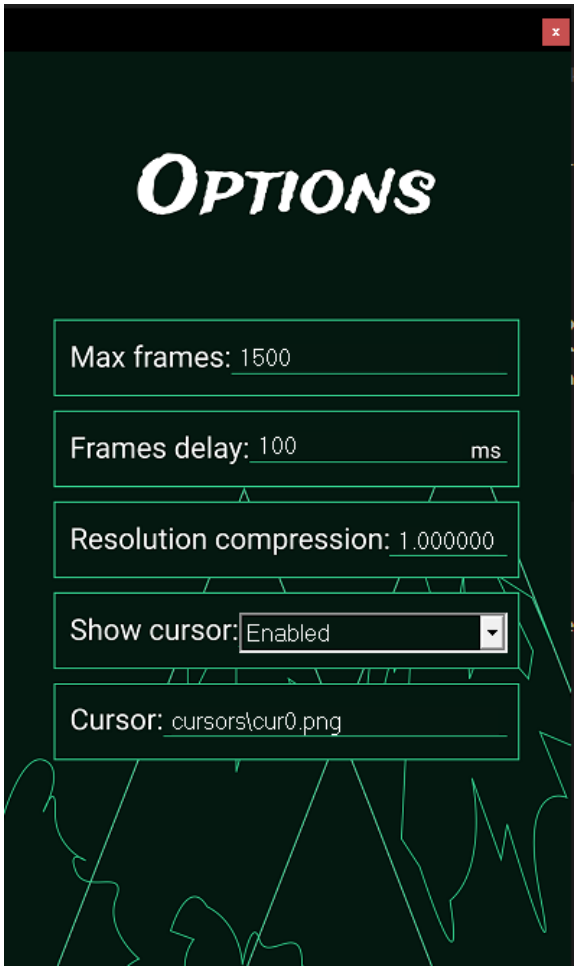

Рисунок 2.2 – Меню настроек программы

В меню настроек вы можете настроить максимальное количество кадров в видео, задержку между кадрами, можете сжать разрешение видео, выбрать любой курсор для его отображения или выключить его вовсе.

2.3 Выделение области для съемки

Область для выделения должна находиться поверх всех окон, чтобы пользователь видел ту область, которую он выделяет. Чтобы выделенная область не мешала нормальной работе программ, необходимо сделать ее прозрачной внутри и полупрозрачной снаружи, рисунок 2.3. Чтобы пользователь понимал о том, что он находиться в режиме выделения области для съемки, необходимо также сменить курсор указателя.

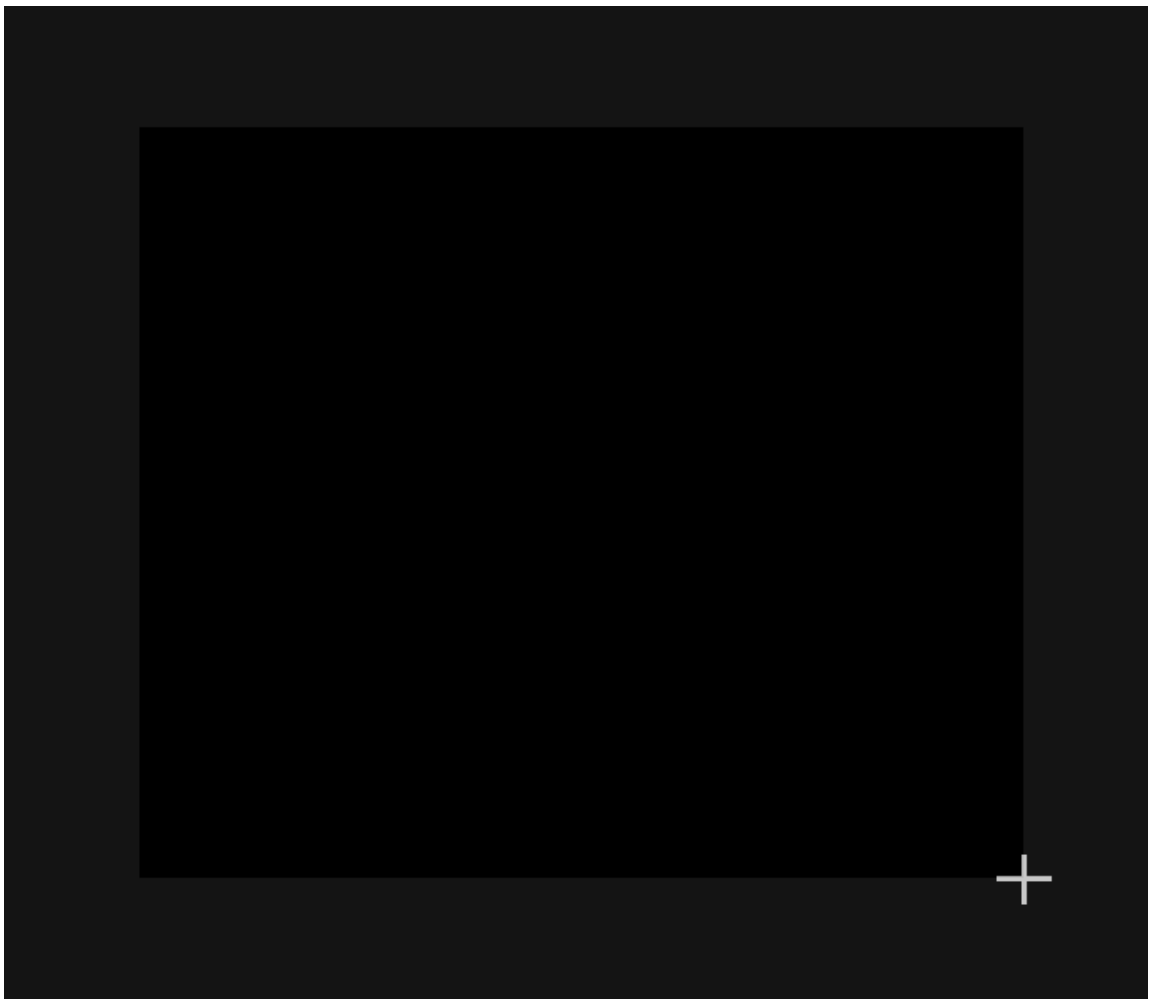

Рисунок 2.3 – Выделение области

Воспользуемся функцией «SetLayeredWindowAttributes», с помощью которой мы сможем настроить прозрачность окна, рисунок 2.4 [4]. Для того чтобы эта функция работала корректно и не перекрывала другие программы, необходимо при создании окна указать флаги, которые указана на рисунке 2.4.

```
areaHWND = CreateWindowEx(WS_EX_TOOLWINDOW | WS_EX_TRANSPARENT | WS_EX_LAYERED | WS_EX_TOPMOST, className, NULL,
    WS_POPUP | WS_VISIBLE | WS_CLIPSIBLINGS | WS_CLIPCHILDREN | WS_MAXIMIZE | WS_MAXIMIZEBOX & ~WS_CAPTION, 0, 0,
     resolutionWH.right, resolutionWH.bottom, NULL, NULL, hInstance, NULL);
SetLayeredWindowAttributes(areaHWND, NULL, WORK_AREA_TRANSPARENCY_DISABLED, LWA_ALPHA);
```

Рисунок 2.4 – Функция создание окна и настройки прозрачности

После чего будет создано прозрачное окно поверх всех и не перекрывающее нажатия мыши и клавиатуры. Следующим шагом необходимо его свернуть, на случай если оно зависнет, т.к. окно находиться поверх всех и его нельзя закрыть.

2.4 Обработка нажатий

Чтобы не перекрывать другие приложения и не мешать их работе, необходимо реализовать большую часть функций по нажатию клавиш. Чтобы это сделать необходимо, воспользоваться «Хуками», т.к. нажатия происходят вне нашего окна. Будем использовать функцию «SetWindowsHookEx» [7]. Эта функция устанавливает определяемую программой подключаемую процедуру в цепочку хук-точек. Прикладная программа устанавливает фильтр – процедуру, чтобы контролировать некоторые типы событий в системе. Подключаемая процедура может контролировать события связанные или с конкретным потоком, или со всеми потоками в системе. Таким образом, мы будем перехватывать нажатия на клавиатуру.

