МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»

Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой ЭВМ
_____ Д.В. Топольский
«___»_____ 2022 г.

РАЗРАБОТКА ANDROID ПРИЛОЖЕНИЯ ДОПОЛНЕННОЙ РЕАЛЬНОСТИ ДЛЯ ОЦЕНКИ БЛЮД ПО QR-КОДУ В МЕНЮ

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУРГУ-090301.2022.254 ПЗ ВКР

Руководитель работы,
к.т.н., доцент каф. ЭВМ
_____П.О. Шабуров
«___»_____2022 г.

Автор работы,
студент группы КЭ-406
_____Я.И. Сотин
«___»_____2022 г.

Нормоконтролёр,
к.п.н., доцент каф. ЭВМ
_____ М.А. Алтухова
«___»_____2022 г.

Челябинск-2022

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

УТВЕРЖДАЮ
Заведующий кафедрой ЭВМ
_____ Д.В. Топольский
«___»_____2022 г.

ЗАДАНИЕ

на выпускную квалификационную работу бакалавра

студенту группы КЭ-406
Сотину Якову Ивановичу,
обучающемуся по направлению
09.03.01 «Информатика и вычислительная техника»

1. **Тема работы:** «Разработка Android приложения дополненной реальности для оценки блюд по QR-коду в меню» утверждена приказом по университету от __ декабря 2021 г. №____

2. **Срок сдачи студентом законченной работы: 1 июня 2022 г.**

3. **Исходные данные к работе:**

Разрабатываемое приложение обеспечивает следующие возможности:

– пользователю – получать интересующую информацию о блюде;

– пользователю – быстро и удобно оставить отзыв о блюде после оплаты чека;

– администрации ресторана – сэкономить место на бумаге в меню;

– администрации ресторана – повысить привлекательность своего заведения за инновационность.

4. **Перечень подлежащих разработке вопросов:**

– анализ существующих программных продуктов, используемых для оценки блюд;

– анализ современных программных технологий и программных решений, для решения задачи определения QR-кодов в реальном времени;

– анализ современных программных технологий и программных решений, для решения задачи отображения плоских AR объектов;

– разработка собственного ПО, для оценки блюд по QR-коду с технологией AR и его модулей;

– оценка работоспособности разработанного программного комплекса и его модулей, для решения практических задач.

5. **Дата выдачи задания:** 1 декабря 2021 г.

Руководитель работы _____/*П.О. Шабуров*/

Студент _____/*Я.И. Сотин*/

КАЛЕНДАРНЫЙ ПЛАН

Этап	Срок сдачи	Подпись руководителя
Введение и обзор литературы	01.01.2022	
Разработка модели, проектирование: – анализ существующих программных продуктов, используемых для оценки блюд;	10.01.2022	
– анализ современных программных технологий и программных решений, для решения задачи определения QR-кодов в реальном;	20.01.2022	
– анализ современных программных технологий и программных решений, для решения задачи отображения плоских AR объектов.	28.02.2022	
Реализация системы: – разработка модуля, фиксирующего визуальные данные с камеры;	2.02.2022	
– разработка модуля обнаружения и захвата изображения QR-кода в поле зрения камеры;	8.02.2022	
– разработка модуля чтения текста и ссылок с QR-кода;	12.02.2022	
– разработка модуля запроса информации из сети при помощи запроса GET HTTP протокола;	16.02.2022	

Этап	Срок сдачи	Подпись руководителя
– разработка модуля отрисовки информации поверх изображения с камеры;	20.02.2022	
– разработка модуля отправки отзывов с помощью POST запросов HTTP протокола на сайт по ссылке;	24.02.2022	
– разработка модуля пользовательского интерфейса для управлением приложением.	31.03.2022	
Тестирование, отладка, эксперименты: – отладка модуля, фиксирующего визуальные данные с камеры;	2.04.2022	
– отладка модуля обнаружения и захвата изображения QR-кода в поле зрения камеры;	10.04.2022	
– отладка модуля чтения текста и ссылок с QR-кода;	18.04.2022	
– отладка модуля запроса информации из сети при помощи запроса GET HTTP протокола;	26.04.2022	
– отладка модуля отрисовки информации поверх изображения с камеры;	1.05.2022	
– отладка модуля отправки отзывов с помощью POST запросов HTTP протокола на сайт по ссылке;	10.05.2022	
– отладка модуля пользовательского интерфейса для управлением приложением.	18.05.2022	
Компоновка текста работы и сдача на нормоконтроль	24.05.2022	

Этап	Срок сдачи	Подпись руководителя
Подготовка презентации и доклада	30.05.2022	

Руководитель работы _____/*П.О. Шабуров*/

Студент _____/*Я.И. Сотин*/

АННОТАЦИЯ

Я.И. Сотин. Разработка Android приложения дополненной реальности для оценки блюд в меню по QR-коду. – Челябинск: ФГАОУ ВО «ЮУрГУ (НИУ)», ВШ ЭКН; 2022, 55 с., 13 ил., библиогр. список – 11 наим.

В рамках выпускной квалификационной работы был проведён анализ существующих аналогов, а также схожих по функционалу систем. В результате анализа были выявлены недостатки этих систем, с целью их ликвидации в выпускной квалификационной работе, а также достоинства этих систем, с целью включения их в выпускную квалификационную работу. Рассмотрены основные технологии, применяющиеся в разработке схожих систем, и были выбраны наиболее подходящие для данного проекта.

После анализа были произведены проектирование, разработка и тестирование приложения.

Результатом работы являются Android приложение дополненной реальности для оценки блюд по QR-коду в меню.

СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

AR – (англ. Augmented Reality – дополненная реальность) воспринимаемая смешанная реальность, создаваемая с помощью компьютера с использованием «дополненных» элементов воспринимаемой реальности, когда реальные объекты монтируются в поле восприятия

3D – (3 Dimensions) трёхмерная реальность

2D – (2 Dimensions) двумерная реальность

HTML – (от англ. HyperText Markup Language – язык гипертекстовой разметки) – стандартизированный язык гипертекстовой разметки документов для просмотра веб-страниц в браузере

HTTP – (англ. HyperText Transfer Protocol – протокол передачи гипертекста) – протокол прикладного уровня передачи данных, в виде гипертекстовых документов в формате HTML, в настоящее время используется для передачи произвольных данных

IDEF0 — методология функционального моделирования (англ. function modeling) и графическая нотация, предназначенная для формализации и описания бизнес-процессов

QR – (англ. Quick Response code – код быстрого отклика) тип матричных штриховых кодов (или двухмерных штриховых кодов)

UI – (англ. User Interface) интерфейс, через который пользователь взаимодействует с системой

UML – (англ. Unified Modeling Language – унифицированный язык моделирования) – язык графического описания для объектного моделирования в области разработки программного обеспечения, для моделирования бизнес-процессов, системного проектирования и отображения организационных структур

URL – (англ. Uniform Resource Locator) – система унифицированных адресов электронных ресурсов, или единообразный определитель местонахождения ресурса (файла)

