ОГЛАВЛЕНИЕ

Н. С. Афанасьев
ПРОГРАММНЫЙ ФРЕЙМВОРК ДЛЯ РЕАЛИЗАЦИИ ВЗАИМОДЕЙСТВИЯ С ПОЛЬЗОВАТЕЛЬСКИМИ ИНТЕРФЕЙСАМИ IOS-ПРИЛОЖЕНИЙ НА ОСНОВЕ ОКУЛОГРАФИИ 198–245

Р. В. Мосолов
ТЕХНОЛОГИЧЕСКИЙ ЦИКЛ РАЗРАБОТКИ ПОИСКОВОЙ СИСТЕМЫ, АГРЕГИРУЮЩЕЙ ЦИТАТЫ ИЗ КНИГ 246–256

Р. Р. Шигапов, А. А. Ференец
РЕКОМЕНДАТЕЛЬНАЯ СИСТЕМА ПОДБОРА ИГРОКОВ В КОМАНДНЫХ ВИДАХ СПОРТА, ПОСТРОЕННАЯ НА ОСНОВЕ МАШИННОГО ОБУЧЕНИЯ 257–280

УДК 004.5, 004.415

ПРОГРАММНЫЙ ФРЕЙМВОРК ДЛЯ РЕАЛИЗАЦИИ ВЗАИМОДЕЙСТВИЯ С ПОЛЬЗОВАТЕЛЬСКИМИ ИНТЕРФЕЙСАМИ IOS-ПРИЛОЖЕНИЙ НА ОСНОВЕ ОКУЛОГРАФИИ

Н. С. Афанасьев[1] **[0000-0002-5306-9672]**

Институт информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета
ул. Кремлевская, 35, г. Казань, 420008

[1]reqenoxus@gmail.com

Аннотация

Использование технологий отслеживания взгляда для взаимодействия с пользовательским интерфейсом iOS-приложений существенно затруднено отсутствием унифицированного подхода к их интеграции. Существующие решения либо жестко ограничены своей предметной областью, либо представляют собой исключительно исследовательские проекты, непригодные для решения прикладных задач. В статье рассматривается создание фреймворка, осуществляющего отслеживание взгляда пользователя на экране Apple-устройств с использованием нативных технологий, а также предоставляющего унифицированный подход к разработке приложений, управляемых при помощи взгляда.

Ключевые слова*: gaze tracking, eye tracking, отслеживание взгляда, окулография, обработка жестов, TrueDepth, ARKit, SceneKit, UIKit, iOS, UX, UI*

ВВЕДЕНИЕ

Задача компьютерной окулографии не нова и имеет множество практических приложений. В их числе решения, помогающие людям с ограниченными возможностями взаимодействовать с компьютерными системами, в том числе с мо-

бильными, посредством отслеживания их взгляда и положения головы [1, 2], решения в области разработки мобильных игр [3], оценки пользовательских предпочтений в области мобильного ритейла [4] и других сферах [5].

В связи с высокими темпами развития камер и датчиков, интегрируемых в мобильные устройства [6, 7], а также повсеместным их распространением [8], применять разработки в области окулографии становится возможным не только в специальных лабораторных условиях, но и в условиях повседневного использования персональных мобильных телефонов, что существенно расширяет возможный круг потребителей подобных технологий. В частности, существует ряд работ, посвященных исследованию этой темы в контексте устройств компании Apple. В их числе специализированные модели машинного обучения для предсказания положения взгляда [9–11], а также ряд прикладных разработок [12—14], но описанные в них решения либо покрывают один конкретный сценарий использования, либо предназначены сугубо для исследовательских целей и не могут быть применимы в прикладных задачах. Иными словами, имеющиеся наработки не предлагают унифицированного подхода к построению приложений, управляемых посредством взгляда пользователя, и сейчас для каждого конкретного сценария использования разрабатывается свое собственное решение, что приводит к дороговизне и, как следствие, малой распространенности этой технологии на рынке мобильных приложений. К аналогичным выводам пришли специалисты из eBay [15], разрабатывая свое решение в смежной области – управление iOS-приложением при помощи движений головы.

Таким образом, целью данной работы стала разработка iOS-фреймворка для окулографии, предоставляющего конкурентоспособную точность распознавания и предлагающего единый подход к проектированию приложений, управляемых при помощи взгляда.

ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ

Изучение окулографии ведется с 1960-х годов, и одним из первых подходов к решению задач в этой области стало использование метода электроокулографии – отслеживания движения глаз при помощи считывания электрических потенциалов с электродов, прикрепленных к уголкам глаз [16]. По мере увеличения

вычислительных мощностей компьютеров и развития технологий в области цифровой обработки изображений стали появляться менее инвазивные, значит, и более предпочтительные для широкого использования варианты, такие как Skyle [14] и Tobii EyeX [17], представляющие собой системы камер, специально предназначенных для отслеживания положения глаз. В «противовес» решениям, основанным на специализированных дополнительных устройствах, можно выделить противоположную группу – решения, основанные исключительно на программном обеспечении и использующие в качестве входных данные, полученные с камер общего назначения – фронтальной камеры смартфона или веб-камеры персонального компьютера или ноутбука [18]. С развитием камер, устанавливаемых в смартфоны, а также ростом популярности AR/VR технологий в мобильных приложениях [19], эту группу пополняют и мобильные решения [20, 21]. В частности, в эту же группу войдет решение, разрабатываемое в рамках данной работы. При этом для определения наиболее важных критериев, которым должно соответствовать разрабатываемое решение, необходимо провести сравнительный анализ непосредственных аналогов – решений для iOS, основанных на программном обеспечении:

- iTracker [9] – представляет собой сверточную нейронную сеть, обрабатывающую входной сигнал с фронтальной камеры телефона. Модель анализирует полное изображение лица для определения его положение относительно камеры, а также изображения каждого из глаз в отдельности для определения их направления относительно лица. Сопоставляя полученные данные, нейронная сеть предсказывает точку на экране, куда направлен взгляд пользователя. Заявленная средняя ошибка предсказанного положения взгляда (расстояние на плоскости между реальным и предсказанным положениями точки) составляет 1.71 см и 2.53 см на телефонах и планшетах без предварительной калибровки. С калибровкой ошибка уменьшается до 1.34 см и 2.12 см соответственно. Из недостатков решения можно выделить сравнительно низкую производительность – созданная модель может обрабатывать данные с камеры только с частотой 10–15 Гц. Также это решение не предоставляет API для интеграции с пользовательским интерфейсом: возможно лишь предсказать точку на экране.

- ScreenGlint [10] – нейронная сеть, предсказывающая положение взгляда пользователя на экране и использующая в качестве ориентира отражение экрана смартфона на поверхности глаза. По заявлению авторов модель достигает средней ошибки в диапазоне от 1.47 см до 1.72 см. Из недостатков можно выделить зависимость точности результатов от освещения, так как используются данные о положении блика от устройства на поверхности глаза, – авторы заявляют, что оценка результатов проводилась исключительно в закрытых помещениях, и при более сложном освещении результаты могут ухудшиться. Также стоит отметить необходимость калибровки для использования модели. Аналогично iTracker, ScreenGlint также не предоставляет возможности реализовывать взаимодействие с пользовательским интерфейсом, решая лишь задачу предсказания точки взгляда.
- SmartEye [11] – система отслеживания взгляда на экране с применением прототипа смартфона с инфракрасной камерой и сверточных нейросетей. Согласно исследованию [11], эта система обеспечивает возможность обрабатывать поток данных с частотой 30 кадров в секунду и на данный момент является самым точным решением среди трекеров взгляда на смартфонах. Главным недостатком решения является то, что оно основано на применении инфракрасного сенсора, что невозможно на iOS устройствах без серьезной модификации – камера TrueDepth, устанавливаемая на iPhone, начиная с модели X, обладает таким сенсором, но система не предоставляет возможности использовать его напрямую [21].
- Hawkeye Access [12] – проприетарное приложение-браузер, полностью управляемое взглядом пользователя. Для реализации отслеживания взгляда используется фреймворк ARKit, предоставляемый компанией Apple. Использование нативного фреймворка дает возможность добиться серьезного улучшения производительности (обработка сигнала с частотой 60 кадров в секунду) и снижения нагрузки на процессор за счет низкоуровневых оптимизаций. Согласно исследованиям [22], использование ARKit позволяет добиться средней ошибки в предсказании точки взгляда пользователя, равной 1.44 см, что соответствует уровню аналогов в этой области. Серьезным недостатком Hawkeye Access является его ограниченная область

применения – решение представляет собой веб-браузер и не может использоваться в других задачах. Access также не дает возможности использовать полученные наработки для интеграции технологии в собственные приложения, а авторы не раскрывают исходный код.

- EyeMU Interactions [13] – JavaScript-приложение, работающее на устройствах под управлением iOS в браузере Safari и представляющее собой комбинацию отслеживания взгляда пользователя с гироскопическими жестами. Приложение позволяет определить, на что пользователь в данный момент смотрит (это может быть, например, системное уведомление), после чего при выполнении определенного гироскопического жеста (например, смахивания влево или вправо) выполнить то или иное действие с объектом, на который направлено внимание пользователя. Распознавание точки взгляда пользователя в этом решении реализовано с помощью сверточной нейросети, по своим принципам схожей с таковой в iTracker и использующей тот же самый датасет для обучения. Заявленная авторами точность предсказания достигает 1.74 см и не требует калибровки при использовании. Стоит отметить, что реализация, по мнению самих разработчиков, является недостаточно эффективной для использования в приложениях – обработка сигналов происходит с частотой 20 Гц, при этом отмечается серьезная нагрузка на батарею смартфона из-за использования Javascript вместо технологий, нативных для платформы. Авторы также не предоставляют возможности реализовывать свою собственную логику для жестов, доступны лишь несколько демонстрационных решений, но нет возможности создавать свои.

На базе аналогов, описанных выше, можно сформулировать список критериев, которым должно соответствовать разрабатываемое решение:

1. Производительность – поддержка решением обработки данных в реальном времени (60 Гц).
2. Универсальность – решение не должно зависеть от конкретного сценария использования и быть применимо для различных задач окулографии.

3. Отсутствие специфических требований к устройству – решение должно быть применимо на устройствах без специальных модификаций или дополнительных аксессуаров.
4. Наличие интерфейса для интеграции в приложения – решение позволяет внедрять в приложения функционал, связанный с отслеживанием взгляда пользователя.
5. Точность – соответствие точности предсказания другим state-of-art решениям.

КОНЦЕПЦИЯ ПРОГРАММНОГО РЕШЕНИЯ

Так как перед разрабатываемым решением ставятся две основные задачи – отслеживание положения взгляда и взаимодействие с его помощью с элементами интерфейса, было принято решение разделить эту ответственность между двумя «слоями» со следующими обязанностями:

- Backend-слой:
    - получение и обработка данных с камеры;
    - вычисление координат точки взора;
    - отслеживание моргания.
- Frontend-слой:
    - преобразование данных о взгляде в события взаимодействия с пользовательским интерфейсом;
    - API для интеграции в клиентское приложение.

Нетрудно видеть, что Frontend-слой является «выходом» для Backend-слоя, а общую схему взаимодействия клиентского приложения и фреймворка можно изобразить следующим образом (рис. 1):

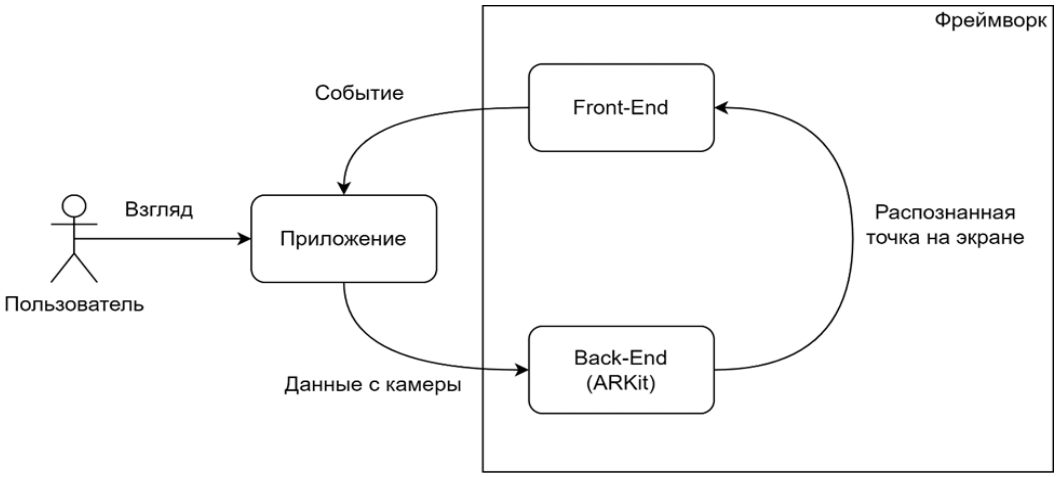


Рисунок 1. Взаимодействие клиента и разрабатываемого фреймворка

Так как задачи, решаемые слоями, независимы, их взаимодействие можно реализовать через абстракцию – Frontend-слой является делегатом Backend-слоя, что позволяет ему работать с данными о положении взгляда, не заботясь об их происхождении (рис. 2):

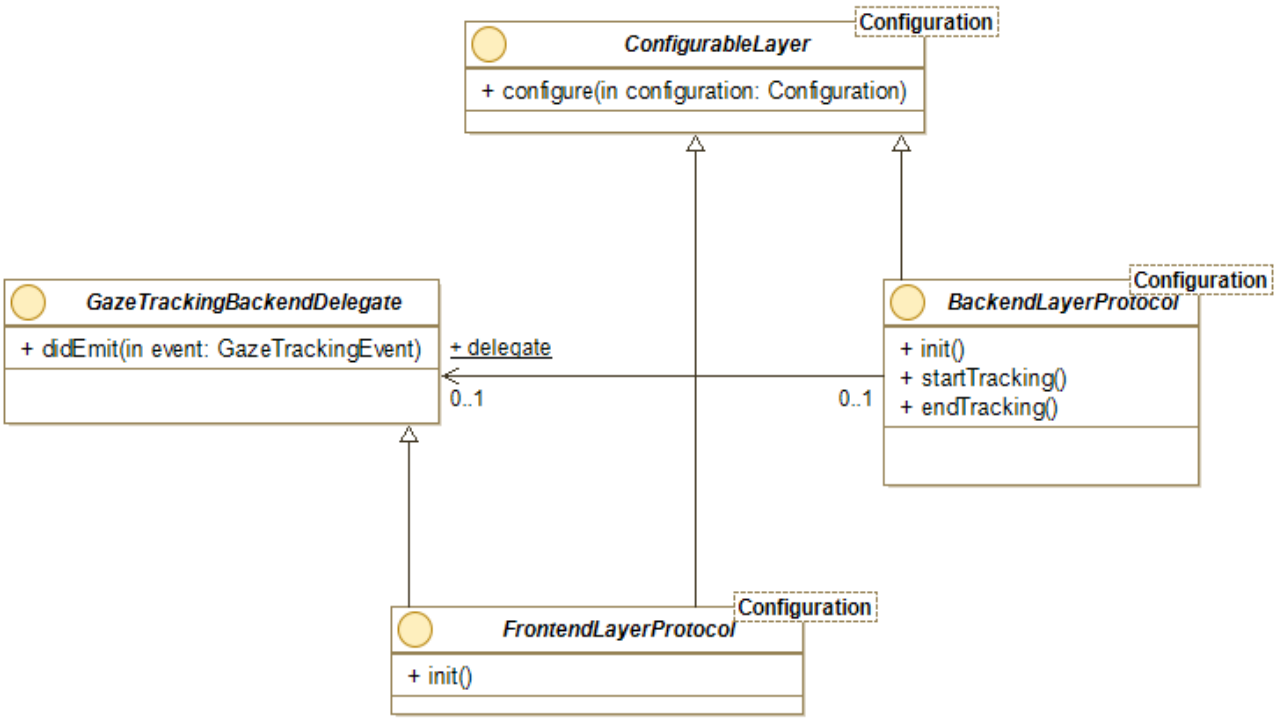


Рисунок 2. Диаграмма слоев решения

Такой подход к построению архитектуры слоев позволяет реализовать единую точку инициализации для создаваемой системы, использующую пару из конфигураций в качестве входного параметра и настраивающую связь между слоями (рис. 3):

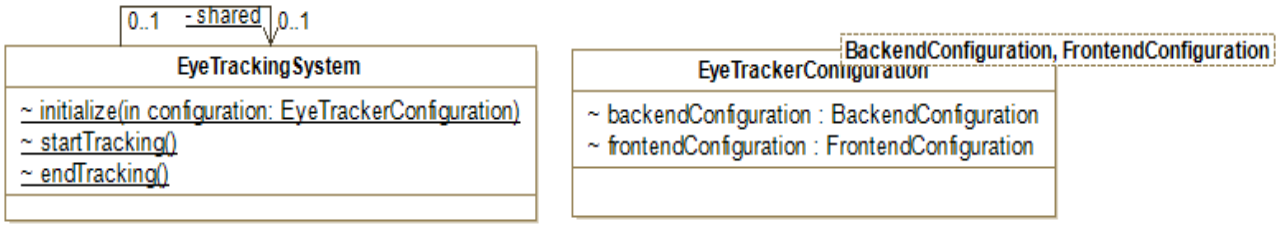


Рисунок 3. Конфигуратор системы отслеживания взгляда

Немаловажным следствием этой архитектурной особенности является возможность пользователя заменять реализации компонентов системы на собственные, если того требует ситуация.

РАСПОЗНАВАНИЕ ВЗГЛЯДА

Исходя из необходимости соответствия разрабатываемого решения критерию точности распознавания, было принято решение использовать фреймворк Apple ARKit для обработки данных с камеры. Согласно исследованию [22], этот инструмент пригоден для использования в задаче окулографии и имеет точность, сравнимую с аналогами. При этом, согласно тому же исследованию, ARKit показывает в данной задаче от 4-х до 6-кратного прироста производительности по сравнению с ранее упомянутым iTracker, что делает его среди рассмотренных самой предпочтительной технологией с точки зрения скорости обработки данных.

Анализируя изображения, получаемое с камеры TrueDepth, ARKit предоставляет для каждого кадра информацию о расположении в пространстве тех или иных объектов, представляя информацию о них в виде объектов ARAnchor. В частности, используя конфигурацию ARFaceTrackingConfiguration, можно получить информацию о расположении в сцене лица пользователя, а также отследить положение глаз и их поворот, характеризующий направление взгляда. Указанные данные о расположении рассчитываются относительно родительского объекта – лица пользователя. Помимо этого система также предоставляет информацию о

камере – ее расположение в пространстве, а также углы поворота. Помимо информации о пространственном положении объектов ARKit позволяет отслеживать и другую информацию – в случае конфигурации для трекинга лиц, в частности, доступна информация о степени «закрытости» глаза в виде значения от 0 до 1, что позволяет отслеживать моргание пользователя.

Расположение тех или иных объектов в пространстве в ARKit описывается с помощью матриц аффинного преобразования следующего вида (рис. 4):

$$M = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Рисунок 4. Матрица аффинного преобразования ARKit

Такой подход позволяет представить всю информацию о расположении объекта с помощью одной структуры (рис. 5 и 6):

$$T = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Рисунок 5. Матрица переноса по осям XYZ

$$R_x(\alpha) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad R_y(\alpha) = \begin{pmatrix} \cos\alpha & 0 & -\sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\alpha & 0 & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$R_z(\alpha) = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 & 0 \\ -\sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Рисунок 6. Матрицы вращения по осям XYZ

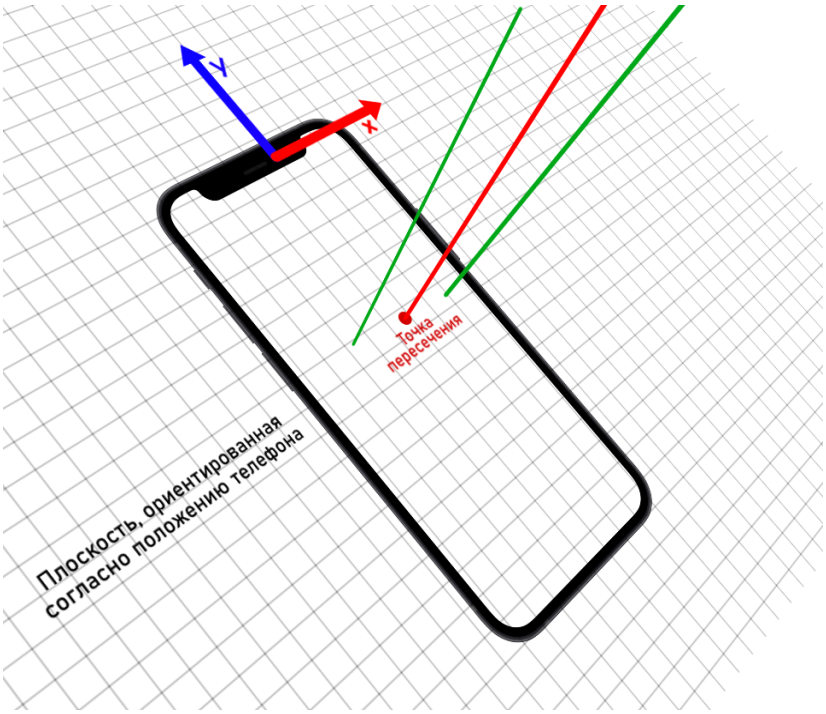


Рисунок 7. Виртуальная сцена для определения положения взгляда

Так как ARKit предоставляет информацию о смещении и повороте головы в координатах всей сцены, а также аналогичную информацию о глазах в координатах лица, появляется возможность сконструировать виртуальную трехмерную сцену (рис. 7), с помощью которой можно будет получить координаты точки взгляда на экране.

Для построения виртуальной сцены было решено воспользоваться другим нативным фреймворком Apple – SceneKit, позволяющим размещать в виртуальном пространстве элементы (Nodes). При этом каждый элемент может иметь свои дочерние элементы, координаты которых будут задаваться в координатной системе родительского элемента. Такой подход явно перекликается с представлением данных в ARKit и позволяет не задумываться о конвертации полученных из него данных в координатную систему всей сцены – будет достаточно лишь воссоздать внутри сцены ту же иерархию элементов. Кроме того, SceneKit оперирует той же системой представления расположения объектов в виде матриц, что опять же позволяет не задумываться о конвертации формата данных и не терять на этом производительность.

