МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное

учреждение высшего образования

«Ярославский государственный университет им. П.Г.Демидова»

Кафедра компьютерной безопасности и

математических методов обработки информации

Курсовая работа

Разработка системы альтернативного управления компьютером.

Научный руководитель

Д-р наук, профессор

В.Г. Дурнев

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2017 г.

Студент группы КБ-51СО

С.М. Соловьев

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2017 г.

Оглавление

[Введение 3](#_Toc452993259)

[Введение в мир систем управления 4](#_Toc452993260)

[Типы управления 4](#_Toc452993261)

[*Управление голосом* 6](#_Toc452993262)

[*Управление «силой мысли»* 7](#_Toc452993263)

[*Управление глазами* 8](#_Toc452993264)

[*Управление языком* 10](#_Toc452993265)

[Разработка системы альтернативного управления ПК 11](#_Toc452993266)

[Введение 11](#_Toc452993267)

[Принцип работы 12](#_Toc452993268)

[*«Слежение» за объектом по цвету* 12](#_Toc452993269)

[*Имплементация курсора* 16](#_Toc452993270)

[*Взаимодействие курсора и ОС* 18](#_Toc452993271)

[Заключение 19](#_Toc452993272)

[Список литературы 20](#_Toc452993273)

[Приложение 21](#_Toc452993274)

[Листинг программы 21](#_Toc452993275)

# Введение

Придумать введение

Тезисно

Реалистично и востребовано ли слежение за глазом?

Кому понадобится

Какие цели и задачи решает приложение

Какова цена

Просто ли использовать

Человек на протяжении своей жизни, по разным данным, получает от 80 % до более 90% информации с помощью зрения. Все привыкли считать глаза неким «устройством» для получения информации, но мало кто задумывался о том, что глаза также могут передавать информацию.

Несколько лет назад такое предложение могло показаться странным, но теперь технологии шагнули далеко вперед, и это стало реальностью. С помощью глаз можно передавать информацию, куда смотрит человек, на чем сфокусирован его взгляд, на чем он сосредоточен.

Использовать данную технологию можно для обучения маленьких детей или детей с заторможенным развитием. На экране одновременно отображаются несколько изображений, обучаемого просят посмотреть на какое-то конкретно по разным характеристикам (цвет, вид, форма и т.д.). Также маркетологи могут измерять эффективность рекламы, демонстрируя ее людям и фиксировать координаты объекта внимания человека. Но, пожалуй, самая главное направление использования подобной технологии это помощь людям с ограниченными физическими возможностями пользоваться ПК для общения, саморазвития, работы или управления другими устройствами.

Безусловно, на рынке технологий существуют подобные системы, но цена их начинается от 10000 рублей. БЛАБЛАБЛА ТУТ ПРО ДЕШЕВИЗНУ И ВОЗМОЖНОСТЬ ОПРОБОВАТЬ ТЕХНОЛОГИЮ УПРАВЛЕНИЯ ГЛАЗАМИ ДОМА.

О ЧУДО, ЭТО ТАК ПРОСТО ИСПОЛЬЗОВАТЬ ЧТО ПРЯМ ЩАС СКАЧАЮ И ВОСПОЛЬЗУЮСЬ

Цель работы

1. Спроектировать приложение для управления ПК взглядом.
2. Разработать прототип приложения.
3. Спланировать дальнейшее развитие приложения.

# Введение в биологическую терминологию

## Введение

Используя ПК или мобильное устройство, человек устремляет взгляд на конкретные секторы экрана. Фиксацию взгляда на конкретном секторе будем называть фокусом взгляда. Для управления ПК требуется постоянный трекинг (слежение) фокуса взгляда. Для понимания физики процесса требуется информация о биологии и поведении глаза. Данный раздел предоставит некоторые базовые знания для создания системы глазного трекинга.

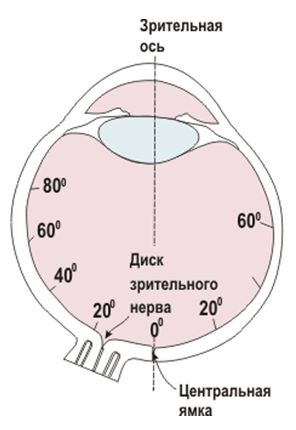
#### Структура человеческого глаза

Для того чтобы человек мог максимально четко рассмотреть объект, ему требуется сфокусировать на нем взгляд. Получение изображения происходит при попадании его через хрусталик на рецепторы глаза. Важным фактором при рассмотрении структуры глаза является то, где располагаются эти рецепторы. В [2] говорится, что если бы рецепторы были расположены равномерно внутри глаза, то была бы проблема понять, как происходит фокусирование. Вместо этого мы имеем различную плотность распределения рецепторов. Наиболее четкое изображение человек получает при попадании его в точку максимального скопления фоторецепторов-колбочек, и эта точка находится в желтой ямке (пятно фовеа).

Фиксация глаза

Но, нельзя забывать, что человеческий зрачок не может замереть на месте для получения стабильного изображения. В [1] используют понятие дрейфа. Дрейф представляет собой неупорядоченное и относительно медленное движение осей глаз, при котором для каждого глаза изображение точки фиксации остается внутри фовеа. Дрейф всегда сопровождается тремором – высоким по частоте, но очень маленьким по амплитуде колебательным движением осей глаз. Проще говоря, глаз даже при фокусировке изображения немного дергается, незаметно для самого человека.

Так как желтая ямка находится внутри глаза и имеет крайне малый размер (диаметр чуть больше 1мм), то фиксировать изменение ее положения не предоставляется возможным. Вместо этого можно отслеживать зрачок и через него выводить точку фокусировки, это становится возможным потому что, объект фокуса находится в центре глаза.



Локализация центральной ямки (0 градусов) и различных областей сетчатки по отношению к ней (в градусах эксцентриситета).

Перемещения глаза

Человеческий глаз может прибывать только в двух состояниях: фокусировка, сопровождающаяся дрейфом и перемещение (саккада).

Саккады (от старинного французского слова, переводимого как «хлопок [паруса](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D1%83%D1%81)») — быстрые, строго согласованные движения глаз, происходящие одновременно и в одном направлении. В работе [3] было обнаружено, что «трансаккадная пространственная память приносится в жертву, чтобы поддерживать перцептивную стабильность »[3]. Это означает, что во время саккадного движения визуальная информация не получается, чтобы сохранить восприятие гладкости и стабильности. Этот эффект известен как саккадическое подавление. Подобным эффектом обладает постоянное использование мыши в качестве интерфейса управления ПК, оно оттачивает навык до сглаженных, точных перемещений курсора в нужное положение.