Браузер – (от англ. web browser) – прикладное программное обеспечение для просмотра страниц, содержания веб-документов, компьютерных файлов и их каталогов; управления веб-приложениями; а также для решения других задач

ГОСТ – (Государственный стандарт) требования государства к качеству продукции, работ и услуг, имеющих межотраслевое значение

Интерфейс – граница между двумя функциональными объектами, требования к которой определяются стандартом, а также совокупность средств, методов и правил взаимодействия (управления, контроля и т.д.) между элементами системы

ИСР – (интегрированная среда разработки) комплекс программных средств, используемый программистами для разработки программного обеспечения (ПО)

Контент – содержание

ОС – комплекс взаимосвязанных программ, предназначенных для управления ресурсами компьютера и организации взаимодействия с пользователем

ПО – программа или множество программ, используемых для управления компьютером

Рендеринг – (англ.Rendering – визуализация, отрисовка) – термин в компьютерной графике, обозначающий процесс получения изображения по модели с помощью компьютерной программы

Штрих. код – (Штриховой код) считываемая машиной оптическая метка, содержащая информацию об объекте, к которому она привязана

ВВЕДЕНИЕ

С появлением средств и ПО для шифрования текстовой информации в таких носителях, как штрих. коды и QR-коды, стало возможным хранить довольно большие объёмы информации в небольшом изображении.

Также существование технологии дополненной реальности позволяет получать дополнительную информацию через визуальные образы, что облегчает жизнь человека.

Во многих ресторанах и кафе в меню блюд указывается только информация о названии блюда, его цене и размере порции. Зачастую этой информации недостаточно, например, когда клиент хочет знать состав блюда, если у него есть аллергия, калорийность, если он на диете или ему хочется знать, какой рейтинг этому блюду поставили другие посетители. Вся эта информация, в основном, записана в каталоге, на сайте ресторана, где каждому блюду могут поставить оценку, но вся это информация не указывается в меню, в угоду удобочитаемости без большого количества информации и, потому что отзывы блюд – это динамический показатель, который не имеет смысла печатать в меню. Также искать всю эту информацию на сайте ресторана может быть утомительно и неудобно, а при отсутствии доступа к интернету вообще невозможно.

Для решения всех этих проблем, мы собираемся создать Android приложение, которое бы считывало всю важную и подробную информацию о блюдах и напитках, сканируя QR-коды, напечатанные рядом с их изображением в меню. Кроме того, при подключении к интернету, эти QR-коды будут хранить дополнительно ссылку на это блюдо в каталоге на сайте ресторана, где будет актуальная оценка о блюде и которую сможет поставить сам пользователь.

Данное приложение будет актуально для сетей ресторанов, кафе и закусочных и т.д. Внедрение этой технологии увеличит их привлекательность, за инновационность и удобство поиска актуальной информации о еде,

которую они предлагают. Также это приложение заинтересует особо избирательных к еде клиентов, как аллергиков и людей на диете.

Цели создания системы:

– облегчить пользователю выбор блюда для заказа;

– сделать кафе и рестораны привлекательнее за счёт внедрения новой технологии.

В результате создания Приложения должны быть улучшены значения следующих показателей:

– повышение осведомлённости клиентов о составе и оценке блюда другими клиентами;

– повышение инновационности и привлекательности ресторанов и кафе.

Задачи в ходе создания системы:

– провести анализ существующих программных продуктов, используемых для оценки блюд;

– провести анализ современных программных технологий и программных решений для решения задачи определения QR-кодов в реальном времени;

– провести анализ современных программных технологий и программных решений для решения задачи отображения плоских AR объектов;

– разработать архитектуру приложения вплоть до модулей и их функций в приложении;

– создать прототип ПО для оценки блюд по QR-коду с технологией AR и его модулей;

– оценить работоспособность разработанного программного комплекса и его модулей, для решения практических задач, с помощью тестирования;

– отладить программный, в случае обнаружения ошибок.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Обзор аналогов

Системы отрисовки AR объектов на месте QR-кодов, как на фидуциарных маркерах, при параллельной обработки информации, закодированной в ней, уже существуют. Однако для целей удобного нахождения информации в меню кафе и ресторанов, при помощи AR ещё не создано. Из приложений использующих связку из сканера QR-кодов и дополненной реальности можно назвать только ARTAR, Aircards, и AR Code.

Из приложений, которые только отрисовывают новое 2D изображение поверх старого, как на маркере можно назвать ScanAR, NAVIDUU, ARVIS, при этом онлайн работает только NAVIDUU. Эти приложения работают только с изображениями, записанными в базу данных приложения и накладывают поверх них, либо потоковое видео по ссылке заранее записанное в базе данных приложения, а не сервера, как NAVIDUU, либо видео, заранее записанное в базу данных приложения, как в оставшихся двух.

Нам же нужно приложение, которое считывает ссылку, зашифрованную в QR-коде, делает запрос по этой ссылке для получения данных о блюде, и отображает эти данные графически поверх QR-кода в режиме реального времени. Полностью такому функционалу реализуют только ARTAR, Vodafone Direct Mail, Aircards, и AR Code, которые используется для создания рекламных AR-визиток и других целей. Эти приложения работают с любыми QR-кодами от разных компаний и отображает 2D изображение, если сервера, по запросу зашифрованной ссылки, предоставляют информацию в виде, доступном для представления в графическом виде.

1.2 Анализ основных технических решений

Приложения ARTAR, Vodafone Direct Mail, Aircards, и AR Code работают на технологии Web AR. WebAR – это веб-технология дополненной реальности, которые позволяют нам испытать AR через свой смартфон без какого-либо дополнительного приложения. Все, что нужно, это настраиваемый URL-адрес, который открывается в браузере смартфона, запрашивает разрешение камеры и доставляет контент AR в реальной среде пользователя. WebAR позволяет пользователям смартфонов открывать для себя технологию AR самым простым способом – через Интернет – без необходимости установки какого-либо приложения. Такая технология преодолевает барьеры и переносит интерактивные 3D-модели в реальный мир, к которым можно получить доступ через QR-код или ссылку и делает технологию универсальной для всех версий платформ.

На данный момент WebAR предлагает ограниченную функциональность по сравнению с AR-приложениями, но уже включает в себя простую 3D-анимацию, видео и определенную степень интерактивности. Кроме того, WebAR поддерживает обнаружение цели-изображения (метки), что является основой механики для демонстрации контента. Эту технологию можно использовать на всех устройствах Android под управлением Android 6.0 и выше с гироскопом и акселерометром. Таким образом, почти 75% устройств Android могут поддерживать WebAR.

Однако у этой технологии есть и свои недостатки. Для нормальной работы требуется приличная скорость Интернета. Кроме того, из-за интенсивной работы с 3D контентом и анимацией кэширование браузера не может использоваться должным образом, что иногда приводит к замедлению работы пользовательского устройства, а также к смещению моделей.

В нашем приложении предусматривается офлайн режим, поэтому мы откажемся от технологии Web AR.

