УДК 004.4, 004.9

ПРОГРАММНОЕ СРЕДСТВО ОПТИМИЗАЦИИ ПРОЦЕССОВ ВИДЕОПРОИЗВОДСТВА

Р. Ф. Давлетшин[1] [0009-0009-5326-3609], И. С. Шахова[2] [0000-0003-1591-5767]

[1, 2]*Институт информационных технологий и интеллектуальных систем, Казанский (Приволжский) федеральный университет, г. Казань*
[1]rustemd02@mail.ru, [2]is@it.kfu.ru

Аннотация

Предложены программные механизмы, направленные на оптимизацию процессов видеопроизводства для авторов художественных видеоматериалов – материалов, предполагающих предварительную постановочную работу. Разработан механизм создания анимированных трехмерных планов съемки (раскадровок) с использованием дополненной реальности для позиционирования и анимации перемещения актеров. С целью преодоления ограничений операционной системы iOS, связанных с доступом к сенсорам, разработан механизм раздельного захвата аудио- и видеопотоков с датчиков устройства для проведения записи, а также их последующей синхронизации по временным меткам для сохранения в память устройства. Отслеживание соблюдения правил композиционного построения и анализ качества изображения на предмет расфокусировки камеры реализованы с использованием технологий компьютерного зрения. Также представлены механизмы работы со сценарием, включающие алгоритмы обработки текста для вывода на экран в виде субтитров, а также распознавания речи актеров и сравнения её с текстом сценария.

Ключевые слова*: видеопроизводство, мобильное кино, дополненная реальность, раскадровка, видеозапись, автоматизация, программное решение.*

ВВЕДЕНИЕ

В последние годы отмечается тенденция к увеличению количества видеоматериалов (в том числе художественных произведений, предполагающих постановочную работу), созданных с использованием мобильных устройств [1, 2]. Данная тенденция частично обоснована регулярным совершенствованием аппаратной составляющей камер смартфонов [3], что приводит к улучшению качества изображения и его приближению к уровню изображения со специализированных камер [4], благодаря чему снижается порог входа в индустрию художественной съемки с точки зрения необходимого бюджета на кинооборудование [5].

Зачастую в процессе записи художественных видеоматериалов на любительском уровне вследствие недостатка бюджета режиссёр вынужден совмещать роли оператора, художника-постановщика и ассистента режиссёра по работе со сценарием, что может вызвать затруднения во время съёмки [6]. Так, режиссёру необходимо следить, чтобы актёры в своих действиях придерживались плана сцены; их реплики совпадали с прописанными фразами в сценарии (пренебрежение этим может вызвать логические проблемы с тавтологией [7], а заметно это станет лишь на этапе монтажа); а также следить за своими действиями как оператора: не допускать попадания в кадр элементов съёмочной площадки, соблюдать общие правила композиции кадра, и чтобы люди в кадре находились в фокусе [8].

Из вышесказанного следует, что ограниченный бюджет и невозможность расширения команды, ответственной за съёмку, не только усложняют процесс производства картины, но и негативно сказываются на итоговом результате. Решением обозначенной проблемы может стать программное средство для мобильных устройств, оптимизирующее указанные процессы. Под оптимизацией в данной работе понимается дополнение процессов отдельными сопроводительными программными механизмами, позволяющими частично автоматизировать отслеживание обозначенных ранее несоответствий плану сцены, реплик, требованиям к построению кадра.

Для анализа разработанности выявленной проблемы были рассмотрены программные решения для мобильных устройств, которые могут быть приме-

нимы на съёмочной площадке. В качестве критериев, которым должно удовлетворять решение проблемы, были сформулированы: возможность создания раскадровки и сохранения двух или более раскадровок, поддержка видеосъёмки, настройка параметров съёмки, механизмы расшифровки речи, поддержка русского языка.

Одним из программных решений, которое может быть потенциально применимо для использования в рассматриваемой предметной области, является LightSpace [9] – инструмент для отрисовки произвольных форм и линий в среде дополненной реальности, который может использоваться для зарисовки общего плана съёмки сцены. Однако в продукте отсутствует возможность анимации раскадровки, а также, ввиду особенностей продукта, нарисованные элементы будут отображаться и на итоговом видеофайле, что ограничивает использование решения в качестве инструмента для художественного кинопроизводства. StoryBoard Animator [10] предполагает создание раскадровок в виде набора зарисовок на белом холсте с возможностью их анимации. К недостаткам этого решения можно отнести отсутствие возможности видеосъёмки, а также тот факт, что созданные в приложении раскадровки ограничены пространством виртуального холста, что может вызвать затруднения, если локация съёмок окажется меньшей, чем предполагалось на этапе их планирования.

Для отслеживания несоответствия реплик актеров тексту сценария может быть использован инструмент SUFLER.PRO [11] – приложение-суфлер с возможностью видеозаписи. В сфере художественного кинопроизводства этот программный инструмент может использоваться оператором в качестве подсказки со сценарием текущей сцены, что позволит человеку за камерой визуально сравнивать фразы, прописанные в тексте, с репликами, которые в действительности произносят актёры. Однако в этом продукте отсутствует возможность настройки таких параметров съёмки, как количество кадров в секунду, уровень баланса белого и др. (за исключением настройки разрешения записи), необходимость контроля которых в художественном кинопроизводстве отмечена в научных изданиях [12].

Аналогичная задача по выявлению несоответствий текста сценария и текста, фактически озвученного актерами, может быть решена с использованием инструмента транскрибирования речи iOS Live Captions [13]. Он может быть применен в

процессе съёмки видео для проведения визуальной оценки наличия или отсутствия в речи актёров отклонений от изначального сценария. Недостатком этого решения является отсутствие поддержки русского языка.

Нашей целью была разработка программного средства, обладающего функциональными возможностями создания и анимации виртуальных раскадровок, осуществления видеозаписи без отображения элементов раскадровки в итоговом файле, анализа картинки на предмет нарушения композиции и расфокусировки, а также возможностью расшифровки произнесенной речи и сравнения ее со сценарием.

АРХИТЕКТУРА ПРОГРАММНОГО РЕШЕНИЯ

В связи с комплексной структурой разрабатываемого программного решения, в качестве архитектурного шаблона для программного продукта было принято решение использовать VIPER.

Архитектура программного решения, а также взаимодействие модулей приложения с прочими элементами программного кода представлены на рисунке 1.

Архитектура программного решения делится на три основные составляющие: модули программных частей решения, сервисы взаимодействия с компонентами устройства и операционной системы, а также набор сущностей.

Набор модулей включает следующие компоненты: главное меню, основной экран видеозаписи и создания раскадровки, а также экран редактирования сценария. Взаимодействие данных модулей с датчиками устройства и элементами базы данных производится при помощи соответствующих сервисов: работы с компонентами видеосъемки, транскрибирования устной речи, работы с базой данных и механизмами персистентности. При этом для согласования единиц различных структур данных сервис видеозаписи производит их конвертацию при помощи соответствующего компонента программного решения.

