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 курсовому проекту

на тему

Программное средство – Pensil-Mouse Controller

БГУИР КР 1-40 01 01 505 ПЗ

Студент: гр.651005 Голуб А.В.

Руководитель: Низовцов Д.В.

Минск 2018

Оглавление

[Введение 3](#_Toc515906304)

[1 Анализ существующих источников 4](#_Toc515906305)

[1.1 Существующие аналоги 4](#_Toc515906307)

[2 Анализ требований к ПС и разработка функциональных требований 7](#_Toc515906308)

[2.1 Теоретический анализ программной логики видеочата 7](#_Toc515906309)

[2.2 Описание функциональности ПС 7](#_Toc515906310)

[2.3 Спецификация функциональных требований 8](#_Toc515906311)

[3 Проектирование программного средства 9](#_Toc515906312)

[4 Создание программного средства 10](#_Toc515906313)

[5 Тестирование, проверка работоспособности и анализ полученных результатов 12](#_Toc515906314)

6 Руководство пользователя…………………………………………………….14

[Заключение 17](#_Toc515906316)

[Список использованной литературы 18](#_Toc515906317)

[Приложение А 19](#_Toc515906318)

[Приложение Б 22](#_Toc515906320)

# ВВЕДЕНИЕ

В современном мире персональный компьютер стал неотъемлемой частью каждого человека. Его используют для различных нужд: просмотр фото, видео, различных других файлов, а также для игры в видеоигры и серфинга в Интернете.

С помощью ПК можно повысить эффективность многих видов работы, чем и активно пользуется человечество. Персональный компьютер с доступом в сеть Интернет даёт возможность получить практически любую информацию и за очень короткий период. Наличие ПК – это очень удобно и практично.

С давних времен люди стремятся к упрощению всего, к достижению большего для себя комфорта. Ярким примером того является автомобиль, который позволил преодолевать большие расстояния намного быстрее, а главное – проще.

Про подобное упрощение можно говорить и в компьютерных технологиях. Так, например, один из примеров - операционная система Windows, в которой с каждой новой версией пользовательский интерфейс менялся на “более дружелюбный к пользователю”.

Но люди постоянно хотят чего-то большего, и теперь в ПК, смартфонах и так далее есть функция голосового управления. И хоть она еще развивается, но результаты развития впечатляют.

Но прогресс IT технологий и на этом не остановился, и всё большую значимость приобретает такая сфера, как графический анализ или анализ изображений.

Разработчики в этой области добились невероятных успехов, начиная от распознавания объектов, лиц и заканчивая анализом здоровья почвы на основе снимков со спутника. Анализ изображений используется в некоторых сигнализациях как детектор движения, а в будущем – для определения злоумышленника по его эмоциям.

Для написания курсового проектирования по предмету Операционные системы и системное программирование я решил попробовать создать программное средство, которое бы реализовывало основные функции курсора на основе визуального управления через веб-камеру.

В качестве языка программирования мною был выбран язык C++, так как он хорошо подходит для написания приложений с графическими задачами, а в качестве среды разработки – Microsoft Visual Studio 2015. Используемый язык и интегрированная среда разработки были изучены мной самостоятельно во время написания лабораторных работ и курсовых проектов в текущем и прошлых семестрах.

# АНАЛИЗ СУЩЕСТВУЮЩИХ ИСТОЧНИКОВ

### Существующие аналоги

В качестве аналогов для анализа я выбрал такие программные средства как *Camera Mouse, Enable Viacam и технологию Intel RealSense Cursor Mode.*

* + 1. **Camera Mouse**

Camera Mouse – бесплатная программа, которая позволяет вам управлять указателем мыши на экране вашего компьютера, просто двигая головой. В перечень функций программы входит и возможность кликов мыши, а так же скролл.



Рисунок 1.1 – Camera Mouse

Программа была разработана в Бостонском колледже, чтобы помочь людям с ограниченными возможностями использовать компьютер. Основная аудитория – люди, которые не имеют надежного контроля над руками.

* + 1. **Enable Viacam**

Enable Viacam – это программное обеспечение для замены мыши, которое перемещает указатель при перемещении головы. Работает на стандартном ПК, оснащенном веб-камерой и не требует никакого дополнительного оборудования.

Среди функций этой программы: эмуляция нажатий клавиши мыши и двойное нажатие.

Недостатком данного ПС является отсутствие функциональности для вращения колёсика вверх и вниз.

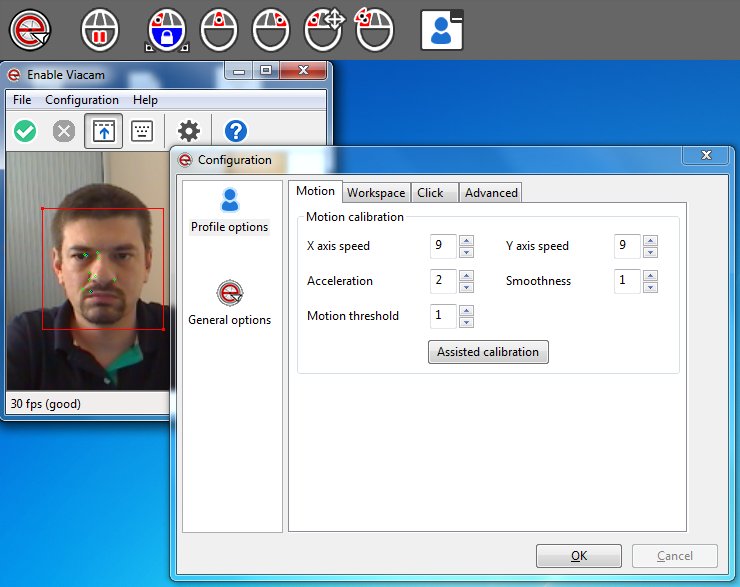


Рисунок 1.2 – Enable Viacam

Главным недостатком двух выше рассмотренных приложений является то, что управление будет затруднено за счёт фактора уставания шеи и того, что слишком затруднительно сильно и много вращать головой для полноценного управления курсором.