Саккады важно отличать от других движений глаз, называемых Saccadic Eye Movements (SACs) в [5] (Преследование, вергентное и вестибулярное). Преследование – движения глаз «преследуют» точку фокусировки, перемещающийся в пространстве, выполняя саккады, чтобы догнать, если точка движется слишком быстро. Движение заметно медленнее, чем у саккад. Вергентные движения (лат. *Vergo* - склоняюсь) - [макродвижения глаз](http://dic.academic.ru/dic.nsf/psihologic/995), приводящие к изменению угла между зрительными осями левого и правого глаза. Вергентные движения – выступают фактором, обеспечивающим [бинокулярное зрение](http://dic.academic.ru/dic.nsf/psihologic/253). Они разделяются на движения конвергенции – сводящие глаза, дивергенции –разводящие глаза, циклофузионные, или [торсионные движения](http://dic.academic.ru/dic.nsf/psihologic/1898), меняющие ориентации сетчаток левого и правого глаз. Вестибулярные движения - это вращения глаза, чтобы компенсировать большие движения головы или тела (стабилизация изображения). Эти три движения менее важны, чем саккады для обработки информации, и как таковые для отслеживания зрачков.

# Методы поиска зрачка

Самые точные методы для обнаружения глаз включают в себя инвазивное или дорогостоящие оборудование. В [2] упоминается метод, который использует контактную линзу, с металлическим кольцом. Фиксация координат происходит с помощью измерения магнитного поля. В [6] в качестве системы фиксации изображения используют шлем с ИК камерами. Минусы данных систем заключаются в том, что они неудобные для использования, не мобильны.

Трудности, возникающие в процессе разработки подобных систем неизбежны. В настоящее время исследования в этой области направлены на создание чувствительного к пользователю и легкодоступного устройства. Идеальным решением для этого является бесконтактное и неинвазивное устройство. Но с такими устройствами возникают новые трудности, такие как, алгоритмы поиска головы, глаз, зрачка. Решения данных проблем будут рассмотрены далее.

# Инфракрасное отражение

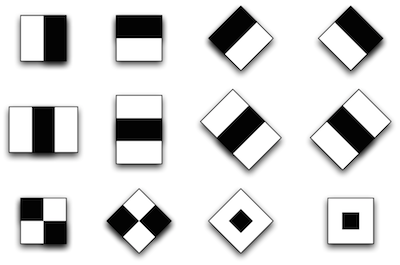
Принцип работы метода ИК отражения заключается в том, что они повышается контраст между зрачком и радужной оболочкой. Ниже приведен пример использования инфракрасного отражения для обнаружения глаз.

В [7] авторы сначала определяют область 60х60 пикселей (исходное разрешение монитора 1280 × 1024 пикселей) с ориентировочной оценкой центра зрачка. Поскольку разные части, глаза имеют разные показатели преломления инфракрасного света, зрачок кажется черным, а остальное нет. Использование простого порогового алгоритма для обработки, отфильтрованной серой шкалы изображения, полученного с камеры, проверяются все точки ниже для определения ROI. Пороговое уравнение, использованное в [7], приведено ниже.

Общая идея использования инфракрасного излучения заключается в том, что отражение инфракрасного излучения упрощает процесс определения центра зрачка. Таким образом, уменьшается сложность расчета и повышается точность обнаружения зрачков, повышается эффективность системы отслеживания глаз. Отрицательная сторона этого заключается в потребности инфракрасного света. Также могут быть другие источники света, или тени, которые нарушают это изображение и создают «шум», что усложняет вычисление, но это проблема актуальна для большинства ненавязчивых систем.

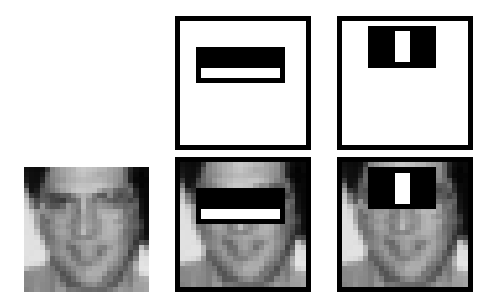
Каскад Хаара

Признаки Хаара — признаки цифрового изображения, используемые в [распознавании образов](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D1%80%D0%B0%D1%81%D0%BF%D0%BE%D0%B7%D0%BD%D0%B0%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F_%D0%BE%D0%B1%D1%80%D0%B0%D0%B7%D0%BE%D0%B2). Своим названием они обязаны интуитивным сходством с [вейвлетами Хаара](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B9%D0%B2%D0%BB%D0%B5%D1%82_%D0%A5%D0%B0%D0%B0%D1%80%D0%B0). Признаки Хаара использовались в первом детекторе лиц, работающем в реальном времени.

Обычно у каждого метода есть основа, то, без чего этот метод не мог бы существовать в принципе, а уже над этой основой строится вся остальная часть. В методе Виолы-Джонса эту основу составляют примитивы Хаара, представляющие собой разбивку заданной прямоугольной области на наборы разнотипных прямоугольных подобластей:  


В оригинальной версии алгоритма Виолы-Джонса использовались только примитивы без поворотов, а для вычисления значения признака сумма яркостей пикселей одной подобласти вычиталась из суммы яркостей другой подобласти [8]. В развитии метода были предложены примитивы с наклоном на 45 градусов и несимметричных конфигураций. Также вместо вычисления обычной разности, было предложено приписывать каждой подобласти определенный вес и значения признака вычислять как взвешенную сумму пикселей разнотипных областей [9]:

Основной причиной, почему в основу метода легли примитивы Хаара, являлась попытка уйти от пиксельного представления с сохранением скорости вычисления признака. Из значений пары пикселей сложно вынести какую-либо осмысленную информацию для классификации, в то время как из двух признаков Хаара строится, например, первый каскад системы по распознаванию лиц, который имеет вполне осмысленную интерпретацию [8]:

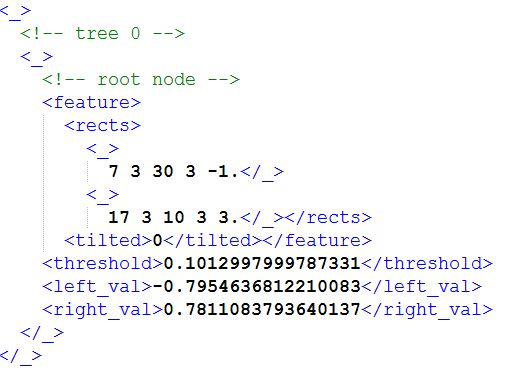


Сложность вычисления признака так же как и получения значения пикселя остается O(1): значение каждой подобласти можно вычислить скомбинировав 4 значения интегрального представления (Summed Area Table — SAT), которое в свою очередь можно построить заранее один раз для всего изображения за O(n), где n — число пикселей в изображении, используя формулу [9]:

Таким образом, создали быстрый алгоритм поиска объектов, который пользуется успехом уже больше десятилетия. Но вернемся к нашим признакам. Для определения принадлежности к классу в каждом каскаде, находится сумма значений слабых классификаторов этого каскада. Каждый слабый классификатор выдает два значения в зависимости от того больше или меньше заданного порога значение признака, принадлежащего этому классификатору. В конце сумма значений слабых классификаторов сравнивается с порогом каскада и выносится решение найден объект или нет данным каскадом.

Для разных видов объектов создаются разные XML классификаторы. Существуют базы и приложения с помощью которых можно получить XML классификатор объекта. Мы воспользуемся каскадом полученным студентами из Modesto Castrillon-Santana (IUSIANI, University of Las Palmas de Gran Canaria, Spain) в 2006 году. Лицензия открыта для некоммерческого использования.