Набор сущностей состоит из сущности данных об актере в виртуальной раскадровке, метаданных непосредственно самой раскадровки, настроек видеоза-

писи, а также вспомогательных сущностей перечисления *enum,* содержащих данные о частоте кадров и скорости перемещения трехмерных моделей по раскадровке.

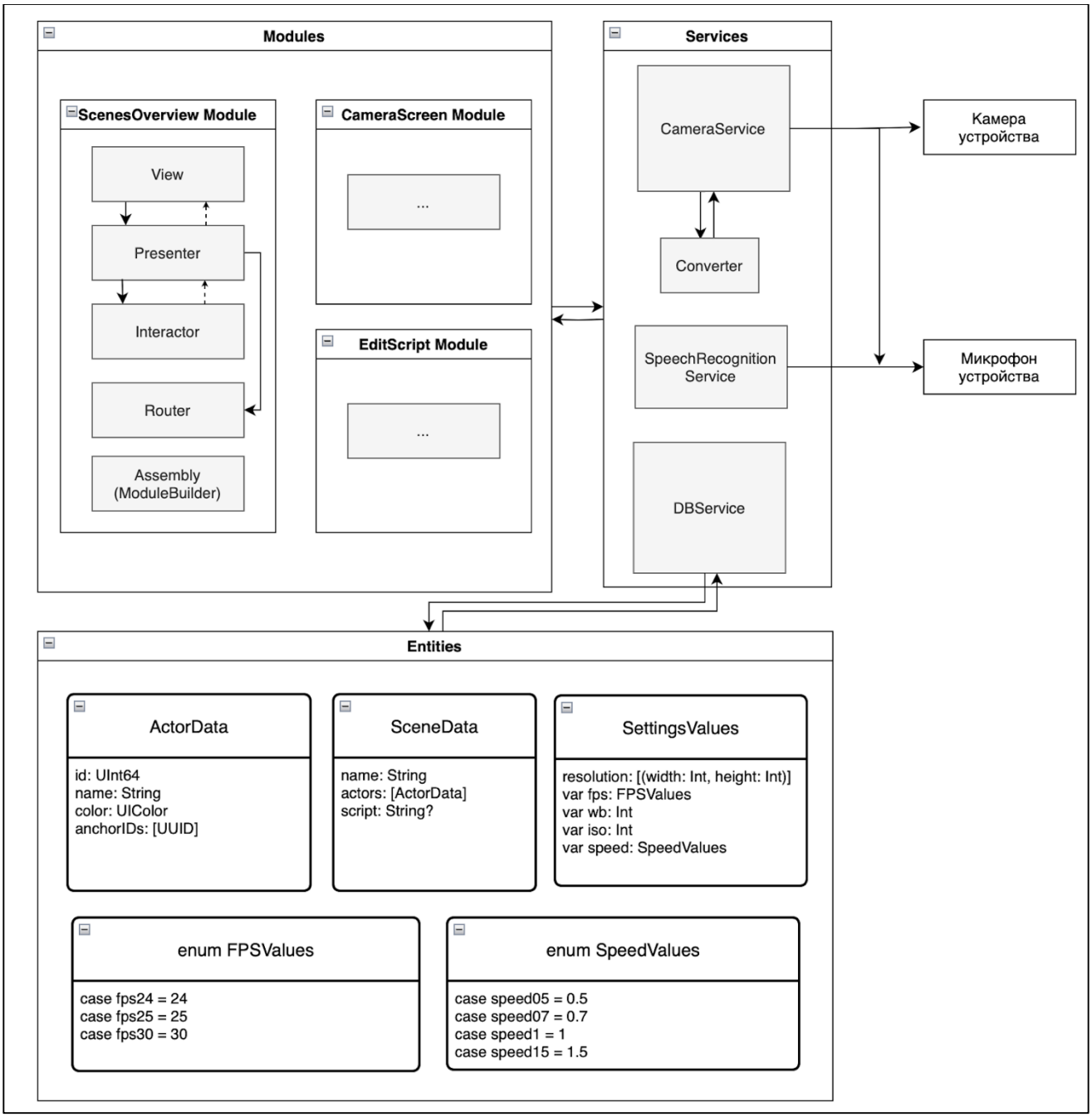

Рис. 1. Упрощенная схема архитектуры программного решения

ПРОГРАММНЫЕ МЕХАНИЗМЫ НА ОСНОВЕ ТЕХНОЛОГИИ ДОПОЛНЕННОЙ РЕАЛЬНОСТИ

При выборе технологии для работы с дополненной реальностью были сформулированы следующие ограничения:

- целевые устройства – устройства, под управлением операционной системы iOS;
- использование программных библиотек и технологий, нативных для экосистемы Apple.

Было выбрано несколько фреймворков, удовлетворяющих поставленным ограничениям: SceneKit [14], ARKit [15], RealityKit [16]. Был проведен анализ доступных в экосистеме Apple фреймворков дополненной реальности, по результатам которого было принято решение использовать фреймворк ARKit в связке с RealityKit, поскольку ARKit уже включает многие необходимые инструменты, такие как, например, автоматическая калибровка по различным поверхностям. Однако он не включает возможность отрисовки трёхмерной графики, поэтому для рендеринга графики была использована библиотека RealityKit.

К основным компонентам используемых фреймворков ARKit и RealityKit относятся:

- ***ARView*** – компонент, который используется для визуального представления среды дополненной реальности на экране устройства;
- ***Scene*** (сцена) – контейнер, содержащий все элементы трехмерного окружения, включая 3D-модели;
- ***ARAnchor*** – якорная точка, которая соотносится фреймворком ARKit с определенной координатой в пространстве и к которой впоследствии прикрепляются 3D-модели;
- ***ModelEntity*** – виртуальный объект, представляющий собой трехмерную модель, которая размещается на сцене и прикреплена к якорной точке.

При инициации пользователем создания на сцене нового трехмерного объекта необходимо вычислить координаты формата *simd_float4x4* для создания ARAnchor на соответствующих координатах и соотнесения модели с данным якорем. Вычисление координат осуществляется путем использования метода *raycast*

фреймворка RealityKit. Этот механизм направляет луч от центра экрана до ближайшей поверхности и, если поверхность найдена, возвращает результат, из которого можно получить координаты точки, на которую упал луч.

С целью визуального разделения трехмерных моделей, обозначающих актеров в виртуальной раскадровке, при создании объекта на сцене также создается и дочерний элемент, прикрепленный к данному объекту и представляющий собой однострочное текстовое представление, которое пользователь может редактировать (поле может использоваться, например, для имени актера). Так как по иерархии компонентов сцены ARKit модель имени актера является дочерним объектом модели актера, то их перемещения синхронизированы.

