УДК 004.93+004.5

СИНХРОНИЗАЦИЯ СЕССИЙ ДОПОЛНЕННОЙ РЕАЛЬНОСТИ В НАТИВНЫХ МОБИЛЬНЫХ ПРИЛОЖЕНИЯХ

Д. А. Евдокименко[1], Р. Г. Ханов[2], И. С. Шахова[3]

Высшая школа информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета

[1]evdodima@gmail.com, [2]rinat@khanov.com, [3]is@it.kfu.ru

Аннотация

Представлена реализация алгоритма синхронизации сессий дополненной реальности в мобильных iOS-приложениях, позволяющего создавать такие сессии с несколькими участниками для их совместного взаимодействия с одними и теми же виртуальными объектами.

***Ключевые слова**: дополненная реальность, augmented reality, AR, ARKit, сессия дополненной реальности, синхронизация сессий дополненной реальности*

ВВЕДЕНИЕ

Дополненная реальность – это среда с прямым или косвенным дополнением физического мира цифровыми данными в режиме реального времени при помощи компьютерных устройств [1]. На мобильных устройствах технологии дополненной реальности могут быть использованы, например, для отображения маячков или иконок на карте в реальном времени при наведении камеры на местность.

В iOS 11 компанией Apple представлен новый фреймворк ARKit, позволяющий применять технологию дополненной реальности (AR, augmented reality) в iOS-приложениях. ARKit, используя информацию со встроенных датчиков устройства, таких, как акселерометр, гироскоп, магнитометр, а также анализируя изображение, полученное с камеры, позволяет определять положение устройства в пространстве относительно других объектов – реальных или виртуальных. Помимо отображения виртуальных объектов данный фреймворк способен распознавать в реальном мире горизонтальные поверхности (начиная с версии 1.5,

и вертикальные). Это позволяет, например, размещать виртуальные объекты на распознанных поверхностях реального мира, благодаря чему виртуальные объекты остаются зафиксированными в конкретных точках на поверхностях реального мира и выглядят более естественными. Он также позволяет, например, проводить измерения расстояний или площадей на данных поверхностях.

Стандартные методы в ARKit предоставляют разработчику информацию о расположении устройства относительно начальной точки отсчета и распознанных горизонтальных и вертикальных плоскостей в реальном мире. В текущей версии ARKit все эти данные хранятся внутри одной сессии дополненной реальности (AR-сессии) на устройстве пользователя, а система координат на каждом устройстве находится в разных точках, в которых сессия была начата. Данные ограничения не позволяют делиться данными сессии с другими устройствами, а также корректно размещать виртуальные объекты из сессий других устройств, так как системы координат расположены по-разному. Таким образом, в текущей версии ARKit не представляется возможным создавать сессии дополненной реальности с несколькими участниками для их совместного взаимодействия с одними и теми же объектами дополненной реальности. Отсутствие возможности создания сессий дополненной реальности с несколькими участниками сильно ограничивает области, в которых технологии дополненной реальности могли бы применяться на мобильных устройствах. В таких сферах, как образование, медицина, проектирование, а также развлечения, дополненная реальность может быть применена гораздо эффективнее, если в процесс взаимодействия с виртуальными объектами будет вовлечено несколько пользователей [2].

В статье представлены реализация алгоритма передачи и синхронизации данных различных AR-сессий между несколькими устройствами и обеспечение совместной работы этих устройств внутри одной сессии дополненной реальности. Для реализации данного алгоритма используются нативные для iOS инструменты и технологии: язык программирования Swift 4.0, фреймворк ARKit 1.5 и системный фреймворк SceneKit для 3D графики, а также системный фреймворк Multipeer Connectivity для организации передачи данных между устройствами.