Пример структуры описания:



Создается впечатление, что здесь куча непонятных цифр и странной информации, но на самом деле все просто:

* Дерево объектов — набор слабых классификаторов, на основе которых выносится решение о том, находится объект на изображении или нет,
* Два объекта в узле (без имени) — это параметры конкретного слабого классификатора.
  + Расшифровка первого объекта слева направо: первые два значения в нашем случае не используется, третье — номер признака в общей таблице признаков (она располагается дальше в XML), четвертое — пороговое значение слабого классификатора. Так как у нас используется классификатор, основанный на одноуровневых решающих деревьях (Decision Stump), то если значение признака Хаара меньше порога слабого классификатора (четвертое значение), выбирается значение left\_val, если больше —right\_val.

Kinect— это горизонтально расположенная коробка на небольшом круглом основании, которую помещают выше или ниже экрана. Размеры — примерно 23 см в длину и 4 см в высоту. Состоит из двух сенсоров глубины, цветной видеокамеры и микрофонной решетки. Проприетарное программное обеспечение осуществляет полное 3-х мерное распознавание движений тела, мимики лица и голоса. Микрофонная решетка позволяет Xbox 360 производить локализацию источника звука и подавление шумов, что дает возможность говорить без наушников и микрофона Xbox Live.

Датчик глубины состоит из инфракрасного проектора, объединенного с монохромной КМОП-матрицей, что позволяет датчику Kinect получать трёхмерное изображение при любом естественном освещении.

Диапазон глубины и программа проекта позволяет автоматически калибровать датчик с учётом условий игры и окружающих условий, например мебели, находящейся в комнат.

Судя по недавнему патенту Microsoft, Kinect будет способен распознавать язык жестов. Пока патент касается только ASL, но, возможно, другие языки будут добавлены позже. Ожидается, что это расширит аудиторию пользователей и поможет обучать немых языку жестов. Однако, согласно официальному комментарию, эта особенность не будет включена в первую версию Kinect из-за сниженного в угоду цене разрешения камер. С другой стороны, Microsoft не отказывается от использования патента — но будет ли это улучшенная версия Kinect или отдельный продукт, пока неизвестно.

### Управление голосом

Управление голосом также является обыденной возможностью на сегодняшний день. Практический каждый современный смартфон поддерживает технологию голосового управления. Автомобили оснащают такой функцией для безопасного управления транспортным средством. Так же данную технологию могут поддерживать абсолютно любые устройства, такие как чайник или газонокосилка. Существуют API (*интерфейс программирования приложений*) и SDK (*комплект средств разработки*) для разработки таких систем самостоятельно.

Примеры: Apple Siri, (Ok’ay Google) Google Speech, Volvo Cars Support.

**Siri** (*Сири*) (англ. *Speech Interpretation and Recognition Interface*) — персональный помощник и вопросно-ответная система, разработанная для IOS. Данное приложение использует обработку естественной речи, чтобы отвечать на вопросы и давать рекомендации. Siri приспосабливается к каждому пользователю индивидуально, изучая его предпочтения в течение долгого времени.

Первоначально Siri стало доступно в App Store как приложение для iOS от Siri Inc. Вскоре, Siri Inc. была приобретена Apple Inc. Но ещё до того, как Apple купила Siri, было объявлено, что их программное обеспечение будет доступно для телефонов BlackBerry и телефонов под управлением Android, затем эти планы были отменены из-за покупки.

Сейчас Siri — неотъемлемая часть IOS и доступна для большинства устройств, выпускаемых компанией: iPhone(4S и старше), iPad(третьего поколения и старше, а также все устройства линейки iPad mini),iPod touch и AppleWatch. Несмотря на это, хакеры смогли приспособить Siri для старых моделей устройств. Apple публично заявила, что у неё нет планов интеграции Siri в более старые продукты в связи с отсутствием в них чипа фильтрации фонового шума.

### Управление «силой мысли»

Нейрокомпьютерный интерфейс (НКИ) (интерфейс «мозг-компьютер») – система, созданная для обмена информацией между мозгом и электронным устройством (например, компьютером). В однонаправленных интерфейсах внешние устройства могут либо принимать сигналы от мозга, либо посылать ему сигналы (например, имитируя сетчатку глаза при восстановлении зрения электронным имплантатом). Двунаправленные интерфейсы позволяют мозгу и внешним устройствам обмениваться информацией в обоих направлениях. В основе нейро-компьютерного интерфейса, часто используется метод биологической обратной связи.

Примеры: NeuroSky MindWave, Brainflight.

Одним из основных элементов биологической обратной связи является электроэнцефалография (ЭЭГ) — метод наблюдения и регистрации электрических процессов в мозгу. Таким образом диагностируется склонность к эпилепсии и наличие опухолей мозга. Во время процедуры «снятия» ЭЭГ на голову пациента надевается прорезиненная шапочка с электродами и далее регистрируется электрическая активность мозга в разных состояниях: например, при восприятии интенсивно мигающего света. ЭЭГ также широко применяется в науке. Благодаря этому методу можно проверять гипотезы о типах мозговой активности при той или иной деятельности.

Ритмы мозга, регистрируемые методом ЭЭГ, могут с относительным успехом контролироваться человеком, а следовательно — могут быть использованы в интерфейсах «человек − машина». Какого-либо практического интереса это почти не вызывает, но есть и выдающиеся исключения из общей тенденции. Например, в феврале этого года был представлен проект Brainflight, в котором человек с помощью ЭЭГ управлял беспилотным летательным аппаратом. Правда, стоит отметить, что «пилот» прошел длительное обучение, в котором не обязательно преуспеет любой другой человек.

### Управление глазами

Существуют системы управление ПК и другими устройствами с помощью глаз. Специальный датчик «следит» за взглядом пользователя и передает информацию о перемещении взгляда на устройство, которое обрабатывает это событие и совершает определенные действия. Такая технология может показаться фантастикой, но уже сегодня обычный пользователь может позволить себе такую за не большую сумму (100 – 200 Евро).

Примеры: The Eye Tribe, Tobii EyeX

Датская компания The Eye Tribe представляет первое в мире коммерческое устройство, позволяющее осуществить управление настольным компьютером, планшетным компьютером или смартфоном с помощью движений глаз пользователя. Система Eye Tribe Tracker состоит из специального программного обеспечения и аппаратной части - небольшого устройства, совместимого с интерфейсом USB 3.0, что позволяет подключить его практически к любому современному цифровому электронному устройству. Вариант системы Eye Tribe Tracker, ориентированный на использование в среде операционной системы Windows, оценивается в 99 долларов, а для разработчиков программного обеспечения существует специализированный комплект разработчика (Software Development Kit), который позволяет значительно упростить интеграцию возможностей системы Eye Tribe Tracker в функции уже существующих и вновь создаваемых программ, включая и компьютерные игры.  
  