Итак, для распознавания положения взгляда строится сцена, состоящая из следующих объектов:

- *Квадратная плоскость со стороной 1 м* – плоскость имитирует собой поверхность экрана и используется для проекции на нее луча взгляда. Расположение плоскости в глобальных координатах, а также поворот должны совпадать с расположением камеры в сцене. Такой подход позволит в будущем нивелировать деградацию точности распознавания при поворотах и наклонах девайса. Размеры плоскости выбраны из соображений нормировки, чтобы не выполнять дополнительных вычислений при последующей конвертации координат из метрической системы в логическую.
- *Виртуальный элемент лица* – элемент, имеющий в координатах сцены то же расположение, что и лицо пользователя. Служит родительским элементом для следующих элементов.
- *Виртуальные элементы глаз* – элементы, являющиеся дочерними для элемента лица. Их расположение в координатной системе родителя равно вычисленному положению глаз относительно лица.
- *Начальная точка взгляда* – дочерний элемент «лицевого» элемента, расположение которого получается путем усреднения соответствующих матриц левого и правого глаз. Служит начальной точкой луча взгляда.
- *Конечная точка взгляда* – дочерний элемент начальной точки, смещенный от последней на 2 метра в положительном направлении по оси Z. При таком расположении начальная и конечная точки взгляда находятся по разные стороны вышеупомянутой плоскости, а прямая, их соединяющая, пересечет плоскость в тех случаях, когда пользователь смотрит на экран.

После построения вышеописанной конфигурации определение положения взгляда сводится к нахождению точки пересечения прямой, проходящей через начальную и конечную точки взгляда и плоскости. Вычисление координат этой точки возможно с помощью встроенного функционала фреймворка SceneKit, а именно, метода, возвращающего координаты точки пересечения в системе координат плоскости. Последним шагом алгоритма является конвертация полученных координат из метрической системы, используемой в SceneKit, в логическую, которой подчиняется UIKit, с помощью которого происходит построение интерфейсов

в iOS-приложениях. Кроме того, необходимо учитывать ориентацию устройства в пространстве, так как при ее смене в системе изменяется ориентация осей координат: положительное направление оси X всегда направлено вправо, а оси Y – вниз, а также изменяется смысл понятий «ширина» и «высота» – при портретной ориентации они совпадают с физическими измерениями устройства, а при альбомной физическая высота становится логической шириной, а физическая ширина, соответственно, логической высотой.

Обозначения, используемые при описании реализации алгоритма:

- *x* – x-координата точки пересечения прямой и плоскости в ее координатах и метрической системе измерения.
- *y* – y-координата точки пересечения прямой и плоскости в ее координатах и метрической системе измерения.
- *d* – диагональ экрана в дюймах, определяется спецификацией устройства.
- *h*, *w* – целочисленные доли высоты и ширины в соотношении сторон экрана; определяются спецификацией устройства.
- *pointWidth* – ширина экрана в логической системе измерения; предоставляется системой.
- *pointHeight* – высота экрана в логической системе измерения; предоставляется системой.
- *metricWidth* – ширина экрана устройства в метрах; вычисляется с использованием логической ширины и диагонали следующим образом:

$$metricWidth = \frac{wd}{\sqrt{(h^2+w^2)}} * 0{,}0254;$$

- *metricHeight* – высота экрана смартфона в метрах; вычисляется с использованием логической высоты и диагонали следующим образом:

$$metricHeight = \frac{hd}{\sqrt{(h^2+w^2)}} * 0{,}0254$$

- *resultX* – x-координата точки взгляда в логической системе координат.
- *resultY* – y-координата точки взгляда в логической системе координат.

Параметры алгоритма, зависящие от спецификации конкретного устройства, такие как соотношение сторон экрана, а также длина диагонали в дюймах предоставляются open-source библиотекой DeviceKit [23].

При помощи параметров, описанных выше, алгоритм вычисления итоговой точки взгляда пользователя представляет собой следующее преобразование координат для каждой конкретной ориентации устройства:

- Портретная ориентация (на рис. 8 красной границей выделена плоскость, на которую проецируется взгляд; масштаб уменьшен для наглядности):

$$resultX = \left(\frac{2x}{metricWidth}\right) * pointWidth + \frac{pointWidth}{2},$$
$$resultY = -\left(\frac{2y}{metricHeight}\right) * pointHeight$$

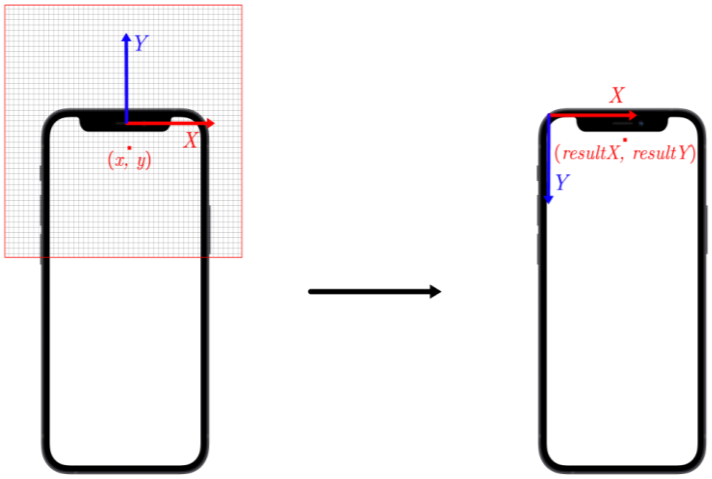


Рисунок 8. Конвертация координат в портретной ориентации

- Левая альбомная ориентация (рис. 9):

$$resultX = \left(\frac{2x}{metricHeight}\right) * pointWidth,$$
$$resultY = -\left(\frac{2y}{metricWidth}\right) * pointHeight + \frac{pointHeight}{2}$$

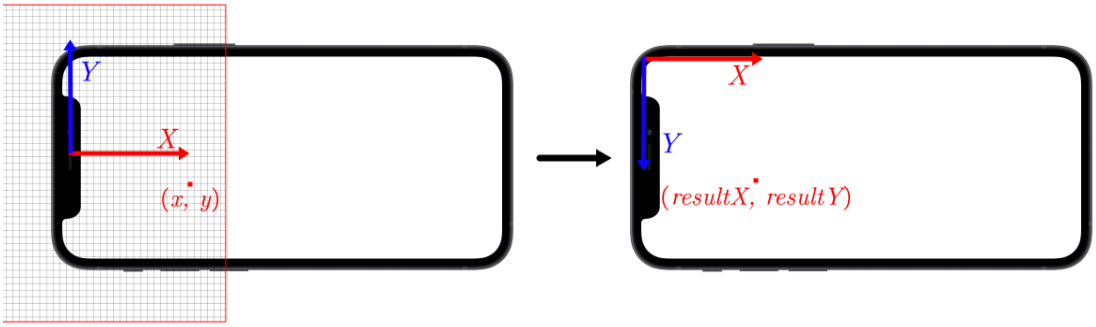


Рисунок 9. Конвертация координат в левой альбомной ориентации

- Правая альбомная ориентация (рис. 10):

$$resultX = \left(\frac{2x}{metricHeight}\right) * pointWidth + pointWidth,$$

$$resultY = -\left(\frac{2y}{metricWidth}\right) * pointHeight + \frac{pointHeight}{2}$$

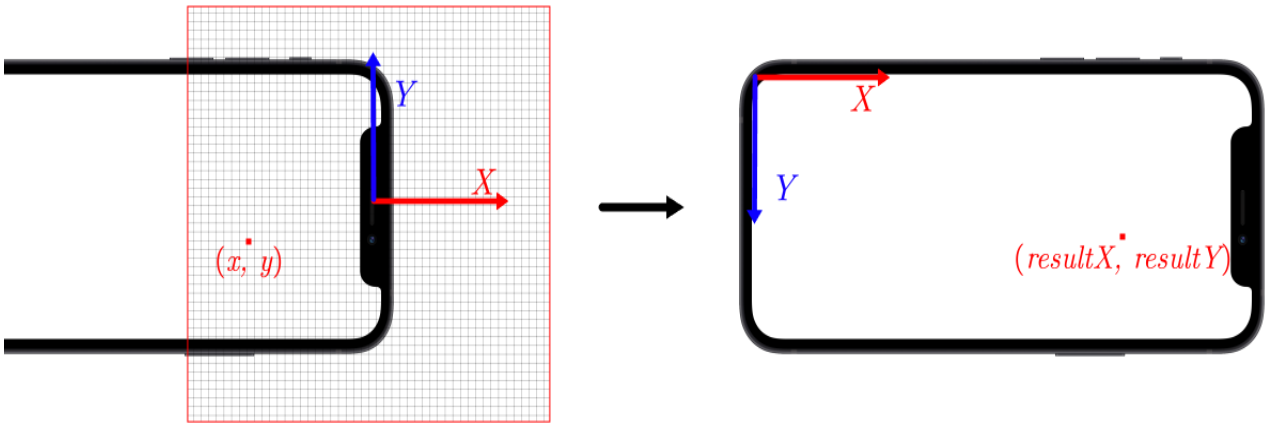


Рисунок 10. Конвертация координат в правой альбомной ориентации

Описанный алгоритм применяется для каждого обработанного кадра в реальном времени (рис. 11).

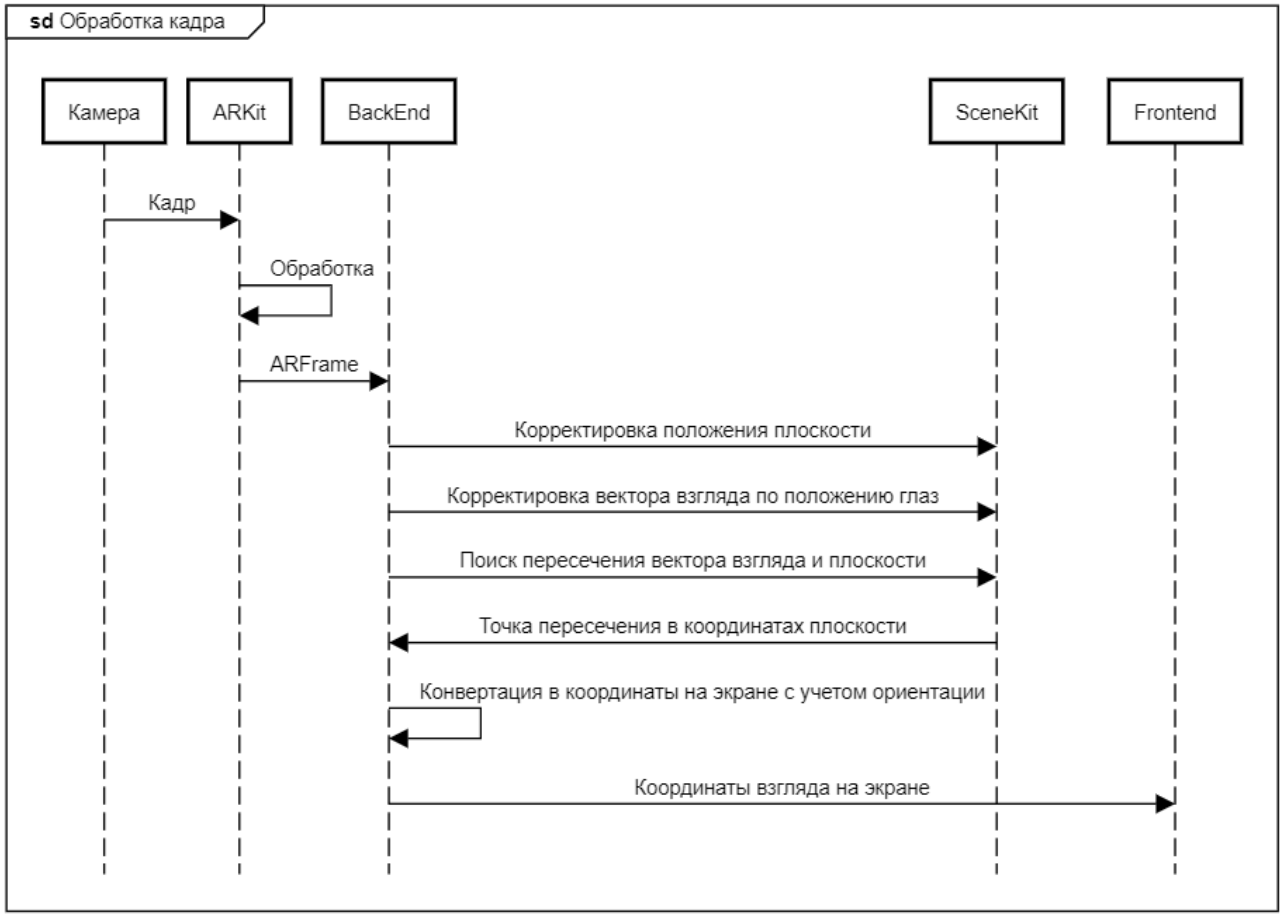


Рисунок 11. Диаграмма последовательности для описанного алгоритма

Помимо отслеживания взгляда пользователя необходимо отслеживать события моргания. Это позволит получить больше вариантов для взаимодействия с элементами интерфейса. Как уже упоминалось ранее, ARKit предоставляет информацию об особенностях выражения лица в виде словаря blendShapes [24]. В частности, этот словарь содержит «степень закрытости» левого и правого глаз в отдельности, выраженную в виде числа в диапазоне от 0 до 1. При превышении порогового значения, определенного заранее, моргание будет считаться состоявшимся, а алгоритм получит возможность проверки вышеуказанных величин вместе с определением положения взгляда. При этом необходимо учитывать, что при моргании направление взгляда кратковременно смещается вниз, поэтому при определении точки, при взгляде на которую произошло моргание, необходимо использовать данные, полученные несколько кадров назад. Этот параметр, как и пороговое значение, задается при конфигурации Backend-слоя. Полный список параметров:

- *blinkFrameOffset* – число кадров, на которые необходимо «вернуться», чтобы получить данные о точке, на которую смотрел пользователь перед тем, как моргнуть.
- *leftEyeBlinkThreshold* – пороговое значение, при превышении которого система считает, что произошло моргание левым глазом.
- *rightEyeBlinkThreshold* – пороговое значение, при превышении которого система считает, что произошло моргание правым глазом.
- *coordinateMapper* – ссылка на конкретную реализацию интерфейса, определяющего функцию перехода между координатами метрической и логической систем; по умолчанию используется реализация алгоритма, приведенного выше.
- *sceneView* – опциональная ссылка на объект ARSCNView, позволяющий вывести на экран визуализацию сцены, описанной выше; используется для целей отладки и демонстрации решения.

КОНЦЕПЦИЯ ОБРАБОТЧИКОВ ЖЕСТОВ

Одной из главных задач нашей работы является разработка публичного API для разработчика, позволяющего использовать данные о положении взгляда пользователя и моргании для реализации произвольной логики. Самым главным приоритетом при решении этой задачи являлось соответствие предлагаемого API нативному подходу к построению пользовательского взаимодействия в iOS.

Для соответствия разрабатываемого решения традиционному подходу к разработке iOS-приложений было принято решение реализовывать внешний API для разработчика как расширение концепции распознавателей жестов (UIGestureRecognizer) – части фреймворка UIKit, унифицирующей событийно-ориентированный подход к проектированию взаимодействия с пользователем в iOS-приложениях.

UIGestureRecognizer представляет собой объект, инкапсулирующий логику обработки определенных событий (чаще всего событий касания экрана). При успешном распознавании того или иного жеста (как правило – совокупности касаний) обработчик с помощью механизма target-action [25] отправляет ассоциированным с ним объектам target сообщение о необходимости выполнения действия

– action. Один или несколько таких объектов можно ассоциировать с любым элементом отображения (наследником UIView), тем самым достигается возможность единообразно добавлять любому элементу на экране возможность обрабатывать те или иные виды нажатий. Внутреннее устройство такого объекта представляет собой конечный автомат из нескольких состояний, переходы между которыми осуществляются при обработке получаемых событий. Существую два основных типа обработчиков – дискретные и непрерывные.

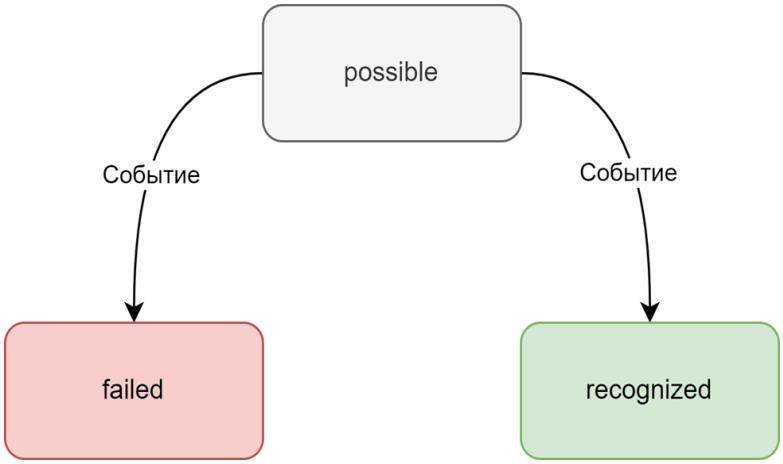


Рисунок 12. Диаграмма состояний дискретного обработчика жестов

Как видно на рис. 12, для дискретных обработчиков используются три состояния:

- *possible* – состояние по умолчанию, означающее, что обработчик готов к получению и обработке событий;
- *recognized* – состояние, означающее, что ожидаемое событие успешно распознано; при достижении этого состояния обработчик вызывает ассоциированную с ним логику через механизм target-action, после чего возвращается в состояние possible;
- *failed* – состояние, означающее, что ожидаемое событие не было распознано; по достижении этого состояния вызов ассоциированного action не происходит, а обработчик возвращается в состояние possible.

К дискретным обработчикам жестов в стандартной библиотеке относятся такие обработчики, как:

- *UITapGestureRecognizer* – обработчик одного или нескольких последовательных нажатий (одним или несколькими пальцами);
- *UILongPressGestureRecognizer* – обработчик длительного касания;
- *UISwipeGestureRecognizer* – обработчик «свайпа» в том или ином направлении.

В случае непрерывных обработчиков число возможных состояний становится больше (рис. 13).

Состояния *possible*, *recognized* и *failed* работают аналогично дискретным обработчиком, но добавляется еще ряд новых:

- *began* – состояние, означающее начало распознавания жеста; при переходе в это состояние происходит вызов соответствующего target-action обработчика;
- *changed* – состояние, означающее очередное изменение в процессе распознавания жеста, еще не приводящее его к завершению, но и не приводящее к ошибке; при переходе в это состояние происходит вызов соответствующего target-action обработчика;

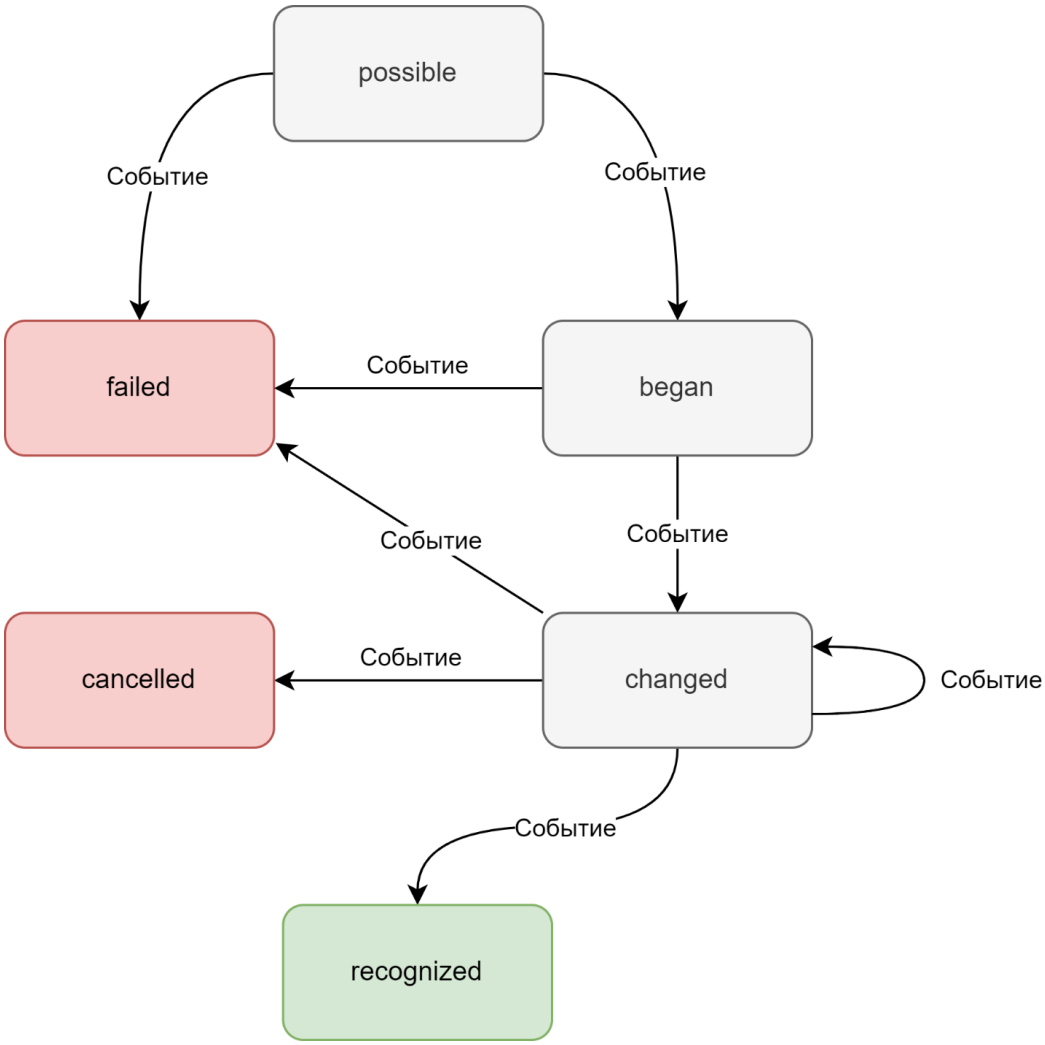


Рисунок 13. Диаграмма состояний непрерывного обработчика жестов

- *cancelled* – состояние, означающее, что в процессе распознавания жеста произошло какое-то событие, которое прервало этот процесс, например, пользователю в этот момент пришел вызов; переход в это состояние прерывает распознавание и сбрасывает состояние обработчика.

ОБОБЩЕНИЕ КОНЦЕПЦИИ НА ОБРАБОТКУ ВЗГЛЯДА

Описанная выше модель состояний универсальна и не зависит от конкретного типа взаимодействия. Для обобщения подхода на «жесты», реализуемые с помощью взгляда, необходимо спроектировать сущность, описывающую то или иное событие, отслеживаемое с помощью окулографии, и использовать его для изменения состояния обработчика.