ПЕРЕДАЧА ДАННЫХ

Объединение устройств в сеть для передачи данных

Multipeer Connectivity Framework – это нативный для системы iOS фреймворк, который позволяет передавать данные, потоки и файлы между iOS-устройствами, находящимися рядом. Для этого используются WiFi сети, прямое WiFi соединение или Bluetooth [3]. Процесс работы фреймворка состоит из двух стадий: стадия обнаружения, во время которой происходит поиск устройств, с которыми будет осуществляться передача данных, и стадия взаимодействия (сессии), во время которой и происходит передача данных.

Для объединения устройств в сеть нужно реализовать метод advertiser(didRecieveInvationFromPeer:), который вызывается у делегата MCNearbyServiceAdvertiser, когда устройство получает приглашение присоединиться к сети. Внутрь этого метода передается функция invitationHandler, которая принимает в качестве параметра булевскую переменную, являющуюся ответом на приглашение (принимается приглашение или нет). Также нужно реализовать метод browser(foundPeer:), который вызывается у делегата MCNearbyServiceBrowser, когда найдено новое устройство. В данном методе нужно пригласить устройство в сеть. Для этого вызывается метод invitePeer. Для начала работы фреймворка нужно вызвать методы startAdvertising и startBrowsing у соответствующих объектов. Далее новые устройства будут находить друг друга и объединяться в сеть автоматически.

Передача данных о расположении устройств

В ARKit для отображения виртуальных объектов и взаимодействия с ними используется фреймворк SceneKit. Информацию о расположении устройства и его ориентации на сцене можно получить из объекта pointOfView, который находится на сцене и обновляется фреймворком ARKit, – этот объект отображает, где сейчас на сцене находится пользователь (наблюдатель).

Расположение устройства хранится в виде трех координат (X,Y,Z) с помощью структуры SCNVector3 [4]. Ориентация устройства (его углы поворота) также хранится в виде SCNVector3, где X, Y и Z – углы поворота вокруг соответствующих осей. Положение этих осей координат показано на Рис. 1, вектор (X, Y, Z).

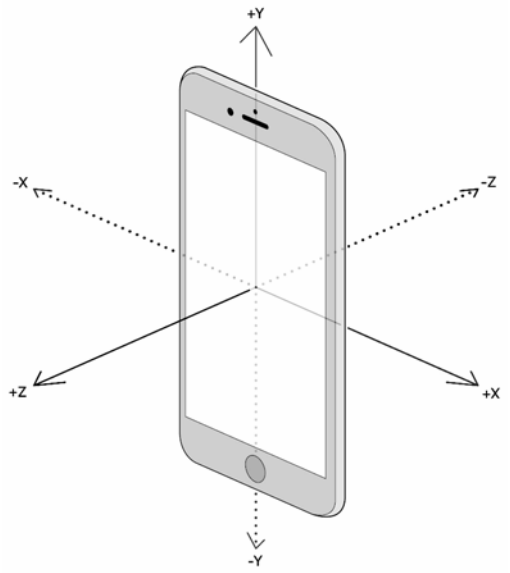

Рис. 1. Расположение осей координат, определяющих поворот устройства

Таким образом, чтобы однозначно определять положения устройств в пространстве, достаточно передавать два вектора: вектор расположения и вектор поворота.