Далее создадим функцию «HookCallback», которая будет обрабатывать наши нажатия, рисунок 2.5.

```
LRESULT __stdcall HookCallback(int nCode, WPARAM wParam, LPARAM lParam)
{
    if (nCode >= 0)
    {
        if (wParam == WM_KEYDOWN)
        {
            kbdStruct = *((KBDLLHOOKSTRUCT*)lParam);
            if (kbdStruct.vkCode == 67 && GetAsyncKeyState(VK_SHIFT) && GetAsyncKeyState(VK_LWIN))···
            else if (kbdStruct.vkCode == VK_ESCAPE && GetAsyncKeyState(VK_LWIN))···
            else if (kbdStruct.vkCode ==  VK_F2)···
            else if (kbdStruct.vkCode == 86 && GetAsyncKeyState(VK_SHIFT) && GetAsyncKeyState(VK_LWIN))···
            else if (kbdStruct.vkCode == 79 && GetAsyncKeyState(VK_SHIFT) && GetAsyncKeyState(VK_LWIN))···
        }
    }

    return CallNextHookEx(_hook, nCode, wParam, lParam);
}
```

Рисунок 2.5 – Функция обработки нажатий

Теперь вне зависимости от того в какой программе мы находимся, мы можем пользоваться нашей программой.

Нажав комбинацию клавиш «Shift + Win + C» у нас появляется полотно для выделения области записи, после того как зона будет выделена пользователь может начать съемки нажав клавишу на клавиатуре «F2». По окончанию записи он может нажать ту же кнопку записи для сохранения выделенной области, или комбинацию клавиш «Win + Esc», для того чтобы прекратить запись и свернуть программу. По нажатию комбинации клавиш «Shift + Win + O», можно открыть или закрыть меню настроек программы.

Чтобы наша программа не зависла по окончанию записи, необходимо убрать наш хук, т.к. программа будет нагружена обработкой записи, после чего хук возвращается.

2.5 Запись видео

Запись осуществляется путем многократного снимка экрана. Снимок экрана делается с использованием библиотеки «Magick++», алгоритм представлен на рисунке 2.6. Снимки должны делаться с определенной задержкой, которую мы устанавливаем перед началом записи с помощью таймера, по окончанию записи таймер удаляется. Новые видеофайлы сохраняются по времени в папку «GIFs» или в папку с программой.

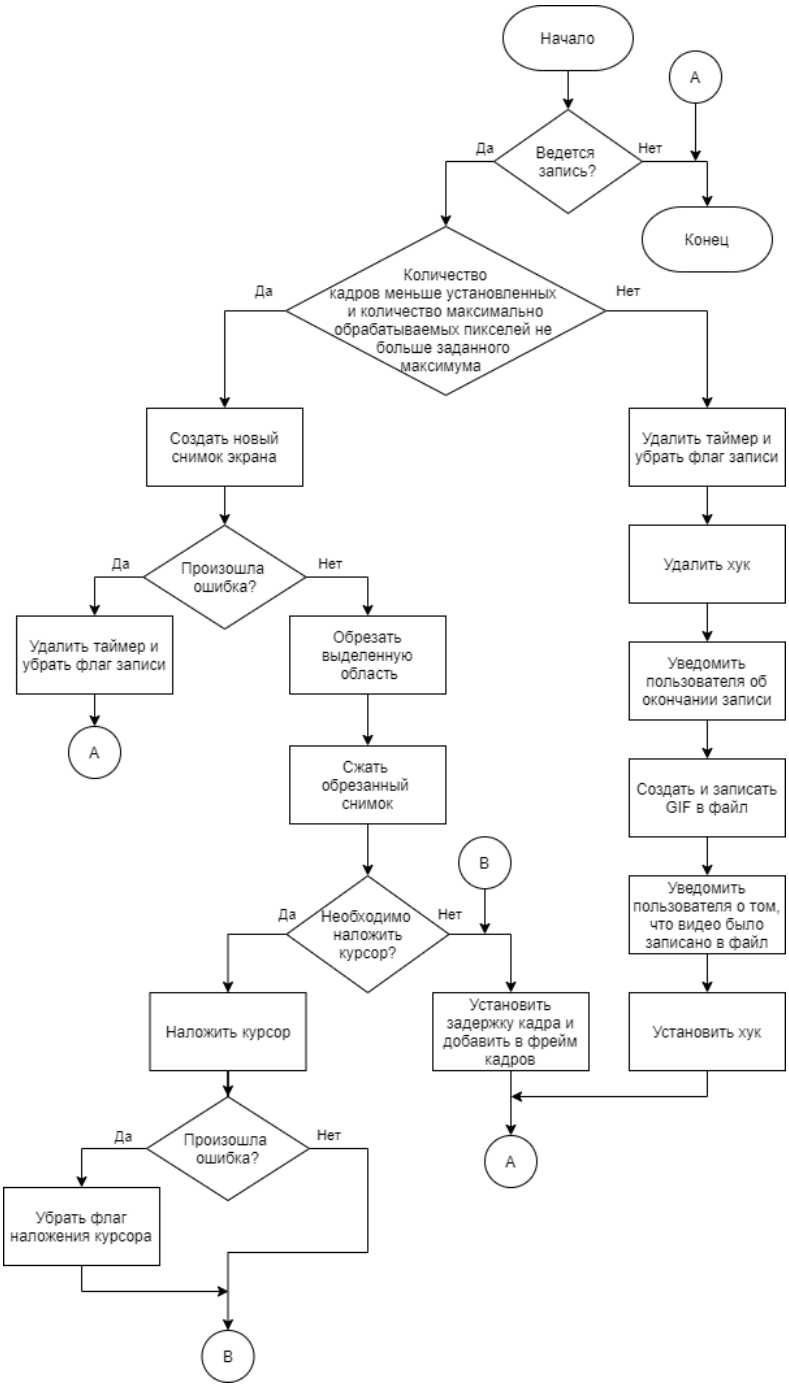

Рисунок 2.6 – Схема алгоритма записи

2.6 Оптимизация размера GIF-файла

Оптимизация размера основана на наложении меньшего фрагмента изображения, а не на полном наложении всего изображения. Это дает меньшее количество пикселей и, следовательно, меньший размер файла на диске. Кроме того, наложение меньшего кадра означает, что клиентскому компьютеру не нужно выполнять столько работы по изменению пикселей на экране.

Однако существуют различные методы удаления, доступные в формате GIF для обработки последнего отображаемого кадра, что может привести к наложению разного размера. Также можно разделить наложения на несколько частей или обновить действия, чтобы получить более сложную, но более оптимизированную анимацию, рисунок 2.7.

Простой оптимизацией по сравнению с наложением, это удаление повторяющихся кадров, рисунок 2.8 [5].