Путь перемещения трехмерных моделей актеров в виртуальной раскадровке задается с помощью меток путей, создаваемых пользователем. Метки путей представляют собой отдельные ModelEntity, прикрепленные к своим элементам ARAnchor. Уникальные идентификаторы меток путей модели сохраняются в виде массива в соответствующем поле, которым обладают объекты структуры данных, созданной для моделей актеров. При этом, поскольку запись происходит в порядке создания пользователем меток пути, данные метки автоматически выступают в качестве контрольных точек, задавая таким образом порядок перемещения модели актера.

Таким образом, диаграмму иерархии компонентов сцены ARView можно представить в виде, приведенном на рисунке 2.

Для наглядности пунктирной линией на рисунке представлена условная связь якорей меток пути и якорей соответствующего актера.

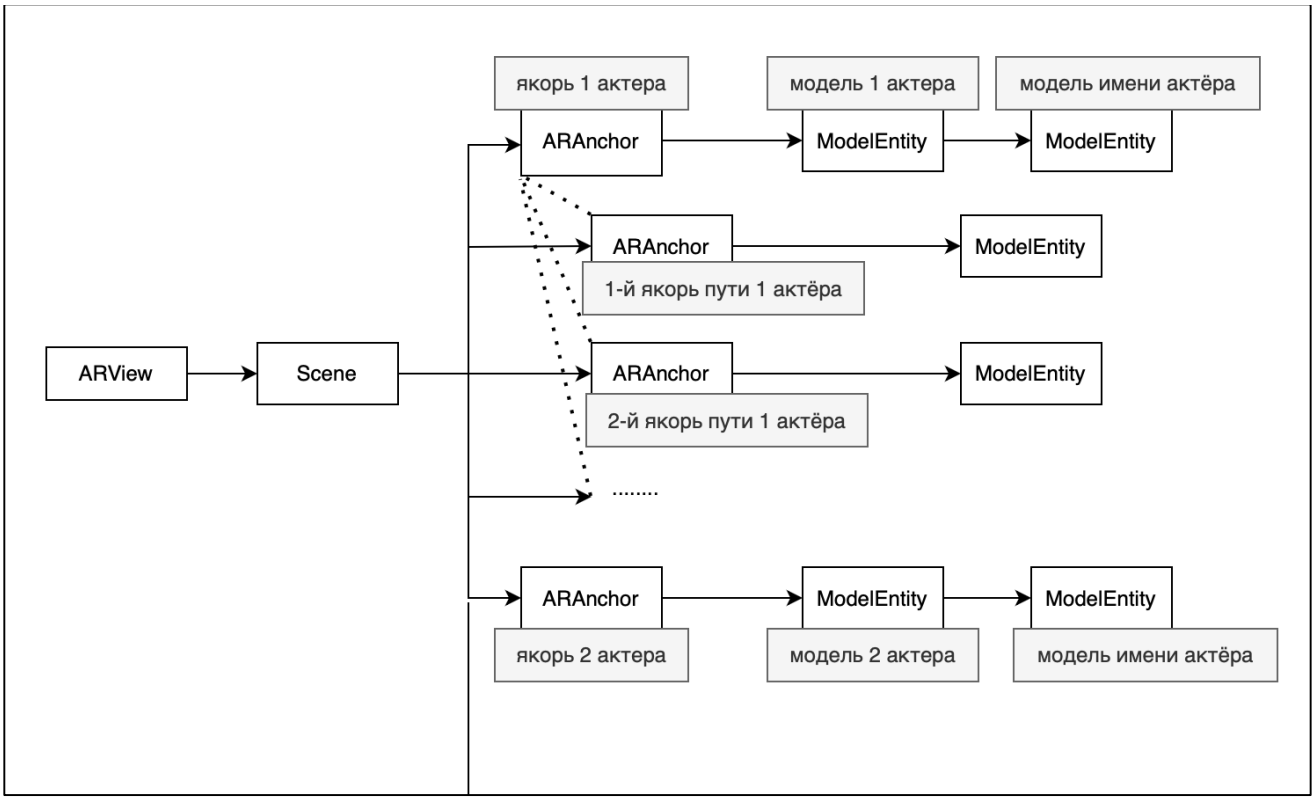

Рис. 2. Диаграмма иерархии компонентов ARView

Для перемещения трехмерных моделей по сцене в ARKit есть соответствующий метод *move()*, который принимает на вход координаты финальной точки перемещения, а также время, за которое анимация должна завершиться. При таком подходе время, за которое модель актера в раскадровке перемещается от точки А до точки Б, было бы идентичным вне зависимости от расстояния, что, очевидно, не соответствовало бы реальной скорости перемещения актеров на площадке. Поэтому был разработан алгоритм, позволяющий динамически вычислять время перемещения трехмерной модели по сцене, исходя из расстояния между контрольными точками и заданной скорости движения.

Для работы алгоритма необходимы координаты начальной точки перемещения, а также конечной (или промежуточной к финальной, в таком случае необходимо выполнить алгоритм n раз, где n – количество промежуточных контрольных точек). Расположение трехмерных моделей в виртуальной сцене ARView описано в виде матрицы размера 4 на 4 [17] (рис. 3).

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}$$

Рис. 3. Матрица координат трехмерных моделей в ARView

Каждый из элементов данной матрицы используется для представления соответствующей координаты объекта в трехмерном пространстве:

- *a11, a12, a13* – представляют вращение вокруг оси X;
- *a21, a22, a23* – представляют вращение вокруг оси Y;
- *a31, a32, a33* – представляют вращение вокруг оси Z;
- *a14, a24, a34* – представляют смещение объекта относительно точки начала координат по осям X, Y, Z соответственно;
- *a41, a42, a43, a44* – координаты проецирования на поверхность изображения.

Соответствующая матрица преобразований имеется у каждого ModelEntity, прикрепленного к ARAnchor. Можно заметить, что для целей перемещения моделей по сцене используются элементы *a14, a24, a34*, обозначающие смещение объекта.

После получения координат начальной и конечной точек перемещения вычисляется расстояние между соответствующими точками каждой из осей, для чего применяется алгоритм нахождения расстояния между двумя точками в n-мерном пространстве, также известного как евклидово расстояние [18]. Для вычисления времени перемещения трехмерной модели (в секундах) найденное расстояние делится на скорость, заданную пользователем. Полученное в результате значение передается в качестве входного параметра в упомянутый ранее метод *move()*.

СОХРАНЕНИЕ И ЗАГРУЗКА СЦЕНЫ ДОПОЛНЕННОЙ РЕАЛЬНОСТИ

Фреймворк ARKit предоставляет возможность получить текущую карту трехмерного окружения с помощью встроенного метода *currentWorldMap()*, который

на выходе дает объект класса ARWorldMap, содержащий текущее состояние окружения в сессии дополненной реальности на момент вызова функции [19]. Полученный объект класса ARWorldMap сохраняется в памяти устройства компонентом класса NSKeyedArchiver. Сохранение производится по уникальному идентификатору сцены в папку в файловой системе устройства, создаваемую программным решением. По идентификатору также происходят последующая загрузка, а также удаление данных. Для сохранения данных о трехмерных моделях используется протокол Codable.