Среди библиотек для ОС Android, позволяющих сканировать QR-коды с камеры смартфона, можно назвать, zxing, OpenCV, Mobile Vision API, и MLKit. Последние две являются разработкой Google, и наиболее тесно интегрированы в Android систему, чем zxing, алгоритмы их работы отработаны и отлажены, а документация, ошибки и пояснения к работе с ними написаны наиболее подробно. Однако создание приложения будет вестись на платформе Unity и языке C#, так как Unity имеет наибольший инструментарий для разработки приложений дополненной реальности и активно поддерживается, как разработчиками, так и сообществом, поэтому библиотекой для обнаружения, захвата и чтения QR-кодов будет OpenCV, так как она единственная написана на языке C# [1]. Кроме того, к этой платформе написано намного больше обучающей литературы и теории. Из недостатков можно назвать небольшую громоздкость приложений из-за большого количества тяжёлых встраиваемых библиотек, но это не существенно, в сравнении с обширным функционалом, предоставляемым этой платформой.

Для создания самих 3D AR объектов мы будем использовать платформу Vuforia вместе с Unity, так как она наиболее удобна в использовании, наименее требовательна к системным требованиям приложения и имеет свои сервисы [2]. Также с помощью сервисов Vuforia можно легко создавать базы данных AR-меток и прикреплённых к ним AR-объектов.

Для решения задачи реализации системы был проведён поиск и анализ уже существующих готовых решений. В ходе поиска готовых программных решений, был найден код на языке C++, для обнаружения, декодирования QR-кодов. Необходимо модифицировать код, чтобы программа обнаруживала и декодировала сразу несколько QR-кодов одновременно и перевести код на язык C# (рисунок 1).

```cpp
#include <opencv2/objdetect.hpp>
#include <opencv2/imgcodecs.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <iostream>

using namespace cv;
using namespace std;

void display(Mat &im, Mat &bbox)
{
  int n = bbox.rows;
  for(int i = 0 ; i < n ; i++)
  {
    line(im, Point2i(bbox.at<float>(i,0),bbox.at<float>(i,1)), Point2i(bbox.at<float>((i+1) % n,0), bbox.at<float>((i+1) % n,1)), Scalar(255,0,0), 3);
  }
  imshow("Result", im);
}

int main(int argc, char* argv[])
{
  // Read image
  Mat inputImage;
  if(argc>1)
    inputImage = imread(argv[1]);
  else
    inputImage = imread("qrcode-learnopencv.jpg");

  QRCodeDetector qrDecoder = QRCodeDetector::QRCodeDetector();

  Mat bbox, rectifiedImage;

  string data = qrDecoder.detectAndDecode(inputImage, bbox, rectifiedImage);
  if(data.length()>0)
  {
    cout << "Decoded Data : " << data << endl;

    display(inputImage, bbox);
    rectifiedImage.convertTo(rectifiedImage, CV_8UC3);
    imshow("Rectified QRCode", rectifiedImage);

    waitKey(0);
  }
  else
    cout << "QR Code not detected" << endl;
}
```

Рисунок 1 – Код QR декодера на C++

Для многопоточного обнаружения и декодирования QR-кодов необходимо модифицировать программный код встроенными в библиотеку OpenCV функцией detectAndDecodeMulty() с соответствующими параметрами (рисунок 2).

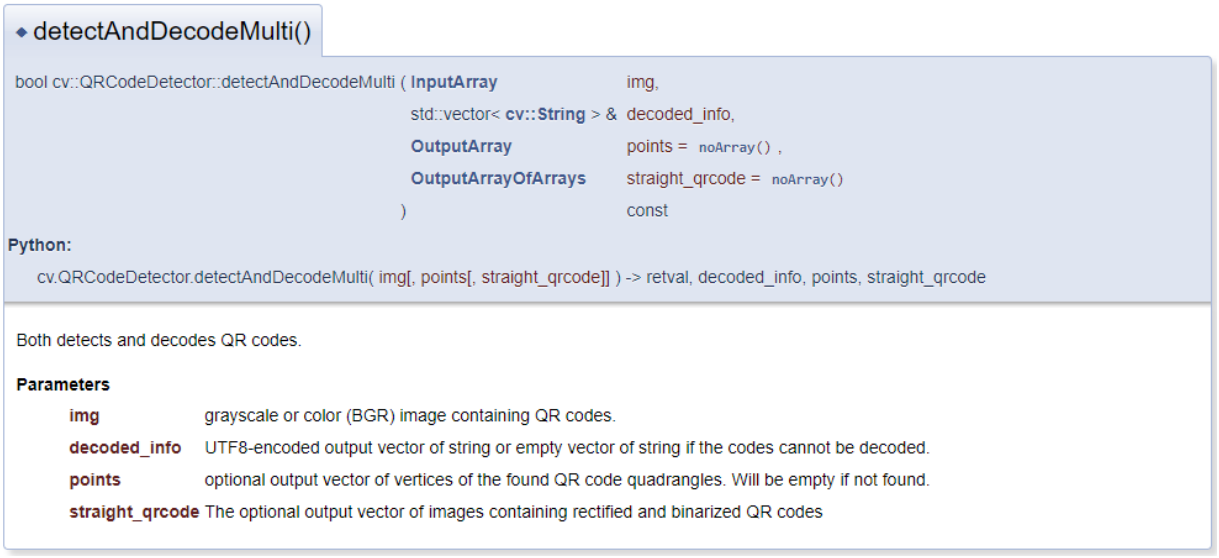

Рисунок 2 – Пояснение к сигнатуре и параметрам функции detectAndDecodeMulty()

Входной параметр img будет содержать зафиксированное изображение с AR Camera, ссылочный тип decoded_info будет содержать декодированный с QR-кода текст, straight_qrcode будет содержать пиксельные изображения QR-кодов, на которые Vuforia будет обнаруживать и накладывать AR объекты.

1.3 Вывод

На основе проведённого сравнительного анализа технологических решений для разработки приложения дополненной реальности для сканирования QR-кодов были выбраны наиболее подходящие элементы.

Было решено, что для создания программы будет использоваться среда Unity, так как она имеет широкий функционал, а её поддержка широко распространена. Также принято решение вести разработку на языке C#, так как именно он поддерживается средой Unity.

Для поддержки технологии дополненной реальности решено использовать платформу Vuforia, так как она наименее требовательна к системным требованиям устройства, по сравнению с другими аналогами, а также активно поддерживается её сообществом.

Для создания системы обнаружения и расшифровывания QR-кодов было решено использовать библиотеку OpenCVForUnity, так как она обладает широким функционалом, бесплатна, адаптирована для среды разработки Unity и языка C#.

Для решения задачи тестирования приложения мы сгенерируем изображение QR-кода с зашифрованным текстом, с перечисленными ингредиентами блинов (Pancakes) и ссылкой на сайт с оценкой блюда, и отобразим это изображение на телефон.

По результатам тестирования, при попадании QR-кода в поле зрения камеры устройства, на котором запущено приложение, на месте QR-кода должна появиться виртуальная табличка с информацией об ингредиентах блинов, расшифрованной из кода, а также актуальной оценкой блюда. Отладка будет вестись в IDE Unity Editor и Visual Studio 2022.