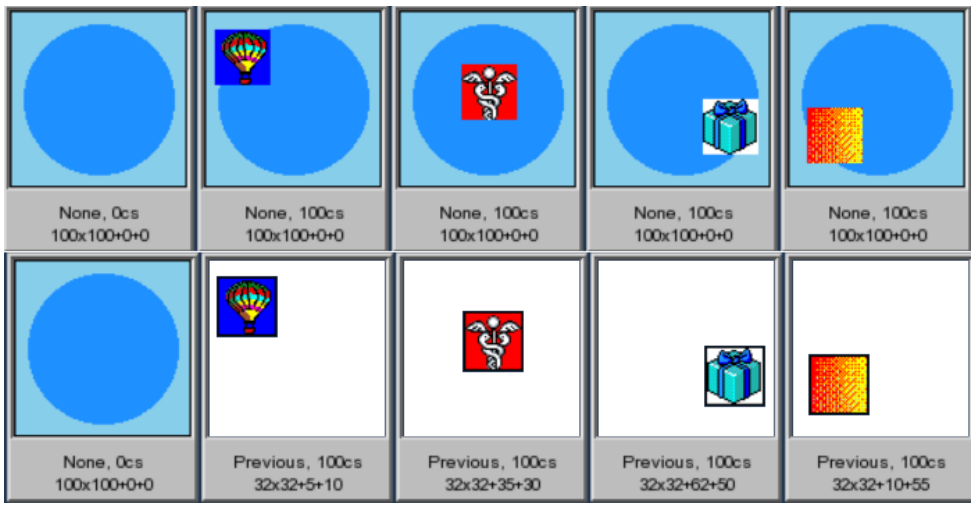

Рисунок 2.7 – Наложение кадров до и после оптимизации

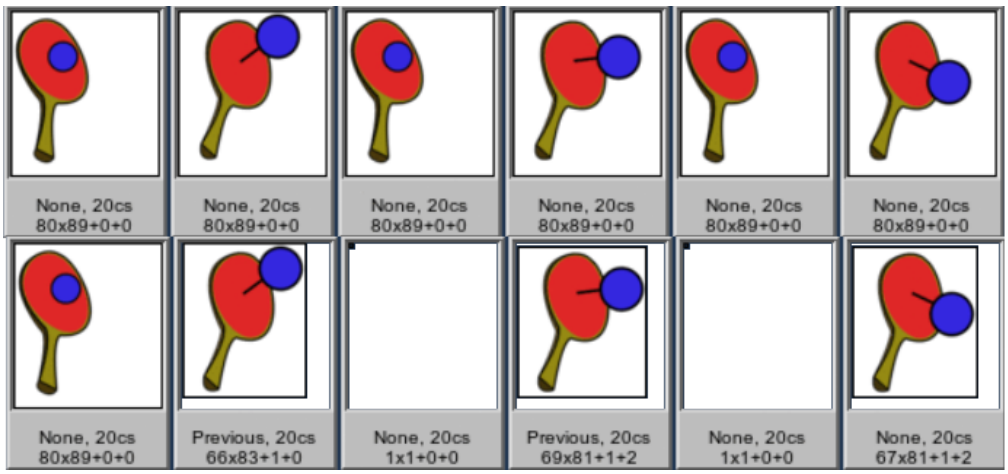

Рисунок 2.8 – Удаление повторяющихся кадров

Таким образом по окончанию записи кадров, конечный GIF-файл оптимизируется по данным алгоритмам.

2.7 Сжатие кадров алгоритмом LZW

Идея, лежащая в основе словарных алгоритмов, заключается в том, что происходит кодирование цепочек элементов исходной последовательности. При этом кодировании используется специальный словарь, который получается на основе исходной последовательности.

Существует целое семейство словарных алгоритмов, но мы рассмотрим наиболее распространённый алгоритм LZW, названный в честь его разработчиков Лепеля, Зива и Уэлча.

Словарь в этом алгоритме представляет собой таблицу, которая заполняется цепочками кодирования по мере работы алгоритма. При декодировании сжатого кода словарь восстанавливается автоматически, поэтому нет необходимости передавать словарь вместе с сжатым кодом.

Словарь инициализируется всеми одноэлементными цепочками, т.е. первые строки словаря представляют собой алфавит, в котором мы производим кодирование. При сжатии происходит поиск наиболее длинной цепочки уже записанной в словарь. Каждый раз, когда встречается цепочка, ещё не записанная в словарь, она добавляется туда, при этом выводится сжатый код, соответствующий уже записанной в словаре цепочки. В теории на размер словаря не накладывается никаких ограничений, но на практике есть смысл этот размер ограничивать, так как со временем начинаются встречаться цепочки, которые больше в тексте не встречаются. Кроме того, при увеличении размеры таблицы вдвое мы должны выделять лишний бит для хранения сжатых кодов. Для того чтобы не допускать таких ситуаций, вводится специальный код, символизирующий инициализацию таблицы всеми одноэлементными цепочками [6].

Рассмотрим пример сжатия алгоритмом. Будем сжимать строку «кукушкакукушонкукупилакапюшон». Предположим, что словарь будет вмещать 32 позиции, а значит, каждый его код будет занимать 5 бит. Изначально словарь заполнен следующим образом, рисунок 2.9.

Эта таблица есть, как и на стороне того, кто сжимает информацию, так и на стороне того, кто распаковывает. Сейчас мы рассмотрим процесс сжатия, рисунок 2.10.

В таблице представлен процесс заполнения словаря. Легко подсчитать, что полученный сжатый код занимает 105 бит, а исходный текст (при условии, что на кодирование одного символа мы тратим 4 бита) занимает 116 бит.

По сути, процесс декодирования сводится к прямой расшифровке кодов, при этом важно, чтобы таблица была инициализирована также, как и при кодировании. Теперь рассмотрим алгоритм декодирования, рисунок 2.11.

Символ	Код
К	00001
У	00010
Ш	00011
А	00100
О	00101
Н	00110
П	00111
И	01000
Л	01001
Ю	01010

Рисунок 2.9 – Исходный словарь для алгоритма LZW

Цепочка из исходной строки	Строка, добавляемая в словарь и её код		Сжатый код
К	КУ	01011	00001
У	УК	01100	00010
КУ	КУШ	01101	01011
Ш	ШК	01110	00011
К	КА	01111	00001
А	АК	10000	00100
КУ	КУК	10001	01011
КУШ	КУШО	10010	01101
О	ОН	10011	00101
Н	НК	10100	00110
КУК	КУКУ	10101	10001
У	УП	10110	00010
П	ПИ	10111	00111
И	ИЛ	11000	01000
Л	ЛА	11001	01001
АК	АКА	11010	10000
А	АП	11011	00100
П	ПЮ	11100	00111
Ю	ЮШ	11101	01010
Ш	ШО	11110	00011
ОН			10011

Рисунок 2.10 – Процесс сжатия

Данные	Символы на выходе	Строка, добавляемая в словарь и её код	
00001	К	К?	01011
00010	У	У?	01100
01011	КУ	КУ?	01101
…			

Рисунок 2.11 – Алгоритм декодирования

Строку, добавленную в словарь на i-ом шаге мы можем полностью определить только на i+1. Очевидно, что i-ая строка должна заканчиваться на первый символ i+1 строки. Т.о. мы только что разобрались, как можно

восстанавливать словарь. Некоторый интерес представляет ситуация, когда кодируется последовательность вида «cScSc», где c — это один символ, а S — строка, причём слово cS уже есть в словаре. На первый взгляд может показаться, что декодер не сможет разрешить такую ситуацию, но на самом деле все строки такого типа всегда должны заканчиваться на тот же символ, на который они начинаются.

2.8 Меню настроек программы

Интерфейс окна меню настроек должен быть максимально простым и интуитивно-понятным, рисунок 2.2. Также меню настроек должно легко вызываться, например, комбинацией клавиш «Shift + Win + O», и той же комбинацией закрываться, или закрываться как обычное окно.

В меню настроек можно настроить пять основных параметров записи, а именно: максимальное количество кадров, задержку между кадрами, степень сжатия разрешения кадров, а также отображение курсора и путь к нему.

Основная часть интерфейса – это простая картинка, поверх которой находятся 5 WinApi элементов. WinApi элементы имеют фиксированные позиции и фон под цвет меню.

После того как вы введете новые данные в меню настроек, они сражу поменяются в программе. При вводе некорректных данных программа не перестает работать.

Все настройки, при выключении программы, сохраняются в файл «options.dat» и при повторном запуске считываются. При первом запуске устанавливаются настройки по умолчанию.

2.9 Автозапуск программы

Автозапуск программы был реализован с помощью простого bat-файла, код представлен на рисунке 2.12.

```
cd C:\Video-recording
start Video-recording.exe
```

Рисунок 2.12 – Код файла для автозапуска

То есть, чтобы наша программа автоматически запускалась при включении компьютера, достаточно расположить файл с программой на диске «C».

3 ТЕСТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

В ходе тестирования приложения не было выявлено недостатков программного средства. Была составлена таблица 3.1, показывающая ожидаемые и реальные результаты, полученные при заданных условиях, она представлена ниже.

Таблица 3.1 – Ожидаемые и реальные результаты тестирования

№	Тестовые случаи	Ожидаемый результат	Полученный результат
1.	Запуск программы	Успешный запуск и инициализация начальных значений параметров программы	Тест пройден
2.	Открытие меню настроек	Успешное отображение меню настроек	Тест пройден
3.	Изменение параметров программы	Изменение параметров программы	Тест пройден
4.	Выключение программы	Успешное выключение программы без ошибок с сохранением параметров	Тест пройден
5.	Выделение области для записи	Выделенная область для записи	Тест пройден
6.	Остановка записи	Новое видео	Тест пройден
7.	Остановка записи с сохранением выделенной области	Новое видео с сохраненной выделенной областью	Тест пройден
8.	Повторная запись с выделенной областью	Новое видео	Тест пройден
9.	Изменение задержки кадров	Изменение задержки кадров	Тест пройден
10.	Указать на не существующую папку с курсором	Сообщение о том, что такого файла не существует	Тест пройден

Продолжение таблицы 3.1.