Так как объект класса ARWorldMap хранит в себе идентификаторы якорей вместе с их координатами в пространстве, а сохраненные данные о трехмерных моделях также включают в себя поле идентификатора якоря, к которому была прикреплена данная модель в сохраненной сессии, то задача загрузки сцены и восстановления предыдущего состояния сессии сводится к сопоставлению информации о моделях по совпадающим идентификаторам, повторному созданию трехмерных моделей на сцене в соответствии с хронологическим порядком расстановки их пользователем, включая прикрепление дочерних элементов к моделям, таких как элемент имени актера и метки пути, а также установке цвета трехмерным объектам.

ЗАХВАТ ВИДЕОПОТОКА С КАМЕРЫ УСТРОЙСТВА

На этапе проработки концепции программного решения предполагалось, что процесс захвата видеопотока будет происходить параллельно процессам, реализующим взаимодействие с технологиями дополненной реальности, однако после проведения ряда испытаний было установлено, что процессы, запущенные на операционной системе iOS, обладают исключительным доступом к датчикам и сенсорам устройства, что исключает возможность одновременного их использования разными приложениями и процессами.

Концепция, реализованная в программном решении, заключается в следующем: фреймворк ARKit обладает механизмом обратного вызова *session(_:didUpdate:)*, который срабатывает при каждом обновлении кадра и предоставляет возможность делегировать обработку кадра стороннему программному коду, предоставляя при этом кадры напрямую с камеры устройства,

то есть до этапа наложения трехмерной графики. Было принято решение создать алгоритм, который после начала записи покадрово принимает на вход изображения с камеры устройства, записывает их в буфер, а по окончании записи производит склейку в единый видеофайл. При этом подразумевалось, что кадры, полученные с помощью метода *session(_:didUpdate:),* являются исключительно визуальными элементами и не содержат аудиоинформации, поэтому необходимо также одновременно производить запись аудио с помощью микрофона мобильного устройства. Это означает, что в дополнение к обозначенной выше задаче склейки отдельных кадров одним из шагов алгоритма перед выдачей готового видеофайла является объединение видеодорожки с записанным аудиопотоком. С целью исключения появления задержки между нажатием пользователем на кнопку начала записи и фактическим стартом захвата видео инициализация компонентов, необходимых для данного процесса, происходит сразу после запуска программного решения (рисунки 4 и 5).