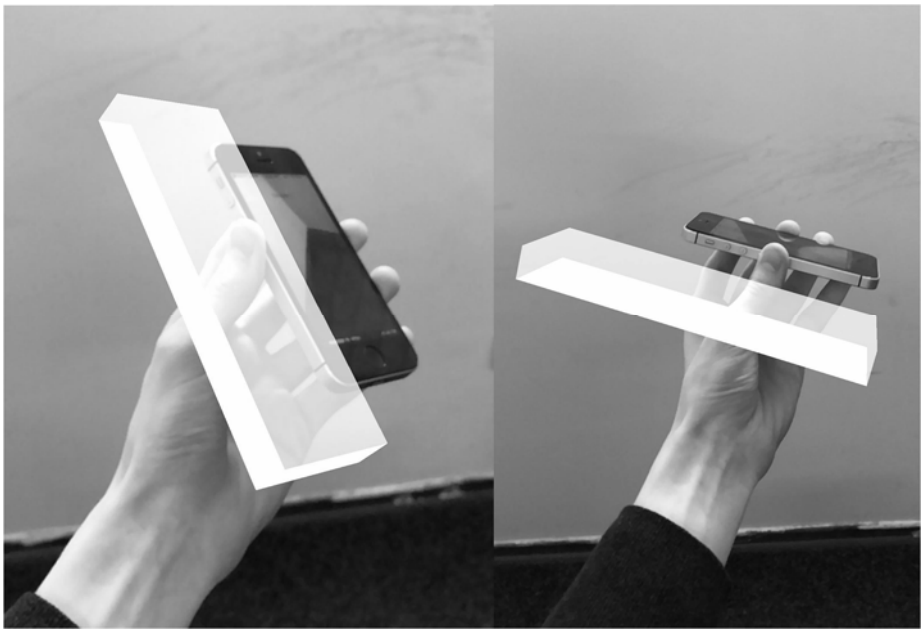

Рис. 2. Пример визуализации положения устройства

Так как данные о положении устройств обновляются постоянно, удобнее и быстрее всего для их передачи использовать потоки данных. В Multipeer Connectivity используются стандартные для языка Swift потоки ввода/вывода Input/OutputStream [5]. Для удобства положения всех устройств на сцене визуализируются в виде параллелепипедов с соотношением сторон, похожим на со-

отношение сторон телефона. В тот момент, когда из потока считываются данные о положении устройства, параллелепипед перемещается в точку с полученными координатами и поворачивается соответственно полученному вектору поворота. На Рис. 2 показаны примеры визуализации положений устройств с помощью данного параллелепипеда.

Передача данных об объектах на сцене

Чтобы обеспечить полную синхронизацию сцен на разных устройствах, необходимо, помимо положений на сцене самих устройств, передавать информацию обо всех объектах сцены и их обновлениях. В SceneKit основным объектом сцены является узел (SCNNode). Основными параметрами узла являются его расположение, ориентация и геометрия. Геометрия в свою очередь имеет определенные вид (сфера, плоскость, куб и т. д.), размер, а также материал [6]. В данном проекте в качестве объектов сцены, которые будут синхронизироваться между устройствами, выступают распознанные с помощью ARKit горизонтальные плоскости, значит, вид и материал геометрии для всех объектов будут одинаковыми.

ARKit предоставляет информацию о распознаваемых плоскостях и позволяет их визуализировать. Для этого существуют три метода делегата ARSCNViewDelegate:

- renderer(didAdd node:) – вызывается, когда была распознана новая плоскость;
- renderer(didUpdate node:) – вызывается, когда для существующей плоскости были обновлены данные (например, размер);
- renderer(didRemove node:) – вызывается, когда ранее распознанная плоскость удаляется (например, если произошло ее слияние с более крупной плоскостью).

Таким образом, можно выделить три вида событий, с помощью которых происходит обновление сцены: add, update и remove. Все эти события так или иначе изменяют состояние сцены и ее объектов.

Удобным и быстрым способом обмена информацией о состоянии сцены является передача по сети данных событий. Такое решение не нагружает сеть постоянной передачей однотипной информации, а также является знакомым и

понятным для разработчиков, так как представляет собой реализацию шаблона проектирования Observer [7].

В Multipeer Connectivity имеется возможность передавать по сети объекты типа Data – это значит, что любой объект, реализующий интерфейс Codable, может быть закодирован с помощью Encoder и передан по сети другим устройствам. В данном случае таким объектом будет объект класса Event, содержащий информацию о передаваемом событии. На Рис. 3 приведена блок-схема алгоритма обработки данных о событии, получаемых устройством.