* + 1. **Intel Realsense**

Intel Realsense – это технология трехмерного зондирования, которая является ключом к тому, чтобы сделать взаимодействие человека с машиной естественным и интуитивным. Как расширение ее функциональных возможностей, у нее существует режим курсора.

Режим курсора - это один из нескольких модулей, которые предоставляют разработчикам приложений расширенные функции отслеживания и распознавания жестов. Он обеспечивает 3D-отслеживание движений рук в режиме реального времени с помощью камеры Intel® RealSense ™.

Модуль Cursor может отслеживать одну или две руки, обеспечивает быстрое и точное отслеживание для отслеживания положения рук в пространстве и преобразует полученные данные в сглаженные и отзывчивые данные курсора руки, сохраненные в данных курсора. Эта информация используется для создания представления курсора на мониторе и может использоваться разработчиками для создания приложений.

Главным недостатком этой технологии является то, что она требует специализированного оборудования, а именно специальной веб-камеры Intel RealSense Camera.



Рисунок 1.3 – Intel Realsense Cursor mode

# 2 АНАЛИЗ ТРЕБОВАНИЙ К ПС И РАЗРАБОТКА ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ

### Теоретический анализ программной логики Pensil-Mouse Controller

При изучении альтернативных программных средств схожего назначения можно выделить несколько особенностей, которые присущи всем современным приложениям такого типа. Это должно быть приложение, поддерживающее такие функции, как:

- Включение графического управления курсором

- Отключение режим графического управления курсором

- Выбор цвета карандаша, а так же цвета точек на них

- Перемещение курсора на экране при перемещении карандаша

- Моделирование левого и правого клика мыши при наклоне карандаша

- Вращение колёсика вверх и вниз

С программной точки зрения было принято решение использовать библиотеку Opencv для получения изображения с камеры, а так же для обработки изображения. Для формирования стандартных компонентов Windows, а так же для моделирования работы мыши использовались стандартные функции WinAPI.

В программном средстве дизайн хоть и важен, однако для простоты использования было решено использовать классический дизайн, с обыкновенным интерфейсом, так как, по сути, это приложение большинство своего времени работы будет проводить в фоновом режиме, чтобы пользователь мог беспрепятственно пользоваться основной функциональностью, предоставленной приложением.

### Описание функциональности ПС

При запуске программного средства пользователь должен произвести калибровку, то есть при помощи 6 ползунков для карандаша и для двух точек, на основании видимости которых происходит вращение колёсика, происходит выбор цвета. По умолчанию, цвет карандаша настроен как голубой, колёсико вверх – красный цвет, а колёсико вниз – жёлтый.

Как только пользователь откалибровал цвет, при помощи клавиатуры он может включить “режим курсора”, и тогда будут доступны функции перемещения курсора на экране, правый и левый клик мыши при наклоне карандаша соответственно вправо и влево, а так же вращение колёсика мыши при видимости на карандаше других цветов, которые задавались калибровкой.

### Спецификация функциональных требований

После анализа различных программных средств, схожей функциональности, мною были выявлены функциональные требования, необходимые для создания современного, достойного и конкурентоспособного программного средства:

- Графический интерфейс для настройки

- Включение и отключение управления через клавиатуру

- Возможность калибровки

- Выход из приложения при помощи клавиатуры

- Управление курсором

В дальнейшем данный функционал можно расширять.

# 3 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

В самом начале разработки программного средства я разработал структуру окон, которая служит для настройки “режима курсора”. Структура состоит из 5 окон, а их взаимосвязь предоставлена в приложении А на рисунке 1.

При открытии программы появятся 5 окон, описанные выше, которые предназначены для предварительной настройки корректной работы программы. Пользователь настраивает цвета так, чтобы в соответствующих окнах они отображались как белый цвет, а остальные как черный (производится бинаризация изображения). После калибровки пользователь может включить “режим курсора” и начать управлять курсором при помощи карандаша.

На рисунке 2 приложения А можно увидеть укрупненную схему программы. В блоке операции пользователя подразумевается именно включение и отключение “режима курсора”, а так же непосредственно его работа.

При разработке программы было решено использовать разбиение программы на классы и использования принципов объектно-ориентированного программирования для того, чтобы разделить программу на отдельные функциональные части, а так же повысить переносимость кода и его легко читаемость.

Работу “режима курсора” можно проследить на рисунке 3 приложения А. Здесь подготовка к клику обозначает наклон карандаша на 7 градусов влево или вправо. Это сделано для того, чтобы повысить точность клика мыши, потому что положение курсора зачастую очень нестабильно, так как человеческая рука находится в постоянном движении. Для решения этой проблемы так же и используется проверка смещения на 5 пикселей, чтобы исключить дребезжание курсора. Блок “Получение состояний карандаша” отображен на рисунке 4 приложения А.

Анализ состояний карандаша происходит на основе анализа изображения, полученного с веб-камеры. Для этого изображение переводится в цветовую модель hsv, так как наиболее приближенна к человеческому восприятия цвета. Затем применяется фильтр для удаления шумов и дальше берется самый большой прямоугольный контур больше заданного значения. Зачастую это и будет наш карандаш. Для вычисления состояния вращения колёсика используется проверка на наличие точки соответствующего цвета в пределах карандаша. Чтобы определить клик, потребовалось вычислять угол наклона карандаша и затем анализировать его.