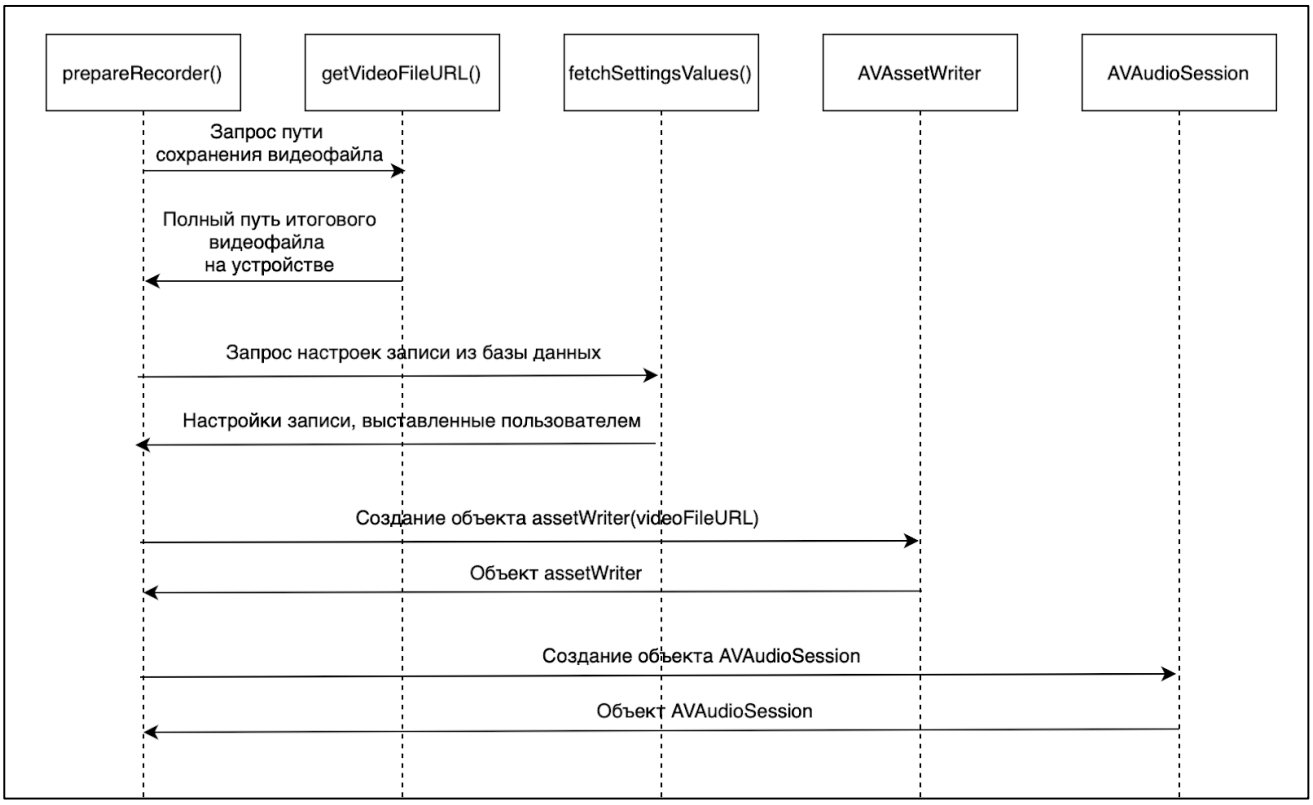

Рис. 4. Схема алгоритма инициализации компонентов видеозахвата

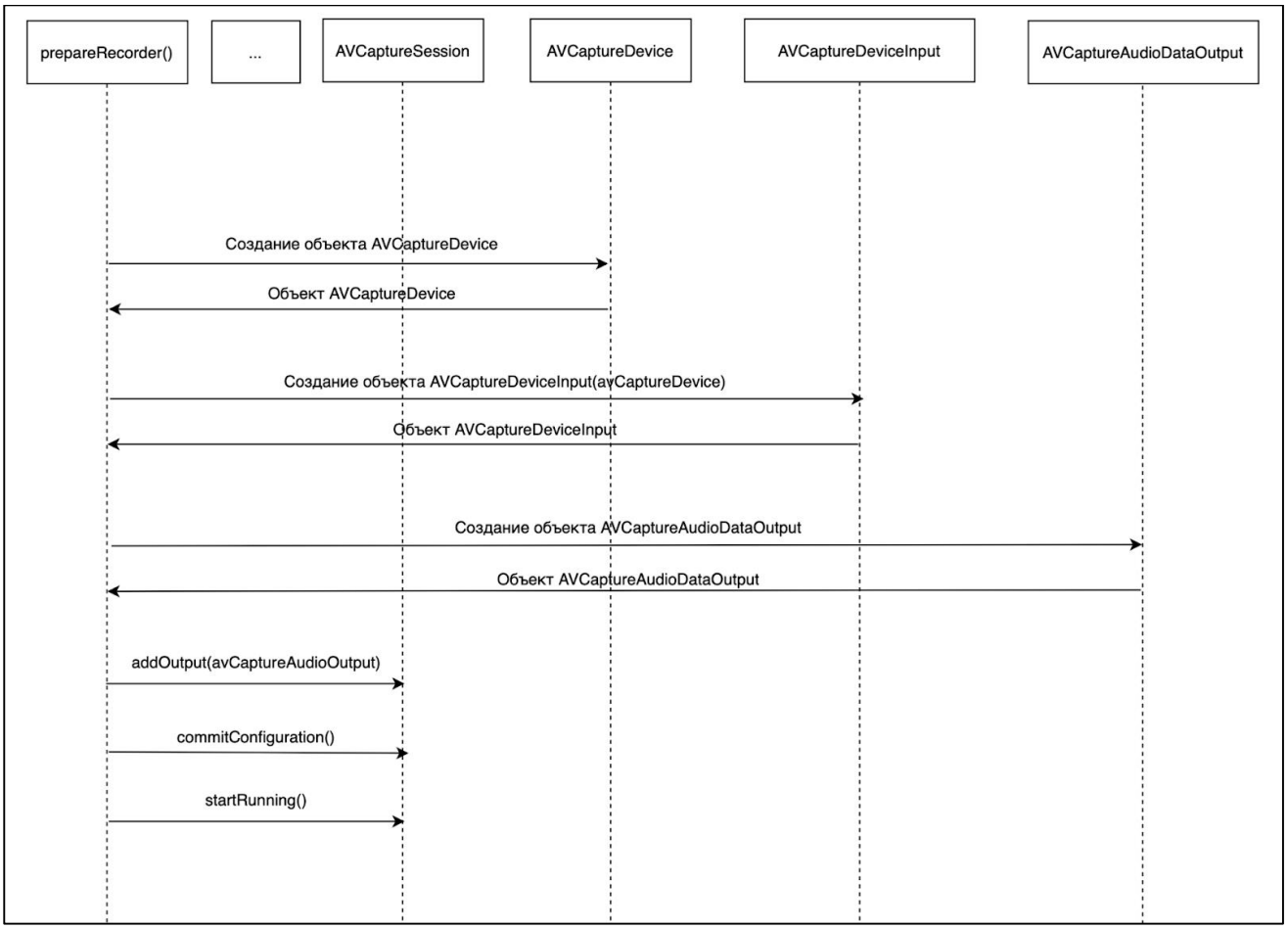

Рис. 5. Схема алгоритма инициализации компонентов видеозахвата (продолжение)

На рисунках 4 и 5 обозначены следующие методы и классы:

- *prepareRecorder()* вызывается при запуске приложения и является стартовой точкой для инициализации объектов видеозахвата;
- *getVideoFileURL()* – метод, производящий генерацию названия будущего видеофайла и возвращающий в качестве выходного параметра путь для его сохранения;
- *fetchSettingsValues()* производит запрос в сервис работы с базой данных для получения пользовательских настроек записи;
- *AVAssetWriter* – часть фреймворка *AVKit*, которая производит запись медиаданных в файл;

- *AVAudioSession* – компонент, выступающий в роли посредника между программным продуктом и механизмами, связанными с воспроизведением и записью аудио в системе iOS в целом;
- *AVCaptureSession* координирует потоки данных с устройств ввода (например, камеры или микрофона) для их записи;
- *AVCaptureDevice* является представлением физического устройства ввода;
- *AVCaptureDeviceInput* – обертка над классом *AVCaptureDevice*, которая отвечает за передачу данных от устройства ввода к *AVCaptureSession*;
- *AVCaptureAudioDataOutput* позволяет проводить аудиозапись и получать доступ к записанным звуковым данным в момент захвата.

ЗАПИСЬ ВИДЕОПОТОКА И СИНХРОНИЗАЦИЯ С ФРАГМЕНТАМИ АУДИО

При нажатии пользователем на кнопку записи начинают работу методы, связанные непосредственно с захватом и записью видеопотока в память устройства. С заранее заданными параметрами происходит создание буфера, куда записывается текущий кадр, полученный от ARView, а также вычисляется временная метка захвата кадра.

Вместе с этим производятся захват аудио и синхронизация с видеорядом. Алгоритм работает следующим образом:

1. сохраняется временная метка полученного буфера, содержащего фрагмент аудио длиной, эквивалентной 1 кадру;
2. вычисляется время представления путем вычитания временной метки начала записи (временной метки первого видеокадра) из временной метки буфера текущего фрагмента;
3. происходит создание нового объекта буфера путем копирования полученного и изменения его временной метки для синхронизации с видеорядом. Для решения данной задачи используется функция *CMSampleBufferCreateCopyWithNewTiming*, в качестве входных параметров принимающая:
    - *allocator* – указывает на то, какой аллокатор памяти будет использован при копировании буфера;
    - *sampleBuffer* – исходный буфер фрагмента аудио;

- *sampleTimingEntryCount* – количество элементов, подаваемых на вход, у которых необходимо изменить временную метку;
- *sampleTimingArray* – указатель на переменную, содержащую информацию о новой временной метке для фрагмента аудио;
- *sampleBufferOut* – указатель на переменную, куда будет произведено копирование исходного буфера.

4. новый объект буфера со временем представления, синхронизированным с видеорядом, записывается в *assetWriterAudioInput*.

В результате работы алгоритма записанные фрагменты аудио синхронизируются по времени с видеокадрами и вместе с ними добавляются в очередь на запись, благодаря чему видеоряд и записанный отдельно аудиоряд итогового видеофайла синхронизированы друг с другом. Визуальное представление работы алгоритма в формате “До – После” представлено на рисунках 6 и 7.

Рис. 6. Визуальное представление видеофайла до работы алгоритма

Рис. 7. Визуальное представление видеофайла после работы алгоритма

При нажатии пользователем на кнопку окончания записи *assetWriterVideoInput* и *assetWriterAudioInput* помечаются соответствующим флагом, а у *assetWriter* вызывается метод завершения работы, после чего происходит сохранение итогового видеофайла в галерею устройства с использованием нативного фреймворка Photos [20].