"Датчики нового поколения с высокой разрешающей способностью позволяют нам реализовать функции слежения за движениями глаз пользователя с точностью, вполне достаточной для управления элементами интерфейса практически любого электронного устройства, работающего под управление одной из популярных операционных систем. Возможности системы Eye Tribe Tracker сопоставимы с возможностями других подобных систем, но основным отличием нашей системы от других систем является ее низкая стоимость, делающая ее чрезвычайно привлекательной для пользователей" - рассказывает Мартин Тол (Martin Tall), соучредитель и один из руководителей компании Eye Tribe.

### Управление языком

Устройство «Альтернативная система управления компьютером без помощи рук» (Alternative Computer Control System, ACCS)  предназначено для помощи людям, находящимся в очень тяжелом состоянии, например, перенесшим травму спинного мозга с последующей утратой функций конечностей. ACCS размещается в ротовой полости человека и содержит модуль управления курсором при помощи движений языка, а также дополнительный модуль на 19 программируемых произвольных команд, облегчающих выполнение тех или иных действий. Основные компоненты ACCS: приемопередатчик, сменный ротовой модуль, программное обеспечение под специализированные задачи. С помощью ACSS полностью неподвижный человек может полноценно управлять компьютером, бытовой электроникой, своим средством передвижения, делать звонки по телефону и отвечать на них.

Сейчас стартап работает над унифицированной платформой, призванной решить задачи по эргономически-продуманной организации рабочих мест глобально и ситуативно. Платформа представляет из себя комплекс систем, подсистем, узлов, навесного и периферийного оборудования, объединенный в единой системе. Основной отличительной особенностью системы является её максимальная эффективность в эргономическом и ортопедическом понимании по отношению к пользователю: все модификации системы помимо основного рабочего положения пользователя предусматривают положение «отдыха», в котором пользователь может расслабиться или даже поспать в перерыве. Данная особенность способна повысить рабочую активность пользователя и сохранить ему здоровье.

# Разработка системы альтернативного управления ПК

## Введение

Для того что бы определиться с типом управления нужно оценить стоимость разработки, целевую аудиторию, технические возможности.

В качестве целевой аудитории выберем пользователей обычного компьютера, не обремененных ограничениями физической активности, так как иначе разработка системы потребует более высокой квалификации разработчика.

В качестве дополнительного оборудования, потребуется веб-камера. С помощью нее будет осуществляться управление ПК.

## Принцип работы

### «Слежение» за объектом по цвету

Веб-камера постоянно захватывает видеоизображение пользователя. На изображение накладываются цветовые фильтры для выделения на изображении объекта конкретного цвета (лучше всего использовать редкие яркие цвета для более точного определения объекта). Программа выделяет объект прямоугольником и начинается «слежение» за объектом. Накладывая на текущий кадр предыдущий, при движении объекта в кадре, можно заметить изменение координат прямоугольника выделяющего объект. Разница этих координат и будет перемещение объекта за 1 кадр.

Для реализации данного функционала на C# используем фреймворк AForge.net. Фреймворк представляет собой набор библиотек, каждая из которых предназначена для решения определенного рода задач:

* **AForge.Imaging** – библиотека с фильтрами и расчетами для обработки изображений;
* **AForge.Vision** – библиотека машинного зрения;
* **AForge.Neuro** – библиотека для работы с нейронными сетями;
* **AForge.Geneti**c – библиотека для работы с генетическими алгоритмами;
* **AForge.Fuzzy** – библиотека нечетких вычислений;
* **AForge.MachineLearning** – библиотека для машинного обучения;
* **AForge.Robotics** – библиотека, предоставляющая подержу некоторых Robotics kits;
* **AForge.Video** – набор библиотек для обработки видео.

AForge.Vision.Motion.IMotionDetector  – интерфейс, позволяющий искать разницу между изображениями. От него унаследован *ColorDetection* класс, выполняющий обработку.  
Для взаимодействия с интерфейсом пользователя был добавлен метод *Initialize*(Image image, Rectangle rect), который инициализирует процесс обработки последующих кадров. Здесь происходит сбор информации о целевом объекте (выделенным прямоугольником на изображении). Собирается информация о доминирующем цвете в избранной области (это свойство и будет в дальнейшем служить основой для слежения). Также запоминается позиция целевого объекта.

IMotionDetector имеет следующие методы:

* *ProcessFrame*(AForge.Imaging.UnmanagedImage) – передает в объект следующий кадр, полученный от устройства захвата изображений. Следующий кадр – объект типа AForge.Imaging.UnmanagedImage, класса, позволяющего подступаться к пикселам намного быстрее, чем *Bitmap.GetPixel(x,y)*.
* *Reset*() – сбрасывает содержимое класса.

Свойства:

* AForge.Vision.Motion.IMotionDetector.*MotionFrame*– свойство типа AForge.Imaging.*UnmanagedImage*, которое отвечает за подсветку региона с объектом.
* AForge.Vision.Motion.IMotionDetector.*MotionLevel*– «уровень движения» (от 0 до 1) – эфимерная величина. Не имплементировал.

Для того, чтобы использовалось не только целевой цвет, но и некоторые оттенки, используется Set Property *DifferenceThreshold*.  
Основная обработка кадра происходит в функции *ProcessFrame*. Алгоритм можно разделить на такие шаги:

1. Наложение на изображение (текущий кадр) фильтра указанного в окне настройки фильтра (рис 1).
2. Расширение региона присутствия объекта. Новое положение будем искать не по всему экрану, а лишь в области, смежной с предыдущим положением. Это делает поиск точнее с той точки зрения, что целевой объект не будет перепутан с другим объектом такого же цвета (в другой части изображения).
3. Вычисление границ объекта в вышеописанной области через определение крайних точек того цвета, который является доминирующим для объекта (здесь также учитывается возможное цветовое отклонение — *DifferenceThreshold*).
4. Создание «маски» *MotionFrame*, которая позволит MotionDetector-у подсветить целевой объект на изображении.
5. Далее вычисляется «средний цвет» и размер нового объекта. Если объект слишком мал (например, на следующем кадре наш целевой объект полностью был закрыт другим объектом) – мы не меняем информации о положении и цвете, которые остались в наследство от обработки предыдущего кадра. Иначе – запоминается новое положение и границы объекта, и, если алгоритм следит за изменением цвета, что устанавливается с помощью свойства *bool DynamicColorTracking*, то также запоминается и новый вычисленный цвет.

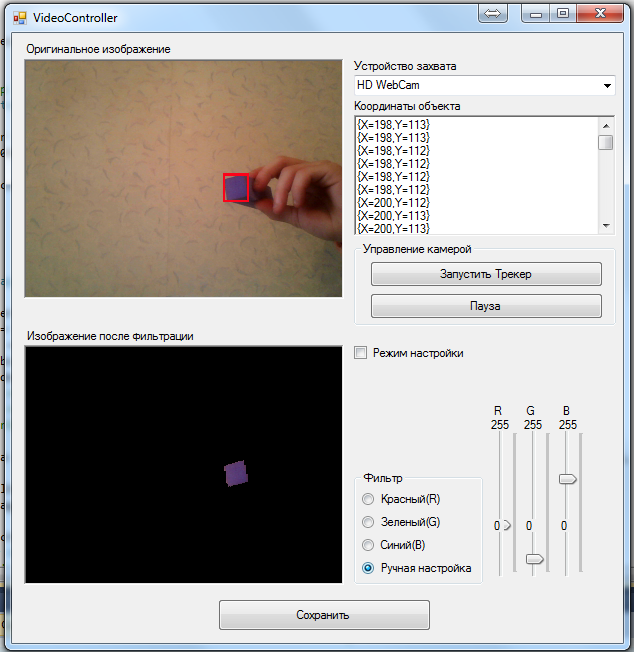


Рис 1.