Обработчики из стандартной библиотеки используют для этой цели объект UITouch, автоматически генерируемый системой при касании экрана. Использование такого же объекта для окулографических жестов невозможно по двум причинам:

- корректное создание объекта UITouch невозможно с помощью открытого API;
- использование этих объектов привело бы к ситуации, когда помимо окулографических обработчиков на них бы реагировали и обычные, так как формат события бы не отличался.

Для этих целей был разработан следующий формат сообщения, позволяющий описать любое событие, связанное с распознаванием взгляда (рис. 14).

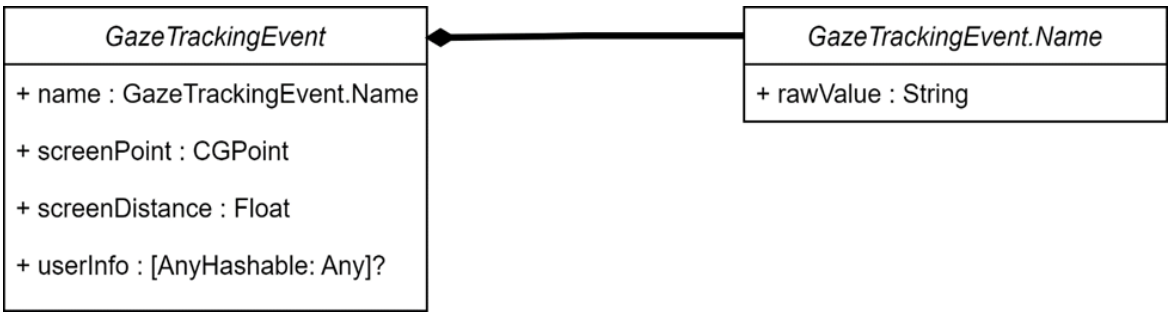


Рисунок 14. Формат событий

Такая модель события позволяет передать обработчику координаты точки взгляда в координатной системе всего экрана, дистанцию от лица пользователя до экрана, а также, опционально, любые другие параметры, переданные в виде словаря. Такой подход позволяет разработчику Backend-слоя добавить дополнительные параметры в событие без нужды изменять основную его модель. Также объект содержит поле Name, задача которого состоит в определении конкретного типа события.

Разрабатываемая реализация включает в себя следующие типы:

- *gazePositionChanged* – изменение положения взгляда;
- *leftEyeBlink* – моргание левым глазом;
- *rightEyeBlink* – моргание правым глазом;
- *bothEyesBlink* – одновременное моргание обоими глазами.

Реализация типа Name кодируется строкой, а не представляется перечислением с целью дать возможность добавлять свои собственные типы, если это понадобится пользователю.

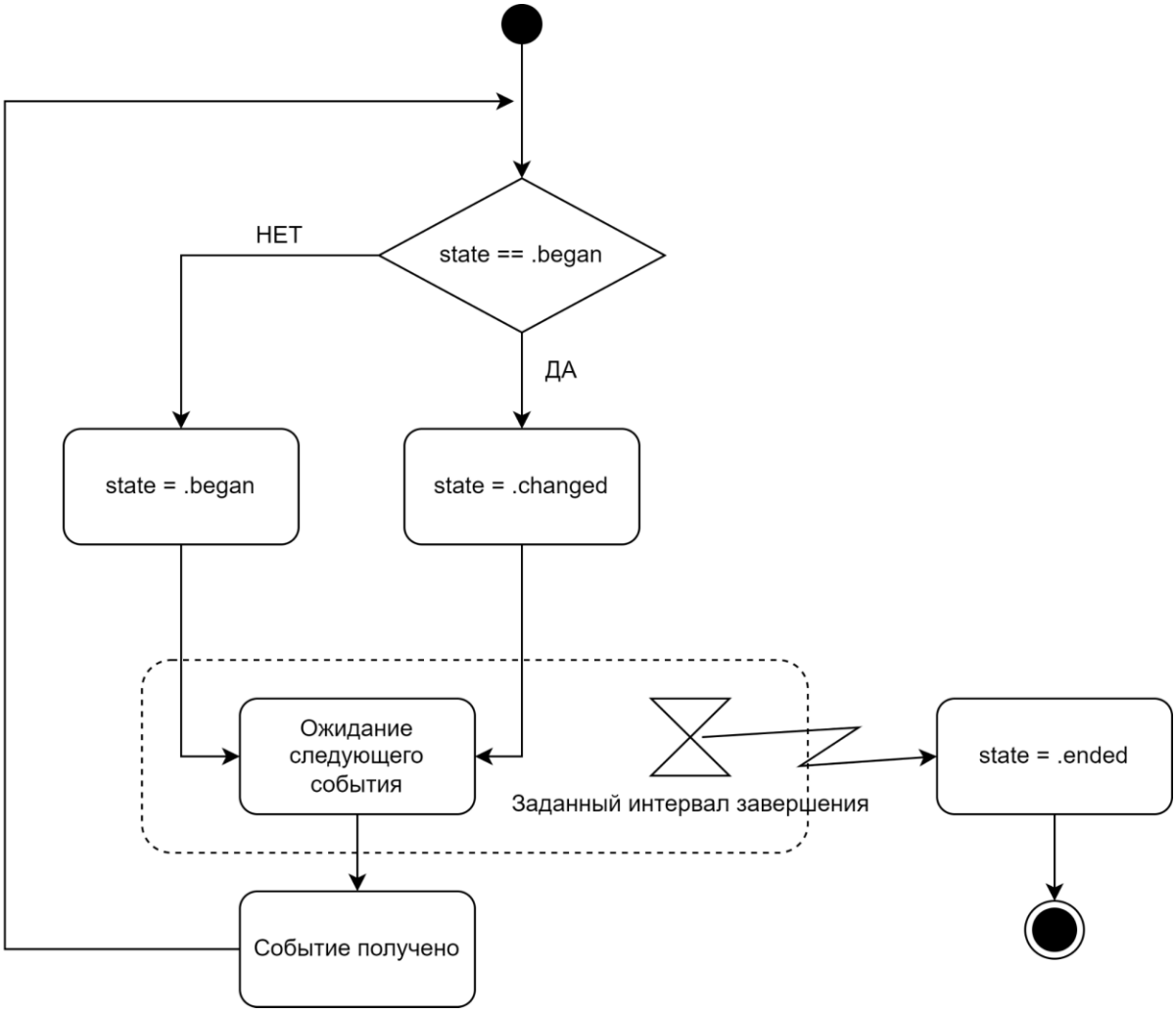


Рисунок 15. Обработка события в *GazeGestureRecognizer*

Таким образом, появляется возможность расширить число системных обработчиков жестов дополнительными вариантами, предназначенными для обработки событий взгляда и использующие вышеописанную модель для изменения своего состояния. В число новых обработчиков входят:

- *GazeGestureRecognizer* – непрерывный жест взгляда, аналог *UIPanGestureRecognizer*. Распознается до тех пор, пока пользователь фокусирует взгляд на определенном элементе отображения. Имеет настраиваемый интервал, в течение которого жест не считается завершенным, даже

если при этом взгляд пользователя переместился – это необходимо для возможности борьбы со случайным переводом взгляда или морганием. Алгоритм обработки события данным обработчиком представлен на рисунке 15.

- *BlinkGestureRecognizer* – дискретный жест моргания, аналог *UITapGestureRecognizer*. Позволяет распознавать одно или несколько морганий подряд при взгляде на определенный интерфейс интерфейса. Допускается распознавание моргания левым и правым глазами в отдельности и одновременного моргания обоими глазами. Кроме того, допускается настройка числа ожидаемых морганий (blinkCount) и максимально возможного временного интервала между морганиями в случае, когда их больше одного. Обработка события представлена на рисунке 16.

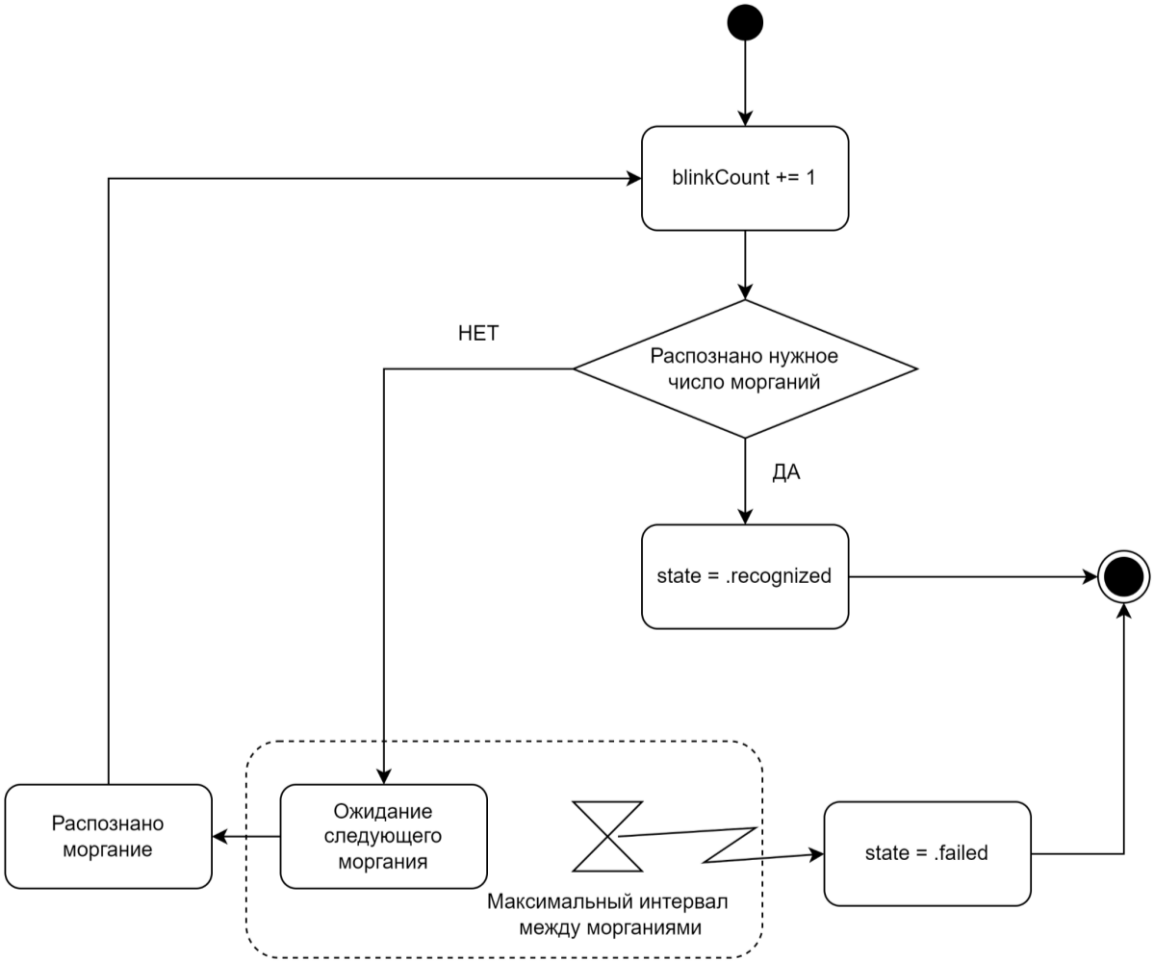


Рисунок 16. Обработка события в *BlinkGestureRecognizer*

- *LongGazeGestureRecognizer* – дискретный жест долгого взгляда на элемент, аналог *UILongPressGestureRecognizer*. Распознается при фокусировке взгляда на элементе в течение заданного периода времени. Также имеется возможность задать интервал времени, в течение которого жест не считается прерванным, даже если фокус взгляда был переведен – в целях борьбы со случайной сменой фокуса или морганием. При этом такой интервал обязательно должен быть меньше, чем интервал, соответствующий распознаванию. Реализация обработки события представлена на рисунке 17 и состоит из двух таймеров. Первое полученное событие активирует оба, а последующие перезапускают таймер, отвечающий за интервал, в течение которого жест не считается нераспознанным. Если он завершится первым, значит, пользователь отвел взгляд на длительное время, тем самым прервав жест. В противном случае первым завершится второй таймер, что будет означать успешное распознавание.

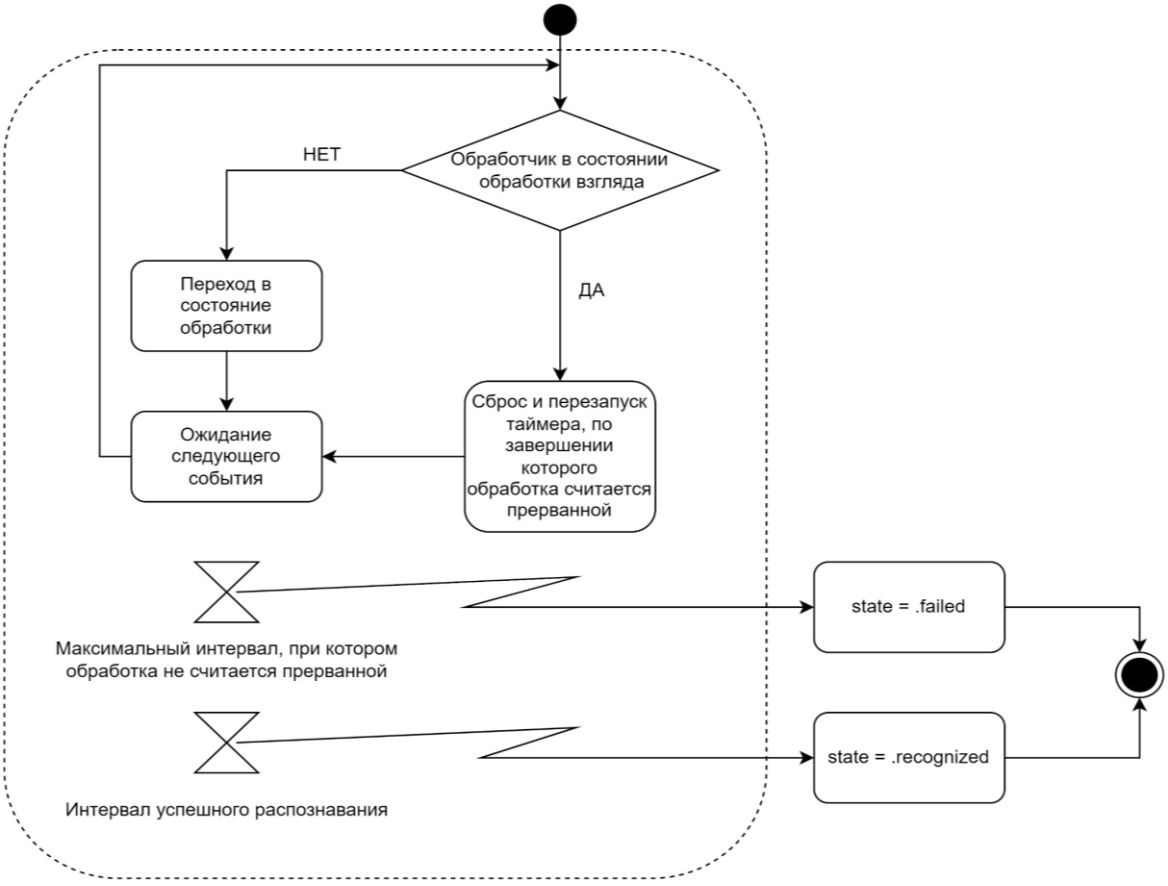


Рисунок 17. Обработка события в *LongGazeGestureRecognizer*

ДИСПЕТЧЕРИЗАЦИЯ СОБЫТИЙ МЕЖДУ ОБРАБОТЧИКАМИ

Помимо реализации самих обработчиков событий, было необходимо реализовать систему, способную доставлять события до этих обработчиков и обеспечивать их взаимосвязанное функционирование. Это является прямой обязанностью Frontend-слоя, который, будучи делегатом Backend-слоя, оповещается о каждом новом распознанном событии. Поскольку разрабатываемое решение является расширением встроенного функционала обработчиков жестов, логика решения этой задачи должна соответствовать таковой в UIKit.

Так как при реализации используется собственная модель события, возможность использовать для этой задачи встроенную логику UIKit отсутствует, и необходимо реализовывать ее отдельно. На верхнем уровне алгоритм обработки события внутри Frontend-слоя выглядит следующим образом (рис. 19).

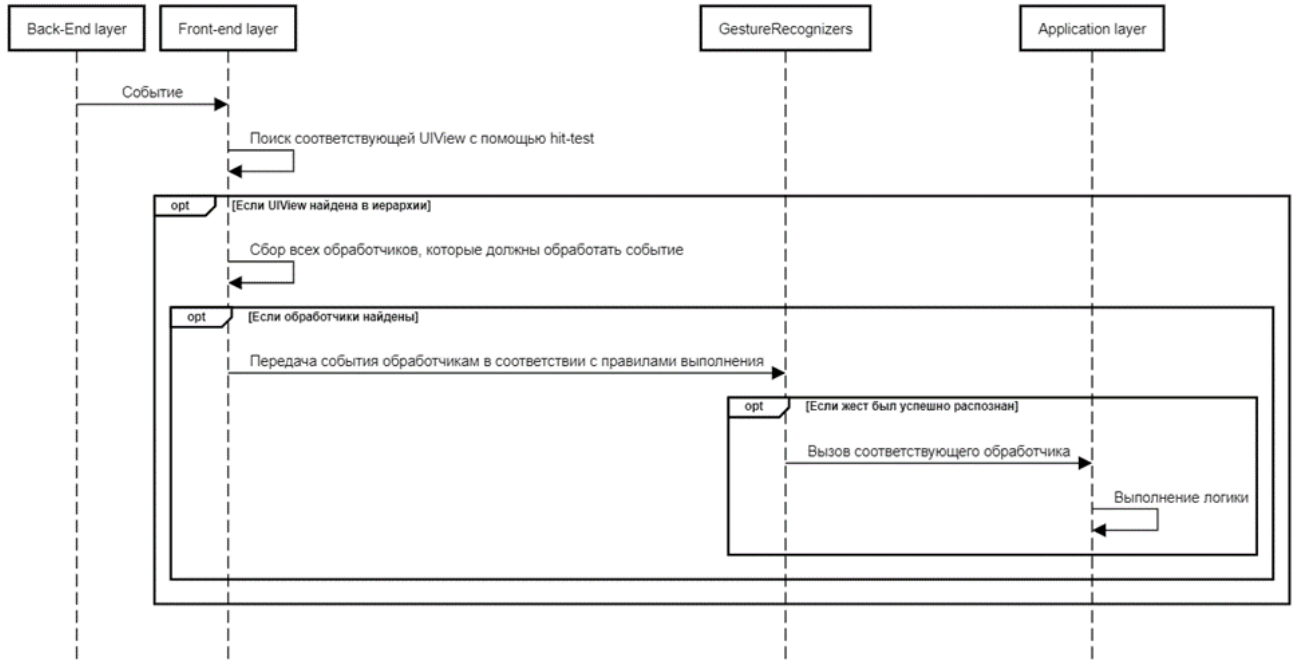


Рисунок 19. Обработка события Frontend-слоем

Первый шаг – поиск элемента, на котором сфокусирован взгляд пользователя, выполняется с помощью поиска в глубину по дереву элементов, начиная от родительского объекта UIWindow – «окна», в котором все визуальные элементы находятся в качестве дочерних с разной степенью вложенности. Все дочерние

элементы окна, будучи наследниками UIView, имеют два вспомогательных метода для выполнения этого поиска:

- `pointInisde(_ point: CGPoint, with event: UIEvent) -> Bool` — метод принимает на вход точку с координатами, выраженными в координатной системе самого элемента, а также объект события и возвращает *true* или *false*, в зависимости от того, находится ли точка внутри границ элемента.
- `hitTest(_ point: CGPoint, with event: UIEvent) -> UIView?` — метод принимает на вход точку с координатами, выраженными в координатной системе самого элемента, а также объект события и возвращает либо сам элемент, если он самый глубокий в иерархии, либо одного из своих потомков, если точка принадлежит ему, либо *nil*, если обход на этом узле должен прекратиться.

С использованием методов, описанных выше, алгоритм поиска нужного элемента реализуется стандартными средствами UIKit и заключается в вызове метода *hitTest* на объекте UIWindow с передачей ему точки в координатах экрана.

На следующем шаге необходимо получить все обработчики, которые должны получить событие. Алгоритм их нахождения полностью аналогичен таковому из UIKit и визуально представлен на рисунке 20.

После того, как все нужные обработчики найдены, нужно определить, каким образом они могут взаимодействовать между собой. По умолчанию в UIKit может происходить распознавание лишь одного жеста за раз, но это поведение можно переопределить с помощью объекта делегата обработчика жеста. Для динамического контроля за взаимодействием различных обработчиков делегат предлагает для реализации следующие методы:

- `gestureRecognizer(UIGestureRecognizer, shouldRecognizeSimultaneouslyWith: UIGestureRecognizer) -> Bool` – метод позволяет динамически определять, должны ли два обработчика обрабатывать события одновременно. При этом даже если один из этой пары вернет в этом методе *false*, это еще не гарантирует отсутствие одно-

временного распознавания, так как оставшийся обработчик может получить от своего делегата значение *true* – в этом случае жесты обработаются одновременно.