МЕХАНИЗМЫ ОБРАБОТКИ ВИДЕОПОТОКА

В кинематографе, как и в других областях визуальных искусств, включая фотографию и графический дизайн, применяется правило третей. Это принцип композиционного построения изображения, основанный на визуальном делении кадра на 9 частей при помощи сетки 3 на 3. Считается, что расположение основного объекта в кадре (например, актера) на пересечении линий сетки позволяет сделать кадр более визуально сбалансированным для глаза человека [21]. В программном решении реализовано два вспомогательных инструмента, связанных с правилом третей:

1. Механизм отслеживания положения головы актера в кадре и отображения предупреждения пользователю при нарушении правила третей, основанный на функционале нативного фреймворка Vision. Для отслеживания головы в пространстве применяется компонент VNFaceObservation, позволяющий покадрово получать координаты лица человека. Далее выполняется проверка на текущее расположение головы человека, и в случае, если граница лица человека пересекает линию пересечения противоположной стороны (для линии пересечения с левой стороны изображения – это правая граница лица, для линии с нижней стороны – верхняя часть лица, и аналогично для остальных сторон), то предполагается, что правило третей нарушено. При обнаружении данного нарушения на экран выводится иконка, обозначающее соответствующее предупреждение для пользователя, а вокруг лица, для которого, предположительно, было нарушено правило третей, происходит отрисовка красной рамки.
2. Сетка правила третей – отрисовка происходит на этапе жизненного цикла UIViewController *viewDidAppear* и производится путем создания на экране прозрачного слоя CALayer() поверх основного изображения и последующего добавления на этот слой линий, являющихся объектами UIBezierPath().

Также был добавлен механизм отслеживания качества изображения на предмет возможного отсутствия фокусировки камеры на лице человека. На небольшом экране устройства может быть затруднительно визуально определить, на чем сфокусирована камера в данный момент, из-за чего лицо актера может оказаться размытым, что станет заметно лишь при просмотре итогового видеофайла на большом экране. Поэтому в программное средство был добавлен вывод предупреждения, если качество картинки участка изображения с лицом человека окажется ниже заданного порогового уровня. Для реализации также был использован компонент VNFaceObservation.

МЕХАНИЗМЫ РАБОТЫ СО СЦЕНАРИЕМ

Сценарий в приложение загружается пользователем в текстовом формате, при этом предполагается, что текст, подаваемый на вход, состоит из набора отдельных реплик, имеющих вид, представленный на рисунке 8.

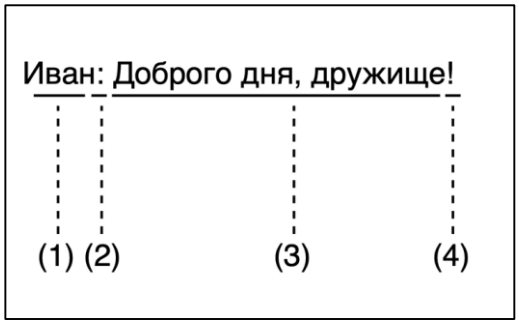

Рис. 8. Возможный вариант реплики, подаваемой на вход

Числовыми значениями на рисунке обозначены:

1. имя персонажа или актера;
2. символ двоеточия;
3. фраза (набор произвольных символов, кроме символов завершения предложения);
4. символ завершения предложения (точка, знак вопроса, восклицательный знак).

После загрузки пользователем текста сценария, удовлетворяющего шаблону, начинает работу алгоритм анализа и разделения текста на составные части. Суть

его работы заключается в следующем: алгоритм посимвольно проходится по тексту, полученному на вход. Как только следующим символом становится двоеточие, прочитанный текст записывается в переменную, отвечающую за имя. Затем происходит посимвольное чтение текста до символа завершения предложения. На данном этапе возможны 2 варианта:

1. реплика состоит из одного предложения, и то, что следует за символом завершения предложения, – следующая реплика;
2. реплика состоит из нескольких предложений, и то, что следует за символом завершения предложения, – продолжение данной реплики.

Соответственно программному решению необходимо учитывать возможность того, что реплика будет удовлетворять одному из двух вышеперечисленных вариантов, для чего алгоритм, как только следующим символом становится символ завершения предложения, предполагает, что следующий за ним текст может быть как новой репликой, так и продолжением текущей реплики. Определяется это следующим образом:

1. если реплика состоит из одного предложения, и символ завершения предложения обозначает ее завершение, то следующим за символом текстом будет имя персонажа, чья реплика идет далее по сценарию, соответственно после имени будет идти символ двоеточия;
2. если реплика состоит из нескольких предложений, и символ завершения предложения разделяет предложения внутри реплики, то следующим за символом текстом будет любой набор символов, после которого может идти любой небуквенный символ, кроме двоеточия.

Таким образом, алгоритм основан на предположении, что текст, следующий за символом завершения предложения, может быть как именем персонажа, кто говорит следующую реплику, именем внутри фразы, так и самой фразой.

МЕХАНИЗМЫ РАСШИФРОВКИ РЕЧИ

В программном решении механизмы расшифровки речи используются в тандеме с компонентом вывода субтитров на экран. При старте записи начинает свою работу инструмент распознавания речи, в основе которого лежат компоненты нативного фреймворка Speech [22] (рис. 9).