Координаты нового положения объекта фиксируются в журнал координат и передаются в другой модуль для перемещения курсора.

### Имплементация курсора

Для работы с курсором используется встроенная библиотека Windows *user32.dll.* Курсор может перемещаться по заданным координатам, с помощью встроенной функции *SetCursorPos*. Так же совершать глобальные действия «Нажатие левой кнопки мыши», «Отпускание левой кнопки мыши», «Нажатие правой кнопки мыши», «Отпускание левой кнопки мыши». Для этого задаются байт коды операций:

* MOUSEEVENTF\_LEFTDOWN = 0x02,
* MOUSEEVENTF\_LEFTUP = 0x04,
* MOUSEEVENTF\_RIGHTDOWN = 0x08,
* MOUSEEVENTF\_RIGHTUP = 0x10

Создаются скрытые кнопки для управления действиями курсора. При наведении курсора на верхнюю часть экрана отображается панель с кнопками выбора действия. Возможные варианты действий:

* Левый клик мыши,
* Двойной левый клик мыши,
* Правый клик мыши,
* Перемещение,
* Клавиатура

Левый клик мыши вызывает два события курсора следующие друг за другом «Нажатие левой кнопки мыши» и «Отпускание левой кнопки мыши».

Двойной левый клик вызывает два раза подряд событие левого клика мыши.

Правый клик вызывает два события курсора следующие друг за другом «Нажатие правой кнопки мыши» и «Отпускание левой кнопки мыши» для вызова контекстного меню, потом вызывается событие левого клика мыши для выбора пункта меню.

Перемещение вызывает событие курсора «Нажатие левой кнопки мыши» для перемещения объекта, потом вызывается событие курсора «Отпускание левой кнопки мыши» для закрепления положения объекта.

Клавиатура вызывает интерфейс стандартной экранной клавиатуры от Windows *osr.exe* с помощью которой курсором может осуществляться ввод текстовой информации.

Встроенная библиотека Windows *user32.dll* позволяет полностью имитировать все действия курсора обычной мыши.

### Взаимодействие курсора и ОС

Так как на кнопки управления курсором нажать возможности нет используем события наведения курсора на объект. При наведении курсора на кнопку панели управления запускается таймер с интервалом в 3000 мс. По истечению данного таймера выбирается действие которое будет совершать курсор и таймер вновь запускается для совершения действия.

Для более простого взаимодействия с пользователем создана форма *CustomCursorForm* которая отображается при выборе действия и при наличии активного выбранного действия. Данная форма отображает *ProgressBar* под курсором мыши дающий понять сколько времени осталось до выбора или совершения действия.

Для удобной отладки программы так же создан механизм логирования действий курсора, нажатых клавиш экранной клавиатуры и координат объекта на изображении полученном с камеры.

# Заключение

В данной работе рассмотрены основные виды систем альтернативного управления ПК. Проведен обзор лидеров в своем классе.

Разработана собственная система управления ПК основанная на слежении за объектом по цвету. Изучены механизмы подключения и использования веб-камеры в качестве «трекера» на языке программирования C#. Реализован механизм взаимодействия с курсом операционной системы через приложение и механизм логирования действий пользователя.

В дальнейшем планируется более подробное изучение технологии слежения за глазом и реализация собственного устройства слежения основанного на ИК камере и алгоритмах вычисления отражения луча в зрачке. Так же планируется создания собственного интерфейса взаимодействия пользователя с физическими ограничениями и ПК, используя технологию слежения за глазом.

# Список литературы

**ОФОРМИ ПРАВИЛЬНО**

1. А.Л. Ярбус Академия наук СССР «Роль движений глаз в процессе зрения». Москва 1996г.
2. Jacob, R. J. K. Eye tracking in advanced interface design, 1995.
3. Klingenhoefer, S., and Bremmer, F. Saccadic suppression of displacement in face of saccade adaptation. Vision Research 51, 8 (2011), 881 – 889. Perception and Action: Part II.
4. Гиппенрейтер Ю. Б. Движения человеческого глаза / Ю. Б. Гиппенрейтер.
5. Grossberg, S., Srihasam, K., and Bullock, D. Neural dynamics of saccadic and smooth pursuit eye movement coordination during visual tracking of unpredictably moving targets. Neural Networks 27, 0 (2012), 1 – 20
6. Mouse cursor control with head and eye movements: A low-cost approach by Yat-Sing Yeung, BSc. 1220 Vienna, 2012
7. Gao, D., Yin, G., Cheng, W., and Feng, X. Non-invasive eye tracking technology based on corneal reflex. Procedia Engineering 29, 0 (2012), 3608 – 3612. 2012 International Workshop on Information and Electronics Engineering.
8. P. Viola and M. Jones. Robust real-time face detection. IJCV 57(2), 2004
9. Lienhart R., Kuranov E., Pisarevsky V.: Empirical analysis of detection cascades of boosted classifiers for rapid object detection. In: PRS 2003, pp. 297-304 (2003)
10. СТАРЫЙ
11. Портал разработчиков Microsoft https://msdn.microsoft.com/
12. Всероссийский ресурс для IT-специалистов <https://habrahabr.ru/>
13. Сайт производителя Tobii <http://www.tobii.com/>
14. Сайт производителя NeuroSky http://www.neurosky.com/
15. Сайт производителя ACCS http://gravitonus.com/
16. Документация по AForge.Net http://www.aforgenet.com/

# Приложение

## Листинг программы

#### *MainFormControl*

Код главной формы *MainFormControl*, осуществляющей взаимодействие программы и курсора, вызов вспомогательных форм для управления ПК.

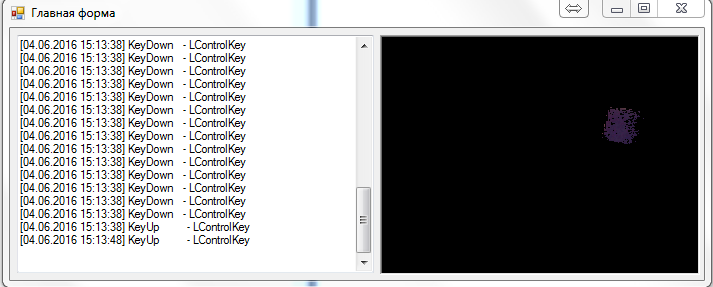
Форма содержит в себе текстовое поле для логирования действий и изображение после фильтрации рис 2.

Рис 2.