11	Запись без остановки	Запись максимального количества кадров	Тест пройден
12	Выключение наложения курсора	Запись видео без курсора	Тест пройден
13	Смена курсора	Смена курсора	Тест пройден
14	Захват и съемка экрана в приложениях, которые открыты в полноэкранном режиме	Видео выделенной области в формате GIF	Тест пройден
15	Автозапуск программы при загрузке операционной системы	Автоматически запущенная программа	Тест пройден
16	Закрытие программы комбинацией клавиш	Закрытая программа	Тест пройден
17	Изменение максимального количества кадров	Изменение максимального количества кадров	Тест пройден

Разработка приложения велась с использованием системы контроля версий GitHub, позволившая сохранять состояние программы на каждом отдельном этапе по ходу добавления нового функционала или изменения уже существующего. Появление новых точек возврата происходит посредством группировки изменённых файлов, затем они объединяются под общим именем «коммита», в котором кратко изложена суть изменений. Также можно добавлять к каждому этапу новые файлы, или удалять устаревшие варианты. После накопления определённого количества групп изменений, их следует отправить на удалённый репозиторий, где видна вся история приложения и разница между каждым новым «коммитом».

Путем тщательной проверки тестами, было выявлено, что ошибок в работе программы нет.

4 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

4.1 Руководство по установке и запуску

Для установки программы необходимо скачать ImageMagick (ImageMagick-7.0.10-37-Q16-HDRI-x64-dll.exe). Затем установите программу по пути «C:\Program Files». При установке нужно выбрать все параметры установки программы, как показано на рисунке 4.1.

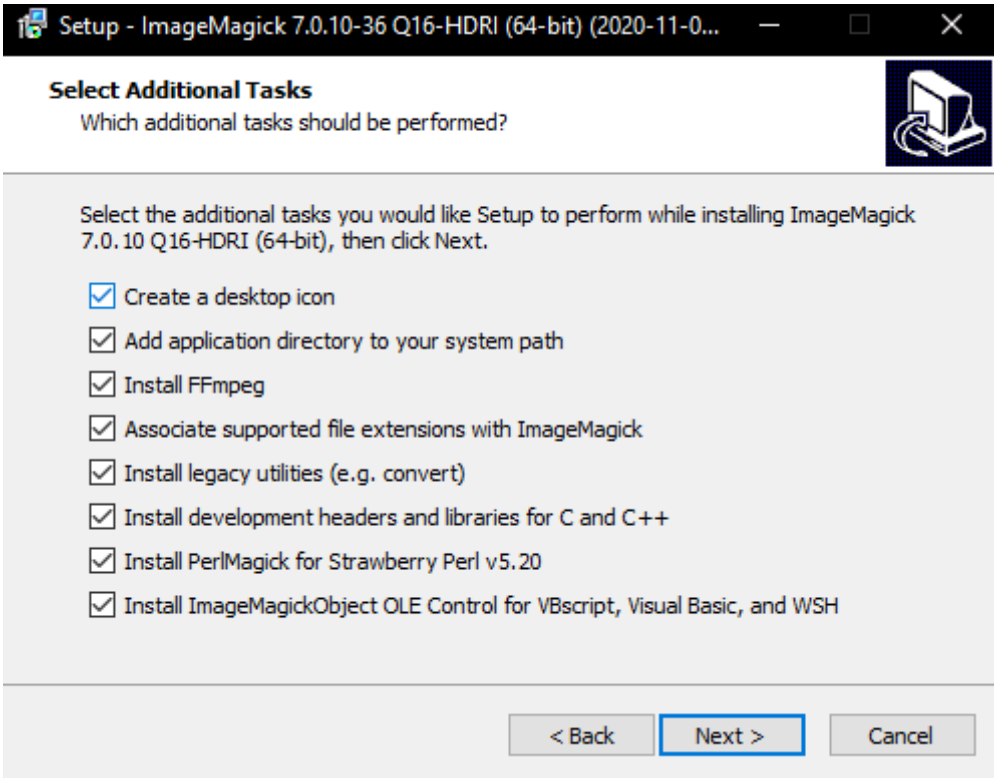

Рисунок 4.1 – Параметры установки ImageMagick

После установки ImageMagick вы можете использовать его как редактор для GIF. Для того чтобы запустить программу достаточно перейти в папку с программой и открыть файл по адресу «Video-recording\Video-recording.exe».

4.2 Установка автозапуска программы

Если вы хотите установить автоматический запуск, то переместите папку с программой на диск C. Затем, нажмите комбинацию клавиш «Win + R», после чего введите «shell:startup» в появившееся окно, затем нажмите «Enter». Затем переместите туда файл с названием «Video-recording» («Video-recording.bat»), который находится в корне папки с программой. После данных действий программа будет запускаться автоматически с включением компьютера.

Установка автозапуска по этапам в картинках:

1. Переместите папку с названием «Video-recording», которая находиться в папке с программой на диск «C», рисунок 4.2.

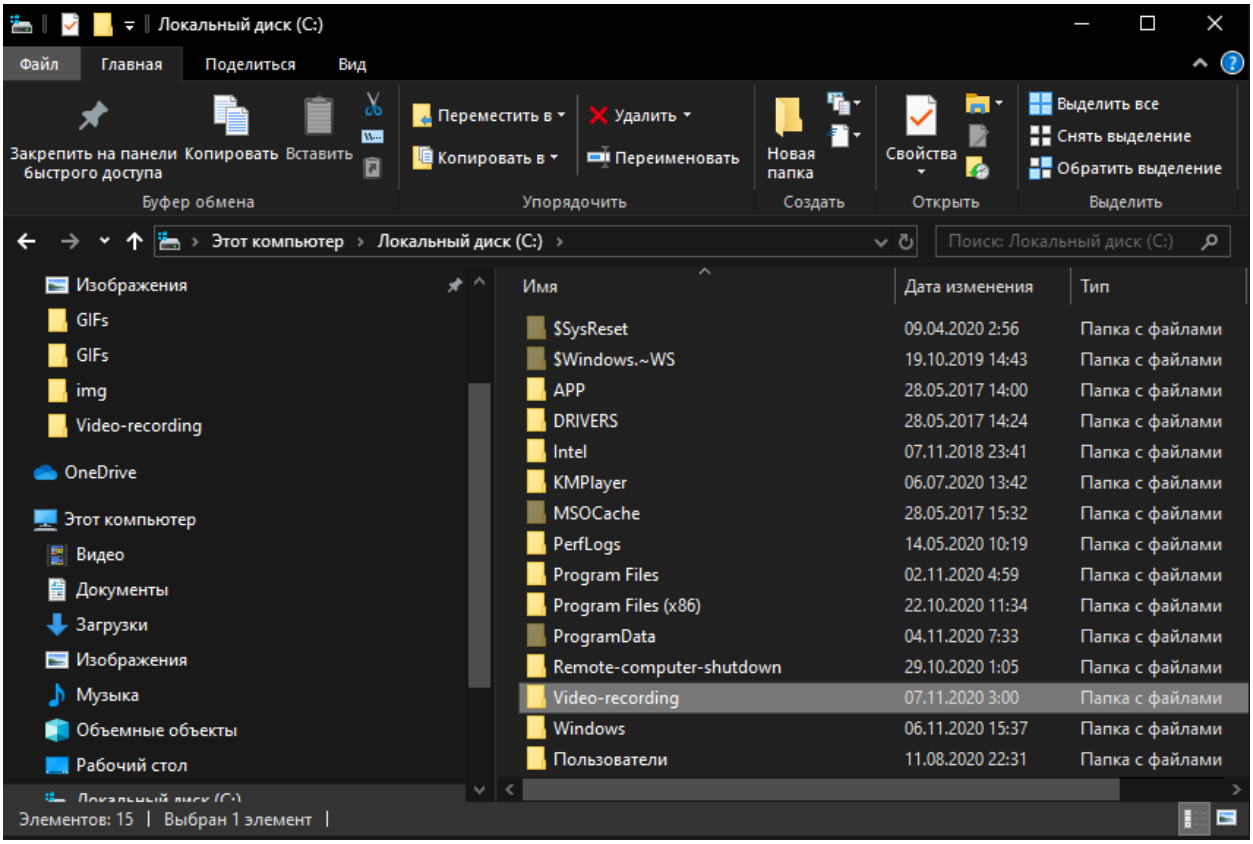

Рисунок 4.2 – Перемещение программы на диск «C»

2. Нажмите комбинацию клавиш «Win + R», после чего введите «shell:startup» в появившееся окно, затем нажмите «Enter», рисунок 4.3.