2 ОПРЕДЕЛЕНИЕ ТРЕБОВАНИЙ

Для реализации данной системы необходим следующий набор подсистем:

– модуль, фиксирующий визуальные данные с камеры;

– модуль обнаружения и чтения QR-кодов;

– модуль со встроенным браузером для отправки запросов по ссылке;

– модуль отрисовки AR информации поверх маркеров (QR-кодов);

– модуль пользовательского интерфейса для управления приложением.

Приложение должно обеспечивать следующие возможности:

– пользователю приложение – найти интересующую его информацию о блюде, чтобы решить, что ему заказать;

– пользователю приложение – получить информацию об оценке блюда другими пользователями без траты времени на посещение сайта ресторана и поиска блюда в его каталоге;

– пользователю приложение – возможность самостоятельно поставить оценку купленному и опробованному блюду без траты времени на посещение сайта ресторана и поиска блюда в его каталоге, чтобы поставить оценку;

– заказчику приложения – повысить иновационность своего ресторана, его привлекательность;

– заказчику приложения – не писать доп. Информацию о блюде на бумаге в меню тем самым экономя её.

При разработке приложения и создании документации должны быть использованы следующие нормативно-технические документы и должна соответствовать следующим стандартам:

1) ГОСТ 34.601-90. Сравнительные испытания мобильных приложений для смартфонов. Стадии создания;

2) ПНСТ 277-2018. Сравнительные испытания мобильных приложений для смартфонов;

3) ГОСТ 28195-89. Оценка качества программных средств. Общие положения;

4) ГОСТ Р ИСО/МЭК 9126-93. Информационная технология. Оценка программной продукции. Характеристики качества и руководства по их применению;

5) ГОСТ Р ИСО/МЭК 10746-1-2004. Информационная технология. Открытая распределенная обработка. Базовая модель. Часть 1. Основные положения;

6) ГОСТ Р ИСО/МЭК 10746-4-2004. Информационная технология. Открытая распределенная обработка. Базовая модель. Часть 4. Архитектурная семантика;

7) ГОСТ Р ИСО/МЭК 15910-2002. Информационная технология. Процесс создания документации пользователя программного средства;

8) ГОСТ Р ИСО/МЭК 12119-2000. Информационная технология. Пакеты программ. Требования к качеству и тестирование;

9) ГОСТ 34.601-90. Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Стадии создания;

10) ГОСТ 34.602-89. Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы;

11) ГОСТ 34.603-92. Информационная технология. Виды испытаний автоматизированных систем;

12) ГОСТ 19. Единая система программной документации;

13) РД 50-34.698-90. Методические указания. Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Требования к содержанию документов.

Приложение должно соответствовать следующим атрибутам качества:

– время запуска приложения и загрузки главного экрана не должно превышать 30 секунд.

Приложение должно реализовывать следующие функциональные требования:

– приложение должно уметь показывать оценку и информацию о калорийности и ингредиентах каждого блюда в меню в режиме дополненной реальности, при попадании QR-кода в поле зрения камеры, на расстоянии не более 0,5 метра от камеры;

– приложение должно давать возможность пользователю поставить свою оценку каждому блюду в листе покупок в чеке, при сканировании QR-кода чека и отправлять эти оценки на сайт ресторана, по ссылкам, указанным в этом же QR-коде для каждой позиции.

Приложение должно соответствовать следующим нефункциональным требованиям:

– приложение должно быть написано на языке C#;

– приложение должно быть разработана на ОС Android 8 и выше;

– приложение должно быть разработано на платформе Unity.

Приложение должно соответствовать следующим системным требованиям:

– все вычисления, кроме обработки запроса на размещение оценки и её получение обрабатываются на самом устройстве, на которое установлено приложение;

– устройство должно предоставлять приложению доступ к своей камере;

– при отсутствии соединения с сетью Интернет, приложение работает в режиме офлайн и отображает только информацию зашифрованную в QR-коде, а именно калории и ингредиенты;

– при наличии соединения с сетью Интернет, приложение работает в режиме онлайн и, помимо статической информации в офлайн режиме, отображает динамическую информацию, а именно оценку актуальную блюда другими пользователями приложения и посетителями сайта ресторана.

Приложение должно соответствовать следующим системным требованиям:

– в приложении должна быть реализована возможность менять режимы отображения информации о блюдах: только калории, только оценка, только ингредиенты.

2.1 Функциональные требования

На рисунке 3 изображена диаграмма вариантов использования системы.

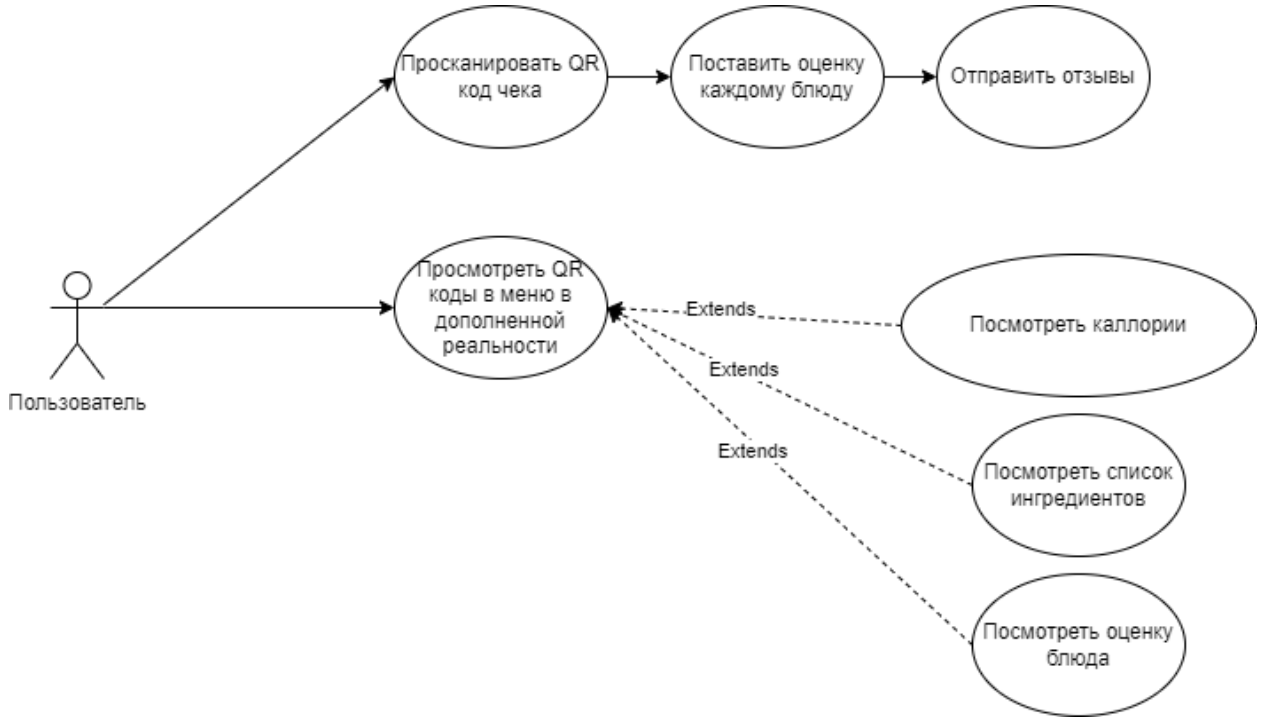

Рисунок 3 – Диаграмма вариантов использования системы

Пользователь: целевая аудитория приложения, чью потребность в поиске дополнительной информации и оценке блюда удовлетворяет приложение.