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using gma.System.Windows;

using System.Runtime.InteropServices;

namespace EyeWindowsController

{

public partial class MainFormControl : Form

{

public int timeInterval {set {} get { return 3000;} }

UserActivityHook actHook;

CustomCursorForm mouse\_form;

int timing;

CustomMouse \_CM;

Timer \_timer, \_progress\_timer;

public string MOUSEEVENTACTION = "LeftClick";

ControlPanelForm \_controlsForm;

bool ActionIsEnd; //Действие совершено

VideoController VCForm;

public MainFormControl()

{

TopMost = true;

this.Hide();

\_CM = new CustomMouse();

\_controlsForm = new ControlPanelForm();

\_controlsForm.Owner = this;

\_controlsForm.setOwner();

\_timer = new Timer();

\_progress\_timer = new Timer();

\_progress\_timer.Interval = (int) timeInterval/12;

\_progress\_timer.Tick += PrintProgress;

\_progress\_timer.Enabled = true;

InitializeComponent();

mouse\_form = new CustomCursorForm();

mouse\_form.Owner = this;

VCForm = new VideoController();

VCForm.Owner = this;

VCForm.Show();

// mouse\_form.Show();

}

[DllImport("user32.dll", CharSet = CharSet.Auto, CallingConvention = CallingConvention.StdCall)]

public static extern void mouse\_event(int dwFlags, int dx, int dy, int cButtons, int dwExtraInfo);

[DllImport("user32.dll", SetLastError = true)]

static extern bool SetCursorPos(int X, int Y);

public void MouseMoved(object sender, MouseEventArgs e)

{

}

public void setMouseCoordinates(int x, int y)

{

SetCursorPos(x, y);

Point tempPoint = new Point(x, y + 15);

mouse\_form.DesktopLocation = tempPoint;

if (\_CM.MouseMoved(x, y))

{

refreshEventHandlers();

// LogWrite("Обновили таймеры");

}

if (y < \_controlsForm.Height && x > (SystemInformation.PrimaryMonitorSize.Width - \_controlsForm.Width) / 2 && x < (SystemInformation.PrimaryMonitorSize.Width + \_controlsForm.Width) / 2)

{

if (!\_controlsForm.IsShowed)

{

\_controlsForm.Show();

mouse\_form.Show();

\_controlsForm.IsShowed = true;

LogWrite("Показываем форму ");

}

}

else

{

if (\_controlsForm.IsShowed)

{

\_controlsForm.Hide();

\_controlsForm.IsShowed = false;

LogWrite("Скрываем форму ");

}

}

}

public void MyKeyDown(object sender, KeyEventArgs e)

{

LogWrite("KeyDown - " + e.KeyData.ToString());

}

enum MouseEvent

{

MOUSEEVENTF\_LEFTDOWN = 0x02,

MOUSEEVENTF\_LEFTUP = 0x04,

MOUSEEVENTF\_RIGHTDOWN = 0x08,

MOUSEEVENTF\_RIGHTUP = 0x10,

}

public void MyKeyPress(object sender, KeyPressEventArgs e)

{

LogWrite("KeyPress - " + e.KeyChar);

}

public void MyKeyUp(object sender, KeyEventArgs e)

{

LogWrite("KeyUp - " + e.KeyData.ToString());

}

public void LogWrite(string txt)

{

logger\_tb.AppendText("[" + DateTime.Now.ToString() + "] "+ txt + Environment.NewLine);

logger\_tb.SelectionStart = logger\_tb.Text.Length;

}

private void MainFormControl\_Load(object sender, EventArgs e)

{

actHook = new UserActivityHook(); // crate an instance with global hooks

// hang on events

actHook.OnMouseActivity += new MouseEventHandler(MouseMoved);

actHook.KeyDown += new KeyEventHandler(MyKeyDown);

actHook.KeyPress += new KeyPressEventHandler(MyKeyPress);

actHook.KeyUp += new KeyEventHandler(MyKeyUp);

actHook.Start();

}

private void DoClick(object sender, EventArgs e)

{

\_CM.DoClick();

setActionEnded(sender, e);

}

private void DoDBLClick(object sender, EventArgs e)

{

\_CM.DoDBLClick();

setActionEnded(sender, e);

}

private void DoRightClick(object sender, EventArgs e)

{

\_CM.DoRightClick();

\_timer.Tick -= DoRightClick;

\_timer.Tick += DoClick;

}

private void DoMove(object sender, EventArgs e)

{

\_CM.DoMoveLeftMouse();

\_timer.Tick -= DoMove;

\_timer.Tick += DoMoveRightMouse;

}

private void DoMoveRightMouse(object sender, EventArgs e)

{

\_CM.DoMoveRightMouse();

\_timer.Tick -= DoMoveRightMouse;

setActionEnded(sender, e);

}

private void PrintProgress(object sender, EventArgs e)

{

mouse\_form.PrintProgress();

}

/// <summary>

/// Скролл

/// </summary>

/// <param name="side">1 - Вверх, 0 - Вниз</param>

public void DoScroll(int side)

{

}

public void refreshEventHandlers()

{

mouse\_form.ClearProgress();

if (ActionIsEnd)

{

ActionIsEnd = false;

LogWrite("Остановили таймеры");

mouse\_form.ClearProgress();

mouse\_form.Hide();

\_progress\_timer.Stop();

MOUSEEVENTACTION = "";

\_timer.Stop();

\_timer.Tick -= DoClick;

\_timer.Tick -= DoDBLClick;

\_timer.Tick -= DoRightClick;

\_timer.Tick -= setActionEnded;

}

else

{

\_timer.Stop();

\_timer.Start();

// \_progress\_timer.Stop();

// \_progress\_timer.Start();

mouse\_form.ClearProgress();

}

}

public void startMouseTimer()

{

//refreshEventHandlers();

LogWrite("Запуск таймеров в главной форме c событием " + MOUSEEVENTACTION);

\_timer.Dispose();

\_timer = new Timer();

\_timer.Interval = timeInterval;

switch (MOUSEEVENTACTION)

{

case "LeftClick":

{

\_timer.Tick += DoClick;

}

break;

case "DBLLeftClick":

{

\_timer.Tick += DoDBLClick;

}

break;

case "RightClick":

{

\_timer.Tick += DoRightClick;

}

break;

case "Move":

{

\_timer.Tick += DoMove;

}

break;

}

//\_timer.Tick += setActionEnded;

\_timer.Enabled = true;

\_timer.Start();

mouse\_form.ClearProgress();

mouse\_form.Show();

\_progress\_timer.Start();

}

public void setActionEnded(object sender, EventArgs e)

{

ActionIsEnd = true;

refreshEventHandlers();

}

public void printControlsForm()

{

}

private void MainFormControl\_FormClosed(object sender, FormClosedEventArgs e)

{

Application.Exit();

}

}

}

#### VideoController

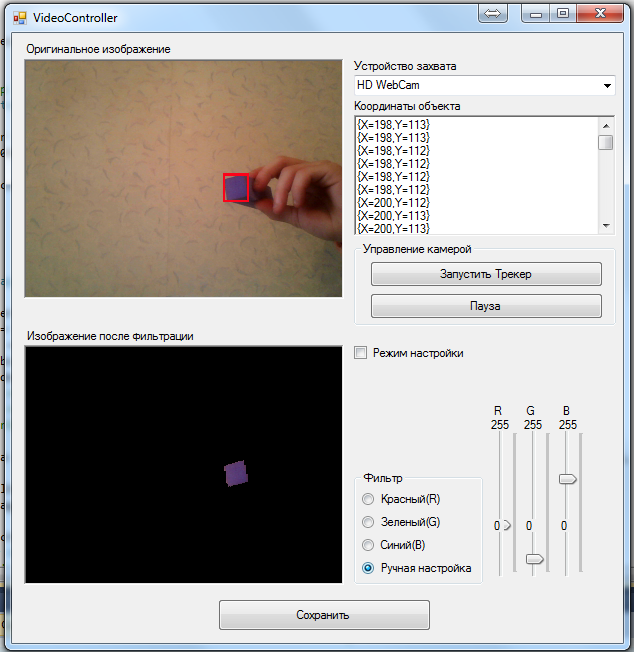
Форма с настройкой трекинга объекта по цвету. В данной форме присутствует оригинальное изображение, изображение после фильтрации, кнопки управления процессом трекинга и настройки фильтрации.