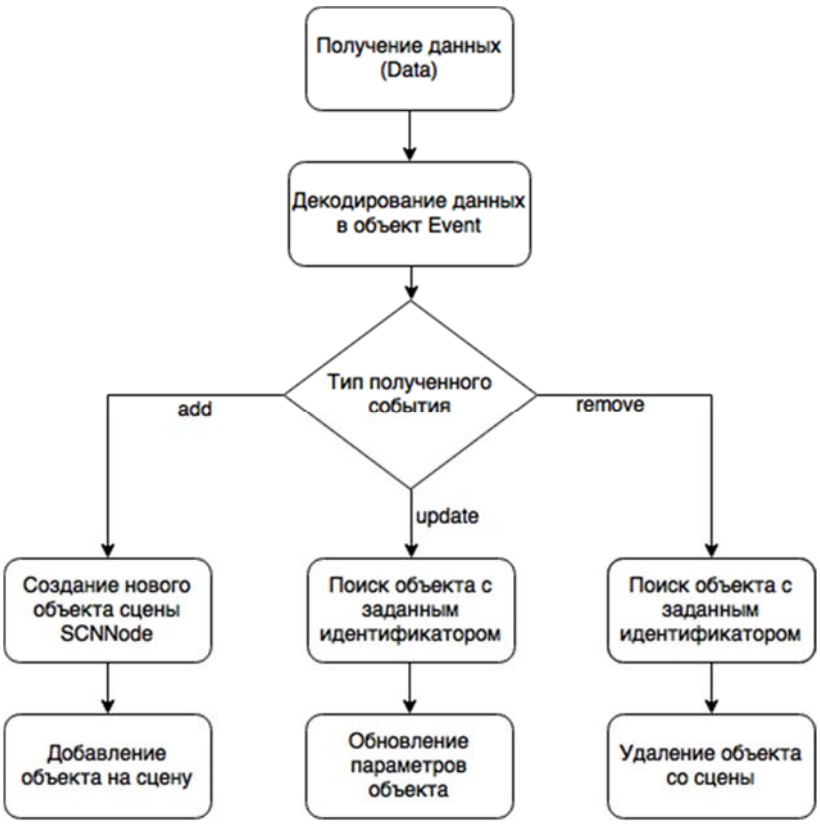

Рис. 3. Блок-схема алгоритма обработки событий

В ARKit удобно визуализировать распознанные плоскости с помощью узлов с геометрией SCNPlane, которая представляет собой плоскость, размер которой можно ограничить по ширине и высоте [8]. В качестве материала плоскости используется сетка из гексагонов (см. Рис. 4).