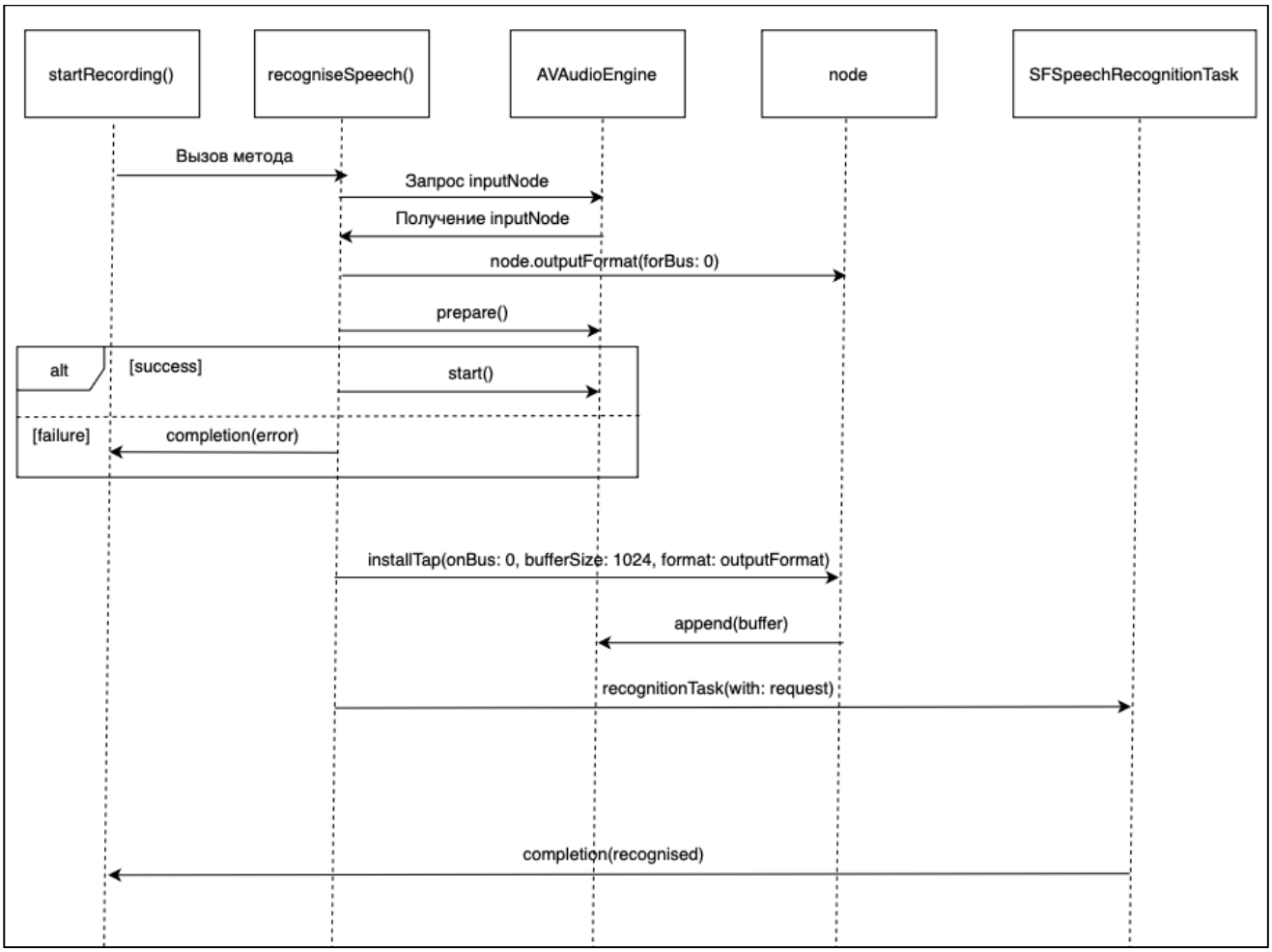

Рис. 9. Схема алгоритма транскрибирования речи

На рисунке 9 обозначены следующие методы и классы:

- *startRecording()* – метод, который вызывается при инициации пользователем начала записи. Является входной точкой компонента распознавания речи;
- *recogniseSpeech()* принимает замыкание completion(), которое вызывается при успешном распознавании или возникновении ошибки;
- AVAudioEngine() – аудиодвижок, отвечающий за получение звукового сигнала от узлов (node);
- *node / inputNode* – узел, производящий захват аудиосигнала;
- SFSpeechRecognitionTask – объект задачи распознавания речи;
- *outputFormat(forBus: 0)* – задание у узла формата записи;
- *prepare()* подготавливает аудиодвижок к запуску;

- *start()* производит запуск аудиодвижка;
- *installTap(onBus: 0, bufferSize: 1024, format: recordingFormat)* производит установку обработчика, захватывающего данные на узле inputNode;
- *append(buffer)* добавляет захваченные данные в буфер для распознавания;
- *recognitionTask(with: request)* запускает задачу распознавания речи;
- *completion(recognised)* вызывает замыкание completion() с распознанным текстом.

Результатом работы алгоритма является объект recognised, содержащий весь транскрибированный текст с момента начала записи. С целью проверки идентичности транскрибированного текста и текста фразы данные элементы обрезаются до двух последних слов, для чего применяются стандартные инструменты работы со строками языка Swift, и уже они проверяются на идентичность. Учитывая специфику работы инструментов распознавания речи, различия в регистре текста, а также расстановка знаков препинания в нем алгоритмом не учитываются.

Предполагается, что если в результате проведенных действий алгоритм выявил сходство распознанного текста и текста, заданного в сценарии, то реплика произнесена, и в субтитре происходит переключение на следующую фразу.

ЗАКЛЮЧЕНИЕ

Разработаны программные механизмы создания и пользовательской настройки анимированных трехмерных раскадровок. Для динамического вычисления скорости перемещения трехмерных моделей в нескольких режимах создана математическая модель с использованием метрики Евклида. Сохранение и загрузка контейнера элементов трехмерного окружения, настроек записи и загруженного сценария реализованы с использованием механизмов конвертации информации о моделях в примитивные типы данных. Для преодоления ограничений операционной системы iOS, связанных с доступом к сенсорам устройства, реализован механизм перехвата кадра у фреймворка дополненной реальности до этапа наложения трехмерной графики с параллельным вычислением временных меток каждого кадра и работающие одновременно с этим механизмы записи аудиопотока и последующей синхронизации видео- и аудиодорожек для форми-

рования итогового видеофайла с сохранением его в галерею устройства. Реализованы механизмы отслеживания положения головы человека в кадре и качества изображения на предмет возможной расфокусировки, основанные на применении технологий компьютерного зрения. Предложены также механизмы работы с загруженным в приложение сценарием, включая алгоритм сериализации сплошного текста для наглядного вывода его на экран, а также компонентов транскрибирования речи, захваченной с аудиопотока микрофона устройства, и последующего сравнения расшифрованной речи с текстом сценария.

СПИСОК ЛИТЕРАТУРЫ

1. *Wang H., Yang S., Dai Z.* The Development of Mobile Short Video Communication in the Context of the Mobile Internet // Science Insights. 2022. Vol. 40. No. 1. P. 421–426. URL: https://doi.org/10.15354/si.22.re011.

2. The Rise of Mobile Filmmaking: A New Era of Cinematic Expression // ECGProductions. 2023.
URL: https://www.ecgprod.com/the-rise-of-mobile-filmmaking/.

3. *Blahnik V., Schindelbeck O.* Smartphone imaging technology and its applications // Advanced Optical Technologies. 2021. Vol. 10, No. 3. P. 145–232.
URL: https://doi.org/10.1515/aot-2021-0023.

4. Smartphones vs Cameras: Closing the gap on image quality // DxOMark. 2021.
URL: https://www.dxomark.com/smartphones-vs-cameras-closing-the-gap-on-image-quality/.

5. *Schleser M.* Smartphone filmmaking: theory and practice. // Bloomsbury Publishing USA. 2021. P. 8–10.

6. *Friis I., Hansen A.* Line-item budgeting and film-production: Exploring some benefits of budget constraints on creativity // Qualitative Research in Accounting & Management. 2015. Vol. 12, No. 4. P. 321–345.
URL: https://doi.org/10.1108/QRAM-01-2015-0016.

7. *Balaban O., Levchenko O., Krupskyy I., Medvedieva A., Mykhalov V.* Script Structures in Modern Audio-Visual Art // Studies in Media and Communication. 2021. Vol. 9, No. 2. P. 45–50. URL: https://doi.org/10.11114/smc.v9i2.5386.

8. *Brown B.* Cinematography: Theory and Practice. 2016. P. 28–44.

9. LightSpace – 3D painting in air.
URL: https://apps.apple.com/ru/app/lightspace-3d-painting-in-ar/id1274597316.

10. Storyboard Animator.
URL: https://apps.apple.com/ru/app/storyboard-animator/id1326518944.

11. SUFLER.PRO.
URL: https://apps.apple.com/ru/app/sufler-pro/id1480258675.

12. *Shaner P.* Digital Filmmaking: An Introduction. // Mercury Learning and Information. 2011. P. 34–50.