Рис 3.

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Drawing.Imaging;

using System.Windows.Forms;

using AForge;

using AForge.Imaging.Filters;

using AForge.Imaging;

using AForge.Video;

using AForge.Video.DirectShow;

using AForge.Math.Geometry;

using System.IO;

using System.Runtime.Serialization.Formatters;

using System.Xml.Serialization;

//Remove ambiguousness between AForge.Image and System.Drawing.Image

using Point = System.Drawing.Point; //Remove ambiguousness between AForge.Point and System.Drawing.Point

namespace EyeWindowsController

{

public partial class VideoController : Form

{

private FilterInfoCollection VideoCapTureDevices;

private VideoCaptureDevice Finalvideo;

private ControllerSettings settings;

MainFormControl owner\_form;

int ScreenWidth = SystemInformation.PrimaryMonitorSize.Width;

int ScreenHeight = SystemInformation.PrimaryMonitorSize.Height;

public VideoController()

{

InitializeComponent();

}

int R; //Trackbarın değişkeneleri

int G;

int B;

int H;

int W;

private void VideoController\_Load(object sender, EventArgs e)

{

VideoCapTureDevices = new FilterInfoCollection(FilterCategory.VideoInputDevice);

foreach (FilterInfo VideoCaptureDevice in VideoCapTureDevices)

{

comboBox1.Items.Add(VideoCaptureDevice.Name);

}

comboBox1.SelectedIndex = 0;

try

{

owner\_form = (MainFormControl)this.Owner as MainFormControl;

deserializeSettings();

StartTracking();

}

catch (Exception exce)

{

}

H = ScreenHeight / pictureBox1.Height + 3;

W = ScreenWidth / pictureBox1.Width;

}

private void StartTracking()

{

if ( Finalvideo != null )

Finalvideo.Stop();

Finalvideo = new VideoCaptureDevice(VideoCapTureDevices[comboBox1.SelectedIndex].MonikerString);

Finalvideo.NewFrame -= Finalvideo\_NewFrame;

Finalvideo.NewFrame += new NewFrameEventHandler(Finalvideo\_NewFrame);

Finalvideo.DesiredFrameRate = 20;// FPS

Finalvideo.DesiredFrameSize = new Size(320, 240);

Finalvideo.Start();

}

private void StartSettingButton\_Click(object sender, EventArgs e)

{

StartTracking();

}

void Finalvideo\_NewFrame(object sender, NewFrameEventArgs eventArgs)

{

Bitmap image = (Bitmap)eventArgs.Frame.Clone();

Bitmap image1 = (Bitmap)eventArgs.Frame.Clone();

pictureBox1.Image = image;

if (rdiobtnRed.Checked)

{

//создадим фильтр

EuclideanColorFiltering filter = new EuclideanColorFiltering();

//Задаем фильтр

filter.CenterColor = new RGB(Color.FromArgb(215, 0, 0));

filter.Radius = 100;

//Применим фильтр

filter.ApplyInPlace(image1);

Tracking(image1);

}

if (rdiobtnBlue.Checked)

{

//создадим фильтр

EuclideanColorFiltering filter = new EuclideanColorFiltering();

//Задаем фильтр

filter.CenterColor = new RGB(Color.FromArgb(30, 144, 255));

filter.Radius = 100;

//Применим фильтр

filter.ApplyInPlace(image1);

Tracking(image1);

}

if(rdiobtnGreen.Checked){

//создадим фильтр

EuclideanColorFiltering filter = new EuclideanColorFiltering();

// set center color and radius

filter.CenterColor = new RGB(Color.FromArgb(0, 215, 0));

filter.Radius = 100;

//Применим фильтр

filter.ApplyInPlace(image1);

Tracking(image1);

}

if (rdiobtnHandSet.Checked)

{

//создадим фильтр

EuclideanColorFiltering filter = new EuclideanColorFiltering();

//Задаем фильтр

filter.CenterColor = new RGB(Color.FromArgb(R, G, B));

filter.Radius = 100;

//Применим фильтр

filter.ApplyInPlace(image1);

Tracking(image1);

}

}

public void Tracking(Bitmap image)

{

BlobCounter blobCounter = new BlobCounter();

blobCounter.MinWidth = 5;

blobCounter.MinHeight = 5;

blobCounter.FilterBlobs = true;

blobCounter.ObjectsOrder = ObjectsOrder.Size;

//Grayscale griFiltre = new Grayscale(0.2125, 0.7154, 0.0721);

//Grayscale griFiltre = new Grayscale(0.2, 0.2, 0.2);

//Bitmap griImage = griFiltre.Apply(image);

BitmapData objectsData = image.LockBits(new Rectangle(0, 0, image.Width, image.Height), ImageLockMode.ReadOnly, image.PixelFormat);

// grayscaling

Grayscale grayscaleFilter = new Grayscale(0.2125, 0.7154, 0.0721);

UnmanagedImage grayImage = grayscaleFilter.Apply(new UnmanagedImage(objectsData));

// unlock image

image.UnlockBits(objectsData);

Bitmap newClone = new Bitmap(image);

//newClone = (Bitmap)image.Clone();

blobCounter.ProcessImage(image);

Rectangle[] rects = blobCounter.GetObjectsRectangles();

Blob[] blobs = blobCounter.GetObjectsInformation();

pictureBox2.Image = image;

owner\_form.videoTranslator.Image = newClone;

if (!chkBoxSettingsMode.Checked)

{

//Трекинг за одним объектом

foreach (Rectangle recs in rects)

{

if (rects.Length > 0)

{

Rectangle objectRect = rects[0];

//Graphics g = Graphics.FromImage(image);

Graphics g = pictureBox1.CreateGraphics();

using (Pen pen = new Pen(Color.FromArgb(252, 3, 26), 2))

{

g.DrawRectangle(pen, objectRect);

}

//Координаты прямоугольника

int objectX = objectRect.X + (objectRect.Width / 2);

int objectY = objectRect.Y + (objectRect.Height / 2);