У пользователя будет возможность воспользоваться двумя экранами приложения: экраном со сканером чека и экраном со множественным сканнером, добавляющим 3D AR объекты поверх QR-кодов, после их сканирования в реальном времени [3, 4].

При нахождении в экране с AR множественным сканнером у пользователя будет возможность выбрать какая именно информация будет отображаться с помощью кнопок изменения режима:

– отображение калорий;

– отображение списка ингредиентов;

– отображение оценки блюда.

Оценка блюда является динамическим показателем, поэтому она должна получаться после HTTP запроса по ссылке зашифрованной в сканируемом QR-коде, который, как правило, должна ссылаться на каталог с этим блюдом на сайте ресторана, который напечатал меню.

Экран со сканером чека должен просто отсканировать один единственный QR-код в чеке без использования AR. В QR-коде чека должны быть зашифрованы называния оплаченных блюд и ссылки на них в каталоге на сайте ресторана, чтобы поставить им оценку. При успешном сканировании приложение должно автоматически переключается на третий экран, где отображается список из купленных блюд, каждому их которых можно добавить оценку от 1 до 10 и отправить эти оценки при нажатии кнопки “Отправить отзывы”.

2.2 Нефункциональные требования

Приложение должно быть написано на языке C# на платформе Unity и должно запускаться на ОС Android 8, поэтому для работы модуля отрисовки AR должная использоваться платформа Vuforia, установленная плагином в среду Unity, через менеджера пакетов, так как только эта платформа обеспечивает наибольшую поддержку AR версиями ОС Android и позволяет запускать приложение на версии Android 8 Nougat [5].

Для обнаружения и декодирования QR-кодов должна использоваться библиотека OpenCVForUnity, являющийся библиотекой OpenCV,

адаптированной для платформы Unity и языка C#, и устанавливающуюся в среду Unity в виде плагина, через менеджера пакетов.

Для разработки AR приложения, также необходимо выбрать в настройках приложения поддержку графического интерфейса OpenGLES2 вместо Vulkan, так как последний не поддерживает рендеринг 3D AR объектов [6, 7, 8].

3 ПРОЕКТИРОВАНИЕ

Программный продукт должен включать в себя следующие компоненты:

– модуль, фиксирующий визуальные данные с камеры;

– модуль обнаружения и чтения QR-кодов;

– модуль со встроенным браузером для отправки запросов по ссылке;

– модуль отрисовки AR информации поверх маркеров (QR-кодов);

– модуль пользовательского интерфейса для управления приложением.

Для проектирования архитектуры приложения была составлена модель взаимодействия компонентов системы в IDEF0 нотации. Модель приложения представленная ниже показывает взаимодействие модулей друг с другом. На первом рисунке изображено представление нулевого уровня, т.е. само приложения, как объекта, на втором – первого уровня, показывающего взаимодействие модулей и компонентов (рисунки 4, 5).