# 4 СОЗДАНИЕ ПРОГРАММНОГО СРЕДСТВА

В среде Visual Studio важнейшим компонентом является окно. На нем можно расположить компоненты и с их помощью осуществляется взаимодействие с пользователем.

В приложении существует 5 окон: 4 из них используются для вывода изображений для калибровки и 5-ая – это калибровочная панель, на которой расположены 18 TrackBar, за хранение значения которого отвечает 18 переменных из класса ColorDetector.

Класс ColorDetector выполняет функцию начального преобразование изображения, а именно фильтрация по цвету и удаление шумов для получения идеального бинарного изображения.

Методы, описанные пользователем, и их назначение приведены в таблице 4.1.

Таблица 4.1 – Методы класса ColorDetector.cpp и их назначение

|  |  |
| --- | --- |
| Имя метода | Назначение |
| GetPensilMask | Получение масок карандаша и точек в нём |
| CreateTrackbars | Создание трэкбаров для калибровки |
| MorphOperations | Морфологические операции для обработки бинарного изображения |

Класс PensilDetector обрабатывает изображение, полученное из класса ColorDetector, анализирует его на предмет нахождения нужного нам карандаша и формирует его состояния, которые сам же и хранит.

Методы, описанные пользователем, и их назначение приведены в таблице 4.2.

Таблица 4.2 – Методы класса PensilDetector.cpp и их назначение

|  |  |
| --- | --- |
| Имя метода | Назначение |
| GetLeftClick | Получение состояния левого клика |
| GetRightClick | Получение состояния правого клика |
| GetWheelUp | Получение состояния движения колёсика вверх |

Продолжение таблицы 4.2

|  |  |
| --- | --- |
| GetWheelDown | Получение состояния движения колёсика вниз |
| GetClickPreparation | Получение состояния подготовка к клику |
| GetMousePosition | Получение позиции курсора |
| CheckClick | Вычисление состояний клика |
| CalculatePositionAndClicks | Расчёт состояний карандаша |
| FindBiggestContour | Поиск наибольшего контура |
| CheckWheels | Получение состояний вращения колёсика |
| GetBottomPoint | Получение самой нижней точки контура |

И в завершении main.cpp, который и использует логику, написанную в предыдущих классах

Глобальные переменные, объявленные пользователем в main.cpp, и их назначение приведены в таблице 4.3.

Таблица 4.3 – Глобальные переменные, используемые в main.cpp

|  |  |  |
| --- | --- | --- |
| Имя переменной | Тип переменной | Назначение |
| cursorFlag | bool | Включён ли “режим курсора” |
| cursorOffsetRange | const int =5 | Смещение, при котором происходит движение курсора |
| x | int | Х координата курсора |
| y | int | Y координата курсора |
| clickFlag | bool | Флаг, хранящий предыдущее состояние клика |

# ТЕСТИРОВАНИЕ, ПРОВЕРКА РАБОТОСПОСОБНОСТИ И АНАЛИЗ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ

Таблица 5.1 – Тестирование приложения

|  |  |  |  |
| --- | --- | --- | --- |
| Номер теста | Проверка | Ожидаемый результат | Полученный результат |
| 1 | Работа ползунков | При перемещении ползунка изменяется изображение в окне | Полученный результат совпал с ожидаемым |
| 2 | Выход из приложения | При нажатии клавиши ESC приложение закроется | Полученный результат совпал с ожидаемым |
| 3 | Движение курсора | При движении карандаша влево курсор движется влево | Если в кард попадают другие динамические объекты, схожие по сигнатуре и цвету с карандашом, то они могут использоваться как источник движения курсора |
| 4 | Левый клик | При наклоне карандаша влево происходит левый клик мыши | Полученный результат совпал с ожидаемым |
| 5 | Правый клик | При наклоне карандаша вправо происходит правый клик мыши | Полученный результат совпал с ожидаемым |
| 6 | Движение колёсика вверх | При нахождении красной точки на карандаше начнётся вращение вверх | Полученный результат совпал с ожидаемым |
| 7 | Движение колёсика вниз | При нахождении жёлтой точки на карандаше начнётся вращение вниз | Полученный результат совпал с ожидаемым |

Продолжение таблицы 5.1.

|  |  |  |  |
| --- | --- | --- | --- |
| 8 | Включение режима курсора | При нажатии на пробел включится режим курсора | Полученный результат совпал с ожидаемым |
| 9 | Выключение режима курсора | При нажатии на пробел при включенном режиме курсора он выключится | Полученный результат совпал с ожидаемым |

# 6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

При запуске программы откроется 5 окон.