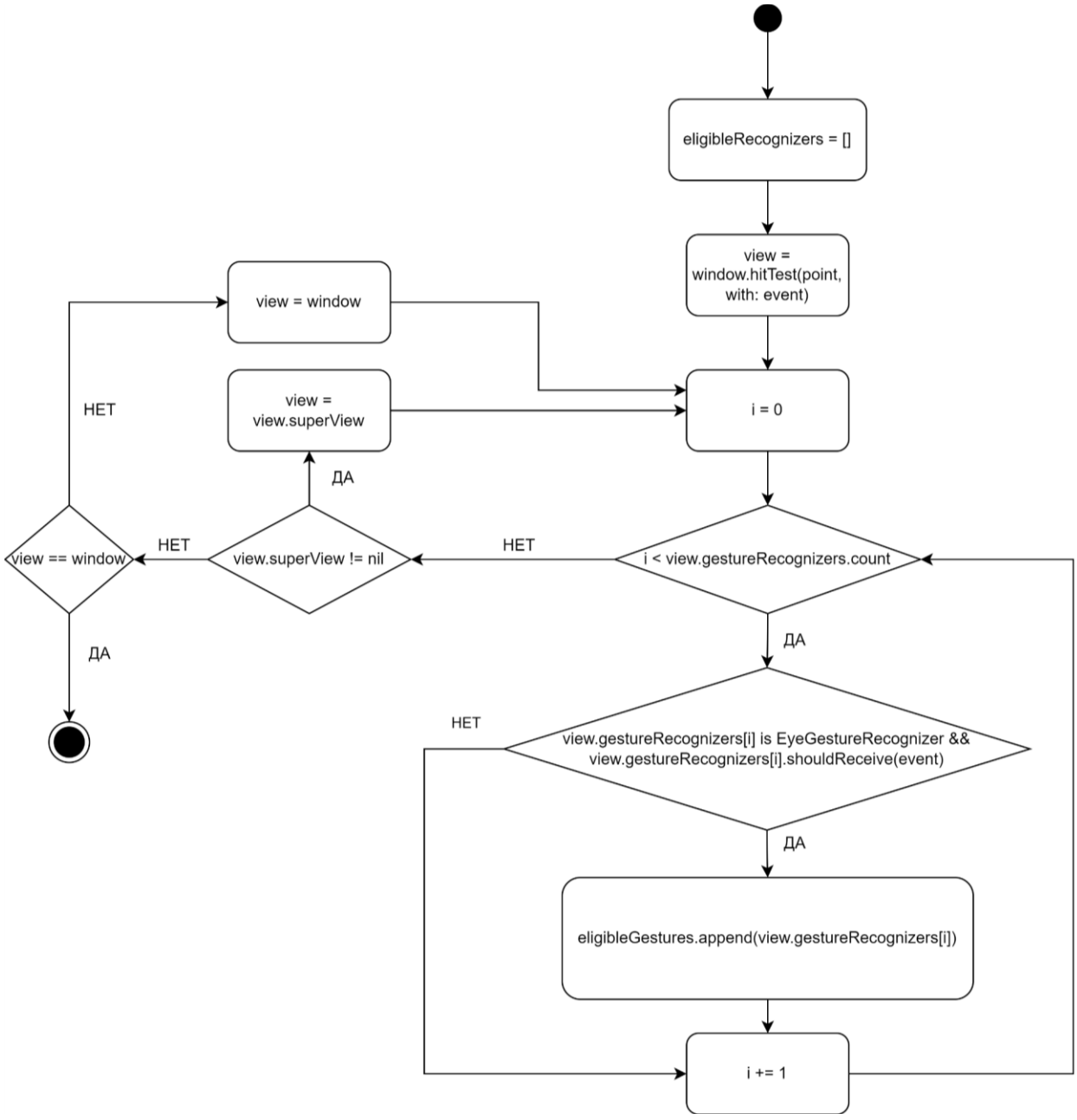


Рисунок 20. Получение всех обработчиков жестов, основанных на отслеживании взгляда

- `gestureRecognizer(UIGestureRecognizer, shouldRequireFailureOf: UIGestureRecognizer) -> Bool` – метод позволяет определить зависимость одного жеста от другого по следующему правилу: если делегат обработчика, передаваемого в метод первым параметром, вернул true, то этот обработчик получит возможность распознать жест только в том случае, если обработчик, переданный вторым параметром в процессе распознавания, перейдет в состояние *failed*. Такое поведение позволяет избегать ситуаций, при которых один из обработчиков перекрывает возможность работы для другого. Пример такой ситуации – элемент имеет обработчик единичного касания и двойного касания одновременно. При такой конфигурации при двойном нажатии не произойдет распознавание соответствующего жеста, вместо этого дважды будет распознан жест единичного касания. Используя этот метод, можно создать правило, по которому жест единичного касания будет распознаваться только тогда, когда жест двойного касания перейдет в состояние *failed.*
- `gestureRecognizer(UIGestureRecognizer, shouldBeRequiredToFailBy: UIGestureRecognizer) -> Bool` – этот метод является обратным к предыдущему и позволяет создать требование, при котором обработчик из второго параметра получит событие лишь при переходе первого в состояние *failed.*

Кроме того, аналоги последних двух методов есть также и у самих обработчиков жестов, что позволяет задавать такие требования на уровне всего подкласса. Пользуясь двумя последними методами делегата и их классовыми аналогами, можно построить между объектами обработчиков граф зависимостей в виде словаря, где ключом является обработчик, а значением – массив обработчиков, неудача распознавания которых должна привести к получению обработчиком-ключом соответствующего события. При построении графа каждый обработчик проверяется на наличие зависимостей только со всеми последующими в списке. При этом проверяется как зависимость первого от второго, так и наоборот. Стоит заметить, что если первый обработчик зависит от второго, то обратная за-

висимость уже не проверяется. С помощью этих мер удается избегать циклических зависимостей между обработчиками. Обработчик A считается зависящим от обработчика B, если выполняются оба следующих условия:

- B не зависит от A.
- Хотя бы одно из следующих условий:
  - `A.delegate?.gestureRecognizer?(A, shouldRequireFailureOf: B) == true`;
  - `B.delegate?.gestureRecognizer?(B, shouldBeRequiredToFailBy: A) == true`;
  - `A.shouldRequireFailure(of: B) == true`;
  - `B.shouldBeRequiredToFail(by: A) == true`.

Из условий, приведенных выше, следует, что приоритет при построении графа зависимостей отдается значению true: если делегат B возвращает false в методе делегата, определяющем зависимость A от B, а делегат A, в свою очередь, такую зависимость разрешает, – зависимость будет создана.

Учитывая все вышеописанное, становится возможным описать общий алгоритм диспетчеризации событий, учитывающий возможность обработчиков выполняться одновременно, а также зависеть от состояния других обработчиков. Терминальными состояниями в алгоритме, изображенном на рис. 22, названы состояния *began*, *changed* и *ended*, то есть все те, что вызывают срабатывание метода action, закрепленного за обработчиком.

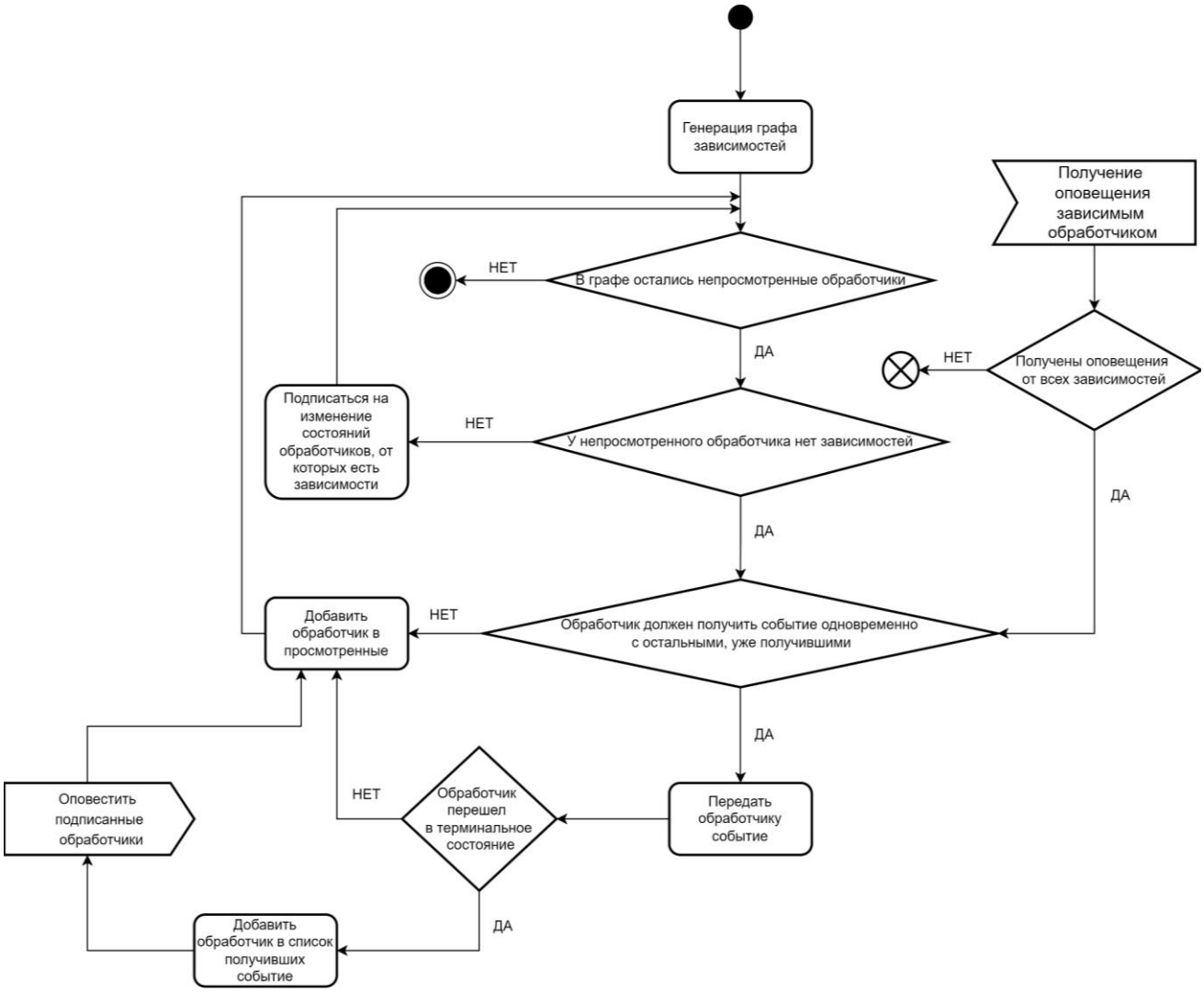


Рисунок 22. Диспетчеризация событий в системе обработчиков

Указанный выше алгоритм завершается в тот момент, когда в системе не остается непросмотренных обработчиков. В силу того, что зависимости удовлетворяются асинхронно, система диспетчеризации сохраняется в памяти до того момента, пока все зависимости не будут тем или иным образом удовлетворены.

ПРИМЕР ИСПОЛЬЗОВАНИЯ

Для подключения разработанного решения к приложению разработчику необходимо выполнить следующую последовательность действий:

1. Подключить фреймворк к проекту приложения с помощью менеджера зависимостей SPM [26].
2. Произвести инициализацию системы при старте приложения:

a. для использования конфигурации по умолчанию достаточно вызвать следующий метод:

```
EyeTrackingSystem.initializeWithDefaultConfiguration()
```

b. если необходимо задать пользовательские параметры для конфигурации, можно воспользоваться следующим методом:

```
EyeTrackingSystem.initializeWithCustomConfiguration<B:
BackendLayerProtocol, F: FrontendLayerProtocol>(_ configura-
tion: EyeTrackingConfiguration<B, F>)
```

при использовании встроенной реализации Frontend и Backend-слоев, вызов метода выше может принимать вид:

```
EyeTrackingSystem.initializeWithCustomConfiguration(
    EyeTrackingConfiguration<ARKitTracker, EventDispatcher>
        .builder()
        .backend(
            config: ARKitTrackerConfiguration(
                 blinkFrameOffset: 20,
                 leftEyeBlinkThreshold: 0.3,
                 rightEyeBlinkThreshold: 0.3
            )
        )
        .frontend(
            config: EventDispatcherConfiguration(
                windowDetectionMethod: .automatic,
                displayGazeLocation: true
            )
        )
        .build()
)
```

Для конфигурации встроенной реализации Frontend-слоя доступны два параметра: windowDetectionMethod, позволяющий передать системе ссылку на объект UIWindow, начиная от которого будет производиться поиск элемента, к которому относится событие, либо предоставить системе возможность определить этот объект самостоятельно, а также параметр displayGazeLocation, отвечающий за отображение предсказываемой точки взгляда на экране. Кроме конфигурации встроенных реализаций слоев, разработчик имеет возможность

передать в этом методе объекты собственных реализаций этих компонентов, если того требует ситуация.

3. Вызвать следующий статический метод в момент, когда нужно начать отслеживание взгляда:

```
EyeTrackingSystem.startTracking().
```

4. Вызвать следующий статический метод в момент, когда отслеживание нужно приостановить (например, для экономии ресурсов): `EyeTrackingSystem.endTracking()`.

После выполнения описанных выше шагов система будет готова к работе. Далее, необходимо добавить нужные обработчики жестов к элементам пользовательского интерфейса таким же образом, как при использовании системных обработчиков из UIKit. Пример добавления обработчика события моргания двумя глазами к кнопке button:

```
      let blinkGestureRecognizer = BlinkGestureRecognizer(
      target: self,
      action: #selector(self.handleBlink)
)
blinkGestureRecognizer.blinkType = .bothEyes
blinkGestureRecognizer.blinkCount = 1
button.addGestureRecognizer(blinkGestureRecognizer)
```

Пример реализации target-action логики для обработчика выше, при которой в результате моргания при взгляде на кнопку в консоль будет выведена строка “hello world”:

```
@objc func handleBlink() {
      print(“hello world”)
}
```

ОЦЕНКА ТОЧНОСТИ РАСПОЗНАВАНИЯ ВЗГЛЯДА

Для оценки эффективности разработанного решения необходимо рассмотреть две основных метрики: точность распознавания взгляда и производительность. Их оценка производится с помощью серии экспериментов и позволяет проверить соответствие решения изначально заявленным критериям. Для оценки точности приведенного алгоритма использовались две метрики – евклидово расстояние между реальной и предсказанной точками в сантиметрах и отклонение в градусах (рис. 23).

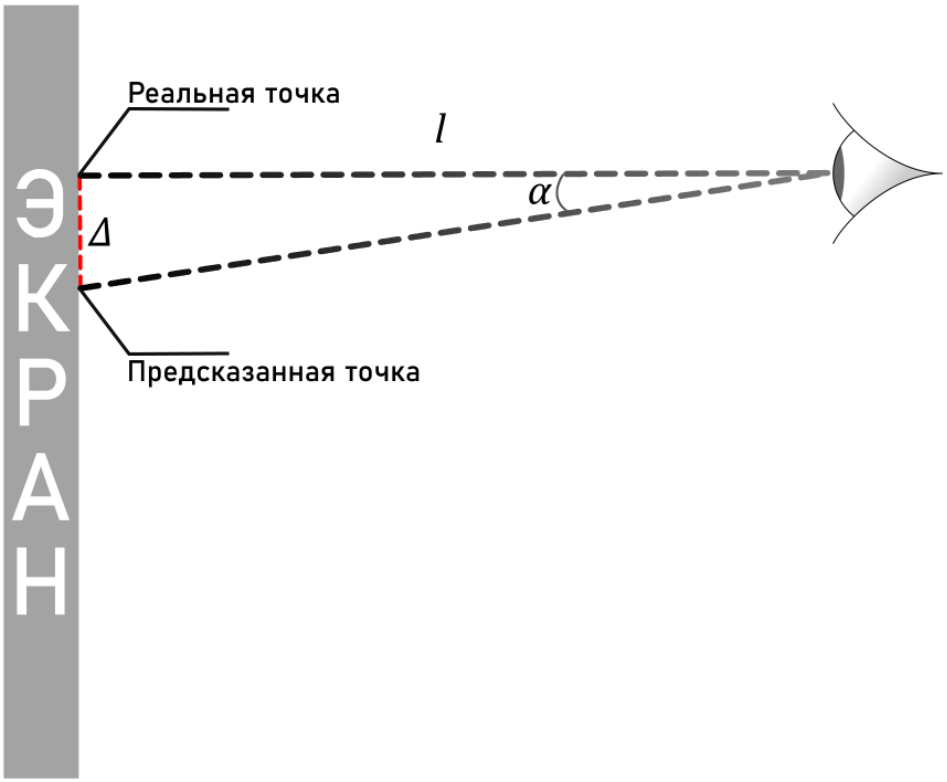


Рисунок 23. Оценка точности распознавания

На рис. 23 использованы следующие обозначения:

- Δ – пространственное отклонение (евклидово расстояние), выраженное в сантиметрах,
- α – угол отклонения, выраженный в градусах,
- *l* – расстояние от лица до экрана, выраженное в метрах.

Вычисление указанных метрик производится следующим образом:

$$\Delta = \sqrt{\left(x_{real} - x_{predicted}\right)^2 + \left(y_{real} - y_{predicted}\right)^2},$$

где x_{real} – x-координата реальной точки на экране; $x_{predicted}$ – x-координата предсказанной точки взгляда на экране; y_{real} – y-координата реальной точки на экране; $y_{predicted}$ – y-координата предсказанной точки взгляда на экране; $\alpha = atan\left(\frac{\Delta}{l}\right)$.

Получение данных, необходимых для вычисления отклонений, производилось с помощью эксперимента, в рамках которого испытуемому предлагалось следить взглядом за точкой на экране. Каждый раз, когда система распознает новую точку взгляда на экране, производится расчет евклидового расстояния между ней и «идеальной» точкой, движущейся по экрану. Кроме того, средствами ARKit фиксируется приблизительное расстояния от лица до экрана смартфона. По окончании эксперимента сохраняются среднее пространственное отклонение и среднее расстояние от лица до экрана. На основе этих данных производится расчет угла отклонения.

Траектория точки показана на рис. 24. Движение начинается от точки 1, а прохождение каждой из стрелок занимает две секунды. Последняя точка располагается в центре экрана. Так как система поддерживает альбомную ориентацию, траектория также автоматически пропорционально подстраивается под текущую ориентацию устройства.

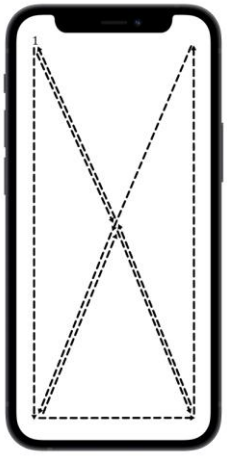


Рисунок 24. Траектория «идеальной» точки на экране

Полный эксперимент занимает 18 секунд и выполняется с частотой обновления экрана в 60 кадров в секунду. Таким образом, одно испытание позволяет получить усредненное отклонение в градусах и сантиметрах на основе около 1080 отдельных измерений.

При проведении эксперимента смартфон размещается на специальном штативе, позволяющем регулировать углы наклона по осям X и Y (рис. 25). Голова испытуемого неподвижна и расположена на заданном расстоянии на уровне экрана смартфона.

Рисунок 25. Установка для фиксации устройства

Измерения точности производились с использованием устройства Apple iPhone 12 Mini под управлением iOS 15.5. Для каждой из ориентаций (портретная, левая альбомная, правая альбомная) рассматривались пять случаев:

- устройство расположено перпендикулярно взгляду;
- устройство наклонено на 15 градусов в положительном направлении оси X;
- устройство наклонено на 15 градусов в отрицательном направлении оси X;
- устройство наклонено на 15 градусов в положительном направлении оси Y;
- устройство наклонено на 15 градусов в отрицательном направлении оси Y.

Для каждого из случаев производилось пять независимых испытаний. Таким образом, серия экспериментов содержит 75 независимых измерений, каж-

дое из которых, как было указано ранее, содержит около 1080 отдельных измерений отклонения. Всего были проведены две серии экспериментов в разных условиях. В первом случае испытания проводились без отображения предсказанной точки на экране, а во втором случае она была видимой. Результаты первой серии представлены в таблицах 1–3:

Таблица 1. Усредненные показатели точности в портретной ориентации

	Δ, pt.	Δ, см.	α, град.	*l*, м.
Перпендикулярно	121.5198449	1.863738925	3.058346105	0.348917902
15˚ по оси X	126.1530491	1.93479796	3.196714584	0.345608822
-15˚ по оси X	119.4069968	1.831334363	2.949938691	0.355282272
15˚ по оси Y	113.2929082	1.737563137	2.810618648	0.35404134
-15˚ по оси Y	112.8362141	1.73055886	2.820730189	0.351573338

Таблица 2. Усредненные показатели точности в левой альбомной ориентации

	Δ, pt.	Δ, см.	α, град.	*l*, м.
Перпендикулярно	99.2640869	1.521467504	2.46230048	0.35409145
15˚ по оси X	99.32916445	1.522464977	2.45649227	0.354626564
-15˚ по оси X	100.4491829	1.539632028	2.564976492	0.343888084
15˚ по оси Y	94.02865067	1.441221501	2.324039943	0.354389908
-15˚ по оси Y	103.6312298	1.588404763	2.526351547	0.360188454

Таблица 3. Усредненные показатели точности в правой альбомной ориентации

	Δ, pt.	Δ, см.	α, град.	*l*, м.
Перпендикулярно	106.9752863	1.639660695	2.610194765	0.359367904
15˚ по оси X	105.3752049	1.615498835	2.625566731	0.351983119
-15˚ по оси X	107.3729781	1.602190452	2.571041685	0.357779693
15˚ по оси Y	111.0812991	1.649823521	2.717085377	0.347994089
-15˚ по оси Y	111.0624266	1.570941563	2.601909458	0.346306784

Из приведенных результатов замеров можно сделать следующие выводы:

- В среднем по всей серии ошибка системы составляет около 108 точек (1.65 см) или 2.68˚.
- Стандартное отклонение в зависимости от ориентации устройства составляет около 9.61 точек (0.15 см) или 0.25˚, что позволяет заключить, что точность алгоритма существенным образом не зависит от ориентации устройства.
- Стандартное отклонение в зависимости от поворота устройства в каждой ориентации представлено в таблице 4 и позволяет заключить, что поворот устройства в заданном диапазоне по осям X и Y существенным образом не влияет на точность алгоритма.

Таблица 4. Стандартное отклонение в зависимости от поворота

	$\sigma(\Delta)$, pt.	$\sigma(\Delta)$, см.	$\sigma(\alpha)$, град.
Портретная	5.648074951	0.08662401718	0.1637479851
Левая альбомная	3.45909131	0.05301912493	0.09177681367
Правая альбомная	2.574311632	0.03132238018	0.05509940652

Результаты второй серии экспериментов (с предсказываемой точкой, видимой испытуемому) представлены в таблицах 5–-7.