// g.DrawString(objectX.ToString() + "X" + objectY.ToString(), new Font("Arial", 12), Brushes.Red, new System.Drawing.Point(250, 1));

g.Dispose();

this.Invoke((MethodInvoker)delegate

{

richTextBox1.Text = objectRect.Location.ToString() + "\n" + richTextBox1.Text + "\n";

try

{

owner\_form.setMouseCoordinates(ScreenWidth - objectRect.Location.X \* W, objectRect.Location.Y \* H);

}

catch (Exception except)

{ }

});

}

}

}

}

// Conver list of AForge.NET's points to array of .NET points

private Point[] ToPointsArray(List<IntPoint> points)

{

Point[] array = new Point[points.Count];

for (int i = 0, n = points.Count; i < n; i++)

{

array[i] = new Point(points[i].X, points[i].Y);

}

return array;

}

private void PauseSettingButton\_Click(object sender, EventArgs e)

{

if (Finalvideo.IsRunning)

{

Finalvideo.Stop();

}

}

private void trackBar1\_Scroll(object sender, EventArgs e)

{

R = trackBar1.Value;

}

private void trackBar2\_Scroll(object sender, EventArgs e)

{

G = trackBar2.Value;

}

private void trackBar3\_Scroll(object sender, EventArgs e)

{

B = trackBar3.Value;

}

private void saveSettingsButton\_Click(object sender, EventArgs e)

{

settings = new ControllerSettings();

settings.Red = R;

settings.Green = G;

settings.Blue = B;

if (rdiobtnRed.Checked)

settings.CheckSettings = "Red";

if (rdiobtnGreen.Checked)

settings.CheckSettings = "Green";

if (rdiobtnBlue.Checked)

settings.CheckSettings = "Blue";

if (rdiobtnHandSet.Checked)

settings.CheckSettings = "HandSet";

using (Stream fStream = new FileStream("XMLSettings.xml",

FileMode.Create, FileAccess.Write, FileShare.None))

{

XmlSerializer xmlFormat = new XmlSerializer(typeof(ControllerSettings));

xmlFormat.Serialize(fStream, settings );

}

}

public void deserializeSettings()

{

FileStream fs = new FileStream("XMLSettings.xml", FileMode.Open);

try

{

XmlSerializer xml = new XmlSerializer(typeof(ControllerSettings));

settings = (ControllerSettings)xml.Deserialize(fs);

R = settings.Red;

G = settings.Green;

B = settings.Blue;

rdiobtnRed.Checked = false;

rdiobtnGreen.Checked = false;

rdiobtnBlue.Checked = false;

rdiobtnHandSet.Checked = false;

if (settings.CheckSettings == "Red")

rdiobtnRed.Checked = true;

if (settings.CheckSettings == "Green")

rdiobtnGreen.Checked = true;

if (settings.CheckSettings == "Blue")

rdiobtnBlue.Checked = true;

if (settings.CheckSettings == "HandSet")

{

rdiobtnHandSet.Checked = true;

trackBar1.Value = settings.Red;

trackBar2.Value = settings.Green;

trackBar3.Value = settings.Blue;

}

}

catch (Exception exc)

{

MessageBox.Show(exc.Message);

}

finally

{

fs.Close();

}

}

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

StartTracking();

}

}

}

#### ControlPanelForm

Форма для выбора действия. Отображается поверх всех окон при наведении мыши в верхнюю часть экрана. Для выбора действия нужно задержать курсор на 3 сек.

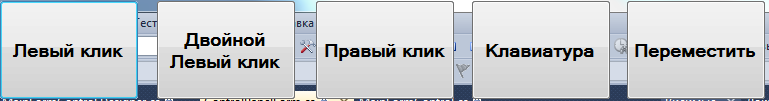


Рис 4.

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.Diagnostics;

using System.IO;

using System.Runtime.InteropServices;

namespace EyeWindowsController

{

public partial class ControlPanelForm : Form

{

int HEIGHT = SystemInformation.PrimaryMonitorSize.Height / 10;

int WIDTH = SystemInformation.PrimaryMonitorSize.Width/ 2;

public bool IsShowed = false;

public ControlPanelForm()

{

InitializeComponent();

this.Width = WIDTH;

this.Height = Height;

this.Left = SystemInformation.PrimaryMonitorSize.Width / 2 - WIDTH / 2;

this.Top = 0;

this.FormBorderStyle = FormBorderStyle.None;

this.AllowTransparency = true;

this.BackColor = Color.AliceBlue;//цвет фона

this.TransparencyKey = this.BackColor;//он же будет заменен на прозрачный цвет

}

MainFormControl main;

Timer \_timer;

bool ActionIsSelected;

public void setOwner()

{

main = (MainFormControl)this.Owner as MainFormControl;

}

private void leftMouseClickBTN\_MouseEnter(object sender, EventArgs e)

{

SetClick();

}

public void refreshTimer()

{

if (!ActionIsSelected)

{

main.MOUSEEVENTACTION = "";

main.setActionEnded(null, null);

}

}

private void DblLeftMouseClickBTN\_MouseEnter(object sender, EventArgs e)

{

SetClick();

}

private void DblLeftMouseClickBTN\_MouseLeave(object sender, EventArgs e)

{

refreshTimer();

}

private void leftMouseClickBTN\_MouseLeave(object sender, EventArgs e)

{

refreshTimer();

}

private void RightMouseClickBTN\_MouseLeave(object sender, EventArgs e)

{

refreshTimer();

}

private void keyboard\_btn\_MouseLeave(object sender, EventArgs e)

{

refreshTimer();

}

private void move\_btn\_MouseLeave(object sender, EventArgs e)

{

refreshTimer();

}

private void SetClick()

{

main.MOUSEEVENTACTION = "LeftClick";

main.startMouseTimer();

ActionIsSelected = false;

}

private void leftMouseClickBTN\_Click(object sender, EventArgs e)

{

ActionIsSelected = true;

main.MOUSEEVENTACTION = "LeftClick";

main.startMouseTimer();

this.Hide();

}

private void DblLeftMouseClickBTN\_Click(object sender, EventArgs e)

{

ActionIsSelected = true;

main.MOUSEEVENTACTION = "DBLLeftClick";

main.startMouseTimer();

this.Hide();

}

private void RightMouseClickBTN\_Click(object sender, EventArgs e)

{

ActionIsSelected = true;

main.MOUSEEVENTACTION = "RightClick";

main.startMouseTimer();

this.Hide();

}

private void keyboard\_btn\_Click(object sender, EventArgs e)

{

try

{

if (!Process.GetProcessesByName("osk").Any())

{

var path64 = @"C:\Windows\winsxs\amd64\_microsoft-windows-osk\_31bf3856ad364e35\_6.1.7600.16385\_none\_06b1c513739fb828\osk.exe";

var path32 = @"C:\windows\system32\osk.exe";

var path = (Environment.Is64BitOperatingSystem) ? path64 : path32;

Process.Start(path);

}

}

catch (Exception exc)

{

main.LogWrite("Ошибка запуска Клавиатуры " + exc);

}

}

private void move\_btn\_Click(object sender, EventArgs e)

{

ActionIsSelected = true;

main.MOUSEEVENTACTION = "Move";

main.startMouseTimer();

this.Hide();

}

private void ControlPanelForm\_Load(object sender, EventArgs e)

{

}

}

}