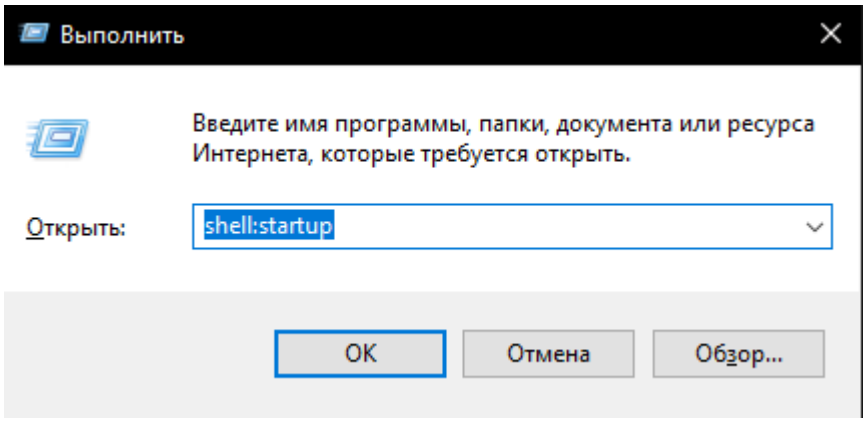

Рисунок 4.3 – Ввод команды в окно «Выполнить»

3. Затем переместите туда файл с названием «Video-recording» («Video-recording.bat»), рисунок 4.4.

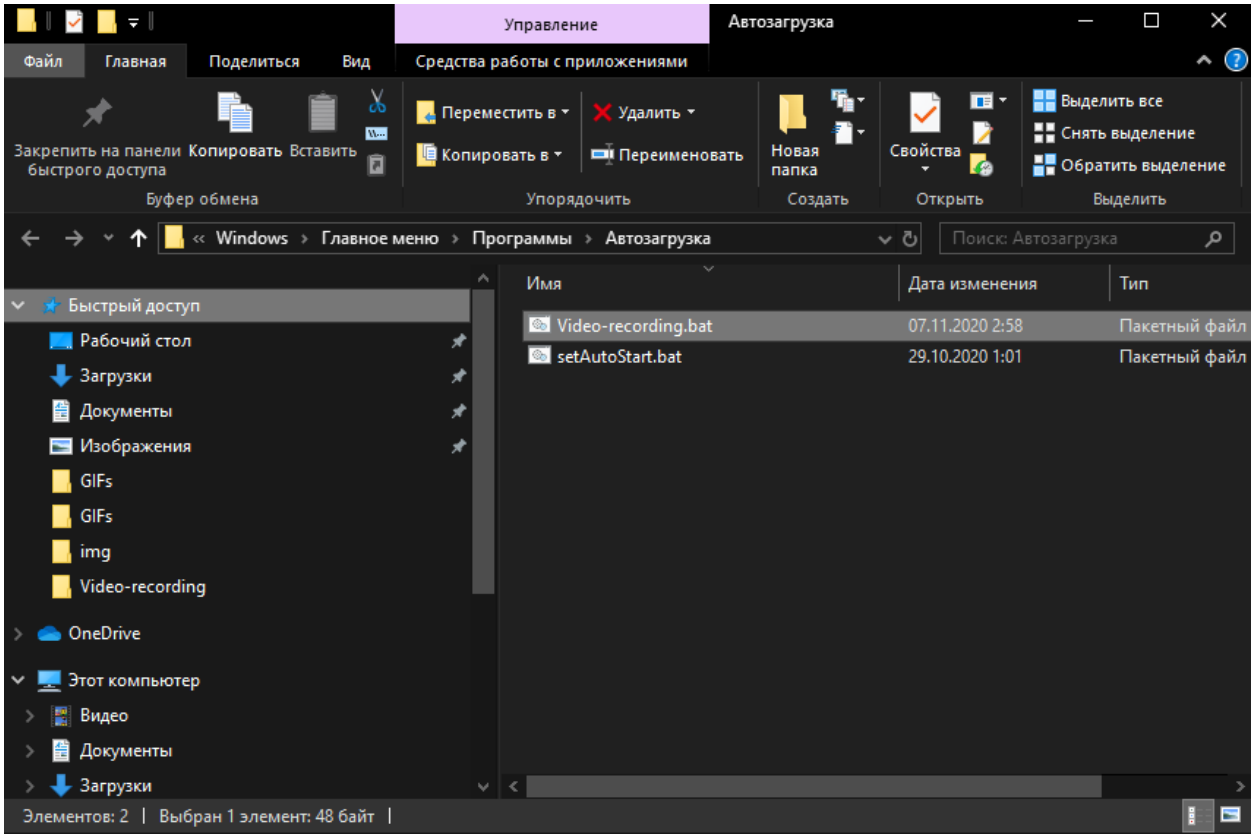

Рисунок 4.4 – Перемещений файла автозагрузки в каталог файлов автозагрузки

4.3 Руководство по использованию

Интерфейс программного средства является интуитивно-понятным и простым.

После установки и запуска программы, чтобы начать записывать определенную область или весь экран, нужно нажать комбинацию клавиш «Shift + Win + C», после чего необходимо выделить с помощью курсора мыши область для записи, рисунок 2.3. После того как вы выделили определенную область экрана достаточно нажать на клавишу «F2», после чего запись начнется.

Запись длиться до того момента покуда вы не нажмете клавишу «F2» или комбинацию клавиш «Win + Esc», или покуда не будет достигнут максимум по количеству кадров, или по максимальному количеству пикселей.

Нажав клавишу «F2», вы сохраняете выделенную область для повторной записи, нажав комбинацию клавиш «Win + Esc» (рекомендуется), вы скрываете программу и прекращаете запись.

Чтобы открыть меню настроек необходимо нажать комбинацию клавиш «Shift + Win + O», после чего появится окно настроек, рисунок 2.2.

В окне настроек вы можете настроить: максимальное количество кадров в видео, задержку между кадрами, степень сжатия разрешения кадров, видимость курсора мыши и изменить его на свой, указав новый путь.

При настройке программы не надо нажимать ни на какие кнопки, вы просто вводите новые параметры, и они сразу применяются. Даже во время записи программы вы можете изменить параметры записи.

По окончанию записи, на экране появляется уведомление о том, что запись была окончена, рисунок 4.5.

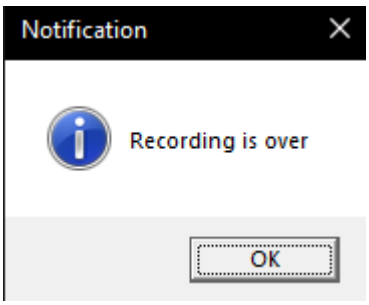

Рисунок 4.5 – Уведомление о завершении записи

Через некоторое время, а именно когда уже будет готов новый GIF-файл, появиться еще одно уведомление, рисунок 4.6.

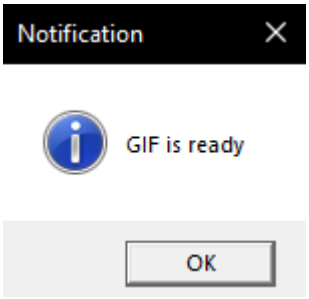

Рисунок 4.6 – Уведомление о завершении обработки файла

Также могут возникнуть и другие уведомления, например, уведомления об ошибках. Так в случае если вы не установите ImageMagick, у вас может запуститься программа, но запись вестись не будет. Или если у вас не будет папки «GIFs» для хранения GIF-файлов и т.д.

Во время обработки программа не отвечает на действия пользователя, поэтому рекомендуется скрывать ее по завершению записи.

Чтобы закрыть программу необходимо нажать комбинацию клавиш «Shift + Win + V».

ЗАКЛЮЧЕНИЕ

Довольно часто бывают моменты, когда проще один раз показать, чем несколько раз написать или рассказать. Но это лишь проще для понимания того, кто слушает, в то время как объясняющему приходиться искать или готовые видео, или самому пытаться найти программу для записи. В таком случае данная программа является простым решением, т.к. с ней в комплекте идет программа для редактирования, с помощью которой можно пометить важные моменты и т.п.

Преимуществом этой программы является то, что она бесплатная и крайне проста в управлении. Стоит ее один раз установить на своем компьютер и вспомнить в момент, когда она будет нужна. Для того, чтобы записать новый GIF-файл, необходимо нажать всего пару клавиш.

В рамках данного курсового проекта было разработано программное средство «Видеозахват экрана», которое обеспечит возможность записи видео в GIF-файл в любое время. Работа была выполнена в полном объеме, были реализованы все поставленные функции.

Существует много возможностей для дальнейшего улучшения приложения. Одним из самых простых направлений является создание программы для установки приложения. Также можно добавить новые форматы записи и расширить меню.