Таблица 5. Усредненные показатели точности при видимой точке в портретной ориентации

	Δ, pt.	Δ, см.	α, град.	*l*, м.
Перпендикулярно	64.93241289	0.995862572	1.608413	0.354550966
15˚ по оси X	65.60818226	1.006226786	1.622489135	0.355389242
-15˚ по оси X	61.48929135	0.943055727	1.513964562	0.356741428
15˚ по оси Y	68.0074467	1.04302408	1.665302015	0.358886768
-15˚ по оси Y	65.0814779	0.998148769	1.62415837	0.352011556

Таблица 6. Усредненные показатели точности при видимой точке в левой альбомной ориентации

	Δ, pt.	Δ, см.	α, град.	*l*, м.
Перпендикулярно	64.66490306	0.991149484	1.586634077	0.357860568
15˚ по оси X	67.7229802	1.038022076	1.654347969	0.359592622
-15˚ по оси X	66.66243389	1.024849087	1.679440628	0.350282642
15˚ по оси Y	65.91835626	1.028582535	1.601117292	0.368524345
-15˚ по оси Y	68.2537732	1.015429365	1.612060424	0.360982671

Таблица 7. Усредненные показатели точности при видимой точке в правой альбомной ориентации

	Δ, pt.	Δ, см.	α, град.	*l*, м.
Перпендикулярно	67.8631078	1.040169878	1.718798882	0.346977592
15˚ по оси X	70.09962755	1.074450071	1.701589532	0.361679694
-15˚ по оси X	67.4719812	1.034174896	1.686856696	0.351101388
15˚ по оси Y	69.77300248	1.069443735	1.792034267	0.34183665
-15˚ по оси Y	70.56571451	1.055339606	1.794716585	0.337062920

Результаты второй серии экспериментов позволяют сделать следующие выводы:

- Средняя ошибка по всей серии составила около 67 точек (1.02 см) или 1.65˚. Таким образом, по сравнению с первой серией точность была увеличена на 61.4%. Это, вероятнее всего, связано с возможностью испытуемого видеть результат работы системы в реальном времени и влиять на него для улучшения результата. Исходя из такого ощутимого прироста точности, было принято решение встроить в реализацию Frontend-слоя опцию включения отображения «курсора».
- Аналогично первой серии, в результатах второй сохраняется малая зависимость точности распознавания от ориентации – стандартное отклонение составляет около 2 точек (0.02 см) или 0.06˚. Аналогичный вывод можно сделать и для зависимости от поворота по осям X и Y в диапазоне от -15˚ до 15˚.

В обеих сериях экспериментов измерения производились при среднем расстоянии между лицом испытуемого и экраном в 35 см. Имеет смысл провести дополнительные испытания с целью выявления зависимости между точностью алгоритма и расстоянием. С этой целью были дополнительно проведены две серии по 5 замеров на расстояниях в 25 см и 45 см соответственно. Замеры производились в портретной ориентации и при включенном отображении предсказываемой точки. Результаты представлены в таблице 8.

Таблица 8. Усредненные показатели ошибки в портретной ориентации в зависимости от расстояния

	Δ, pt.	Δ, см.	α, град.
0.25 м.	63.31681119	0.9710842346	2.107622504
0.35 м.	64.9324129	0.9958625725	1.608413
0.45 м.	67.43337716	1.034219636	1.317330087

Стандартное отклонение по всей выборке составляет 8 точек или около 0.12 см, что позволяет сделать вывод о незначительности зависимости точности алгоритма от расстояния в диапазоне от 25 до 45 см. В приведенной таблице отклонение, выраженное в градусах, изменяется значительно сильнее остальных метрик, так как она напрямую зависит от расстояния, и близкое евклидово отклонение на различных дистанциях дает более сильное различие в градусной мере.

Также необходимо провести сравнение разработанного решения с рассмотренными state-of-art аналогами [9–13]. Согласно данным из соответствующих работ и проведенным замерам можно видеть, что разработанное решение при отображении указателя в реальном времени опережает все рассмотренные аналоги (рис. 26). При отключенном отображении указателя решение опережает два из четырех рассмотренных аналогов. Учитывая, что проведенные в рамках данной работы замеры могут не в полной мере быть аналогичны таковым в аналогичных работах, из полученных данных можно сделать вывод, что решение обладает точностью, как минимум сравнимой с state-of-art реализациями, что позволяет говорить о выполнении критерия точности, заявленного в начале работы.

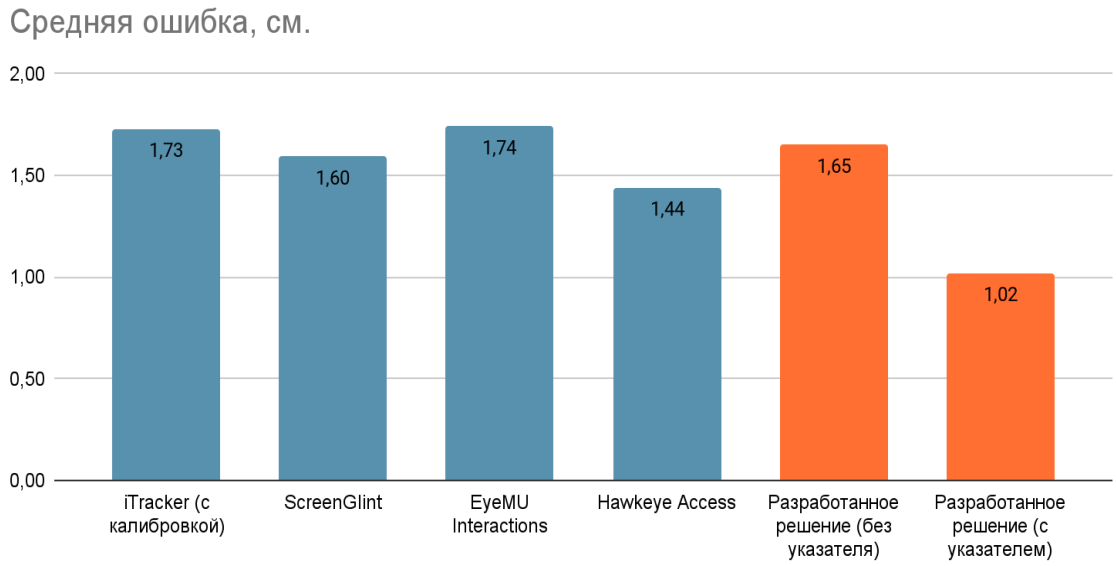


Рисунок 26. Сравнение точности с state-of-art аналогами

ОЦЕНКА ПРОИЗВОДИТЕЛЬНОСТИ

Помимо соответствия точности разрабатываемого решения state-of-art аналогам необходимо учесть также критерий производительности. Производительность решения складывается из скорости обработки данных в каждом из слоев:

- Backend-слой выполняет единственную задачу – предсказание точки взгляда на экране по алгоритму, описанному в главе 3. Этот процесс не зависит от конкретного сценария, шаги алгоритма всегда одинаковы, поэтому он поддается оценке в абсолютных единицах. На устройстве Apple iPhone 12 Mini под управлением iOS 15.5 получены следующие данные на основе выполнения алгоритма для 5000 кадров:
    - Среднее время обработки кадра – 0.00045 с.
    - Минимальное время обработки кадра – 0.000083 с.
    - Максимальное время обработки кадра – 0.0075 с.

  С учетом полученных данных можно заключить, что алгоритм достаточно производителен для обработки сигнала с частотой в 60 Гц, что согласуется с характеристиками камеры *TrueDepth*, используемой в качестве устройства ввода. Аналогичные выводы о производительности сделаны авторами статьи о применимости ARKit к задаче окулографии [22].
- Frontend-слой выполняет задачу преобразования данных, сформированных Backend-слоем, и диспетчеризации событий между обработчиками. Очевидно, что оценить производительность слоя в абсолютных единицах не представляется возможным, так как реальное число шагов алгоритма различается в зависимости от конфигурации каждого конкретного экрана приложения. Абсолютное время обработки каждого события линейно зависит от глубины иерархии элементов, так как выполняется их обход в глубину, а также квадратически от числа используемых обработчиков в силу необходимости обработки зависимостей между ними. Так как алгоритм концептуально повторяет таковой для системных обработчиков, имеются основания полагать, что в силу того, что UIKit поддерживает работу с интерфейсом с частотой кадров 60 Гц и более [27], тем же свойством обладает и Frontend-слой разрабатываемого решения.

ЗАКЛЮЧЕНИЕ

Разработанный фреймворк предоставляет разработчикам интерфейс, дающий возможность реагировать на данные о текущем положении взгляда на экране смартфона и выполнять на его основе произвольные действия. Созданное решение абстрагировано от конкретной предметной области и может быть применено для решения задач в любой из вышеперечисленных областей. В то же время, оно реализовано исключительно программными средствами и не требует каких-либо дополнительных аксессуаров, что позволяет увеличить потенциальный охват пользователей технологий окулографии при одновременном снижении трудозатрат и, как следствие, стоимости их внедрения.

Программный код разработанного фреймворка опубликован в открытом доступе в репозитории GitHub [28].

СПИСОК ЛИТЕРАТУРЫ

1. *Esiyok C. et al.* Novel hands-free interaction techniques based on the software switch approach for computer access with head movements // Universal Access in the Information Society. 2020. P. 1–15. https://doi.org/10.1007/s10209-020-00748-1

2. *Roig-Maimó M.F. et al.* Evaluation of a mobile head-tracker interface for accessibility // International Conference on Computers Helping People with Special Needs. Springer, Cham, 2016. P. 449–456.
https://doi.org/10.1007/978-3-319-41267-2_63

3. *Abbaszadegan M., Yaghoubi S., MacKenzie I.S.* TrackMaze: A comparison of head-tracking, eye-tracking, and tilt as input methods for mobile games // International Conference on Human-Computer Interaction. Springer, Cham, 2018. P. 393–405. https://doi.org/10.1007/978-3-319-91250-9_31

4. *Tupikovskaja-Omovie Z., Tyler D.* Clustering consumers' shopping journeys: eye tracking fashion m-retail // Journal of Fashion Marketing and Management: An International Journal. 2020. Т. 24. №. 3. P. 381–398.
https://doi.org/10.1108/JFMM-09-2019-0195

5. *Garbutt M. et al.* The embodied gaze: Exploring applications for mobile eye tracking in the art museum // Visitor Studies. 2020. Vol. 23. No. 1. P. 82–100.
https://doi.org/10.1080/10645578.2020.1750271

6. *Vogt M., Rips A., Emmelmann C.* Comparison of iPad Pro®'s LiDAR and TrueDepth capabilities with an industrial 3D scanning solution // Technologies. 2021. Vol. 9. No. 2. P. 25. https://doi.org/10.3390/technologies9020025

7. *Breitbarth A. et al.* Measurement accuracy and dependence on external influences of the iPhone X TrueDepth sensor // Photonics and Education in Measurement Science 2019. International Society for Optics and Photonics, 2019. Vol. 11144. P. 1114407. https://doi.org/10.1117/12.2530544

8. Number of smartphone users worldwide from 2016 to 2023 // Statista – The Statistics Portal for Market data, Market Research and Market Studies.
URL: https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/

9. *Krafka K. et al.* Eye tracking for everyone // Proceedings of the IEEE Conference On Computer Vision And Pattern Recognition. 2016. P. 2176–2184.
https://doi.org/10.1109/CVPR.2016.239

10. *Huang M. X. et al.* Screenglint: Practical, in-situ gaze estimation on smartphones // Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems. 2017. P. 2546–2557. https://doi.org/10.1145/3025453.3025794

11. *Brousseau B., Rose J., Eizenman M.* Smarteye: An accurate infrared eye tracking system for smartphones // 2018 9th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON). IEEE, 2018. P. 951–959.
https://doi.org/10.1109/UEMCON.2018.8796799

12. Hawkeye Access | Control your iOS device using your eyes.
URL: https://www.usehawkeye.com/accessibility

13. *Kong A. et al.* EyeMU Interactions: Gaze+ IMU Gestures on Mobile Devices // Proceedings of the 2021 International Conference on Multimodal Interaction. 2021. P. 577–585. https://doi.org/10.1145/3462244.3479938

14. Skyle 2 for iPad – eyeV. URL: https://eyev.de/en/ipad/

15. *Cicek M. et al.* Mobile head tracking for ecommerce and beyond //Electronic Imaging. 2020. Vol. 2020. No. 3. P. 303-1–303-12.
https://doi.org/10.48550/arXiv.1812.07143

16. *Kaufman A.E., Bandopadhay A., Shaviv B.D.* An eye tracking computer user interface // Proceedings of 1993 IEEE Research Properties in Virtual Reality Symposium. IEEE, 1993. P. 120–121. https://doi.org/10.1109/VRAIS.1993.378254

17.*Gibaldi A. et al.* Evaluation of the Tobii EyeX Eye tracking controller and Matlab toolkit for research // Behavior Research Methods. 2017. Vol. 49. No. 3. P. 923–946. https://doi.org/10.3758/s13428-016-0762-9

18.*Xu P. et al.* Turkergaze: Crowdsourcing saliency with webcam based eye tracking //arXiv preprint arXiv:1504.06755. 2015. https://doi.org/10.48550/arXiv.1504.06755

19.*Qiao X. et al.* A new era for web AR with mobile edge computing // IEEE Internet Computing. 2018. Vol. 22. No. 4. P. 46–55.
https://doi.org/10.1109/MIC.2018.043051464

20.*Taaban R.A., Croock M.S., Korial A.E.* Eye Tracking Based Mobile Application // International Journal of Advanced Research in Computer Engineering & Technology (IJARCET). 2018. Vol. 7. No. 3. P. 246–250.

21.*Heryadi Y. et al.* Mata: An Android Eye-Tracking Based User Interface Control Application // Journal of Games, Game Art, and Gamification. 2016. Vol. 1. No. 1. P. 35–40. https://doi.org/10.21512/jggag.v1i1.7249

22.*Greinacher R., Voigt-Antons J.N.* Accuracy Assessment of ARKit 2 Based Gaze Estimation // International Conference on Human-Computer Interaction. Springer, Cham, 2020. P. 439–449. https://doi.org/10.1007/978-3-030-49059-1_32

23.devicekit/DeviceKit: DeviceKit is a value-type replacement of UIDevice. URL: https://github.com/devicekit/DeviceKit

24.blendShapes | Apple Developer Documentation. URL: https://developer.apple.com/documentation/arkit/arfaceanchor/2928251-blendshapes

25.init(target:action:) | Apple Developer Documentation.
URL: https://developer.apple.com/documentation/uikit/uigesturerecognizer/1624211-init

26.Swift.org – Package Manager. URL: https://www.swift.org/package-manager/

27.Optimizing ProMotion Refresh Rates for iPhone 13 Pro and iPad Pro | Apple Developer Documentation.
URL: https://developer.apple.com/library/archive/technotes/tn2460/_index.html

28.GitHub – ReQEnoxus/gaze-tracker: UIGestureRecognizer extension based on GazeTracking. URL: https://github.com/ReQEnoxus/gaze-tracker

SOFTWARE FRAMEWORK FOR IMPLEMENTING USER INTERFACE INTERACTION IN IOS APPLICATIONS BASED ON OCULOGRAPHY

Nikita Afanasev[1][0000-0002-5306-9672]

Institute of Information Technology and Intelligent Systems, Kazan Federal University, 5 Kremlevskaya str., Kazan, 420008

[1]reqenoxus@gmail.com

Abstract

Usage of gaze tracking technologies for the purpose of user interface interaction in iOS applications is significantly hampered by the absence of a unified approach to their integration. Current solutions are either strictly limited to their own use-case or made solely for research purposes and thus inapplicable to real-world problems. The focus of this article is the development of a software framework that performs gaze tracking using native technologies and suggests a unified approach to the development of gaze-driven iOS applications.

Keywords*: gaze tracking, eye tracking, oculography, gesture recognizers, TrueDepth, ARKit, SceneKit, UIKit, iOS, UX, UI*

REFERENCES

1. *Esiyok C. et al.* Novel hands-free interaction techniques based on the software switch approach for computer access with head movements // Universal Access in the Information Society. 2020. P. 1–15. https://doi.org/10.1007/s10209-020-00748-1

2. *Roig-Maimó M.F. et al.* Evaluation of a mobile head-tracker interface for accessibility // International Conference on Computers Helping People with Special Needs. Springer, Cham, 2016. P. 449–456. https://doi.org/10.1007/978-3-319-41267-2_63

3. *Abbaszadegan M., Yaghoubi S., MacKenzie I.S.* TrackMaze: A comparison of head-tracking, eye-tracking, and tilt as input methods for mobile games // International Conference on Human-Computer Interaction. Springer, Cham, 2018. P. 393–405. https://doi.org/10.1007/978-3-319-91250-9_31

4. *Tupikovskaja-Omovie Z., Tyler D.* Clustering consumers' shopping journeys: eye tracking fashion m-retail // Journal of Fashion Marketing and Management: An International Journal. 2020. Т. 24. №. 3. P. 381–398.
https://doi.org/10.1108/JFMM-09-2019-0195

5. *Garbutt M. et al.* The embodied gaze: Exploring applications for mobile eye tracking in the art museum // Visitor Studies. 2020. Vol. 23. No. 1. P. 82–100.
https://doi.org/10.1080/10645578.2020.1750271

6. *Vogt M., Rips A., Emmelmann C.* Comparison of iPad Pro®'s LiDAR and TrueDepth capabilities with an industrial 3D scanning solution // Technologies. 2021. Vol. 9. No. 2. P. 25. https://doi.org/10.3390/technologies9020025

7. *Breitbarth A. et al.* Measurement accuracy and dependence on external influences of the iPhone X TrueDepth sensor // Photonics and Education in Measurement Science 2019. International Society for Optics and Photonics, 2019. Vol. 11144. P. 1114407. https://doi.org/10.1117/12.2530544

8. Number of smartphone users worldwide from 2016 to 2023 // Statista – The Statistics Portal for Market data, Market Research and Market Studies.
URL: https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/

9. *Krafka K. et al.* Eye tracking for everyone // Proceedings of the IEEE Conference On Computer Vision And Pattern Recognition. 2016. P. 2176–2184.
https://doi.org/10.1109/CVPR.2016.239

10.*Huang M. X. et al.* Screenglint: Practical, in-situ gaze estimation on smartphones // Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems. 2017. P. 2546–2557. https://doi.org/10.1145/3025453.3025794

11.*Brousseau B., Rose J., Eizenman M.* Smarteye: An accurate infrared eye tracking system for smartphones // 2018 9th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON). IEEE, 2018. P. 951–959.
https://doi.org/10.1109/UEMCON.2018.8796799

12.Hawkeye Access | Control your iOS device using your eyes. URL: https://www.usehawkeye.com/accessibility

13.*Kong A. et al.* EyeMU Interactions: Gaze+ IMU Gestures on Mobile Devices // Proceedings of the 2021 International Conference on Multimodal Interaction. 2021. P. 577–585. https://doi.org/10.1145/3462244.3479938

14.Skyle 2 for iPad – eyeV. URL: https://eyev.de/en/ipad/

15.*Cicek M. et al.* Mobile head tracking for ecommerce and beyond //Electronic Imaging. 2020. Vol. 2020. No. 3. P. 303-1–303-12. https://doi.org/10.48550/arXiv.1812.07143

16. *Kaufman A.E., Bandopadhay A., Shaviv B.D.* An eye tracking computer user interface // Proceedings of 1993 IEEE Research Properties in Virtual Reality Symposium. IEEE, 1993. P. 120–121. https://doi.org/10.1109/VRAIS.1993.378254

17.*Gibaldi A. et al.* Evaluation of the Tobii EyeX Eye tracking controller and Matlab toolkit for research // Behavior Research Methods. 2017. Vol. 49. No. 3. P. 923–946. https://doi.org/10.3758/s13428-016-0762-9

18.*Xu P. et al.* Turkergaze: Crowdsourcing saliency with webcam based eye tracking //arXiv preprint arXiv:1504.06755. 2015. https://doi.org/10.48550/arXiv.1504.06755

19.*Qiao X. et al.* A new era for web AR with mobile edge computing // IEEE Internet Computing. 2018. Vol. 22. No. 4. P. 46–55. https://doi.org/10.1109/MIC.2018.043051464

20.*Taaban R.A., Croock M.S., Korial A.E.* Eye Tracking Based Mobile Application // International Journal of Advanced Research in Computer Engineering & Technology (IJARCET). 2018. Vol. 7. No. 3. P. 246–250.

21.*Heryadi Y. et al.* Mata: An Android Eye-Tracking Based User Interface Control Application // Journal of Games, Game Art, and Gamification. 2016. Vol. 1. No. 1. P. 35–40. https://doi.org/10.21512/jggag.v1i1.7249

22.*Greinacher R., Voigt-Antons J.N.* Accuracy Assessment of ARKit 2 Based Gaze Estimation // International Conference on Human-Computer Interaction. Springer, Cham, 2020. P. 439–449. https://doi.org/10.1007/978-3-030-49059-1_32

23.devicekit/DeviceKit: DeviceKit is a value-type replacement of UIDevice. URL: https://github.com/devicekit/DeviceKit

24.blendShapes | Apple Developer Documentation. URL: https://developer.apple.com/documentation/arkit/arfaceanchor/2928251-blendshapes

25.init(target:action:) | Apple Developer Documentation.
URL: https://developer.apple.com/documentation/uikit/uigesturerecognizer/1624211-init

26.Swift.org – Package Manager. URL: https://www.swift.org/package-manager/

27.Optimizing ProMotion Refresh Rates for iPhone 13 Pro and iPad Pro | Apple Developer Documentation.
URL: https://developer.apple.com/library/archive/technotes/tn2460/_index.html

28.GitHub – ReQEnoxus/gaze-tracker: UIGestureRecognizer extension based on GazeTracking. URL: https://github.com/ReQEnoxus/gaze-tracker

СВЕДЕНИЯ ОБ АВТОРЕ

АФАНАСЬЕВ Никита Станиславович – бакалавр Института информационных технологий и интеллектуальных систем КФУ, г. Казань.

Nikita AFANASEV – undergraduate student of the Institute of Information Technologies and Intelligent Systems, Kazan (Volga region) Federal University, Kazan.

e-mail: reqenoxus@gmail.com

ORCID: 0000-0002-5306-9672

Материал поступил в редакцию 25июня 2022 года

УДК 004.413.2

ТЕХНОЛОГИЧЕСКИЙ ЦИКЛ РАЗРАБОТКИ ПОИСКОВОЙ СИСТЕМЫ, АГРЕГИРУЮЩЕЙ ЦИТАТЫ ИЗ КНИГ

Р. В. Мосоло́в[0000-0002-4399-4397]

Корпорация X5 Retail Group (Москва)

R.V.Mosolov@ya.ru

Аннотация

Описан технологический цикл разработки поисковой системы по 14 книгам философской направленности Л.А. Секлитовой и Л.Л. Стрельниковой, состоящий из 6 этапов работ. Идеи статьи могут быть полезны при проектировании и разработке программного обеспечения, агрегирующего цитаты из серий книг, монографий, научных публикаций или научно-периодических изданий, например, в целях хранения персональных ссылок на вторичные источники, часто пригождающиеся при написании научных статей и оформлении презентаций в педагогике. Поисковая система является результатом года работ автора и группы из примерно 30 волонтёров. Система представляет собой сервис, встроенный в веб-приложение. Технологический стек: Jade, CSS, JS, Node.js, Express.js, ESLint, Jest.