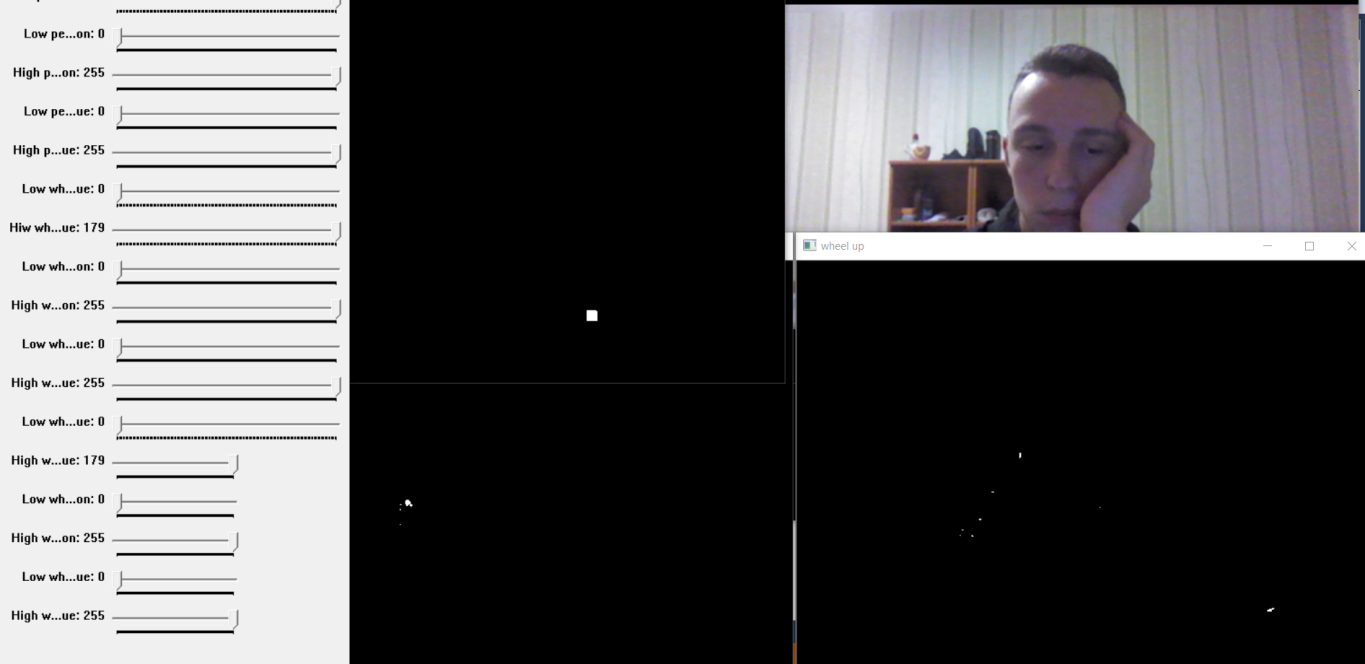


Рисунок 6.1 – стартовый запуск

Следующий шаг работы приложения – это калибровка. По умолчанию приложение откалибровано на голубой цвет карандаша, и красный с жёлтым для движение колёсика вверх и вниз соответственно.

Однако для того, чтобы можно было, например, изменить цвет карандаша достаточно изменить положения 6-ти верхних ползунков в панели калибровки и отслеживать изменения в окне под названием “pensil”.

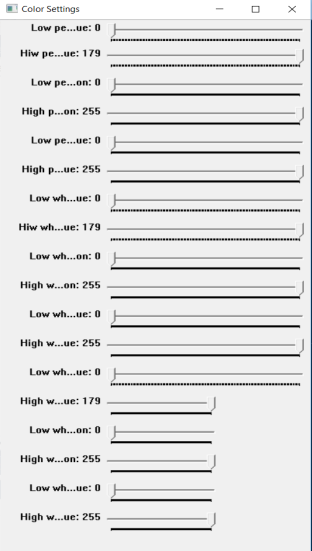


Рисунок 6.2 – панель калибровки

Для того, чтобы можно было изменить цвета движения колёсика, достаточно по тому же принципу изменить положения с 7 по 12 ползунки для движения вверх и отслеживать изменения в окне “wheel up” и для движения колёсика вниз – с 13 по 18 и отслеживать изменения в окне “wheel down”.

После калибровки пользователь может включить режим курсора нажав клавишу пробел. Чтобы его отключить, нужно нажать опять пробел.

А затем нужно просто держать карандаш в зоне видимости камеры на расстоянии примерно 30 сантиметров.

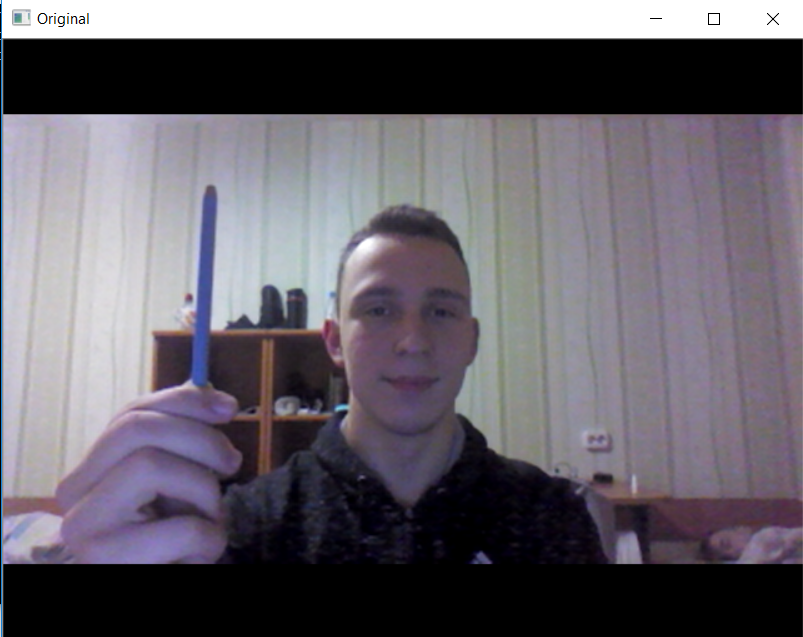


Рисунок 6.3 – режим курсора

Для того, чтобы осуществить левый или правый клик мыши нужно наклонить карандаш влево или вправо на угол 45 градусов.