Использование данного приложения позволит в любой момент времени записать видео простой комбинацией клавиш, после чего вы можете отредактировать полученное видео в формате GIF и показать.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] recordit.co [Электронный портал]. – Электронные данные. – Режим доступа: https://recordit.co/

[2] www.bandicam.com [Электронный портал]. – Электронные данные. – Режим доступа: https://www.bandicam.com/ru/

[3] fraps.com [Электронный портал]. – Электронные данные. – Режим доступа: https://fraps.com/

[4] www.vsokovikov.narod.ru [Электронный портал]. – Электронные данные. – Режим доступа: http://www.vsokovikov.narod.ru/New_MSDN_API/Window/fn_setlayeredwindowattribut.htm

[5] www.imagemagick.org [Электронный портал]. – Электронные данные. – Режим доступа: https://www.imagemagick.org/Usage/anim_opt/

[6] neerc.ifmo.ru [Электронный портал]. – Электронные данные. – Режим доступа: https://neerc.ifmo.ru/wiki/index.php?title=%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_LZW

[7] docs.microsoft.com [Электронный портал]. – Электронные данные. – Режим доступа: https://docs.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-setwindowshookexa

ПРИЛОЖЕНИЕ 1
Исходный код программы

Файл Video-recording.cpp:

```
#define _CRT_SECURE_NO_WARNINGS
#define TIMER_ID 1
#define WORK_AREA_TRANSPARENCY_ACTIVE 20
#define WORK_AREA_TRANSPARENCY_DISABLED 0
#define MAIN_DISABLED 0
#define MAIN_ACTIVE 255
#define MAX_NUMBER_OF_PIXELS 250000000

#include "framework.h"
#include <Windowsx.h>
#include "Video-recording.h"
#include "windowFunctions.cpp"
#include "options.cpp"
#include <Magick++.h>
#include <ctime>
#include <fstream>

HWND areaHWND;
RECT resolutionWH;
RECT rcSize;
HDC hdcBackBuffer, hdcArea;
PAINTSTRUCT ps;

std::vector<Magick::Image> frames;

int maxFrames = 1000;
LONG delay = 250;
double resolution = 1;
bool flagCursorShow = true;
std::string pathToCursor = "cursors/cur0.png";
std::string prevPathToCursor;
bool flagRecording = false;
bool flagMouseDown = false;
bool areaIsReady = false;
bool flagMainHWND = false;
bool flagInit = false;
bool flagEscKey = false;

POINT startPoint;
POINT endPoint;
Magick::Geometry selectedArea;
HDC secondHdc;
LONG width, height, offSetX, offSetY;
LONG numberOfPixelsPerFrame;
POINT cursorPos;
```

```
Magick::Image cursorIco;

HWND optionsHWND;
HINSTANCE hInstanceGlobal;
HBITMAP hBitmap;
HWND hwndMaxFrames;
HWND hwndFramesDelay;
HWND hwndResolutionCompression;
HWND hwndFlagCursor;
HWND hwndPathToCursor;

std::string buffStr;
std::wstring buffWStr;
wchar_t buffW[1024];

HHOOK _hook;
KBDLLHOOKSTRUCT kbdStruct;

LRESULT __stdcall HookCallback(int nCode, WPARAM wParam, LPARAM lParam)
{
   if (nCode >= 0)
   {
      if (wParam == WM_KEYDOWN)
      {
         kbdStruct = *((KBDLLHOOKSTRUCT*)lParam);
         if (kbdStruct.vkCode == 67 && GetAsyncKeyState(VK_SHIFT) &&
GetAsyncKeyState(VK_LWIN))
         {
            endPoint.x = 0;
            endPoint.y = 0;
            startPoint.x = 0;
            startPoint.y = 0;
            ResizeWnd(areaHWND);
            flagRecording = false;
            SetLayeredWindowAttributes(areaHWND, NULL,
WORK_AREA_TRANSPARENCY_ACTIVE, LWA_ALPHA);
            SetWindowLong(areaHWND, GWL_EXSTYLE, WS_EX_TOOLWINDOW |
WS_EX_LAYERED | WS_EX_TOPMOST);
            SetWindowPos(areaHWND, NULL, 0, 0, resolutionWH.right, resolutionWH.bottom,
SWP_SHOWWINDOW | SWP_NOZORDER | SWP_NOMOVE);
         }
         else if (kbdStruct.vkCode == VK_ESCAPE && GetAsyncKeyState(VK_LWIN))
         {
            flagRecording = false;
            flagMouseDown = false;
            flagEscKey = true;
            HideAreaHWND();
         }
         else if (kbdStruct.vkCode ==  VK_F2)
         {
            if (!flagMouseDown && areaIsReady)
```

```
            {
               flagRecording = !flagRecording;
               if (flagRecording) SetTimer(areaHWND, TIMER_ID, delay, NULL);
            }
         }
         else if (kbdStruct.vkCode == 86 && GetAsyncKeyState(VK_SHIFT) &&
GetAsyncKeyState(VK_LWIN))
         {
            DestroyWindow(areaHWND);
         }
         else if (kbdStruct.vkCode == 79 && GetAsyncKeyState(VK_SHIFT) &&
GetAsyncKeyState(VK_LWIN))
         {
            if (!flagMainHWND)
            {
               SetLayeredWindowAttributes(optionsHWND, NULL, MAIN_ACTIVE,
LWA_ALPHA);
               SetWindowPos(optionsHWND, NULL, 0, 0, 413, 673, SWP_SHOWWINDOW |
SWP_NOZORDER | SWP_NOMOVE);
               SetWindowLong(optionsHWND, GWL_EXSTYLE, WS_EX_TOOLWINDOW |
WS_EX_LAYERED | WS_EX_TOPMOST);
            }
            else
            {
               SetLayeredWindowAttributes(optionsHWND, NULL, MAIN_DISABLED,
LWA_ALPHA);
               SetWindowPos(optionsHWND, NULL, 0, 0, 413, 673, SWP_HIDEWINDOW);
               SetWindowLong(optionsHWND, GWL_EXSTYLE, WS_EX_TOOLWINDOW |
WS_EX_TRANSPARENT | WS_EX_LAYERED | WS_EX_TOPMOST);
            }
            flagMainHWND = !flagMainHWND;
         }
      }
   }

   return CallNextHookEx(_hook, nCode, wParam, lParam);
}

void SetHook()
{
   _hook = SetWindowsHookEx(WH_KEYBOARD_LL, HookCallback, NULL, 0);
}

void UnHook()
{
   UnhookWindowsHookEx(_hook);
}

LRESULT CALLBACK AreaWndProc(HWND hWnd, UINT msg, WPARAM wParam,
LPARAM lParam)
{
```

```
   switch (msg)
   {
   case WM_TIMER:
   {
      if (frames.size() > 0 || flagRecording)
      {
         using namespace Magick;
         if (frames.size() < maxFrames && frames.size() * numberOfPixelsPerFrame / (resolution
* resolution) < MAX_NUMBER_OF_PIXELS && flagRecording)
         {
            Image img;
            try
            {
               img = Image("screenshot:");
            }
            catch (...)
            {
               flagRecording = false;
               KillTimer(areaHWND, TIMER_ID);
               MessageBox(NULL, L"Can't start recording", L"ERROR", MB_ICONERROR);
               return 0;
            }
            img.crop(selectedArea);
            img.repage();
            img.compressType(MagickCore::LZWCompression);
            if (flagCursorShow)
            {
               GetCursorPos(&cursorPos);
               img.composite(cursorIco, (cursorPos.x - offSetX), (cursorPos.y - offSetY),
MagickCore::PlusCompositeOp);
            }
            if (resolution > 1)
            {
               img.resize(Geometry(std::to_string(((double) width) / resolution)));
            }
            img.animationDelay(delay / 10);
            frames.push_back(img);
         }
         else
         {
            KillTimer(areaHWND, TIMER_ID);
            MessageBox(NULL, L"Recording is over", L"Notification",
MB_ICONINFORMATION);
            UnHook();
            areaIsReady = false;

            std::time_t time = std::time(0);
            std::tm* now = std::localtime(&time);
            std::string pathToFile = "GIFs/" + std::to_string(now->tm_mday) + "." +
std::to_string(now->tm_mon) +
```

```
                "." + std::to_string(now->tm_year + 1900) + " (" + std::to_string(now->tm_hour) +
"-" +
                std::to_string(now->tm_min) + "-" + std::to_string(now->tm_sec) + ").gif";

            try
            {
                writeImages(frames.begin(), frames.end(), pathToFile);
            }
            catch (...)
            {
                MessageBox(NULL, L"Make a GIFs folder to store GIF files", L"ERROR",
MB_ICONERROR);
                pathToFile = "" + std::to_string(now->tm_mday) + "." + std::to_string(now-
>tm_mon) +
                    "." + std::to_string(now->tm_year + 1900) + " (" + std::to_string(now->tm_hour)
+ "-" +
                    std::to_string(now->tm_min) + "-" + std::to_string(now->tm_sec) + ").gif";
                writeImages(frames.begin(), frames.end(), pathToFile);
            }

            frames.clear();
            if (!flagRecording && !flagEscKey) areaIsReady = true;
            flagRecording = false;
            SetHook();
            flagEscKey = false;
            MessageBox(NULL, L"GIF is ready", L"Notification", MB_ICONINFORMATION);
        }
    }
    return 0;
}
case WM_CREATE:
{
    ReadOptionsFile();
    ResizeWnd(hWnd);
    HideAreaHWND();
    return 0;
}
case WM_MOUSEMOVE:
{
    endPoint.x = GET_X_LPARAM(lParam);
    endPoint.y = GET_Y_LPARAM(lParam);
    if (flagMouseDown) ResizeWnd(hWnd);
    return 0;
}
case WM_LBUTTONDOWN:
{
    flagMouseDown = true;
    startPoint.x = GET_X_LPARAM(lParam);
    startPoint.y = GET_Y_LPARAM(lParam);
    return 0;
}
```

```
    case WM_LBUTTONUP:
    {
        flagMouseDown = false;
        endPoint.x = GET_X_LPARAM(lParam);
        endPoint.y = GET_Y_LPARAM(lParam);
        selectedArea = Magick::Geometry(width, height, offSetX, offSetY);
        ResizeWnd(hWnd);
        SetWindowLong(hWnd, GWL_EXSTYLE, WS_EX_TOOLWINDOW |
WS_EX_TRANSPARENT | WS_EX_LAYERED | WS_EX_TOPMOST);
        areaIsReady = true;
        numberOfPixelsPerFrame = width * height;
        return 0;
    }
    case WM_SIZE:
    {
        return 0;
    }
    case WM_CLOSE:
    {
        return 0;
    }
    case WM_DESTROY:
    {
        UnHook();
        WriteOptionsFile();
        PostQuitMessage(0);
    }
    case WM_PAINT:
    {
        BeginPaint(hWnd, &ps);
        FillRect(hdcBackBuffer, &rcSize, (HBRUSH)GetStockObject(WHITE_BRUSH));
        BitBlt(hdcBackBuffer, offSetX, offSetY, rcSize.right - rcSize.left, rcSize.bottom -
rcSize.top, hdcArea, 0, 0, SRCCOPY);
        BitBlt(ps.hdc, 0, 0, rcSize.right - rcSize.left, rcSize.bottom - rcSize.top, hdcBackBuffer, 0,
0, SRCCOPY);
        EndPaint(hWnd, &ps);
        return 0;
    }
    }

    return DefWindowProc(hWnd, msg, wParam, lParam);
}

int WINAPI WinMain(HINSTANCE hPrevInstance, HINSTANCE hInstance, LPSTR
lpCmdLine, int nShowCmd)
{
    static TCHAR className[] = TEXT("RecorderAreaClass");
    static TCHAR windowName[] = TEXT("AreaRecorder");

    WNDCLASSEX wcex;
```

```
    wcex.cbClsExtra = 0;
    wcex.cbSize = sizeof(WNDCLASSEX);
    wcex.cbWndExtra = 0;
    wcex.hbrBackground = NULL;
    wcex.hCursor = LoadCursor(hInstance, IDC_CROSS);
    wcex.hIcon = LoadIcon(hInstance, IDI_APPLICATION);
    wcex.hIconSm = NULL;
    wcex.hInstance = hInstance;
    wcex.lpfnWndProc = AreaWndProc;
    wcex.lpszClassName = className;
    wcex.lpszMenuName = NULL;
    wcex.style = 0;

    if (!RegisterClassEx(&wcex))
        return 0;

    hInstanceGlobal = hInstance;
    CreateMainHWND();
    GetWindowRect(GetDesktopWindow(), &resolutionWH);

    areaHWND = CreateWindowEx(WS_EX_TOOLWINDOW | WS_EX_TRANSPARENT |
WS_EX_LAYERED | WS_EX_TOPMOST, className, NULL,
        WS_POPUP | WS_VISIBLE | WS_CLIPSIBLINGS | WS_CLIPCHILDREN |
WS_MAXIMIZE | WS_MAXIMIZEBOX & ~WS_CAPTION, 0, 0,
        resolutionWH.right, resolutionWH.bottom, NULL, NULL, hInstance, NULL);
    SetLayeredWindowAttributes(areaHWND, NULL,
WORK_AREA_TRANSPARENCY_DISABLED, LWA_ALPHA);

    if (!areaHWND) return 0;

    ShowWindow(areaHWND, nShowCmd);
    UpdateWindow(areaHWND);
    SetHook();

    MSG msg;
    while (GetMessage(&msg, NULL, 0, 0))
    {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }

    return msg.wParam;
}
```