Ключевые слова*: цитаты писателей, агрегатор книг, цитаты философов, поисковые системы книги, цитаты из литературы, философские цитаты, оптимизация и продвижение в поисковых системах книг, лучшие цитаты из книг, разработка поисковой системы.*

ВВЕДЕНИЕ

В мае 2021 года в рамках деятельности региональной общественной организации «Центр Духовного Развития Человека «Золотая Раса»» (далее – ЦЗР; https://www.rusprofile.ru/id/1217700649473) была начата разработка сервиса, агрегирующего цитаты из 79 книг Л.А. Секлитовой и Л.Л. Стрельниковой (https://gold-race.org/searcher). Автор статьи отвечал за проектирование и программирование алгоритмов данного сервиса, разрабатывал как клиентскую (см. рис. 1), так и серверную части сервиса.

Цель создания сервиса – помочь пользователям найти ответ на философский вопрос при вводе определённого запроса в поисковую строку данного сервиса. Первый коммит сервиса был сделан 15 мая 2021 года (https://github.com/R-Mosolov/spircent/tree/7c190f7bc9919a50816001a230a87b2b2b23836d). В целях функциональной безопасности сервиса, а также защиты авторских прав авторов цитат исходный код сервиса хранится в приватном репозитории.

Все работы волонтёрами велись на благотворительной основе и, как правило, в свободное от работы время. В среднем один волонтёр был готов посвящать работам в Центре около 8 часов в неделю.

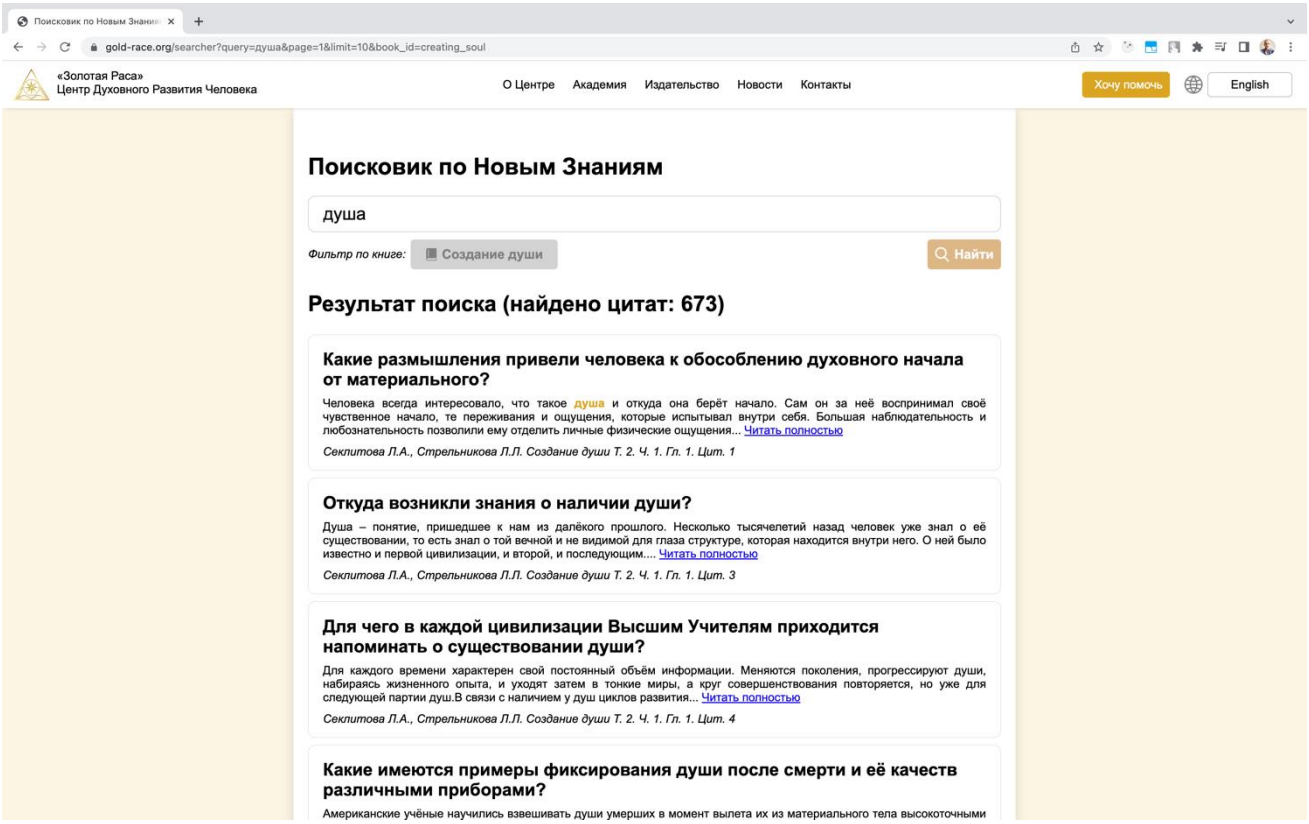


Рисунок 1. Пользовательский интерфейс сервиса по поиску книжных цитат

ТЕХНОЛОГИЧЕСКИЙ СТЕК

Следует отметить, что сами разработанные поисковые алгоритмы, с нашей точки зрения, не представляют существенной сложности, поскольку основная часть усилий была сосредоточена, скорее, на создании руководств пользователей, организационных условий для переноса цитат на сайт, а также обеспечении жизнеспособности IT-инфраструктуры.

В таблице ниже представлен список основных технологий, использованных при разработке сервиса. Отметим, что это неполный список того, чем автору статьи довелось заниматься в процессе разработки сервиса. Также в силу благотворительного характера работ в организации, способствовавшей привлечению и удержанию программистов узкого профиля (речь идёт о разработчиках серверной части и DevOps), автору требовалось освоить работу с циклом CI/CD. Если в первый месяцы существования веб-приложения данную настройку можно было спокойно делегировать PaaS-платформе Heroku (https://heroku.com), то после введения ряда санкций в отношении России, связанных с военными действиями на Украине (https://habr.com/ru/post/653605/) и как следствие приведших к появлению «Указа о мерах по обеспечению ускоренного развития отрасли информационных технологий в России», введённого Президентом В.В. Путиным (http://kremlin.ru/acts/news/67893), потребовалось осваивать, как вручную поднять и настроить виртуальный приватный сервер (VPS), чтобы перенести веб-приложение на российский сервер для обеспечения дальнейшей работоспособности созданного сервиса. Потребность эта особенно остро проявилась тогда, когда из-за ошибок, появляющихся в интерфейсе Heroku, стало невозможным публиковать какие-либо изменения на сайте. Предполагаем, что последнее было связано с поломкой интеграции Heroku с GitHub. Хотя руководство GitHub публично утверждало, что будет защищать права отдельно взятых разработчиков на свободное распространение кода независимо от страны их проживания (https://github.blog/2022-03-02-our-response-to-the-war-in-ukraine/), факт поломки, коррелирующий с политическими событиями, свидетельствовал об обратном. В итоге веб-приложение было перенесено на сервер под ОС Ubuntu 20.04 LTS (3 Гб ОЗУ, 3 ядра, 60 Гб SSD).

Таблица 1. Технологический стек сервиса «Поисковик по Новым Знаниям»

№	**Название технологии**	**Тип технологии**
1	Jade	Препроцессор
2	CSS	Язык стилизации
3	JS	Язык программирования

4	Node.js	Серверное окружение
5	Express.js	Серверный фреймворк

ТЕХНОЛОГИЧЕСКИЙ ЦИКЛ ПЕРЕНОСА ЦИТАТ

Перенос цитат состоял из 5–6 этапов работ (см. рис. 2). Вариативность с одним этапом была обусловлена тем, что работающий над переносом цитат волонтёр мог либо использовать Эксель-таблицы, либо отказаться от них. Эксель был выбран не просто так. Благодаря жёсткой структуре ячеек, Эксель-таблицы быстро преобразуются в формат JSON – основной формат хранения цитат в проекте. Однако, к нашему сожалению, выяснилось, что для большинства волонтёров работа с Экселем представляла техническую сложность. Поскольку проект реализовывался на благотворительной основе, то для его осуществления привлекались работники разных профессий, имеющие сильно различающийся уровень владения компьютером, и разных возрастов.

Приведём некоторые цифры:

– формально в проекте числилось около 20–30 волонтёров на разных стадиях его реализации;

– по факту, регулярно работало около 6–7 волонтёров;

– цитаты в Эксель-таблице прислал только 1 человек.

Предпочтение Ворда Экселю отдавалось не только из-за технических сложностей при переносе цитат. Также оно было связано с тем, что Эксель (по крайней мере, по умолчанию) не проверяет орфографию и пунктуацию. Последние являются критическим бизнес-требованием при работе с книгами, поскольку вопросы для цитат составлялись самими волонтёрами, а не брались в готовом виде из книг.

Ближе к завершению работ над сервисом, когда параллельно велись работы по интернационализации интерфейса сайта, нами эмпирически была обнаружена *«золотая середина» между Экселем и Вордом*, позволяющая, с одной стороны, использовать жёсткую систему ячеек таблиц, с другой стороны, включающая проверку орфографии и пунктуации в текстах. Данной «золотой серединой» стало заполнение текстов прямо в Ворде, но с предварительным созданием таблицы в нём.

Остановимся на подборе ключевых слов. Поскольку архитектурно сайт, на котором расположен сервис, задумывался с целью поисковой оптимизации его страниц, или SEO (отсутствие большого маркетингового бюджета – критический фактор для региональной общественной организации), что влияло на выбор его технологического стека, то для его страниц требовалось подбирать ключевые слова. Изначально мы планировали поручить данную задачу самим волонтёрам, чтобы снизить нагрузку на программистов, но, столкнувшись с тем, что для большинства из них освоение Экселя представляло сложность, мы решили отказаться от этой идеи и сосредоточить данную компетенцию в рамках Отдела Информационных Технологий организации. К сожалению, и это оказалось неверным решением, поскольку впоследствии, в процессе наращивания компетенции по SEO, мы узнали, что «поисковые пауки» не заполняют форм (тогда как её заполнение необходимо для отправки поискового запроса браузеру) [1, с. 245].

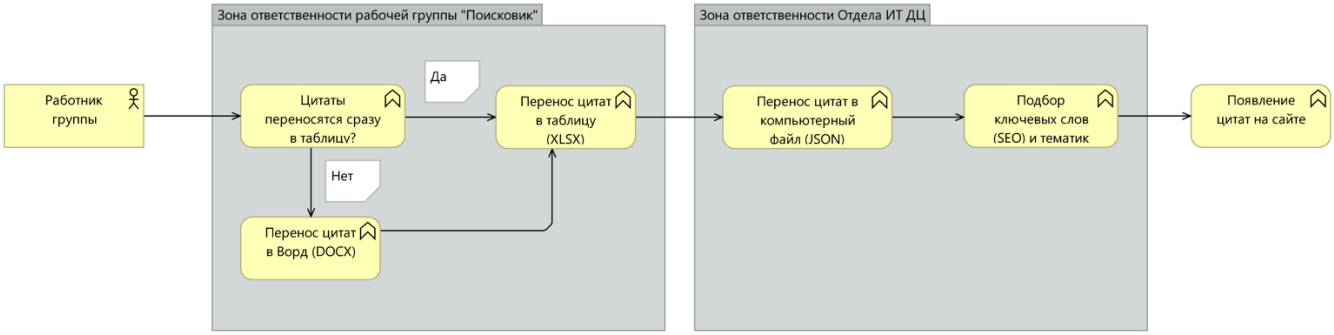


Рисунок 2. Блок-схема технологического цикла переноса цитат

ОРГАНИЗАЦИОННЫЕ И ТЕХНОЛОГИЧЕСКИЕ ОШИБКИ

Отметим ряд ошибок, совершённых на технологическом и организационном уровнях работы над проектом. Возможно, эта информация поможет разработчикам/заказчикам аналогичных сервисов избежать их впоследствии, учась на опыте других.

Первая ошибка (организационная) происходила из мотивационной основы, в рамках которой работали составители цитат. Дело в том, что книги делились на цитаты исключительно на благотворительной основе, что создавало ряд ограничений в механизмах мотивации работников. Так, например, у нас не было столь распространённых в России рычагов отсылки к окладу или премии. В связи

с этим была совершена одна из наиболее крупных, на наш взгляд, ошибок. Постановка задачи волонтёрам в первые 1–3 месяца работ утрированно звучала следующим образом: «Вот вам книга, у Вас есть год, будем ожидать вашего возвращения, когда будут готовы все цитаты».

Подобная постановка задачи впоследствии могла привести к осуществлению на практике «каскадной модели» управления проектом [2, с. 46]. Хотя такая модель использовалась бы не для управления группой разработчиков, а для управления специалистами в предметной области, тем не менее, как показала практика переноса цитат в первый месяц работ, она была не столь эффективна, как если бы мы использовали отдельные аспекты гибких (Agile) методологий, например, SCRUM.

В связи с этим в дальнейшем было принято **решение** разделить каждую книгу на смысловые части и просить волонтёров отправлять промежуточные результаты работ, тем самым распределяя объёмы работ по смысловым частям книги (главам), или спринтам (см. рис. 3). Это позволило сократить срок между началом работ волонтёров и выводом сервиса в продакшен примерно в 4 раза.

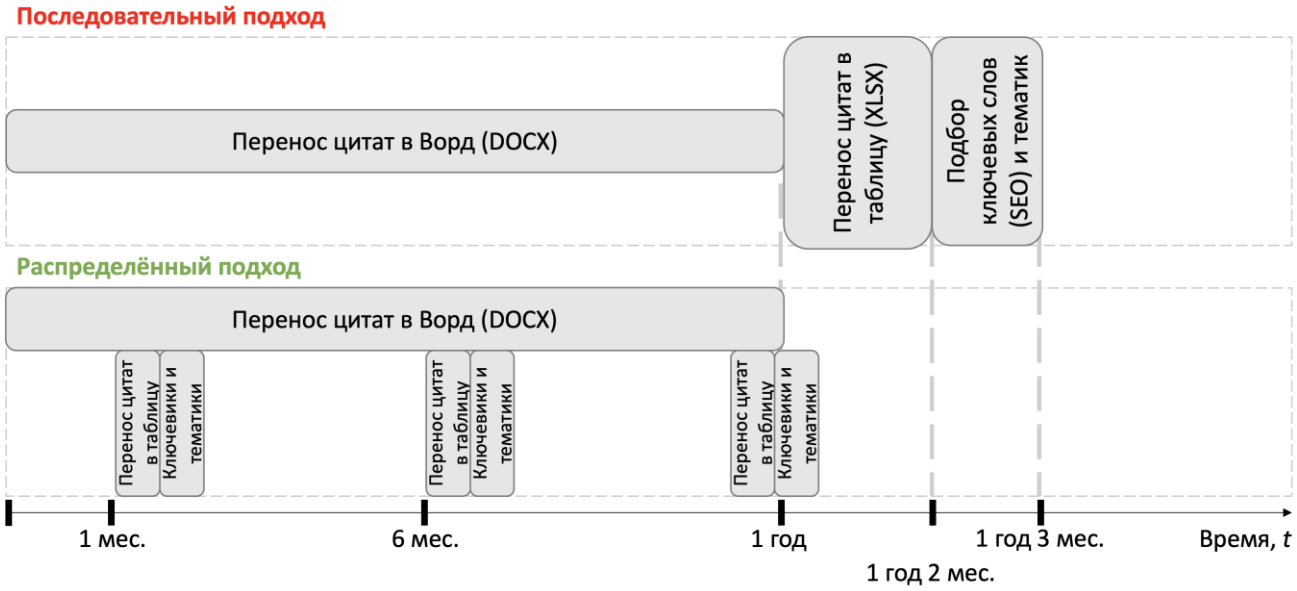


Рисунок 3. Различия последовательного и распределённого подходов при разделении книг на цитаты

Вторая ошибка (технологическая) заключалась в том, что поскольку сервис разрабатывался для региональной общественной организации, то последнее

накладывало существенный отпечаток на финансовую составляющую. Так, например, организация существует на членские взносы и пожертвования, что приводит к относительно нестабильному рекламному бюджету, а также, в принципе, малости его доли, могущей быть выделенной на организацию рекламных кампаний. В связи с этим большой упор на этапе проектирования сайта организации делался на поисковую оптимизацию страниц сайта (SEO). Это подводило к мысли о том, чтобы и страницы поисковой системы сделать оптимизированными под поисковые запросы пользователей. Однако данный замысел не получилось осуществить в полной мере вследствие архитектурной ошибки, допущенной на этапе проектирования сервиса, и отсутствия учёта того фактора, что «поисковые пауки» не заполняют никаких форм [1, с. 245], тогда как ввод поискового запроса пользователем являлся обязательным условием для начала работ с сервисом.

Решением, учитывающим названный фактор, стала разработка нового сервиса, процесс которого вёлся параллельно с разработкой поисковика и занял не менее 10 месяцев работ и подключения 6 из 12 отделов организации. Этим сервисом стал обучающий курс (https://gold-race.org/remote-format), спроектированный по всем канонам поисковой оптимизации [1], т. е. содержащий такие страницы сайта, которые с большей вероятностью позволят пользователю найти ответ на свой вопрос, вводимый в поисковой системе Яндекса/Гугла/др.

ОБ ОГРАНИЧЕНИЯХ ИСКУСТВЕННОГО ИНТЕЛЛЕКТА В РАСПОЗНАВАНИИ НОВОГО ПОНЯТИЙНОГО АППАРАТА

Основная сложность, с которой мы столкнулись при стремлении полностью автоматизировать процесс переноса цитат, состояла в том, что цитаты из переносимых книг содержали новый понятийный аппарат, ранее малораспространённый даже в кругах философов, например, «программа жизни», «энергокомпозит души», «ячейка матрицы души», «отрицательная Система Бога» и др. Более полный их список можно посмотреть в [3]. В связи с этим построение интеллектуальных алгоритмов нейронных сетей, базирующихся на анализе семантической составляющей абзацев и агрегирующих на их основе заголовки цитат, представлялось процессом маловозможным, поскольку чем более сложным является поня-

тийный аппарат агрегируемых текстов, тем менее вероятно, как показал эмпирический опыт, можно встретить хорошо размеченные корпуса текстов для него, как, например, в (https://ling.hse.ru/krut).

ОБ АЛГОРИТМИЧЕСКОЙ СЛОЖНОСТИ ПОИСКОВОЙ СИСТЕМЫ

Разработанная нами поисковая система, на наш взгляд, не является алгоритмически сложным программным решением. Поскольку для поиска цитаты мы не использовали ни парсинга PDF-страниц посредством технологии Elasticsearch [4] или Solr, ни сложных методик формирования предложений для автозаполнений поисковых запросов, реализованных, например, в поисковой системе Google [1], ни оптимизаций производительности за счёт сокращения количества циклов [5, с. 294] на низкоуровневых языках программирования вроде C/C++, встраиваемых в веб через WebAssembly [6]. Функции поиска были выполнены на высокоуровневом языке программирования JavaScript посредством использования исключительно нативных методов данного языка, таких как Array#filter() (https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Array/filter) для фильтрации цитат по заголовку и основному тексту, Array#slice() (https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Array/slice) для построения пагинации и String#includes() (https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Array/slice) для поиска точного, регистронезависимого совпадения с запросом пользователя.

Основная сложность нашего проекта состояла, скорее в том, чтобы:

1. Технически содействовать в организации работы волонтёров и консультировать о технических возможностях и ограничениях реализации новых идей;
2. Управлять небольшой командой, состоящей из 2 разработчиков, 2 специалистов по заполнению JSON-модулей текстами цитат и 2 веб-дизайнеров. Учитывая, что разработчиков было всего 2 – первый (автор данной статьи) на тот момент имел 3 года опыта разработки клиентской части веб-приложений, а второй до перехода в проект писал на Ассем-

блере, имея опыт на JavaScript не более 1 года – требовалось мобилизовать силы и вести full-stack разработку, занимаясь видами работ сразу нескольких специалистов. В числе осваиваемых специальностей были фронтенд-разработка, бэкенд-разработка, DevOps и роль владельца продукта, принимавшего участие на собраниях членов Совета Центра «Золотая Раса» с последующим сбором требований и оповещением о технических ограничениях специалистов Отдела Информационных Технологий Центра «Золотая Раса».

ЗАКЛЮЧЕНИЕ

На июнь 2022 года в поисковую систему были добавлены текстовые модули по цитатам из 14 книг разного объёма представленности на сайте.

Благодарности

Выражаем благодарность Екатерине Вербовской (Германия), автору идеи поисковой системы по книгам Л.А. Секлитовой и Л.Л. Стрельниковой. Также благодарим М.А. Горкавенко (Москва) и В.Г. Шмакова (Московская обл.) за их помощь в организации волонтёрской работы по составлению цитат и Е.Е. Белковского (Беларусь) за разработку утилиты, проверяющей качество цитат на этапе их интеграции в виде текстовых модулей в кодовую базу (из расширений XLS/XLSX в JSON).

СПИСОК ЛИТЕРАТУРЫ

1. *Энж Э., Спенсер С. Стрикчиола Д.* SEO – искусство раскрутки сайтов. 3-е изд. СПб: 2017. 816 с.

2. *Ларман К.* Применение UML 2.0 и шаблонов проектирования, 3-е изд. : Пер. с англ. СПб.: ООО «Диалектика», 2020. 736 с. : ил. Парал. тит. англ.

3. Словарь космической философии. М.: Свет, 2021. 304 с. (Серия «За гранью непознанного»).

4. *Тарнбулл Д., Берримен Дж.* Релевантный поиск с использованием Elasticsearch и Solr. / пер. с англ. Киселев А. Н. М.: ДМК Пресс, 2018. 408 с.: ил.

5. *Таненбаум Э., Остин Т.* Архитектура компьютера. 6-е изд. СПб.: Питер, 2013. 816 с.: ил.

6. *Галлан Жерар*. WebAssembly в действии. СПб.: Питер, 2022. 496 с.: ил. (Серия «Библиотека программиста»).

DEVELOPING TECHNOLOGICAL CYCLE OF SEARCH SYSTEM THAT AGREGATES CITATIONS BY BOOKS

R. V. Mosolov[0000-0002-4399-4397]

Corporation "X5 Retail Group" (Moscow)

R.V.Mosolov@ya.ru

Abstract

In this article, we have described the technological cycle to develop the search system by 14 philosophical books by L.A. Seklitova, and L.L. Strelnikova. The cycle contained 6 steps of work. The ideas from the article may be useful to project, and develop a software, aggregating citations from books series, monographs, scientific periodicals, or scientific articles. For example, this experience may be useful for creating customized links on secondary sources that needs at a stage of writing scientific articles and design of presentations in Pedagogy. The search system is the result of 1 year work by the article author, and the group of around 30 volunteers. The system is represented a service, integrating in the web application. The technological stack contains Jade, CSS, JS, Node.js, Express.js, ESLint, Jest.