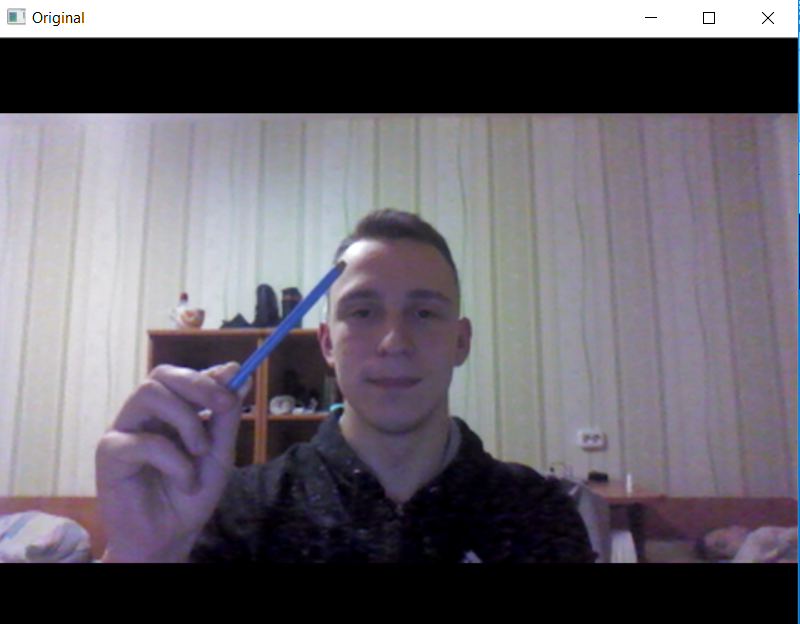


Рисунок 6.4 – левый клик мыши

А для того, чтобы можно было “вращать колёсико”, нужно развернуть карандаш стороной с точкой, цвет которой определяет вращение вверх, или точкой, цвет которой определяет вращение вниз.

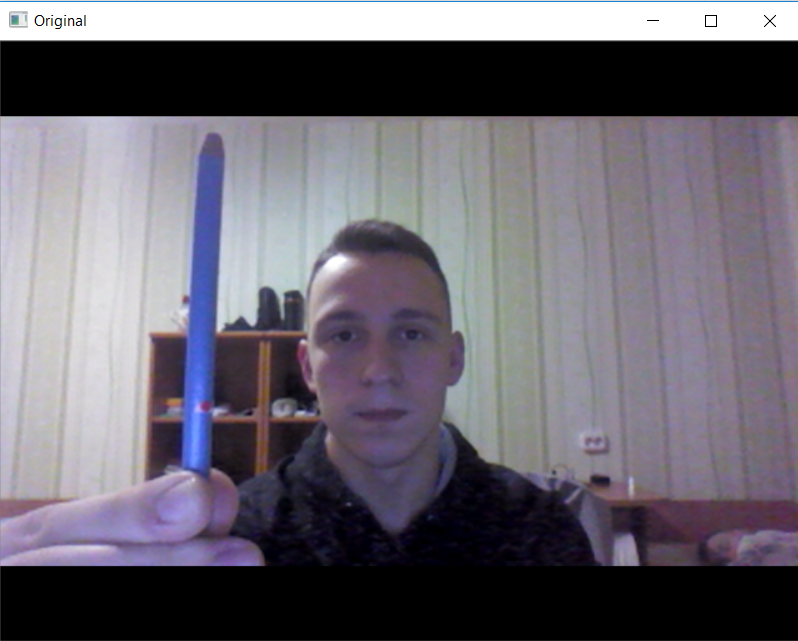


Рисунок 6.5 – вращение колёсика вверх

Чтобы закрыть приложение достаточно нажать клавишу ESC.

# ЗАКЛЮЧЕНИЕ

В ходе разработки данного программного средства был создан продукт, рассчитанный на широкий круг пользователей. Простота приложения способствуют этому.

Во время разработки мною были закреплены знания о программировании на WinAPI в Visual Studio 2015.

Из-за недостатка опыта, некоторые дополнительные функции, такие как двойной щелчок, плавное и быстрое перемещение курсора, а так же возможность выделения объектов не были реализованы. Возможно добавление различных языков интерфейса или различных тем оформления или полноценное расширение функционала и превращения данной программы в полноценный проект для повседневного использования с открытым исходным кодом.

Но, не смотря на всё это, мною был разработан продукт, который успешно справляется с поставленными задачами и отражает уровень знаний, полученный мною за семестр обучения.

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

[1] Википедия [Электронный ресурс]. – Электронные данные. – Режим доступа: https://ru.wikipedia.org/wiki/.

[2] ГОСТ 19.701–90 (ИСО 5807–85) [Текст]. – Единая система программ-ной документации: Сб. ГОСТов. – М.: Стандартинформ, 2005 с.

[3] Документация winapi[Электронный ресурс]. – Электронные данные. – Режим доступа: https://msdn.microsoft.com.

[4] Документация OpenCV[Электронный ресурс]. – Электронные дан-ные. – Режим доступа: https://www.opencv.org/.