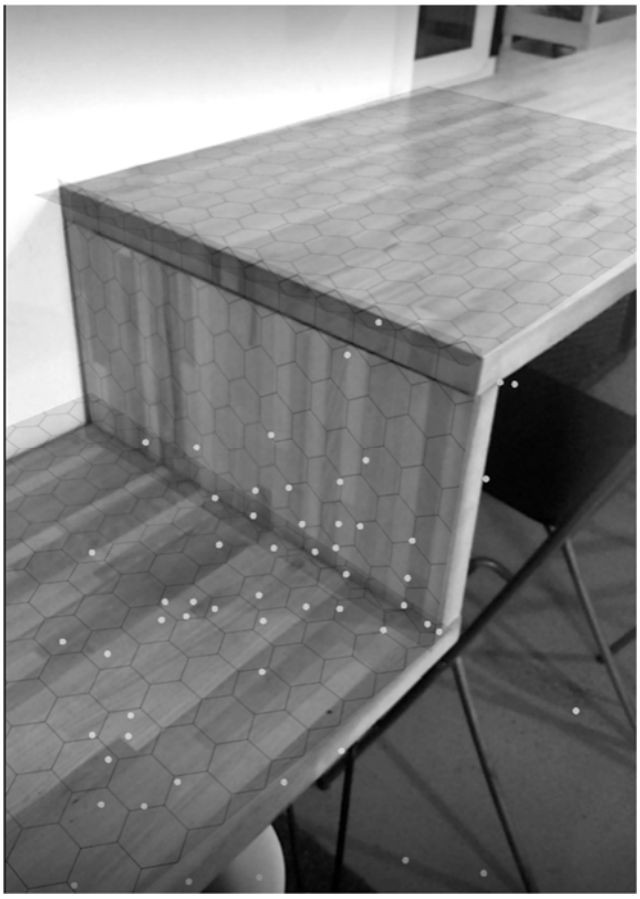

Рис. 4. Визуализация распознанных плоскостей

СИНХРОНИЗАЦИЯ СИСТЕМ КООРДИНАТ

Распознавание и локализация изображений

Начиная с версии 1.5, в ARKit появилась возможность распознавать и локализовывать (определять расположение и ориентацию в пространстве) заданные изображения [9]. Информация о распознанных изображениях, как и информация о распознанных плоскостях, будет передаваться приложению с помощью методов ARSCNViewDelegate. Пример работы этой технологии представлен на Рис. 5.

Синхронизация систем координат двух устройств

Данная технология может быть применена не только для распознавания статичных изображений в реальном мире, но и для таких изображений, расположение которых изменяется. Это позволяет, например, расположить заданную картинку на экране другого мобильного телефона или планшета, что позволит,

распознавая данное изображение, получать информацию о расположении устройства в пространстве и его ориентации. Чтобы такое изображение было корректно распознано на экране устройства, нужно правильно задать его физические размеры и, соответственно, отобразить его на экране таким образом, чтобы оно соответствовало заданным размерам. Также изображение должно соответствовать следующим условиям [10]:

Рис. 5. Распознавание и локализация изображений

- не должно иметь центральную симметрию (например, изображение круглой мишени будет плохо распознаваться, так как не ясно, где у него верх, а где низ);
- не должно иметь однородных цветовых областей;
- должно иметь высокую контрастность;
- должно быть хорошо текстурировано.

Когда устройства объединяются в сеть для передачи данных, их системы координаты не синхронизированы: начало координат сессии дополненной реальности в каждом устройстве находится в том месте, где был телефон, когда началась сессия, а поворот системы координат – такой же, как у устройства в момент начала сессии. Поэтому недостаточно просто передавать информацию о

расположении объектов и устройств на сцене, так как в разных устройствах одни и те же объекты сцены будут расположены в разных местах относительно объектов реального мира. Поэтому, прежде чем начинать взаимодействие пользователей с дополненной реальностью, необходимо синхронизировать системы координат их устройств.

Для синхронизации систем координат сессий дополненной реальности подходит технология, описанная ранее: она позволяет определить реальное расположение устройства, с которым происходит синхронизация, и сопоставить его реальные координаты с теми, которые присылает данное устройство. Далее, зная реальные координаты устройства и те координаты, которые получены от данного устройства, нужно переместить систему координат устройства таким образом, чтобы эти координаты совпали.

Таким образом, для синхронизации систем координат двух устройств подходит следующий алгоритм:

- объединение устройств в сеть для передачи данных;
- начало передачи данных о расположении и ориентации устройств между собой;
- определение главного устройства, с которого будет производиться распознавание изображения, и второстепенного, на экране которого будет показано изображение для распознавания;
- распознавание главным устройством изображения, показанного на экране второстепенного;
- получение на главном устройстве координат распознанного изображения и, соответственно второго устройства;
- нахождение различия в углах поворота систем координат.
- поворот системы координаты главного устройства на вычисленный угол;
- нахождение вектора разности положений систем координат устройств;
- перемещение начала системы координат главного устройства на вычисленный вектор.

В результате работы данного алгоритма начало системы координат главного устройства поворачивается согласно повороту системы координат второстепенного устройства, а начало системы координат помещается в ту же точку, где находится начало системы координат второстепенного устройства.