Keywords*: search system, searching system, search by books, search by book, search by citations, citations aggregator, aggregator by citations, books aggregate, citations data aggregators, develop search engine.*

REFERENCES

1. J*enzh Je., Spenser S. Strikchiola D.* SEO – iskusstvo raskrutki saĭtov. 3-e izd. SPb: 2017. 816 s.

2. *Larman K.* Primenenie UML 2.0 i shablonov proektirovanija, 3-e izd. : Per. s angl. SPb.: OOO «Dialektika», 2020. 736 s. : il. Paral. tit. angl.

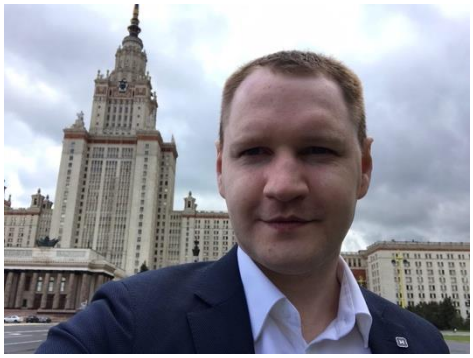
3. Slovar' kosmicheskoj filosofii. M.: Svet, 2021. 304 s. (Serija «Za gra-n'ju nepoznannogo»).

4. *Tarnbull D., Berrimen Dzh.* Relevantnyj poisk s ispol'zovaniem Elasticsearch i Solr. / per. s angl. Kiselev A. N. M.: DMK Press, 2018. 408 s.: il.

5. *Tanenbaum Je., Ostin T.* Arhitektura komp'jutera. 6-e izd. SPb.: Piter, 2013. 816 s.: il.

6. *Gallan Zherar.* WebAssembly v dejstvii. SPb.: Piter, 2022. 496 s.: il. (Serija «Biblioteka programmista»).

СВЕДЕНИЯ ОБ АВТОРЕ



МОСОЛО́В Роман Валерьевич – бакалавр социологии (Казанский федеральный университет (КФУ)), магистр компьютерных наук (Институт информационных технологий и интеллектуальных систем КФУ), старший разработчик в корпорации X5 Retail Group (Москва). В 2021 г. разрабатывал клиентскую часть сервисов изменения персональных данных и сбора данных (опросов) для 259 тыс. работников ТС «Пятёрочки», с мая 2022 г. по настоящее время участвует в разработке клиентской части сервиса базы знаний для 40 тыс. работников ТС «Перекрёстка».

Roman Valerievich MOSOLOV – Bachelor of Sociology (Kazan Federal University), Master of Computer Science (HS ITIS), senior developer at corporation X5 Retail Group (Moscow). Roman has developed client side of changing personal data and grabbing data (polls) services for 259K employees of “Pyaterochka” (supermarket chain) in 2021. At current time, he is developing client side of knowledge base service for 40K employees of “Perekrestok” (supermarket chain).

email: R.V.Mosolov@ya.ru

ORCID: 0000-0002-4399-4397

Материал поступил в редакцию 25 июля 2022 года

УДК 004.5, 004.415

РЕКОМЕНДАТЕЛЬНАЯ СИСТЕМА ПОДБОРА ИГРОКОВ В КОМАНДНЫХ ВИДАХ СПОРТА, ПОСТРОЕННАЯ НА ОСНОВЕ МАШИННОГО ОБУЧЕНИЯ

Р. Р. Шигапов[1][0000-0002-3163-0959], А. А. Ференец[2][0000-0002-7859-9901]

Институт информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета
ул. Кремлевская, 35, г. Казань, 420008

[1] shigapov.rinat.2000@gmail.com, [2] ist.kazan@gmail.com

Аннотация

Описана разработка на основе машинного обучения рекомендательной системы подбора игроков на примере хоккея с возможностью расширения ее использования в различных командных видах спорта. Для каждого вида спорта рассмотрены амплуа и характеристики игроков, которые были структурированы и разделены на общие группы. Проанализирована информация о хоккее, футболе, баскетболе и волейболе. Для каждого из рассмотренных параметров выведены коэффициенты, показывающие их влияние на результат матча. Протестированы модели, построенные на основе различных алгоритмов машинного обучения. Создан веб-интерфейс приложения.

Ключевые слова*: спорт, хоккей, подбор игроков, рекомендательная система, машинное обучение*

ВВЕДЕНИЕ

Переход игрока из одного спортивного клуба в другой (трансфер) является неотъемлемой частью командных игр. За последние несколько лет количество трансферов неуклонно росло, как и общая сумма денег, потраченная клубами на покупку новых игроков [1].

Спортивные клубы заинтересованы в новых игроках для улучшения результатов команды. Но нередко случается, что игрок переоценён и не приносит ожи-

даемую от него пользу [2]. Из-за неудачной трансферной политики команда может проигрывать матчи, клуб — потерять большие деньги, а карьеры купленных игроков — пойти на спад. В связи с этим сейчас в основных спортивных лигах как мира, так и России автоматизируется сбор данных о статистике определённых игроков, матчей или же команд, а их последующий анализ является актуальной задачей [3]. Так появилась гипотеза, что можно на основе машинного обучения реализовать рекомендательную систему для подбора игроков в командных видах спорта.

Для реализации подобной системы требуется решить следующие задачи: изучить влияние различных доступных статистических параметров игроков на результаты команды, подготовить набор данных (датасет), обучить модели машинного обучения и выбрать наилучшую из них, а также создать алгоритм рекомендации и реализовать пользовательский интерфейс.

ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ

Проанализированы списки наиболее популярных в мире видов спорта [4, 5] и выделены четыре: футбол, хоккей, волейбол, баскетбол.

Для поиска новых талантливых игроков клубы создают специальные скаутинговые отделы. Как правило, в этих отделах работают люди с опытом в игре, например, игроки, которые закончили свою спортивную карьеру. Раньше в скаутских отделах полагались только на мнение авторитетных лиц. В таких отделах при проведении анализа игроков основное влияние на результат имел человеческий фактор, который не всегда объективен: для анализа требуется очень качественное представление информации, так как объём статистических данных весьма большой. Сейчас же с каждым годом новые технологии все чаще стали внедряться в ведение статистики и аналитики в спорте. Появляются новые системы, которые предлагают различные функции для усовершенствования команды:

- InStat — веб-приложение, направленное на получение статистики матчей, создание специальных тренировок и отчётов по командам соперников. Основным видом спорта является футбол, также некоторые функции развиты для баскетбола и хоккея. Отметим, что для сбора статистики каждый матч просматривают два человека, что делает данный способ трудозатратным. Особенностью является то, что статистика привязана к видео [6]. Главная

особенность платформы — возможность просматривать статистику вместе с эпизодами игры.

- IceBerg — средство для сбора статистики с помощью машинного обучения и специального оборудования. На основе полученных данных система может создавать отчеты по играм и специальные тренировки как для определенного игрока, так и для всей команды [7].
- DataVolley4 — программное обеспечение, с помощью которого команды создают отчёты по командам соперника и изменяют тактики своей игры [8].

Исходный код названных продуктов — коммерческая тайна, а результаты точности реализуемых ими функций неизвестны. Отметим также, что в данных системах отсутствует функция рекомендаций.

Помимо общепринятых характеристик выступления команды, в каждом виде спорта у игроков есть свои характеристики, которые показывают, насколько успешно выступает данный игрок. Важность этих характеристик зависит не только от вида спорта, но и от позиции, на которой выступает игрок (амплуа).

Для футбола (ф), хоккея (х), баскетбола (б) и волейбола (в) характерны свои позиции игроков:

- нападающий (х, ф),
- полузащитник (ф),
- защитник (х, ф),
- вратарь (х, ф),
- либеро (в),
- связующий (в),
- доигровщик (в),
- диагональный (в),
- центральный связующий (в),
- игрок передней линии (б),
- игрок задней линии (б).

В данных видах спорта прослеживаются закономерности в позициях игроков, которых можно разделить на игроков нападения и обороны, а также вратарей.

В хоккее для полевых игроков выделяют 10 основных параметров, а для вратарей – 7 [9]. В футболе для полевых игроков выделяют 13 основных параметров, а для вратарей – 8 [10]. В баскетболе используют 9 параметров для игроков [11], а в волейболе — 7 [12]. Отметим, что для каждого вида спорта и каждого амплуа игрока есть свои характерные параметры.

Если сравнить статистику по разным видам спорта, то можно найти много общих характеристик, однако среди них есть и свойственные каждому отдельному спорту. Можно разделить все признаки на несколько групп:

- игровые (количество проведённых матчей, время на площадке),
- вратарские (для футбола и хоккея),
- атакующие (голы, передачи, набранные очки),
- оборонительные (перехваты, блок-шоты, отборы, блокированные удары, эффективность приёма),
- штрафные (штрафные минуты, фолы, красные и желтые карточки).

Во всех выбранных видах спорта для победы необходимо набрать больше очков, чем соперник. Наиболее отличается волейбол: игра ограничивается не временем, а количеством очков — для победы необходимо выиграть 3 сета, в каждом из которых набрать 25 очков, но с разницей в 2 очка. В остальных играх матч ограничивается временем:

- футбол — 2 тайма по 45 минут;
- хоккей — 3 периода по 20 минут;
- баскетбол — 4 периода по 10 минут.

Другим важным аспектом командных игр являются замены. В этом компоненте наиболее жесткие правила в футболе. С 2020 года из-за COVID-19 разрешены пять замен за весь матч, до этого было всего три. В волейболе разрешены 6 замен за партию. В баскетболе и хоккее ограничений на замены нет. Правила замен влияют на количество игроков в команде:

- В футболе за команду играют 10 полевых игроков плюс вратарь, и на замене сидят от 3 до 7 человек в зависимости от лиги [13].
- В волейболе в команде может быть максимум 14 человек, в игре участвуют 6 человек и 8 запасных [14].
- В баскетболе на площадке находятся 5 человек и от 5 до 7 – на замене [15].

- В хоккее на площадке находятся 5 игроков и вратарь. На замене сидят примерно еще 3 пятёрки (игроки, играющие вместе на площадке) и запасной вратарь, то есть всего в хоккейный состав входят примерно 22 игрока [16].

Если сравнивать игры по скорости изменения ситуации на площадке, то безоговорочно впереди будет хоккей, так как в отличие от остальных видов спорта игроки тут катаются на коньках, что в разы быстрее бега [17].

Мы можем сделать вывод, что несмотря на то, что в каждом виде спорта существуют свои особенности, в целом структура игр имеет общую базу и строится на единых принципах. Благодаря этому, создав алгоритм рекомендательной системы, можно будет расширить его и на другие виды спорта.

В данной работе показана реализация рекомендательной системы на основе хоккея с шайбой.

ПРОЕКТИРОВАНИЕ РЕКОМЕНДАТЕЛЬНОЙ СИСТЕМЫ

Игровые характеристики — это основные характеристики для любого амплуа. Для более релевантных данных был сделан выбор игроков, которые сыграли определённое количества игр в сезоне, в каждой из которых провели минимальное среднее время на площадке. Эти данные зависят от выбранного вида спорта.

В Континентальной хоккейной лиге (КХЛ) в регулярном сезоне 56 игр. При этом в команде больше игроков, чем может быть заявлено на матч. Из-за постоянных ротаций в составе или же, например, травм игрок может сыграть не в каждом матче. Для корректных рекомендаций игрок, сыгравший в 25% матчей сезона, может учитываться.

Хоккейный матч длится 60 минут, как правило в команде 4 пятёрки (5 человек) полевых игроков. Так как время между игроками делится не поровну (лидеры обычно играют больше), то можно считать минимальным необходимым

временем, проведённым на площадке, 15% общего времени всех игр. Таким образом, если игрок сыграл в 14 матчах сезона более 9 минут, то он может быть рекомендован системой.

Для полевых игроков (нападающих и защитников) параметры почти одинаковы, но оказывают разное влияние. Для нападающих более существенны атакующие характеристики, так как их основная задача забивать голы в ворота противников. В настоящее время, если игрок не умеет или не хочет отрабатывать в защите, существует практика не ставить его в состав. Также последние несколько лет стали популярны такие термины, как «нападающий оборонительного плана» и «атакующий защитник» [18]. Эти термины означают, что игроки выполняют функции, не связанные напрямую со своим амплуа. Например, «нападающий оборонительного плана» может забивать мало голов и не набирать большого количества очков, но зато такой игрок может выходить в меньшинстве и блокировать собой броски. Как правило, такие игроки «таскают рояль», именно поэтому их действия могут оставаться незамеченными, и их часто недооценивают. Чтобы в разрабатываемой системе не было таких случаев, для полевых игроков необходимо рассматривать все доступные характеристики, как атакующие, так и оборонительные.

Если предыдущие параметры добавляли баллы в оценку, то штрафные характеристики вычитают их, так как это действия, которые отрицательно сказываются на игре команды. Зачастую в игре двух равных команд основным фактором успеха является именно игра в неравных составах, и если игроки не дисциплинированы, заработав удаление, они могут подвести команду.

Для оценки нападающего необходимо понять, с какими весами брать каждый параметр.

На официальном сайте КХЛ [19] представлены 25 параметров полевых игроков.

Так как некоторые характеристики могут повторять друг друга, например, заброшенные шайбы плюс передачи – это очки, а в других лигах и видах спорта может не быть такой подробной статистики, приведем основные параметры:

- Количество проведенных игр — игровая, сколько игрок отыграл матчей в сезоне;
- Заброшенные шайбы — атакующая, количество голов за сезон или матч;
- Передачи — атакующая, количество результативных передач, после которых были забиты голы;
- Плюс/Минус — игровая, коэффициент полезности игрока: при участии в атаке, которая завершилось голом, прибавляется балл, при нахождении на поле и взятии ворот командой соперника балл вычитается;
- Штрафное время — штрафная, удаление игрока с поля за нарушение, в зависимости от вида нарушения игрок может получить 2, 5, 10 или 20 минут штрафного времени;
- Время на площадке — игровая, время, которое игрок непосредственно играл;
- Броски по воротам — атакующая, количество бросков, произведенное игроком в створ ворот;
- Вбрасывания — атакующая, характеристика, как правило, относящаяся к центральному нападающему; при вводе шайбы судьёй в игру происходит борьба центральных нападающих за шайбу;
- Выигранные вбрасывания — атакующая характеристика;
- Силовые приемы — оборонительная, количество силовых приёмов в рамках правил против соперника;

- Фолы против игрока — атакующая, количество заработанных игроком штрафов соперника на себе;
- Блокированные броски — оборонительная, броски, которые игрок заблокировал, не дав дойти шайбе до ворот.

Чтобы доказать предположение об атакующих, оборонительных и штрафных характеристик, проделаем следующие действия:

1. Для каждой команды и каждого игрока выводится среднее значение выбранных параметров, кроме количества проведённых игр и времени на площадке из-за того, что они нужны для составления выборки, а не рекомендации игроков. Возьмем для примера казанский «Ак Барс» (см. таблицу 1).

Таблица 1 — Параметры нападающих казанского «Ак Барса» в среднем за матч

голы	передачи	плюс/минус	Штрафы	Броски	вбрасыв.	выиг. вбрасыв.	силов. прием.	фолы против	блок. броски
2.5	2.7	1.9	6.7	22.3	38.5	31.4	9.7	2.8	5.6

Таблица 2 — Параметры нападающих в среднем для игрока

голы	передачи	плюс/минус	Штрафы	Броски	вбрасыв.	выиг. вбрасыв.	силов. прием.	фолы против	блок. броски
0.14	0.17	0.32	0.53	1.23	0.43	3.7	0.57	0.13	0.35

2. Проверяется, как коррелируют данные команд из предыдущего пункта с результатами команды. В таблице 3 можно заметить верность рассуждений по поводу параметров: забитые шайбы, передачи, плюс/минус, штрафы, броски, выигранные вбрасывания. Параметр вбрасывания не должен учитываться, так как количество вбрасываний никак не может повлиять на игру, в отличие от выигранных. Хотя параметры «блокированные броски» и «фолы против» слабо коррелируют с результатами, их не учитывать нельзя. Можно предположить, что слабая корреляция связана со сложностью выявления заблокированных бросков — бросок заблокирован, гола нет. С «фолами против» объяснить сложнее, но, если вспомнить о важности большинства/меньшинства, этот параметр нельзя не учитывать. Возможно, влияет подготовка «спецбригад» для большинства и меньшинства, и этот параметр теряется.

Таблица 3 — Корреляция параметров нападающих в среднем по клубам и результатами матчей

голы	передачи	плюс/минус	Штрафы	броски	вбрасыв.	выиг. вбрасыв.	силов. прием.	фолы против	блок. броски
0.75	0.71	0.74	-0.66	0.53	-0.11	0.30	0.10	0.16	0.37

Для оценки игрока воспользуемся данными, полученными в доказательствах. Разделим данные таблицы 3 на данные таблицы 2, убрав вбрасывания (см. таблицу 4).

Таблица 4 — Весовые коэффициенты игровых характеристик для нападающих

голы	передачи	плюс/минус	Штрафы	Броски	вбрасыв.	выиг. вбрасыв.	силов. прием.	фолы против
5.35	4.18	2.31	-0.65	0.43	0.08	0.17	1.23	0.54

В отличие от нападающих у защитников меньше действий, которые можно зафиксировать. В хоккейной среде есть фраза: «Лучший защитник тот, которого не видно», то есть главная задача игрока обороны – не пропустить шайбу. Именно поэтому у защитников обычно смотрят на параметр плюс/минус, который показывает, насколько успешно защитник сыграл в обороне. Среди игроков обороны также есть свои исключения – «атакующие защитники», которые иногда могут набрать очков больше, чем нападающие. Эта тенденция пришла из-за океана из Национальной хоккейной лиги (НХЛ) [20]. Поэтому, как и у нападающих, при выводе оценки берутся оборонительные, атакующие и штрафные характеристики. Метод оценки игроков защиты схож с нападающими, за исключением параметра «выигранные вбрасывания», так как защитники редко принимают в них участие. В таблицах 5 и 6 мы получили ожидаемые параметры, учитывая результаты нападающих. Для более релевантных оценок параметры «фолы против» и «броски» берутся с другими весами.

Таблица 5 — Параметры защитников в среднем для игрока

голы	пере-дачи	плюс/минус	Штра-фы	Брос-ки	выиг. вбра-сыв.	силов. прием	фолы против
0.08	0.14	0.31	0.49	0.87	0.67	0.09	0.62

Таблица 6 — Корреляция параметров защитников в среднем по клубам и результатами матчей

голы	пере-дачи	плюс/минус	Штра-фы	Брос-ки	выиг. вбра-сыв.	силов. прием	фолы против
0.23	0.57	0.74	-0.24	0.25	0.18	0.04	0.35

Таблица 7 — Весовые коэффициенты игровых характеристик для защитников

голы	пере-дачи	плюс/минус	Штра-фы	Брос-ки	выиг. вбра-сыв.	силов. прием	фолы против
2.87	4.07	2.39	-0.49	0.28	0.27	0.44	0.56

Позиция вратаря есть не во всех видах спорта. В выбранных видах спорта позиция вратаря встречается только в футболе и хоккее. Вратарь защищает ворота, чтобы соперник не забил гол. На успешности защиты основана статистика вратарей. У них совсем другие параметры, из-за чего их характеристики были вынесены в отдельную вратарскую категорию. В неё входят:

- отражённые броски,
- штрафное время,
- выигрыши,
- броски в створ ворот,
- процент отражённых бросков,
- время на площадке.
- проигрыши,
- пропущено шайб,
- коэффициент надежности (60 минут × пропущенные шайбы / время на площадке),
- «сухие» игры (игры, в которых вратарь пропустил 0 голов).

Также в хоккее, в отличие от полевых игроков, как правило, замена вратаря встречается крайне редко. Это может произойти в случае травмы или если у основного вратаря не пошла игра.

Другой важной особенностью в построении рекомендательной системе стало количество вратарей. Если в каждой хоккейной команде может быть около 15 нападающих, 10 защитников, то вратарей обычно максимум 3. К тому же третий вратарь, обычно только номинальный, например, заканчивающий карьеру или, наоборот, очень молодой. Это происходит из-за того, что третий вратарь крайне редко играет, на матч заявляются только два вратаря. Существуют разные тактики использования вратарей, но зачастую в клубе есть один основной вратарь, который играет 90% матчей, и второй вратарь, который хуже по классу и выходит на игру лишь иногда. Поэтому некоторые тренеры дают резервным вратарям дополнительные задания, такие как ведение статистики [21].

Основываясь на вышесказанном, можно сделать вывод, что система для вратарей не будет пользоваться спросом, а количество статистических параметров не даст её развить, как этого хотелось бы.

Рекомендации для команд строятся на основе игроков, входящих в состав этой команды. С помощью машинного обучения предполагается оценка игроков на будущий сезон, матч или временной интервал. Далее для каждой команды рассчитывается средний рейтинг для каждого амплуа. Затем формируется список из определённого количества игроков, рейтинг которых выше среднего по команде в этом амплуа. Тем самым выбранные игроки будут положительно влиять на команду. Также к рекомендации прикрепляется ссылка на статистику каждого игрока.

Исходя из наблюдений за трансферами за прошлые года, перед заключением основного контракта клуб заключает с игроком «пробный» или «просмотровый» контракт [22]. Обычно это контракт со спортсменом, который ни разу не работал с тренерским штабом и не знаменит. Это происходит из-за того, что часто на игрока могут влиять неспортивные причины, например, межличностные. Поэтому при использовании клубами созданной системы, если игрок не знаком руководству, то рекомендуется подписывать игрока изначально на «пробные» контракты.

Из выявленных проблем можно указать, что на рынке свободных агентов очень мало вратарей, и, как правило, команды укомплектованы этим амплуа:

вряд ли какая-то команда захочет отдавать вратаря при успешных результатах, ведь считается, что «вратарь — это половина команды».

РЕАЛИЗАЦИЯ И ТЕСТИРОВАНИЕ РЕКОМЕНДАТЕЛЬНОЙ СИСТЕМЫ

Для создания оценок и работы системы необходимы были данные игроков со статистикой за каждый сыгранный матч. В интернете необходимого набора данных (датасета) в открытом доступе найдено не было.

Для создания датасета были взяты данные с официального сайта Континентальной хоккейной лиги [19]. КХЛ — это международная лига, которая считается сильнейшей в Европе и второй в мире. Также на основе результатов российских команд в этой лиге выбирается победитель чемпионата России.

Для решения этой задачи был реализован парсер на языке программирования Python с использованием библиотек BeautifulSoup4 и requests [23–25].

