РЕДАКТИРОВАНИЕ ИЗОБРАЖЕНИЙ НА МОБИЛЬНОЙ ПЛАТФОРМЕ IOS

Харитоненко Д.А.

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Савёнок В.А. – ст. преподаватель

В работе рассмотрены возможности мобильной платформы iOS для редактирования изображений, а также методы интеграции и оптимизации производительности. Проанализированы инструменты Core Image и Metal, использование 3D LUT. Описаны подходы к улучшению пользовательского опыта приложения фоторедактора, включая удобное взаимодействие с интерфейсом, а также архитектурные принципы организации кода для разделения логики и упрощения поддержки приложения.

Редактирование изображений на компьютерах началось с появлением программы Photoshop [1]. Со временем, с развитием мобильных устройств и сенсорных экранов, а также улучшением качества мобильных камер, редакторы изображений стали доступны и на смартфонах. Сегодня такие приложения являются незаменимыми для активных пользователей соцсетей и любителей фотографии, предлагая широкий набор инструментов и удобный пользовательский интерфейс, понятный даже непрофессионалам.

Мобильная платформа iOS включает четыре слоя: Cocoa Touch, Media, Core Services и Core OS. Слой Media содержит фреймворки, предназначенные для работы с изображениями, видео и текстом, в том числе Core Image, который предоставляет инструменты для обработки.

Для обсуждения Core Image необходимо сначала ввести понятие фильтра. Фильтр – это объект, который имеет определённое количество входных и выходных параметров и выполняет преобразование данных.

Core Image – это технология обработки изображений, обеспечивающая высокую точность, работу вблизи реального времени. Она упрощает применение фильтров, скрывая низкоуровневые операции с пикселями, что позволяет разработчикам легко добавлять эффекты без сложного кода.

Фильтры применяются к изображению без изменения исходных данных, что позволяет редактировать изображение недеструктивно. Пользователь может настраивать параметры фильтров, добиваясь нужного эффекта [2].

Core Image предоставляет обширный набор встроенных фильтров для обработки изображений, охватывающих различные категории эффектов. К примеру, фильтры размытия, такие как CIBoxBlur, создают эффекты размытия изображения. Фильтры коррекции цвета позволяют изменять параметры цвета изображения, такие как экспозиция, гамма-коррекция и сепия. Фильтры композиции, такие как CIAdditionCompositing, позволяют комбинировать два изображения с различными режимами наложения. Фильтры искажения создают эффекты искажения, такие как закручивание или воронка [3].

Помимо использования встроенных фильтров, разработчики могут создавать собственные с помощью CIKernel – объекта в Core Image, который содержит программный код для обработки изображения. Он позволяет реализовывать кастомные фильтры для более сложных операций, не поддерживаемых стандартными фильтрами Core Image. Для создания таких фильтров используется Metal Shading Language (MSL), который предоставляет возможность писать шейдеры для работы с изображениями, обеспечивая высокую производительность через вычисления на графическом процессоре.

Core Image поддерживает наложение нескольких фильтров одновременно. Вместо последовательного применения каждого фильтра создаётся оптимизированный конвейер инструкций, позволяющий выполнить все операции за один цикл обработки пикселей. Это ускоряет вычисления, минимизируя задержки. Обработку выполняет либо CPU, либо GPU, в зависимости от того, какое устройство обеспечит лучшую производительность, используя компиляцию «на лету» (JIT). [4]

Для данного вида программного обеспечения важен быстрый отклик. Например, при изменении интенсивности параметра на слайдере изменения должны мгновенно отображаться на изображении. Это требует высокой производительности обработки и отображения изображений. Одним из решений для реализации такого отклика является использование представления MTKView. В отличие от UIImage, который в основном используется для отображения статичных изображений и работает на CPU, использование Metal с MTKView позволяет задействовать графический процессор для рендеринга и обработки изображений в реальном времени. Данное представление упрощает интеграцию высокопроизводительных графических операций в приложения. MTKView автоматически управляет слоем CAMetalLayer, который отвечает за отображение рендеренного содержимого на экране. Для работы MTKView требуется объект MTLDevice, представляющий собой интерфейс к GPU. Этот объект используется для создания и управления другими объектами Metal, такими как очереди команд и буферы команд.