# Приложение А

# Схемы алгоритмов



Рисунок 1 – Структура окон программы



Рисунок 2 – Общая схема работы программы

Рисунок 3 – Работа режима курсора





Рисунок 4 – Получение состояний карандаша

# Приложение Б

# Исходный текст программы

Код main.cpp

#include "opencv2/opencv.hpp"

#include "opencv2/imgcodecs.hpp"

#include "opencv2/imgproc.hpp"

#include "opencv2/videoio.hpp"

#include <opencv2/highgui.hpp>

#include <opencv2/video.hpp>

#include <Windows.h>

#include "ColorDetector.h"

#include "PensilDetector.h"

#pragma warning(disable:4996)

using namespace cv;

using namespace std;

bool cursorFlag = false;

const int cursorOffsetRange = 5;

int x;

int y;

bool clickFlag = false;

int main(int argc, char\*\* argv)

{

cv::VideoCapture videoCapture(1);

videoCapture.set(CV\_CAP\_PROP\_SETTINGS, 1);

if (!videoCapture.isOpened())

{

cout << "Can't find camera!" << endl;

return -1;

}

ColorDetector skinDetector;

PensilDetector pensilDetector;

cv::Mat frame,pensilFrame,wheelUpFrame,wheelDownFrame;

VideoCapture cap(1);

if (!cap.isOpened())

{

cout << "Cannot open the web cam" << endl;

return -1;

}

cv::Point point;

cv::Point prevPoint;

point.x = 0;

point.y = 0;

while (true)

{

videoCapture >> frame;

pensilFrame = frame.clone();

wheelUpFrame = frame.clone();

wheelDownFrame = frame.clone();

skinDetector.getPensilMask(pensilFrame,wheelUpFrame,wheelDownFrame);

cv::imshow("pensil", pensilFrame);

cv::imshow("wheel down", wheelDownFrame);

cv::imshow("wheel up", wheelUpFrame);

if (cursorFlag)

{

pensilDetector.CalculatePositionAndClicks(pensilFrame, wheelUpFrame, wheelDownFrame);

pensilDetector.CheckClick();

if (!pensilDetector.GetClickPreparation())

{

prevPoint = point;

point = pensilDetector.GetMousePosition();

if (((point.x < prevPoint.x + cursorOffsetRange) && (point.x > prevPoint.x - cursorOffsetRange) && (point.y < prevPoint.y + cursorOffsetRange) && (point.y > prevPoint.y - cursorOffsetRange)))

{

cv::Size size = frame.size();

POINT cursorPoint;

if (point.x < 150 || point.x>size.width - 150 || point.y<150 || point.y>size.height - 50)

{

GetCursorPos(&cursorPoint);

x = cursorPoint.x;

y = cursorPoint.y;

}

else

{

int temp = GetSystemMetrics(SM\_CXSCREEN);

x = temp - temp\*(point.x - 150) / (size.width - 300);

y = GetSystemMetrics(SM\_CYSCREEN)\*(point.y - 150) / (size.height - 200);

}

SetCursorPos(x, y);

}

}

if (pensilDetector.GetLeftClick())

{

mouse\_event(MOUSEEVENTF\_LEFTDOWN, x, y, 0, 0);

clickFlag = true;

puts("Left click");

continue;

}

else

{

if (clickFlag)

{

mouse\_event(MOUSEEVENTF\_LEFTUP, x, y, 0, 0);

}

clickFlag = false;

}

if (pensilDetector.GetRightClick())

{

mouse\_event(MOUSEEVENTF\_RIGHTDOWN, x, y, 0, 0);

mouse\_event(MOUSEEVENTF\_RIGHTUP, x, y, 0, 0);

puts("Right click");

}

if (pensilDetector.GetWheelUp())

{

mouse\_event(MOUSEEVENTF\_WHEEL, 0, 0, DWORD(WHEEL\_DELTA), 0);

puts("wheel up");

}

if (pensilDetector.GetWheelDown())

{

mouse\_event(MOUSEEVENTF\_WHEEL, 0, 0, DWORD(-WHEEL\_DELTA), 0);

puts("wheel down");

}

}

imshow("Original", frame);

int key = waitKey(1);

if (key == 27)

{

cout << "esc key is pressed by user" << endl;

break;

}

else

{

if (key == 32)

{

if (!cursorFlag)

{

cursorFlag = true;

}

else

{

cursorFlag = false;

}

}

}

}

return 0;

}

ColorDetector.h

#pragma once

#include<opencv\cv.h>

using namespace std;

class ColorDetector

{

public:

ColorDetector(void);

void getPensilMask(cv::Mat &pensil,cv::Mat &wheelUp,cv::Mat &wheelDown);

void CreateTrackbars();

private:

int iLowPensilH = 0;

int iHighPensilH = 179;

int iLowPensilS = 0;

int iHighPensilS = 255;

int iLowPensilV = 0;

int iHighPensilV = 255;

int iLowWUH = 0;

int iHighWUH = 179;

int iLowWUS = 0;

int iHighWUS = 255;

int iLowWUV = 0;

int iHighWUV = 255;

int iLowWDH = 0;

int iHighWDH = 179;

int iLowWDS = 0;

int iHighWDS = 255;

int iLowWDV = 0;

int iHighWDV = 255;

void MorphOperations(cv::Mat &threshold);

};

ColorDetector.cpp

#include "ColorDetector.h"

#include"opencv2\opencv.hpp"

ColorDetector::ColorDetector(void)

{

CreateTrackbars();

iLowPensilH = 75;

iLowPensilS = 113;

iLowPensilV = 23;

iHighPensilH = 125;

iLowWUH = 160;

iLowWUS = 122;

iLowWUV = 50;

iLowWDH = 15;

iHighWDH = 35;

iLowWDS = 140;

iLowWDV = 50;

}

void ColorDetector::CreateTrackbars()