13. Get live captions in real time on iPhone.
URL: https://support.apple.com/en-in/guide/iphone/iphe0990f7bb/ios.

14. Providing 3D virtual content with SceneKit.
URL: https://developer.apple.com/documentation/arkit/arscnview/providing3dvirtualcontentwith_scenekit.

15. ARKit Documentation.
URL: https://developer.apple.com/documentation/ arkit/.

16. Improving the performance of a RealityKit app.
URL: https://developer.apple.com/documentation/realitykit/improving-the-performance-of-a-realitykit-app.

17. SIMD Float4x4.
URL: https://developer.apple.com/documentation/ accelerate/simd_float4x4.

18. *Saito T., Toriwaki J.I.* New algorithms for euclidean distance transformation of an n-dimensional digitized picture with applications // Pattern Recognition. 1994. Vol. 27, No. 11. P. 1551–1565.

19. ARWorldMap.
URL: https://developer.apple.com/documentation/arkit/ arworldmap.

20. PhotoKit. URL: https://developer.apple.com/documentation/photokit.

21. Cinematography Techniques: The Different Types.
URL: https://rudafilms.com/gallery/Cinematography%20Techniques%20The%20Different%20Types.pdf.

22. Speech. URL: https://developer.apple.com/documentation/speech.

SOFTWARE TOOL FOR VIDEOPRODUCTION OPTIMISATION

R. F. Davletshin[1] [0009-0009-5326-3609], I. S. Shakhova[2] [0000-0003-1591-5767]

[1, 2]*Institute of Information Technology and Intelligent Systems, Kazan Federal University, Kazan*

[1]rustemd02@mail.ru, [2]is@it.kfu.ru

Abstract

The paper proposes software mechanisms aimed at enhancing video production processes for the authors of artistic video materials. We propose a mechanism for creating animated three-dimensional shooting plans (storyboards) using augmented reality to position and animate the movement of actors. In order to overcome the limitations of the iOS operating system related to access to sensors, we developed a mechanism for separately capturing audio and video streams from device sensors for recording and their subsequent synchronization by timestamps for saving to device memory. Computer vision technologies are used to ensure compliance with the rules of compositional construction and image quality analysis. The paper also presents mechanisms for working with the script, including text processing algorithms for displaying subtitles on the screen, and speech recognition algorithms for comparing speech recognition of actors with the text of the script.

Keywords: *video production, movie making, mobile cinema, augmented reality, storyboard, video recordings, automation, software solution.*

REFERENCES

1. *Wang H., Yang S., Dai Z.* The Development of Mobile Short Video Communication in the Context of the Mobile Internet // Science Insights. 2022. Vol. 40. No. 1. P. 421–426. URL: https://doi.org/10.15354/si.22.re011.

2. The Rise of Mobile Filmmaking: A New Era of Cinematic Expression // ECGProductions. 2023.
URL: https://www.ecgprod.com/the-rise-of-mobile-filmmaking/.

3. *Blahnik V., Schindelbeck O.* Smartphone imaging technology and its applications // Advanced Optical Technologies. 2021. Vol. 10, No. 3. P. 145–232.

URL: https://doi.org/10.1515/aot-2021-0023.

4. Smartphones vs Cameras: Closing the gap on image quality // DxOMark. 2021.
URL: https://www.dxomark.com/smartphones-vs-cameras-closing-the-gap-on-image-quality/.

5. *Schleser M.* Smartphone filmmaking: theory and practice. // Bloomsbury Publishing USA. 2021. P. 8–10.

6. *Friis I., Hansen A.* Line-item budgeting and film-production: Exploring some benefits of budget constraints on creativity // Qualitative Research in Accounting & Management. 2015. Vol. 12, No. 4. P. 321–345.
URL: https://doi.org/10.1108/QRAM-01-2015-0016.

7. *Balaban O., Levchenko O., Krupskyy I., Medvedieva A., Mykhalov V.* Script Structures in Modern Audio-Visual Art // Studies in Media and Communication. 2021. Vol. 9, No. 2. P. 45–50. URL: https://doi.org/10.11114/smc.v9i2.5386.

8. *Brown B.* Cinematography: Theory and Practice. 2016. P. 28–44.

9. LightSpace – 3D painting in air.
URL: https://apps.apple.com/ru/app/lightspace-3d-painting-in-ar/id1274597316.

10. Storyboard Animator.
URL: https://apps.apple.com/ru/app/storyboard-animator/id1326518944.

11. SUFLER.PRO.
URL: https://apps.apple.com/ru/app/sufler-pro/id1480258675.

12. *Shaner P.* Digital Filmmaking: An Introduction. // Mercury Learning and Information. 2011. P. 34–50.

13. Get live captions in real time on iPhone.
URL: https://support.apple.com/en-in/guide/iphone/iphe0990f7bb/ios.

14. Providing 3D virtual content with SceneKit.
URL: https://developer.apple.com/documentation/arkit/arscnview/providing3dvirtualcontentwith_scenekit.

15. ARKit Documentation.
URL: https://developer.apple.com/documentation/ arkit/.

16. Improving the performance of a RealityKit app.

URL: https://developer.apple.com/documentation/realitykit/improving-the-performance-of-a-realitykit-app.

17. SIMD Float4x4.

URL: https://developer.apple.com/documentation/ accelerate/simd_float4x4.

18. *Saito T., Toriwaki J.I.* New algorithms for euclidean distance transformation of an n-dimensional digitized picture with applications // Pattern Recognition. 1994. Vol. 27, No. 11. P. 1551–1565.

19. ARWorldMap.

URL: https://developer.apple.com/documentation/arkit/ arworldmap.

20. PhotoKit. URL: https://developer.apple.com/documentation/photokit.

21. Cinematography Techniques: The Different Types.

URL: https://rudafilms.com/gallery/Cinematography%20Techniques%20The%20Different%20Types.pdf.

22. Speech. URL: https://developer.apple.com/documentation/speech.

СВЕДЕНИЯ ОБ АВТОРАХ

ДАВЛЕТШИН Рустем Фаридович – бакалавр, выпускник Института информационных технологий и интеллектуальных систем Казанского федерального университета, г. Казань.

Rustem Faridovich DAVLETSHIN – Bachelor's degree graduate from the Institute of Information Technology and Intelligent Systems at Kazan Federal University, Kazan.

email: rustemd02@mail.ru

ORCID: 0009-0009-5326-3609

ШАХОВА Ирина Сергеевна – старший преподаватель кафедры программной инженерии Института информационных технологий и интеллектуальных систем Казанского федерального университета, г. Казань.

Irina Sergeevna SHAKHOVA – Senior Lecturer at the Department of Software Engineering, Institute of Information Technology and Intelligent Systems, Kazan Federal University, Kazan.

email: is@it.kfu.ru

ORCID: 0000-0003-1591-5767

Материал поступил в редакцию 18 июля 2024 года