Файл options.cpp:

```
#include "options.h"

inline LRESULT CALLBACK OptionsWndProc(HWND hWnd, UINT msg, WPARAM
wParam, LPARAM lParam)
{
    switch (msg)
```

```
    {
    case WM_CREATE:
    {
        hBitmap = (HBITMAP)LoadImage(hInstanceGlobal, L"img/background.bmp",
IMAGE_BITMAP, 0, 0, LR_LOADFROMFILE);
        break;
    }
    case WM_PAINT:
    {
        PAINTSTRUCT     ps;
        HDC             hdc;
        BITMAP          bitmap;
        HDC             hdcMem;
        HGDIOBJ         oldBitmap;

        hdc = BeginPaint(hWnd, &ps);

        hdcMem = CreateCompatibleDC(hdc);
        oldBitmap = SelectObject(hdcMem, hBitmap);

        GetObject(hBitmap, sizeof(bitmap), &bitmap);
        BitBlt(hdc, 0, 0, bitmap.bmWidth, bitmap.bmHeight, hdcMem, 0, 0, SRCCOPY);

        SelectObject(hdcMem, oldBitmap);
        DeleteDC(hdcMem);

        EndPaint(hWnd, &ps);
        break;
    }
    case WM_CLOSE:
    {
        SetLayeredWindowAttributes(optionsHWND, NULL, MAIN_DISABLED,
LWA_ALPHA);
        SetWindowLong(optionsHWND, GWL_EXSTYLE, WS_EX_TOOLWINDOW |
WS_EX_TRANSPARENT | WS_EX_LAYERED | WS_EX_TOPMOST);
        SetWindowPos(optionsHWND, NULL, 0, 0, 413, 673, SWP_HIDEWINDOW);
        flagMainHWND = false;
        return 0;
    }
    case WM_CTLCOLOREDIT:
    {
        HDC hdcEdit = (HDC)wParam;
        SetTextColor(hdcEdit, RGB(255, 255, 255));
        SetBkColor(hdcEdit, RGB(5, 24, 16));
        return (LONG)GetStockObject(BLACK_BRUSH);
    }
    break;
    case WM_COMMAND:
    {
        if (!flagInit) break;
```

```
        GetWindowText(hwndMaxFrames, buffW, 1024);
        buffWStr = std::wstring(buffW);
        buffStr = std::string(buffWStr.begin(), buffWStr.end());
        try
        {
           maxFrames = std::stoi(buffStr);
        }
        catch (...) {}

        GetWindowText(hwndFramesDelay, buffW, 1024);
        buffWStr = std::wstring(buffW);
        buffStr = std::string(buffWStr.begin(), buffWStr.end());
        try
        {
           delay = std::stol(buffStr);
        }
        catch (...) {}

        GetWindowText(hwndResolutionCompression, buffW, 1024);
        buffWStr = std::wstring(buffW);
        buffStr = std::string(buffWStr.begin(), buffWStr.end());
        try
        {
           resolution = std::stol(buffStr);
        }
        catch (...) {}

        GetWindowText(hwndPathToCursor, buffW, 1024);
        buffWStr = std::wstring(buffW);
        pathToCursor = std::string(buffWStr.begin(), buffWStr.end());

        ComboBox_GetLBText(hwndFlagCursor, ComboBox_GetCurSel(hwndFlagCursor),
buffW);
        buffWStr = std::wstring(buffW);

        if (buffWStr == L"Enabled")
        {
           flagCursorShow = true;
        }
        else
        {
           flagCursorShow = false;
        }

        if (prevPathToCursor != pathToCursor)
        {
           prevPathToCursor = pathToCursor;
           try
           {
              cursorIco = Magick::Image(pathToCursor);
           }
```

```
            catch (...) {}
        }

        return 0;
    }
    }

    return DefWindowProc(hWnd, msg, wParam, lParam);
}

inline void CreateMainHWND()
{
    WNDCLASSEX wcex;

    wcex.cbClsExtra = 0;
    wcex.cbSize = sizeof(WNDCLASSEX);
    wcex.cbWndExtra = 0;
    wcex.hbrBackground = NULL;
    wcex.hCursor = LoadCursor(NULL, IDC_ARROW);
    wcex.hIcon = NULL;
    wcex.hIconSm = NULL;
    wcex.hInstance = hInstanceGlobal;
    wcex.lpfnWndProc = OptionsWndProc;
    wcex.lpszClassName = L"Video-Recoding";
    wcex.lpszMenuName = NULL;
    wcex.style = 0;

    RegisterClassEx(&wcex);
    optionsHWND = CreateWindowEx(WS_EX_TOOLWINDOW | WS_EX_TRANSPARENT |
WS_EX_LAYERED | WS_EX_TOPMOST,
        L"Video-Recoding", NULL, WS_VISIBLE | WS_CLIPSIBLINGS | WS_CLIPCHILDREN
| WS_SYSMENU & ~WS_CAPTION, 0, 0, 413, 673, NULL, NULL, NULL, NULL);
    SetLayeredWindowAttributes(optionsHWND, NULL, MAIN_DISABLED, LWA_ALPHA);

    LOGFONT logFont;
    logFont.lfHeight = -16;
    strcpy((char*)logFont.lfFaceName, "Aria");
    auto hfont = CreateFontIndirect(&logFont);

    hwndMaxFrames = CreateWindow(L"EDIT", L"", WS_VISIBLE | ES_NUMBER |
WS_TABSTOP | ES_AUTOHSCROLL | WS_CHILD,
        167, 202, 180, 20, optionsHWND, NULL, hInstanceGlobal, NULL);
    SendMessage(hwndMaxFrames, WM_SETFONT, (WPARAM)hfont, (LPARAM)0);

    hwndFramesDelay = CreateWindow(L"EDIT", L"", WS_VISIBLE | ES_NUMBER |
WS_TABSTOP | ES_AUTOHSCROLL | WS_CHILD,
        180, 263, 142, 20, optionsHWND, NULL, hInstanceGlobal, NULL);
    SendMessage(hwndFramesDelay, WM_SETFONT, (WPARAM)hfont, (LPARAM)0);

    hwndResolutionCompression = CreateWindow(L"EDIT", L"", WS_VISIBLE | ES_NUMBER
| WS_TABSTOP | ES_AUTOHSCROLL | WS_CHILD,
```

```
        277, 329, 70, 20, optionsHWND, NULL, hInstanceGlobal, NULL);
    SendMessage(hwndResolutionCompression, WM_SETFONT, (WPARAM)hfont,
(LPARAM)0);

    hwndFlagCursor = CreateWindow(L"COMBOBOX", L"", WS_VISIBLE | WS_TABSTOP |
CBS_HASSTRINGS | CBS_DROPDOWNLIST | WS_CHILD,
        167, 389, 186, 100, optionsHWND, NULL, hInstanceGlobal, NULL);
    SendMessage(hwndFlagCursor, WM_SETFONT, (WPARAM)hfont, (LPARAM)0);

    SendMessage(hwndFlagCursor, CB_ADDSTRING, 0, (LPARAM)TEXT("Enabled"));
    SendMessage(hwndFlagCursor, CB_ADDSTRING, 0, (LPARAM)TEXT("Disabled"));

    hwndPathToCursor = CreateWindow(L"EDIT", L"", WS_VISIBLE | WS_TABSTOP |
ES_AUTOHSCROLL | WS_CHILD,
        120, 454, 227, 20, optionsHWND, NULL, hInstanceGlobal, NULL);
    SendMessage(hwndPathToCursor, WM_SETFONT, (WPARAM)hfont, (LPARAM)0);
}

inline void ReadOptionsFile()
{
    std::ifstream file;
    try
    {
        cursorIco = Magick::Image(pathToCursor);
    }
    catch (...)
    {
        MessageBox(NULL, L"Cursor not found", L"ERROR", MB_ICONERROR);
        flagCursorShow = false;
    }

    try
    {
        file.open("options.dat");
        file >> maxFrames >> delay >> resolution >> flagCursorShow >> pathToCursor;
    }
    catch (...) {}

    file.close();
    SetWindowText(hwndMaxFrames, (LPCWSTR)std::to_wstring(maxFrames).c_str());
    SetWindowText(hwndFramesDelay, (LPCWSTR)std::to_wstring(delay).c_str());
    SetWindowText(hwndResolutionCompression,
(LPCWSTR)std::to_wstring(resolution).c_str());
    SetWindowText(hwndPathToCursor, (LPCWSTR)(std::wstring(pathToCursor.begin(),
pathToCursor.end())).c_str());
    if (flagCursorShow)
    {
        SendMessage(hwndFlagCursor, CB_SETCURSEL, 0, 0);
    }
    else
    {
```

```
        SendMessage(hwndFlagCursor, CB_SETCURSEL, 1, 0);
    }

    flagInit = true;
    prevPathToCursor = pathToCursor;
}

inline void WriteOptionsFile()
{
    std::ofstream myfile;
    myfile.open("options.dat");
    myfile << maxFrames << " " << delay << " " << resolution << " " << flagCursorShow << " "
<< pathToCursor << " ";
    myfile.close();
}
```