{

cv::namedWindow("Color Settings", 0); //create a window called "Control"

cv::resizeWindow("Color Settings", 400, 900);

cvCreateTrackbar("Low pensil hue", "Color Settings", &iLowPensilH, 179); //Hue (0 - 179)

cvCreateTrackbar("Hiw pensil hue", "Color Settings", &iHighPensilH, 179);

cvCreateTrackbar("Low pensil saturation", "Color Settings", &iLowPensilS, 255); //Saturation (0 - 255)

cvCreateTrackbar("High pensil saturation", "Color Settings", &iHighPensilS, 255);

cvCreateTrackbar("Low pensil value", "Color Settings", &iLowPensilV, 255); //Value (0 - 255)

cvCreateTrackbar("High pensil value", "Color Settings", &iHighPensilV, 255);

cvCreateTrackbar("Low wheel up hue", "Color Settings", &iLowWUH, 179); //Hue (0 - 179)

cvCreateTrackbar("Hiw wheel up hue", "Color Settings", &iHighWUH, 179);

cvCreateTrackbar("Low wheel up saturation", "Color Settings", &iLowWUS, 255); //Saturation (0 - 255)

cvCreateTrackbar("High wheel up saturation", "Color Settings", &iHighWUS, 255);

cvCreateTrackbar("Low wheel up value", "Color Settings", &iLowWUV, 255); //Value (0 - 255)

cvCreateTrackbar("High wheel up value", "Color Settings", &iHighWUV, 255);

cvCreateTrackbar("Low wheel down hue", "Color Settings", &iLowWDH, 179); //Hue (0 - 179)

cvCreateTrackbar("High wheel down hue", "Color Settings", &iHighWDH, 179);

cvCreateTrackbar("Low wheel down saturation", "Color Settings", &iLowWDS, 255); //Saturation (0 - 255)

cvCreateTrackbar("High wheel down saturation", "Color Settings", &iHighWDS, 255);

cvCreateTrackbar("Low wheel down value", "Color Settings", &iLowWDV, 255); //Value (0 - 255)

cvCreateTrackbar("High wheel down value", "Color Settings", &iHighWDV, 255);

}

void ColorDetector::MorphOperations(cv::Mat &thresh)

{

cv::Mat erodeElement = cv::getStructuringElement(cv::MORPH\_RECT,cv::Size(5,5));

cv::Mat dilateElement = cv::getStructuringElement(cv::MORPH\_RECT, cv::Size(8,8));

cv::erode(thresh, thresh, erodeElement);

cv::dilate(thresh, thresh, dilateElement);

cv::dilate(thresh, thresh, dilateElement);

cv::erode(thresh, thresh, erodeElement);

}

void ColorDetector::getPensilMask(cv::Mat &pensil,cv::Mat &wheelUp,cv::Mat &wheelDown)

{

cvtColor(pensil, pensil, CV\_BGR2HSV);

inRange(

pensil,

cv::Scalar(iLowPensilH, iLowPensilS, iLowPensilV),

cv::Scalar(iHighPensilH, iHighPensilS, iHighPensilV),

pensil);

MorphOperations(pensil);

cvtColor(wheelDown, wheelDown, CV\_BGR2HSV);

inRange(

wheelDown,

cv::Scalar(iLowWDH, iLowWDS, iLowWDV),

cv::Scalar(iHighWDH, iHighWDS, iHighWDV),

wheelDown);

cvtColor(wheelUp, wheelUp, CV\_BGR2HSV);

inRange(

wheelUp,

cv::Scalar(iLowWUH, iLowWUS, iLowWUV),

cv::Scalar(iHighWUH, iHighWUS, iHighWUV),

wheelUp);

}

PensilDetector.h

#pragma once

#include "opencv\cv.h"

using namespace std;

class PensilDetector

{

public:

PensilDetector(void);

bool GetLeftClick();

bool GetRightClick();

bool GetWheelUp();

bool GetWheelDown();

bool GetClickPreparation();

cv::Point GetMousePosition();

void CheckClick();

void CalculatePositionAndClicks(cv::Mat frame, cv::Mat wheelUp, cv::Mat wheelDown);

private:

int FindBiggestContour(vector<vector<cv::Point>> contours);

void CheckWheels(vector<cv::Point> pensil,cv::Mat wheelUp, cv::Mat wheelDown);

cv::Point GetBottomPoint(vector<cv::Point> points);

bool leftClickFlag = false;

bool rightClickFlag = false;

bool wheelUpFlag = false;

bool wheelDownFlag = false;

bool clickPreparationFlag = false;

bool timerLeft = false;

bool timerRight = false;

int timer = 0;

int angel = 0;

const int timerClick = 1000;

cv::Point mousePosition;

};

PensilDetector.cpp

#include "PensilDetector.h"

#include "opencv2\opencv.hpp"

#pragma warning(disable:4996)

PensilDetector::PensilDetector(void)

{

}

cv::Point PensilDetector::GetBottomPoint(vector<cv::Point> points)

{

cv::Point tempPoint = cv::Point(0, 0);

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

{

if (points[i].y > tempPoint.y)

{

tempPoint = points[i];

}

}

return tempPoint;

}

void PensilDetector::CalculatePositionAndClicks(cv::Mat frame,cv::Mat wheelUp,cv::Mat wheelDown)

{

vector<vector<cv::Point> > contours;

vector<cv::Vec4i> hierarchy;

findContours(frame, contours, hierarchy, CV\_RETR\_TREE, CV\_CHAIN\_APPROX\_SIMPLE, cv::Point(0, 0));

if (contours.size() == 0)

return;

int big = FindBiggestContour(contours);

if (cv::contourArea(contours[big]) < 500)

{

return;