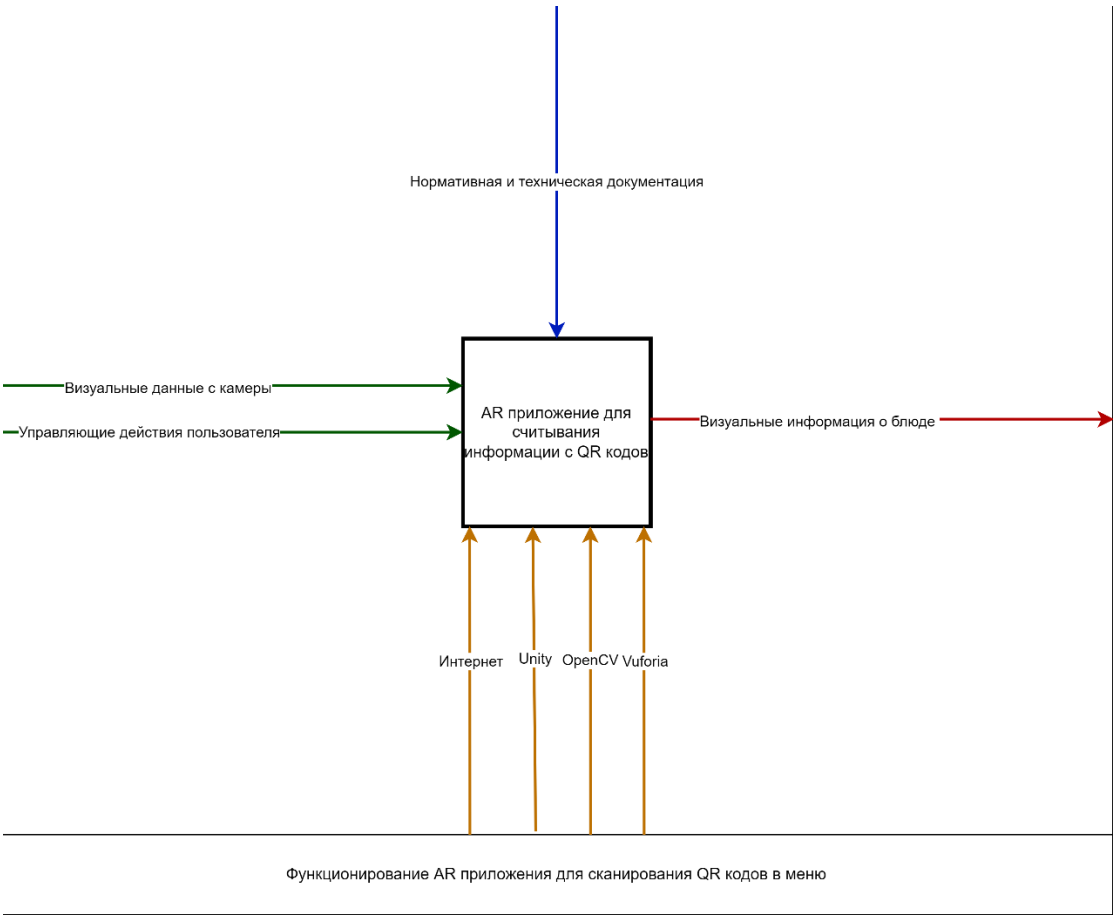

Рисунок 4 – Представление приложения 0 уровня

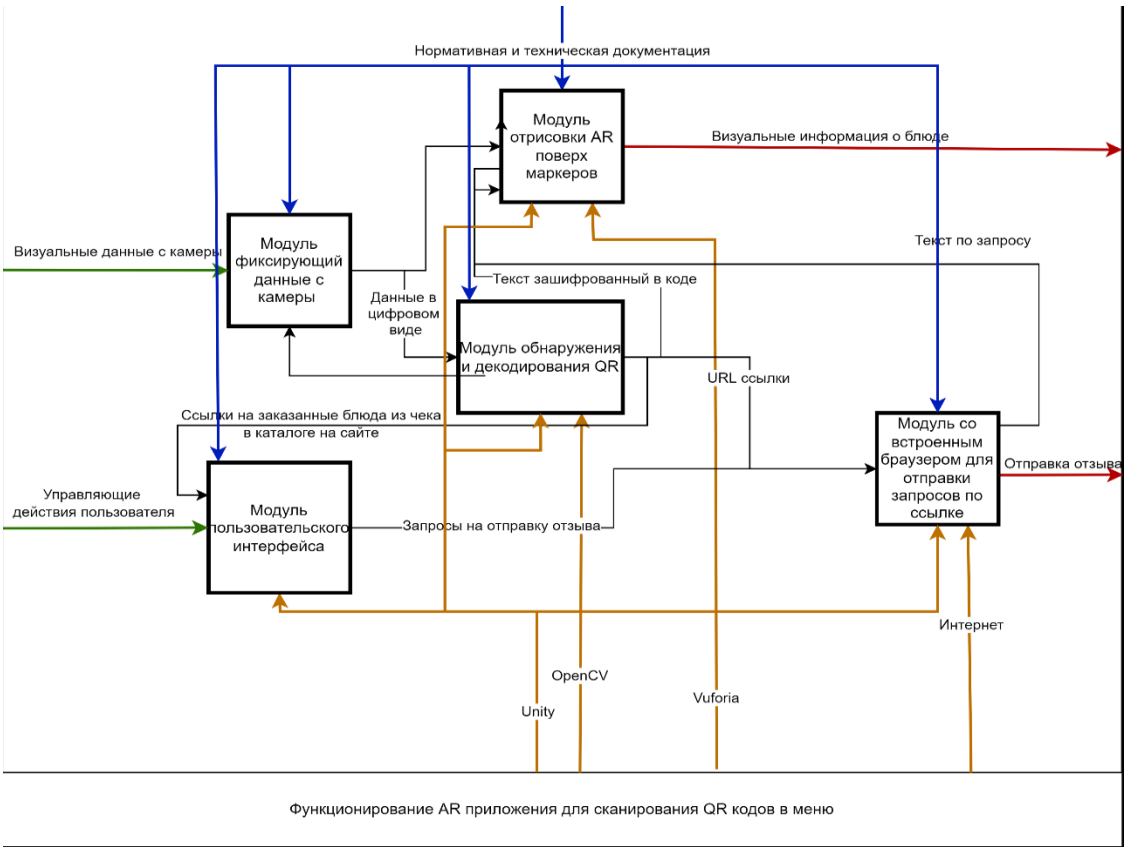

Рисунок 5 – Компоненты системы приложения и их взаимодействие в представлении 1 уровня

Также в ходе моделирования и проектирования системы была сформирована UML диаграммы последовательности для нашего приложения (рисунок 6).

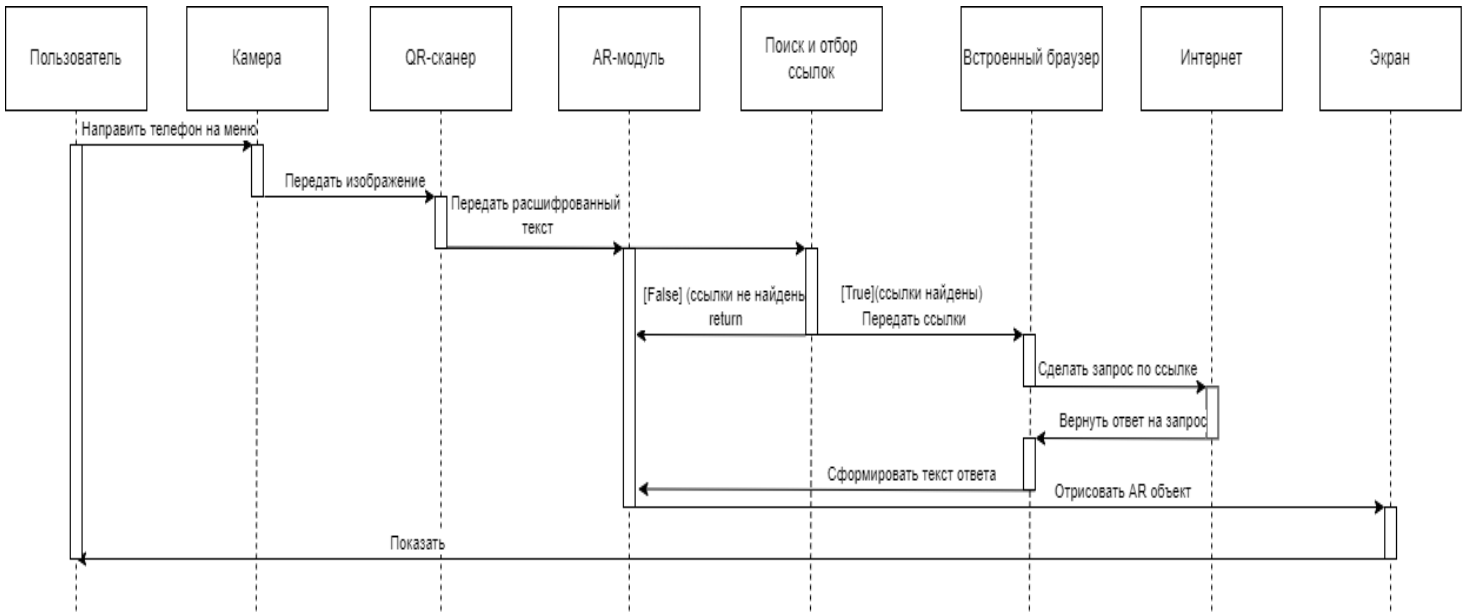

Рисунок 6 – UML диаграмма последовательности

4 РЕАЛИЗАЦИЯ

Для реализации приложения, нами был создан 3D проект на Unity и несколько сцен внутри проекта:

1) YummyBookSaved;

2) SetScoresMenu;

3) ReciptScanner.

4.1 Сцена YmmyBookSaved

В сцене YmmyBookSaved реализуется главный экран приложения. Он отвечает за отрисовку объектов дополненной реальности, а также за обнаружение, сканирование и чтение множества QR-кодов одновременно в реальном времени, с переключением режимов отображения информации о блюдах.

Роль камеры для отрисовки AR объектов выполняет камера ARCameraYummyBookSaved из библиотеки Vuforia. Объект CreatorItems, контролирует генерацию и инициализацию объектов класса ImageTarget и объектов Item вложенных в них. Для выполнения этой операции в CreatorItems встроен скрипт CreatingItems.cs [9].