Обобщенный алгоритм синхронизации

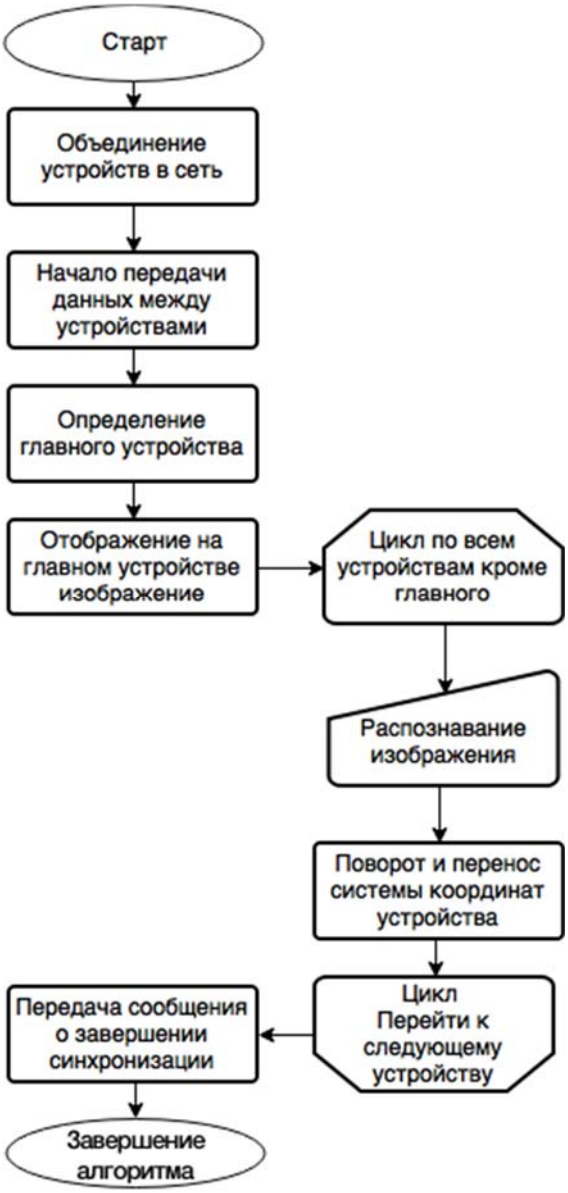

Рис. 6. Блок-схема обобщенного алгоритма синхронизации

Описанный алгоритм подходит только для синхронизации двух устройств, однако его можно обобщить на случай любого числа устройств. Для этого необходимо выделить главное устройство, к системе координат которого будут приведены системы координат остальных устройств. Затем, применив алгоритм

синхронизации из предыдущего параграфа для каждого второстепенного устройства, путем распознавания изображения, показанного на экране главного устройства, нужно переместить систему координат этого устройства, чтобы она соответствовала системе координат главного. На Рис. 6 приведена блок-схема работы обобщенной версии алгоритма.

ЗАКЛЮЧЕНИЕ

Построенный алгоритм выполняет следующие функции:

- объединение устройств в сеть для передачи данных;
- синхронизация информации о положении всех устройств на сцене;
- синхронизация информации о распознаваемых плоскостях.

Таким образом, данный алгоритм позволяет находящимся рядом устройствам работать внутри одной AR-сессии. На Рис. 7 приведен пример работы алгоритма для двух устройств.