}

vector<int> convex;

cv::convexHull(cv::Mat(contours[big]), convex, false);

vector<cv::Vec4i> offsets;

cv::convexityDefects(cv::Mat(contours[big]), convex, offsets);

if (offsets.size() != 0)

{

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

{

cv::Point tempPoint = cv::Point((contours[big][offsets[i].val[0]].x + contours[big][offsets[i].val[1]].x) / 2, (contours[big][offsets[i].val[0]].y + contours[big][offsets[i].val[1]].y) / 2);

if (sqrt(pow(abs(tempPoint.x - contours[big][offsets[i].val[2]].x), 2) + pow(abs(tempPoint.y - contours[big][offsets[i].val[2]].y), 2))>50)

{

return;

}

}

angel = cv::fitEllipse(contours[big]).angle - 90;

mousePosition = GetBottomPoint(contours[big]);

CheckWheels(contours[big], wheelUp, wheelDown);

}

}

void PensilDetector::CheckWheels(vector<cv::Point> pensil,cv::Mat wheelUp, cv::Mat wheelDown)

{

vector<vector<cv::Point> > contours;

vector<cv::Vec4i> hierarchy;

findContours(wheelUp, contours, hierarchy, CV\_RETR\_TREE, CV\_CHAIN\_APPROX\_SIMPLE, cv::Point(0, 0));

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

{

int j;

for (j = 0; j < contours[i].size(); j++)

{

int temp = cv::pointPolygonTest(pensil, contours[i][j], false);

if (temp == -1 || temp == 0)

{

break;

}

}

if (j == contours[i].size())

{

if (!wheelDownFlag)

{

wheelUpFlag = true;

}

break;

}

else

{

if (wheelUpFlag)

{

wheelUpFlag = false;

}

}

}

findContours(wheelDown, contours, hierarchy, CV\_RETR\_TREE, CV\_CHAIN\_APPROX\_SIMPLE, cv::Point(0, 0));

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

{

int j;

for (j = 0; j < contours[i].size(); j++)

{

if (cv::pointPolygonTest(pensil, contours[i][j], false) == -1)

{

break;

}

}

if (j == contours[i].size())

{

if (!wheelDownFlag)

{

wheelDownFlag = true;

}

break;

}

else

{

if (wheelDownFlag)

{

wheelDownFlag = false;

}

}

}

}

int PensilDetector::FindBiggestContour(vector<vector<cv::Point> > contours)

{

int indexOfBiggestContour = -1;

int sizeOfBiggestContour = 0;

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

{

if (contours[i].size() > sizeOfBiggestContour)

{

sizeOfBiggestContour = contours[i].size();

indexOfBiggestContour = i;

}

}

return indexOfBiggestContour;

}

void PensilDetector::CheckClick()

{

if ((angel > 1 && angel < 83) || (angel > -83 && angel < -1))

{

if (!clickPreparationFlag)

{

clickPreparationFlag = true;

}

}

else

{

if (clickPreparationFlag)

{

clickPreparationFlag = false;

}

}

if (angel > 1 && angel<75)

{

if (leftClickFlag)

{

leftClickFlag = false;

}

if (timerRight)

{

rightClickFlag = false;

}

else

{

rightClickFlag = true;

}

if (timer == 0)

{

timerRight = true;

}

if (timerRight)

{

timer++;

}

if (timerRight > timerClick)

{

timer = 0;

timerRight = false;

}

}

else

{

if (angel > -75 && angel<-1)

{

if (rightClickFlag)

{

rightClickFlag = false;

}

if (timerLeft)

{

leftClickFlag = false;

}

else

{

leftClickFlag = true;

}

if (timer == 0)

{

timerLeft = true;

}

if (timerLeft)

{

timer++;

}

if (timerLeft > timerClick)

{

timer = 0;

timerLeft = false;

}

}

else

{

rightClickFlag = false;

timerRight = false;

timer = 0;

leftClickFlag = false;

timerLeft = false;

timer = 0;

}

}

}

bool PensilDetector::GetLeftClick()

{

return leftClickFlag;

}

bool PensilDetector::GetRightClick()

{

return rightClickFlag;

}

bool PensilDetector::GetClickPreparation()

{

return clickPreparationFlag;

}

bool PensilDetector::GetWheelUp()

{

return wheelUpFlag;

}

bool PensilDetector::GetWheelDown()

{

return wheelDownFlag;

}

cv::Point PensilDetector::GetMousePosition()

{

return mousePosition;

}

ВЕДОМОСТЬ ДОКУМЕНТОВ

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Обозначение | | | | Наименование | | | | Дополнительные сведения | | | |
|  | | | | Текстовые документы | | | |  | | | |
|  | | | |  | | | |  | | | |
| БГУИР КР 1–40 01 01 618 ПЗ | | | | Пояснительная записка | | | | 44 с. | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | | Графические документы | | | |  | | | |
|  | | | |  | | | |  | | | |
| ГУИР 651006 18 СП | | | | Общая схема программы | | | | Формат А1 | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  |  |  |  |  | БГУИР КР 1-40 01 01 505 Д1 | | | | | | |
|  |  |  |  |  |
|  |  |  |  |  |
| Изм. | Л. | № докум. | Подп. | Дата | Pensil-Mouse Controller |  | | | | Лист | Листов |
| Разраб. | | Голуб А.В. |  | 04.06.18 | Т |  | |  | 44 | 44 |
| Пров. | | Низовцов Д.В. |  | 04.06.18 | Кафедра ПОИТ  гр. 651006 | | | | | |
|  | |  |  |  |
|  | |  |  |  |
|  | |  |  |  |