Для вариации возможных улучшений точности модели машинного обучения данные были собраны для матчей по отдельности и для всего сезона вместе. В общем было получено: 451 107 записей об игроках в матчах; 19 675 записей об игроках в сезонах; 3 578 игроков.

Для предобработки датасета были проведены следующие действия:

1. Время, проведённое на площадке, для более удобного расчёта было переведено в секунды из минут плюс секунд.
2. Из-за переименования некоторых клубов была проделана работа для унифицирования названий в датасете. Например, в 2010 году клубы «ХК МВД» и «ХК Динамо» объединились в «ОХК Динамо», для каждого игрока с одним из этих клубов решено записывать клуб «Динамо Москва» [26].
3. Были удалены некоторые данные, которые могут испортить статистику, например, матч между Авангардом и Витязем от 9 января 2010 года, закончившийся массовой дракой. Команды не доиграли матч и заработали 14 штрафных часов [27]. В подобных случаях в датасете вместе результата матча прописано «-:-». Такие матчи были проанализированы и удалены.

4. Были выбраны данные, начиная с сезона 2014/2015, из-за того, что до этого параметры «фолы против», «силовые приёмы» и «блокированные броски» не считались.
5. Были выбраны игроки, которые минимум в 14 матчах сыграли больше 9 минут (540 секунд).
6. Датасет был разделён по амплуа игроков.
7. Набор данных для нападающих и защитников был разделён на данные, включающие игры плей-офф и не включающие эти игры.

Для обучения модели использовались стандартные для этой задачи технологии: язык программирования Python и библиотека scikit-learn [23, 28]. Были рассмотрены следующие алгоритмы машинного обучения, решающие задачу регрессии:

- Линейная регрессия [29] — самый простой алгоритм машинного обучения, который строит модель зависимости одного параметра от одного или нескольких других.
- Случайный лес [30] — алгоритм, реализованный с помощью решающих деревьев.
- XGBoost [31] — алгоритм, построенный на ансамбле слабых предсказывающих моделей
- Регрессия LASSO [32] — линейная регрессия с добавленной регуляризацией, которая заключается во введении дополнительного параметра. В этом алгоритме используется регуляризация L1, которая взвешивает ошибки по абсолютному значению.
- Ридж-регрессия [33] — алгоритм, аналогичный предыдущему, но он использует регуляризацию L2, то есть взвешивает ошибки по их квадрату.

Для оценки моделей была выбрана характеристика MAPE (средняя абсолютная ошибка в процентах) [34]. Этот коэффициент легко интерпретируется и часто используется для сравнения моделей (см. рис.3).

$$\mathrm{MAPE} = \frac{100\%}{n} \sum_{t=1}^{n} \left| \frac{A_t - F_t}{A_t} \right|$$

Рисунок 3. Формула подсчёта MAPE

Таблица 8 — Средняя абсолютная ошибка в процентах для моделей нападающих

Алгоритм/ Параметр	Линейная регрессия	Случайный лес	XGBoost	Регрессия Lasso	Ридж-регрессия
За сезон	0.43128452	0.43101564	0.42478811	0.41351516	0.49690804
По времени на площадке	0.27168860	0.27341709	0.27544888	0.26920587	0.27161855
Без игр плей-офф	0.30768135	0.31707778	0.30556947	0.30856364	0.30743493

Таблица 9 — Средняя абсолютная ошибка в процентах для моделей защитников

Алгоритм/ Параметр	Линейная регрессия	Случайный лес	XGBoost	Регрессия Lasso	Ридж-регрессия
За сезон	0.45104690	0.46619587	0.47591978	0.46723438	0.44972414
По времени на площадке	0.36275719	0.35827988	0.39512220	0.35803927	0.36216254
Без игр плей-офф	0.31173705	0.26844788	0.26821248	0.26116363	0.31264784

Из результатов таблиц 8 и 9 можно сделать вывод, что наилучшим способом является регрессия LASSO, причём для защитников лучше использовать датасет без игр плей-офф. Это объясняется тем, что в плей-офф тактики команд сильно изменяются и становятся более оборонительными, защитники реже рискуют и идут в нападение, это влияет на оценку защитников. Для нападающих наилучшим образом обучается модель на датасете по времени, проведённому на льду. Из-за того, что игроки проводят разное время на льду, не всегда объективно сравнивать

игроков по матчам. Поэтому мы смотрим параметры за определённое время. Клуб может узнать, сколько в среднем проводит игрок времени на льду в текущей команде, и оценить необходимость его покупки.

Для создания программы, предоставляющей пользовательский интерфейсЮ были выбраны современные популярные технологии: Java, Spring, Hibernate, PostgreSQL.

Для рекомендации игроков пользователь должен выбрать команду и амплуа игрока. Поэтому было реализовано веб-приложение, в котором пользователю на выбор предоставляются команды и позиции игроков. После выбора игроков система предоставляет список рекомендованных игроков. Также для окончательного выбора игроков есть ссылки на статистику игроков на сайте КХЛ.

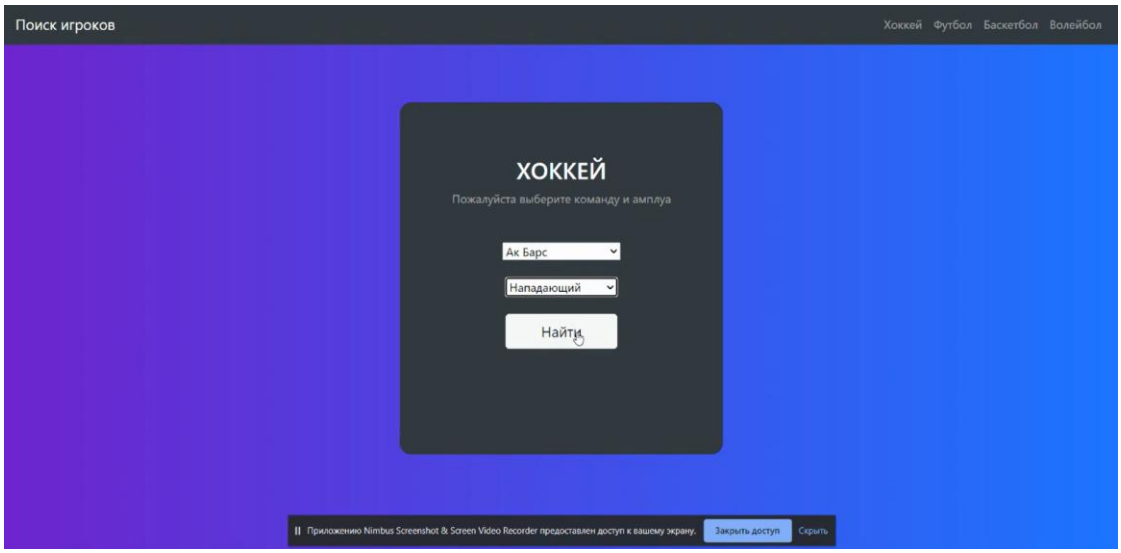


Рисунок 4. Выбор параметров для рекомендательной системы

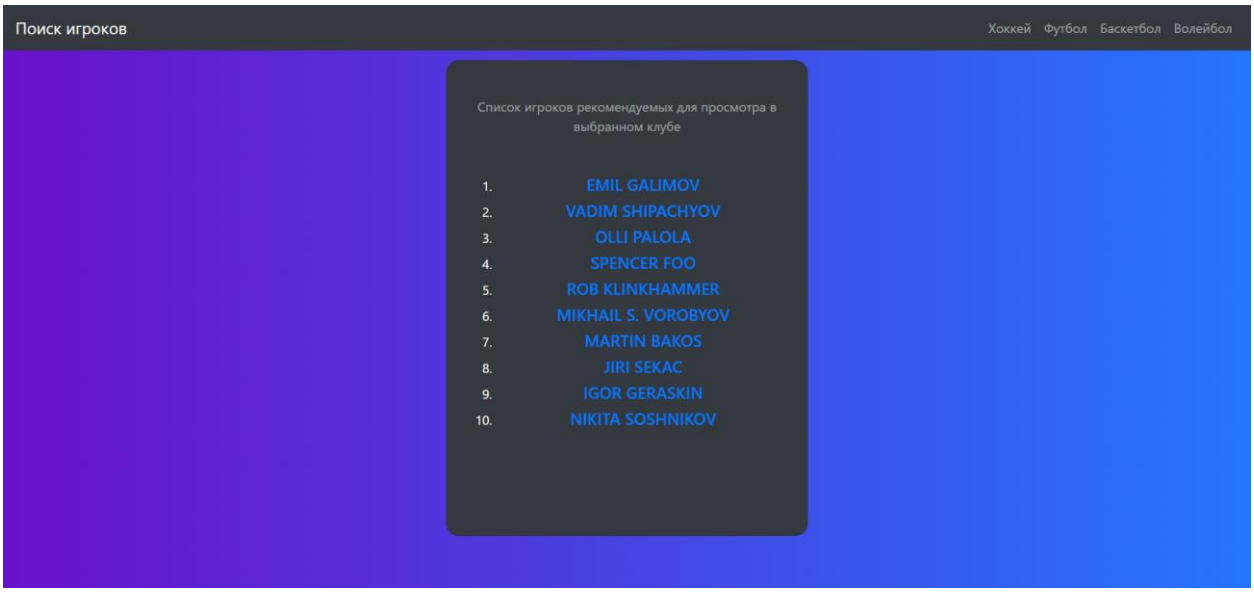


Рисунок 5. Рекомендация игроков нападения для хоккейного клуба «Ак Барс»

Для примера рассмотрим рекомендацию нападающих для казанского хоккейного клуба «Ак Барс».

Среди предложенных игроков есть Иржи Секач и Роб Клинкхаммер. Эти игроки уже играли за выбранный клуб и занимали ведущие роли в нём. Более того, в сезоне 2017/2018 оба игрока выступали за казанский клуб и выиграли вместе с ним кубок Гагарина. Можно предположить, что система выбрала игроков, которые достаточно хорошо вписывались в систему клуба и приносили большую пользу в игре команды.

Также среди предложенных игроков есть Вадим Шипачёв. В сезоне 2021/2022 он был признан самым ценным игроком лиги. В начале мая 2022 года стало известно, что игрок переходит в «Ак Барс». Значит, рекомендации системы совпали с необходимыми параметрами нападающего для казанской команды и ожидаемыми результатами от Вадима Шипачёва.

Среди остальных игроков есть как опытные игроки (как Никита Сошников), так и молодые и перспективные, например, Игорь Гераськин и Михаил Воробьёв.

Также для тестирования были просмотрены предложения рекомендательной системы за прошлый сезон. В списке предложенных был Дмитрий Кагарлицкий, который в этом сезоне стал одним из наиболее удачных приобретений ка-

занского клуба. Можно отметить, что в списке не было, например, Даниила Тарасова, который также стал игроком «Ак Барса» в этом сезоне, но не оправдал ожиданий казанской команды.

ЗАКЛЮЧЕНИЕ

Изучено влияние различных доступных статистических параметров игроков на результаты команды, собран датасет и обучены модели машинного обучения, а также созданы алгоритм рекомендации и пользовательский интерфейс.

Результатом работы является рекомендательная система на основе игроков КХЛ, полученная с применением технологий машинного обучения. Системой могут пользоваться спортивные клубы, агенты, тренеры для поиска игроков, а также болельщики для изучения возможностей трансферов клубов.

В будущем планируется использовать систему для других видов спорта и расширить источники данных для обучения системы.

СПИСОК ЛИТЕРАТУРЫ

1. *Гореликов В.А.* Трансфер игрока в профессиональном спорте как маркетинговый продукт спортивного клуба // Наука и спорт: современные тенденции. 2021. Т. 9, № 3. С. 115–124. https://doi.org/10.36028/2308-8826-2021-9-3-115-124

2. Зачем переходили? Летние трансферы КХЛ, которые пока не принесли никакой пользы. URL: https://www.championat.com/hockey/article-4497349-igroki-neudachno-vystupayuschie-za-novye-kluby-v-khl-lindholm-tarasov-bartoshak-bryukvin-shalunov-dansk-rejdeborn.html

3. Уровень зрелости российского спорттеха. URL: https://rb.ru/analytics/russia-sporttech/

4. Топ-10 самых популярных видов спорта в мире в 2021 году. URL: https://themoney.co/ru/top-10-des-sports-les-plus-populaires-au-monde-en-2021/

5. Россияне назвали самые популярные виды спорта. URL: https://www.magram.ru/news/rossiyane_nazvali_samye_populyarnye_vidy_sporta.html

6. Официальный сайт InStat. URL: https://instatsport.com/

7. Официальный сайт IceBerg. URL: https://www.icebergsports.com/

8. Официальный сайт DataVolley4. URL: https://www.dataproject.com/Products/GLOBAL/en/Volleyball/DataVolley4

9. Хоккейная статистика. URL: https://www.eliteprospects.com/

10. Футбольная статистика. URL: https://ru.whoscored.com/Statistics

11. Статистика на баскетбол. Где искать?
URL: https://bukmekerov.net/stati/cool/statistika-na-basketbol-gde-iskat/

12. Волейбольная статистика. URL: http://www.volleyservice.ru/

13. Международный совет футбольных ассоциаций (IFAB).
URL: https://www.theifab.com/

14. Правила волейбола.
URL: https://www.sportzone.ru/sport/rules.html?sport=volleyball

15. Официальные правила баскетбола 2020. URL:
https://russiabasket.ru/Files/Documents/%d0%9e%d1%84%d0%b8%d1%86%d0%b8%d0%b0%d0%bb%d1%8c%d0%bd%d1%8b%d0%b5%20%d0%9f%d1%80%d0%b0%d0%b2%d0%b8%d0%bb%d0%b0%20%d0%91%d0%b0%d1%81%d0%ba%d0%b5%d1%82%d0%b1%d0%be%d0%bb%d0%b0%202020%20v.1.1.pdf

16. Официальные правила хоккея 2021–2022.
URL: https://fhr.ru/upload/iblock/0a2/Ofitsialnaya-kniga-Pravil-IIKHF.pdf

17. Самые быстрые хоккеисты.
URL: https://www.sport-express.ru/hockey/nhl/reviews/bure-ili-makdevid-samye-bystrye-hokkeisty-v-istorii-nhl-1344222/

18. Позиции игроков в хоккее и их функции.
URL: https://dcntrsport.com/poziczii-igrokov-v-hokkee-i-ih-funkczii/

19. Официальный сайт Континентальной хоккейной лиги (КХЛ).
URL: https://www.khl.ru/

20. Официальный сайт Национальной хоккейной лиги.
URL: https://www.nhl.com/

21. Резервные вратари по-разному убивают время.
URL: https://www.nhl.com/ru/news/rezervnye-vratari-po-raznomu-ubivaiut-vremia/c-874329

22. Хоккеисты на просмотровых контрактах в КХЛ. URL: https://www.championat.com/hockey/article-3808815-hokkeisty-na-prosmotrovyh-kontraktah-v-khl-v-sezone-201920.html

23. Официальный сайт языка программирования Python.

URL: https://www.python.org/

24. Документация библиотеки requests.

URL: https://requests.readthedocs.io/en/latest/

25. Документация библиотеки Beautiful Soup.

URL: https://www.crummy.com/software/BeautifulSoup/bs4/doc/

26. ХК «Динамо» и ХК МВД прекратили свое существование.

URL: https://sportrbc.ru/news/5755069a9a79474acad120d9

27. Самая безумная бойня в истории российского хоккея.

URL: https://www.sport-express.ru/hockey/khl/reviews/10-let-massovym-drakam-v-matche-vityaz-avangard-1629757/

28. Официальный сайт библиотеки scikit-learn.

URL: https://scikit-learn.org/stable/

29. Линейная регрессия. URL: https://neurohive.io/ru/osnovy-data-science/linejnaja-regressija/

30. Случайный лес.

URL: https://dya-konov.org/2016/11/14/%D1%81%D0%BB%D1%83%D1%87%D0%B0%D0%B9%D0%BD%D1%8B%D0%B9-%D0%BB%D0%B5%D1%81-random-forest/

31. XGBoost. URL: https://neerc.ifmo.ru/wiki/index.php?title=XGBoost

32. Регрессия Lasso.

URL: https://ranalytics.github.io/data-mining/042-Regularization.html

33. Ридж-регрессия.

URL: https://www.mygreatlearning.com/blog/what-is-ridge-regression/

34. MAPE.

URL: https://www.statisticshowto.com/mean-absolute-percentage-error-mape/

SOFTWARE FRAMEWORK FOR IMPLEMENTING USER INTERFACE INTERACTION IN IOS APPLICATIONS BASED ON OCULOGRAPHY

R. R. Shigapov[1][0000-0002-3163-0959], A. A. Ferenetz[2][0000-0002-7859-9901]

Institute of Information Technology and Intelligent Systems, Kazan Federal

University, 35 Kremlevskaya str., Kazan, 420008

[1] shigapov.rinat.2000@gmail.com, [2] ist.kazan@gmail.com

Abstract

This article describes the development of a recommender system for selecting players based on machine learning. The system introduced the example of hockey with the possibility of expanding its use in various team sports. For each sport different roles and characteristics of the players were considered. The article analyzes information about hockey, football, basketball and volleyball. The characteristics of the players are structured and divided into general groups. For each parameter coefficients are displayed that show the impact on the result of the match. Various machine learning algorithms were used to build the model. The web interface of the application has been created.

Keywords*: sports, hockey, selection of players, recommender system, machine learning*

REFERENCES

1. *Gorelikov V.A.* Transfer of a player in professional sports as a marketing product of a sports club // Science and sport: current trends. 2021. V. 9, No. 3. S. 115–124. https://doi.org/10.36028/2308-8826-2021-9-3-115-124
2. Why did you switch? KHL summer transfers that have not brought any benefit yet. URL: https://www.championat.com/hockey/article-4497349-igroki-neudachno-vystupayuschie-za-novye-kluby-v-khl-lindholm-tarasov-bartoshak-bryukvin-shalunov-dansk-rejdeborn.html
3. The level of maturity of Russian sports technology.
URL: https://rb.ru/analytics/russia-sporttech/
4. Top 10 most popular sports in the world in 2021.
URL: https://themoney.co/ru/top-10-des-sports-les-plus-populaires-au-monde-en-2021/
5. Russians named the most popular sports.
URL: https://www.magram.ru/news/rossiyane_nazvali_samye_pop-ulyarnye_vidy_sporta.html
6. InStat official website. URL: https://instatsport.com/

7. IceBerg official website. URL: https://www.icebergsports.com/
8. DataVolley4 official website.
URL: https://www.dataproject.com/Products/GLOBAL/en/Volleyball/DataVolley4
9. Hockey statistics. URL: https://www.eliteprospects.com/
10. Football stats. URL: https://en.whoscored.com/Statistics
11. Basketball statistics. Where to look?
URL: https://bukmekerov.net/stati/cool/statistika-na-basketbol-gde-iskat/
12. Volleyball statistics. URL: http://www.volleyservice.ru/
13. International Football Association Board (IFAB).
URL: https://www.theifab.com/
14. Volleyball Rules.
URL: https://www.sportzone.ru/sport/rules.html?sport=volleyball
15. Official Basketball Rules 2020. URL: https://russiabasket.ru/Files/Documents/%d0%9e%d1%84%d0%b8%d1%86%d0%b8%d0%b0%d0%bb%d1%8c%d0%bd%d1%8b%d0%b5%20%d0%9f%d1%80%d0%b0%d0%b2%d0%b8%d0%bb%d0%b0%20%d0%91%d0%b0%d1%81%d0%ba%d0%b5%d1%82%d0%b1%d0%be%d0%bb%d0%b0%202020%20v.1.1.pdf
16. Official Hockey Rules 2021–2022.
URL: https://fhr.ru/upload/iblock/0a2/Ofitsialnaya-kniga-Pravil-IIKHF.pdf
17. The fastest hockey players. URL: https://www.sport-express.ru/hockey/nhl/reviews/bure-ili-makdevid-samye-bystrye-hokkeisty-v-istorii-nhl-1344222/
18. Positions of players in hockey and their functions.
URL: https://dcntrsport.com/poziczii-igrokov-v-hokkee-i-ih-funkczii/
19. Official website of the Kontinental Hockey League (KHL).
URL: https://www.khl.ru/
20. Official website of the National Hockey League. URL: https://www.nhl.com/
21. Reserve goalkeepers kill time in different ways.
URL: https://www.nhl.com/ru/news/rezervnye-vratari-po-raznomu-ubivaiut-vremia/c-874329
22. Hockey players on viewing contracts in the KHL.
URL: https://www.championat.com/hockey/article-3808815-hokkeisty-na-prosmotrovyh-kontraktah-v-khl-v-sezone-201920.html

23. The official website of the Python programming language.
URL: https://www.python.org/
24. Requests. URL library documentation: https://requests.readthedocs.io/en/latest/
25. Beautiful Soup Library Documentation.
URL: https://www.crummy.com/software/BeautifulSoup/bs4/doc/
26. HC Dynamo and HC MVD ceased to exist.
URL: https://sportrbc.ru/news/5755069a9a79474acad120d9
27. The craziest massacre in the history of Russian hockey
28. Official website of the scikit-learn library. URL: https://scikit-learn.org/stable/
29. Linear Regression.
URL: https://neurohive.io/en/osnovy-data-science/linejnaja-regressija/
30. Random Forest.
URL: https://dya-konov.org/2016/11/14/%D1%81%D0%BB%D1%83%D1%87%D0%B0%D0%B9%D0%BD%D1%8B %D0%B9-%D0%BB%D0%B5%D1%81-random-forest/
31. XGBoost. URL: https://neerc.ifmo.ru/wiki/index.php?title=XGBoost
32. Regression Lasso.
URL: https://ranalytics.github.io/data-mining/042-Regularization.html
33. Ridge Regression.
URL: https://www.mygreatlearning.com/blog/what-is-ridge-regression/
34. MAPE.
URL: https://www.statisticshowto.com/mean-absolute-percentage-error-map/

СВЕДЕНИЯ ОБ АВТОРАХ

ШИГАПОВ Ринат Рустемович – выпускник бакалавриата Института информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета, г. Казань.

Rinat SHIGAPOV – graduate student of the Institute of Information Technologies and Intelligent Systems, Kazan (Volga region) Federal University, Kazan.

e-mail: shigapov.rinat.2000@gmail.com

ORCID: 0000-0002-3163-0959

ФЕРЕНЕЦ Александр Андреевич – старший преподаватель кафедры программной инженерии Института информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета, г. Казань.

Alexander FERENETS – senior lecturer of Software Engineering of Institute of Information Technologies and Intelligent Systems KFU

e-mail: ist.kazan@gmail.com

ORCID: 0000-0002-7859-9901

Материал поступил в редакцию 20 августа 2022 года