Объекты Item, содержат в себе элементы Plane и Text (TMP), в совокупности, формирующие таблички с информацией о блюде. Под инициализацией объекта Item подразумевается инициализация текста во вложенном элементе Text (TMP). Сам объект класса ImageTarget от Vuforia, хранит изображение маркера, который система будет искать в поле зрения камеры устройства, и определяет как именно вложенный в него 3D объект будет отображаться на месте маркера.

Метод AddItem из скрипта CreatingItems.cs, вызывается извне – из скрипта QRCodeScanner.cs объекта QRCodeScanner. Листинг кода скрипта CreatingItems.cs. представлен в приложении А.

На рисунке 7 изображена модель ImageTarget.

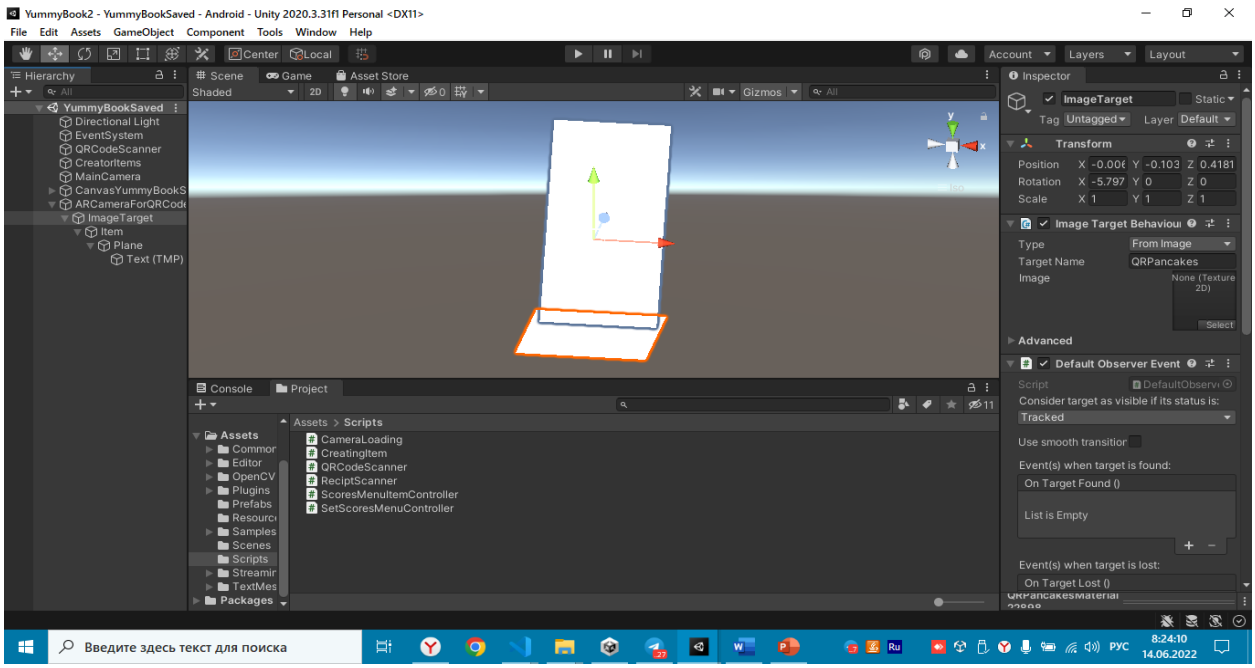

Рисунок 7 – Модель ImageTarget

Игровой объект QRCodeReader со встроенным скриптом QRCodeScanner.cs. отвечает за сканирование, обнаружение и расшифровку QR-кодов с изображения с главной камеры MainCamera. Для сканирования используется метод detectAndDecodeMulti() класса QRCodeDetector из библиотеки OpenCVForUnity.

Для реализации экрана интерфейса пользователя, реагирующего на взаимодействие с ним, в иерархии сцены создан объект CanvasYummyBookSaved класса Canvas. На рисунке 8 изображена модель экрана интерфейса CanvasYummyBookSaved и иерархия вложенных элементов сцены YummyBookSaved.

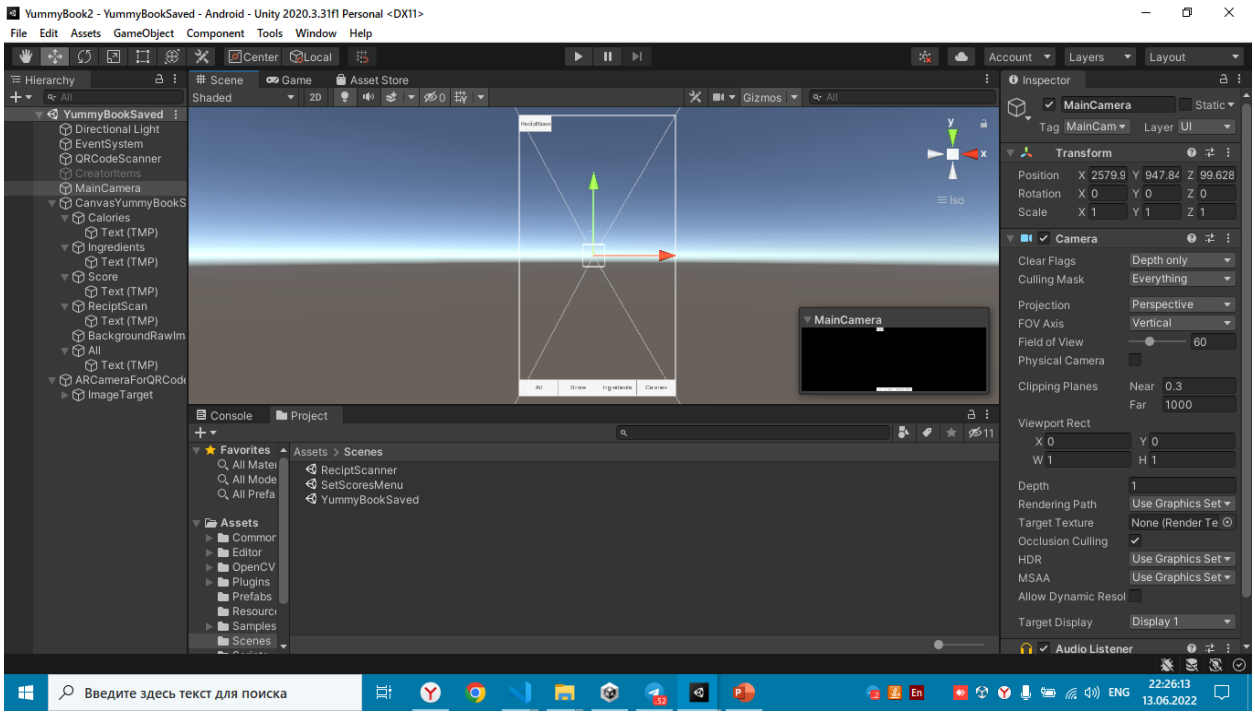

Рисунок 8 – Модель пользовательского интерфейса и иерархия вложенных элементов сцены YummyBookSaved

В объект CanvasYummyBookSaved вложен объект игрового интерфейса RawImage, служащий контейнером для плоских изображений на сцене [10, 11]. Он назван BackgroundRawImage и отображает (рендеринг), как изображение реального мира с камеры MainCamera, так и 3D объектов из дополненной реальности, которые обрабатывает камера ARCameraYummyBookSaved.