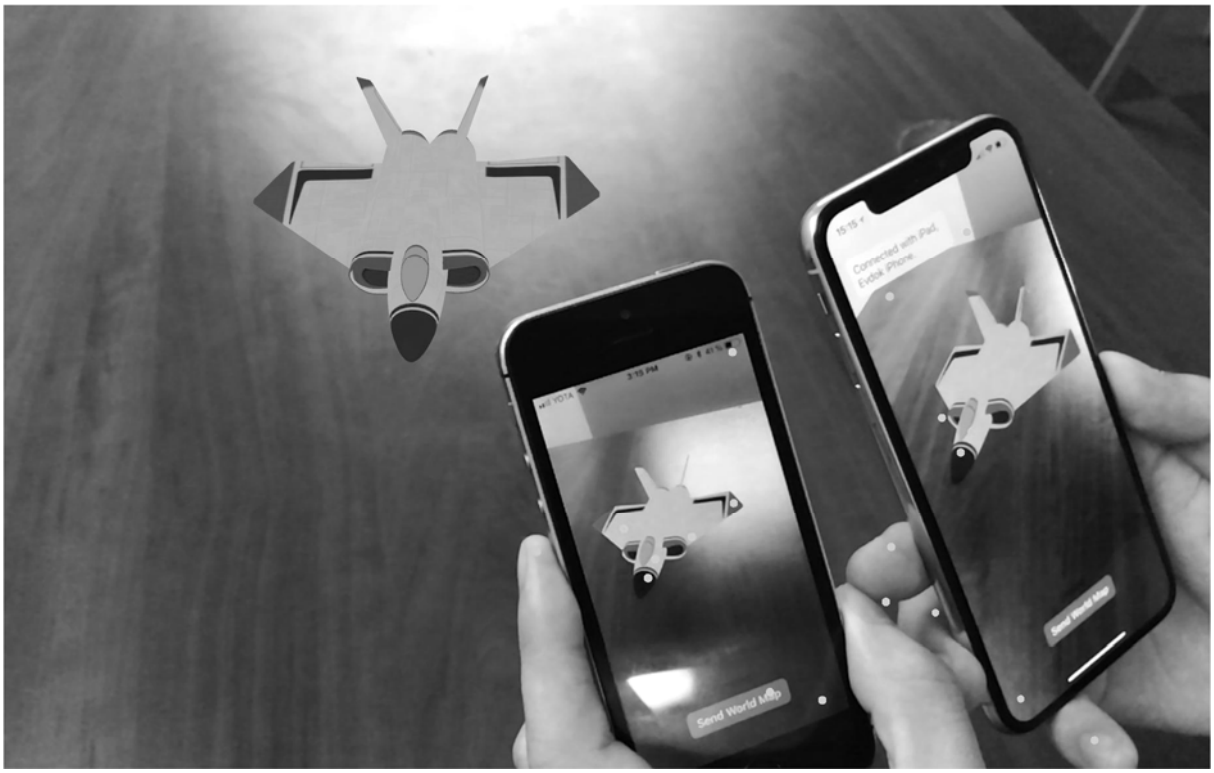

Рис. 7. Пример работы алгоритма

Описанный алгоритм имеет широкий спектр применения, так как позволяет реализовать многопользовательские AR-приложения – игры, приложения для совместного просмотра 3D-контента и взаимодействия с ним, навигационные

приложение и другие. Описанный алгоритм может быть выделен в отдельную библиотеку, что позволит разработчикам, подключая данную библиотеку с помощью CocoaPods [11], без проблем объединять устройства для совместной работы в дополненной реальности.

СПИСОК ЛИТЕРАТУРЫ

1. История VR/AR. URL: https://rb.ru/story/vsyo-o-vr-ar/

2. Towards Massively Multi-User Augmented Reality on Handheld Devices. URL: https://pdfs.semanticscholar.org/05be/d977601c84ae581a9a8b2 054ce484b342 e10.pdf

3. Multipeer Connectivity Framework iOS 7. URL: https://habrahabr.ru/company/touchinstinct/blog/198814/

4. SceneKit. URL: https://www.raywenderlich.com/83748/ beginning-scene -kit-tutorial

5. Класс Stream – документация Apple. URL: https://developer. apple.com/documentation/foundation/stream

6. Структура сцены в SceneKit. URL: https://www.invasivecode.com/ weblog/scenekit-tutorial-part-1/?doing_wp_cron=1513691954. 507189998908996582031 25