Файл windowFunctions.cpp:

```
#include "windowFunctions.h"

inline void ResizeWnd(HWND hWnd)
{
    HDC hdcWindow = GetDC(hWnd);

    GetClientRect(hWnd, &rcSize);

    width = endPoint.x - startPoint.x;
    height = endPoint.y - startPoint.y;
    offSetX = startPoint.x;
    offSetY = startPoint.y;

    if (width < 0)
    {
        width = -width;
        offSetX = endPoint.x;
    }

    if (height < 0)
    {
        height = -height;
        offSetY = endPoint.y;
    }

    if (hdcBackBuffer) DeleteDC(hdcBackBuffer);
    hdcBackBuffer = CreateCompatibleDC(hdcWindow);
    HBITMAP hbmBackBuffer = CreateCompatibleBitmap(hdcBackBuffer, rcSize.right -
rcSize.left, rcSize.bottom - rcSize.top);
    SelectObject(hdcBackBuffer, hbmBackBuffer);
    DeleteObject(hbmBackBuffer);

    if (hdcArea) DeleteDC(hdcArea);
    hdcArea = CreateCompatibleDC(hdcWindow);
```

```
    HBITMAP hbmArea;
    hbmArea = CreateCompatibleBitmap(hdcArea, width, height);
    SelectObject(hdcArea, hbmArea);
    DeleteObject(hbmArea);
    RECT rcSprite;
    SetRect(&rcSprite, 0, 0, width, height);
    FillRect(hdcArea, &rcSprite, (HBRUSH)GetStockObject(BLACK_BRUSH));
    InvalidateRect(hWnd, &rcSize, true);

    ReleaseDC(hWnd, hdcWindow);
}

inline void HideAreaHWND()
{
    endPoint.x = 0;
    endPoint.y = 0;
    startPoint.x = 0;
    startPoint.y = 0;
    ResizeWnd(areaHWND);
    SetLayeredWindowAttributes(areaHWND, NULL,
WORK_AREA_TRANSPARENCY_DISABLED, LWA_ALPHA);
    SetWindowLong(areaHWND, GWL_EXSTYLE, WS_EX_TOOLWINDOW |
WS_EX_TRANSPARENT | WS_EX_LAYERED);
    SetWindowPos(areaHWND, NULL, 0, 0, resolutionWH.right, resolutionWH.bottom,
SWP_HIDEWINDOW);
    areaIsReady = false;
}
```

Обозначение	Наименование	Дополнительные сведения
	Текстовые документы	
БГУИР КП 1–40 01 01 605 ПЗ	Пояснительная записка	40 с.
	Графические документы	
ГУИР 851006 01 ПД	Схема программы на А2	Формат А1

					БГУИР КП 1-40 01 01 605 Д1		
Изм.	Л.	№ докум.	Подп.	Дата		Лист	Листов
Разраб.		Верещагин Н.В.			Видеозахват экрана	Т	40 40
Пров.		Жиденко А. Л.				Кафедра ПОИТ гр. 851006	