Взаимодействие с GPU в Metal осуществляется через отправку команд, которые организуются в буферы команд (MTLCommandBuffer). Для управления этими буферами используется очередь команд (MTLCommandQueue). Разработчик создаёт буфер команд, записывает в него необходимые команды с помощью кодировщиков команд (MTLCommandEncoder), а затем отправляет буфер на выполнение GPU. Такая организация позволяет эффективно управлять процессом рендеринга и выполнять сложные графические операции с высокой производительностью [1].

Фильтры в контексте пользовательского приложения – это предустановленные настройки цветокоррекции, позволяющие мгновенно изменить тон и стиль фотографии. Они представляют собой комбинацию цветовых изменений, таких как баланс белого, насыщенность, контраст и оттенки теней и светов. Один из популярных способов реализации таких фильтров – использование 3D LUT (Look-Up Table). LUT – это таблица соответствия цветов, используемая для преобразования исходных цветовых значений в новые, согласно заданной схеме. В случае 3D LUT такая таблица представляется в виде цветового куба, где каждому входному цвету соответствует откорректированный выходной цвет. В Core Image фильтр CIColorCube позволяет накладывать 3D LUT на изображения, трансформируя их цветовую гамму в соответствии с таблицей соответствия.

В фоторедакторах список фильтров обычно представлен в виде миниатюр с предварительно наложенными эффектами. Для оптимизации обработки сначала уменьшают изображение, а затем применяют фильтры, так как миниатюры имеют небольшой размер, и обработка полного изображения была бы избыточной.

Важную роль в мобильных приложениях играет удобство интерфейса и пользовательский опыт. Инструменты следует грамотно сгруппировать, чтобы упростить навигацию. Например:

– обрезка, кадрирование, поворот;

– наложение фильтров и эффектов;

– настройка цветовых и световых параметров.

При использовании каждого инструмента желательно переходить на отдельный экран, чтобы сосредоточиться на редактировании. Для регулировки параметров можно использовать различные варианты слайдеров или управление жестами. [5]

Пресеты – это заранее настроенные параметры редактирования, которые можно мгновенно применять к изображениям. Они могут включать в себя фильтры, изменения цвета, освещения, контраста, и другие коррекции. Данный функционал делает редактирование в едином стиле удобнее.

В архитектуре редактора изображений важно разделять логику пользовательского интерфейса и бизнес-логику. Для этого стоит использовать ViewModel, которая отвечает за управление состоянием редактирования, а ViewController (View) – только за отображение данных. Эффективным решением будет внедрение объекта без состояния EditingStack, который управляет состоянием редактирования изображения, позволяя применять различные изменения. Этот объект загружает изображение, сохраняет его состояние и поддерживает историю изменений. Важной частью класса является его способность работать с многозадачностью и потоками с использованием разных очередей, что позволяет эффективно обрабатывать изображение. В EditingStack используется архитектура с хранением состояния, где каждое изменение фиксируется в истории редактирования. ViewModel должна управлять EditingStack, обновляя превью изображения при каждом изменении. ViewController, при этом, должен подписываться на изменения в ViewModel, оставаясь максимально лёгким и не содержащим состояние. Такой подход не только упрощает реализацию отмены и повтора изменений, но и делает код более чистым, модульным и удобным для расширения.

Список использованных источников:

1. Adobe Blog – URL: https://blog.adobe.com/en/publish/2025/02/19/celebrating-35-years-of-creativity-community-innovation-with-adobe-photoshop (дата обращения: 25.03.2025).

2. Apple Developer Documentation – URL: https://developer.apple.com (дата обращения: 25.03.2025).

3. Objc.io. An Introduction to Core Image – URL: https://www.objc.io/issues/21-camera-and-photos/core-image-intro (дата обращения: 25.03.2025).

4. Kodeco. Core Image Tutorial for iOS – URL: https://www.kodeco.com/25658084-core-image-tutorial-for-ios-custom-filters (дата обращения: 25.03.2025).

5. Resolve Cafe. Cube LUT Format Specification – URL: https://resolve.cafe/developers/luts (дата обращения: 25.03.2025)

6. User Interface Design of Mobile Photo Editors / Irma Rochmawati // Proceedings of the International Conference on Business, Economic, Social Science, and Humanities (ICOBEST 2018). – 2018. – P. 79.