7. Шаблон проектирования Observer. URL: http://design-pattern.ru/ patterns/observer.html

8. Свойства геометрии SCNPlane. URL: http://spec-zone.ru/RU/OSX/ documentation/SceneKit/Reference/SCNPlane_Class/index.html

9. Распознавание изображений в ARKit. URL: https://developer. apple.com/documentation/arkit/recognizing_images_in_an_ar_experience

10. Требования к распознаваемому изображению. URL: https://developer. apple.com/documentation/arkit/recognizing_images_in_an_ar_experience

11. Менеджер зависимостей Cocoapods. URL: https://cocoapods.org

AUGMENTED REALITY SESSIONS SYNCHRONIZATION ALGORITHM IN MOBILE APPLICATIONS

D. A. Evdokimenko[1], R. G. Khanov[2], I. S. Shakhova[3]

Higher School of Information Technologies and Intelligent Systems at Kazan (Volga region) Federal University

[1]evdodima@gmail.com, [2]rinat@khanov.com, [3]is@it.kfu.ru

Abstract

Implementation of augmented reality session synchronization algorithm in mobile iOS applications. Algorithm allows to create augmented reality sessions with several participants for their joint interaction with the same virtual objects.

Keywords: *augmented reality, AR, ARKit, augmented reality session, augmented reality session synchronization*

REFERENCES

1. History of VR/AR. URL: https://rb.ru/story/vsyo-o-vr-ar/
2. Towards Massively Multi-User Augmented Reality on Handheld Devices. URL: https://pdfs.semanticscholar.org/05be/d977601c84ae581a9a8b2054ce484b342e10.pdf
3. Multipeer Connectivity Framework iOS 7. URL: https://habrahabr.ru/company/touchinstinct/blog/198814/
4. SceneKit. URL: https://www.raywenderlich.com/83748/beginning-scene-kit-tutorial
5. Class Stream – *Apple Documentation.* URL: https://developer. apple.com/ documentation/foundation/stream
6. SceneKit scene structure. URL: https://www.invasivecode.com/weblog/scenekit-tutorial-part-1/?doing_wp_cron=1513691954. 5071899890899658203125
7. Observer design pattern. URL: http://design-pattern.ru/ patterns/observer.html
8. SCNPlane geometry properties. URL: http://spec-zone.ru/RU/OSX/ documentation/SceneKit/Reference/SCNPlane_Class/index.html
9. Image recognition in ARKit. URL: https://developer. apple.com/documentation/arkit/recognizing_images_in_an_ar_experience

10. Requirements for the recognized image. URL: https://developer.apple.com/documentation/arkit/recognizing_images_in_an_ar_experience

11. Cocoapods dependencies manager. URL: https://cocoapods.org

СВЕДЕНИЯ ОБ АВТОРАХ

ХАНОВ Ринат Гафурович – студент Высшей школы информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета, направление «Прикладная информатика».

Rinat Gafurovich KHANOV, student of the Higher School of Information Technologies and Intelligent Systems of Kazan Federal University.

email: rinat@khanov.com

ЕВДОКИМЕНКО Дмитрий Андреевич – студент Высшей школы информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета, направление «Прикладная информатика».

Dmitriy Andreevich EVDOKIMENKO, student of the Higher School of Information Technologies and Intelligent Systems of Kazan Federal University.

email: evdodima@gmail.com

ШАХОВА Ирина Сергеевна – ассистент кафедры программной инженерии Высшей школы информационных технологий и интеллектуальных систем Казанского федерального университета. Сфера научных интересов - цифровые образовательные системы, индивидуализация образования, мобильное обучение.

Irina Sergeevna SHAKHOVA – teacher of the Higher School of Information Technologies and Intelligent Systems at Kazan Federal University. Research interests include digital educational systems, individualization of education, mobile learning.

email: is@it.kfu.ru

Материал поступил в редакцию 28 мая 2018 года