В объект CanvasYummyBookSaved также вложены элементы пользовательского интерфейса (UI):

– кнопка “Calories”, для включения режима отображения калорий блюд в табличках;

– кнопка “Ingridients”, для включения режима отображения ингредиентов блюд в табличках;

– кнопка “Score”, для включения режима отображения оценки блюд пользователями в табличках;

– кнопка “All”, для включения режима отображения всей информации о блюде в табличках;

– кнопка “ReciptScan”, для переключения приложения в режим сканирования чека, т.е. загрузки сцены ReciptScanner.

В каждый из этих кнопок встроен объект QRCodeReader со скриптом QRCodeScanner.cs активирующийся при совершении события On Click(), т.е. нажатии. Так, при нажатии кнопки Ingredients, запускается метод OnClickIngridients() из скрипта QRCodeScanner.cs, при нажатии Calories – OnClickCalories(), при нажатии Score – OnClickScore() и т.д. Эти методы инициализируют флаги, которые сообщают, какую информацию из расшифрованных QR-кодов записывать в таблички внутри объектов Item, вложенных в ImageTarget.

За кнопкой ReciptScann закреплён метод OnClickReciptScanner(), который загружает и переключается на сцену ReciptScanner, при событии совершении события OnClick, т.е. нажатии. Листинг кода скрипта QRCodeScanner.cs представлен в приложении Б.

4.2 Сцена ReciptScanner

В сцене ReciptScanner реализуется возможность сканирования QR-кода чека уже оплаченных блюд, для их последующей оценки. За процесс сканирования QR-кода в поле зрения камеры MainCamera отвечает игровой объект ReciptScanner с встроенным скриптом ReciptScanner.cs. Данный скрипт имеет поле для встраивания объекта RawImage в области которого и будет сканироваться QR-код, при попадании в неё. В данном случае объект RawImage служит прицелом для сканирования и обозначен розовым квадратом. Для сканирования используется метод detectAndDecode() класса QRCodeDetector из библиотеки OpenCVForUnity.

Также внутри сцены создан элемент пользовательского интерфейса CanvasForReciptScanner класса Canvas, обеспечивающего взаимодействие пользователя с приложением в режиме сканирования чека. Внутри

CanvasForReciptScanner создан вложенный элемент пользовательского интерфейса Scanning – кнопка с текстом Text (TMP) с надписью “Scan”. В этот элемент встроен игровой объект ReciptScanner и при нажатии на неё активируется метод OnClickScan(), который расшифровывает QR-код в области поля RawImage, извлекает данные из полученного сообщения, загружает сцену SetScoresMenu и передаёт туда данные, как параметры, для формирования списка оплаченных блюд на оценку. Листинг кода скрипта ReciptScanner.cs представлен в приложении В.

На рисунке 9 изображена модель пользовательского интерфейса и иерархия вложенных элементов сцены ReciptScanner.

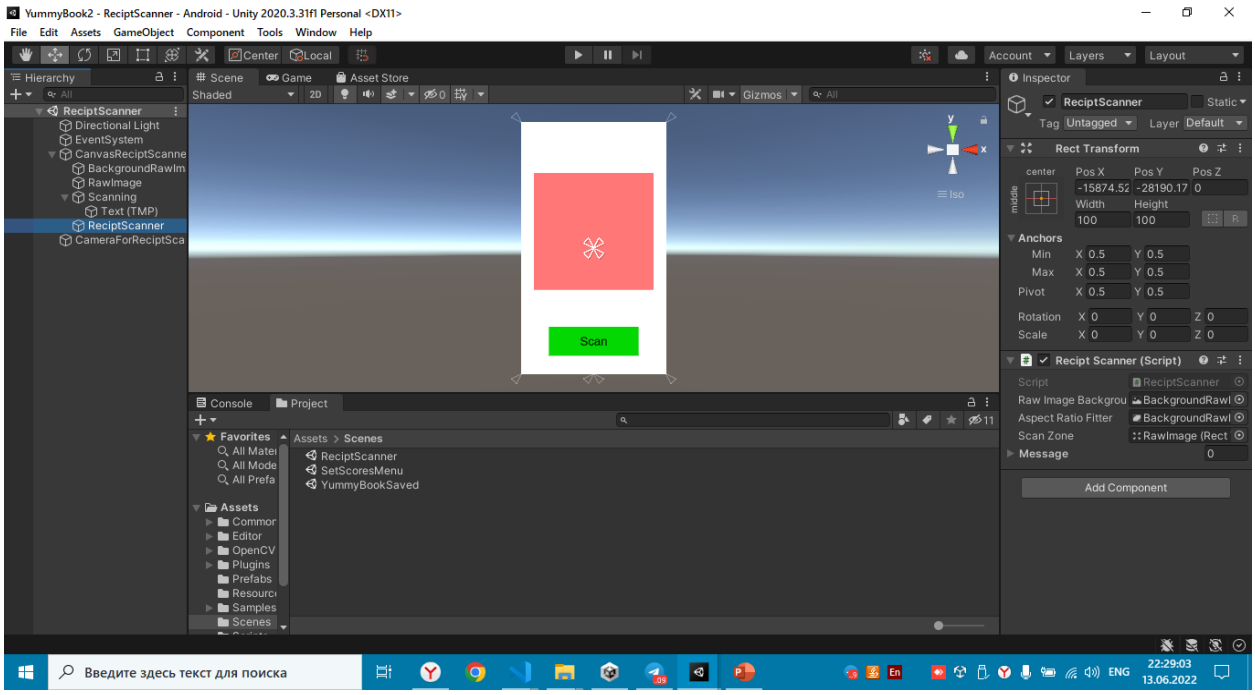

Рисунок 9 – Модель пользовательского интерфейса и иерархия вложенных элементов сцены ReciptScanner

4.3 Сцена SetScoresMenu

В сцене SetScoresMenu реализуется возможность пользователя устанавливать свои оценки каждому оплаченному блюду. Информация оплаченных блюд получается при сканировании QR-кода чека в сцене ReciptScanner и передается в SetScoresMenu, после чего она обрабатывается

игровым объектом SetScoresMenuConttroller и встроенным в него скриптом SetScoresMenuConttroller.cs.

В иерархии создан объект пользовательского интерфейса CanvasSetScoresMenu класса Canvas, реализующий управление экраном пользователем. В Canvas вложены элементы Button (кнопка) и ScrollArea (зона прокрутки содержимого), а внутри ScrollArea игровой объект Content (содержимое), внутрь которого вложена группа игровых объектов Item.

Элемент Item реализует в себе само оплаченное блюдо – в нём отображается имя оплаченного блюда, через UI элемент Text (TMP) и дана возможность поставить блюду ему оценку. Для реализации возможности задания оценки в Item также вложены два элемента типа “кнопка”: ScorePlus, прибавляющее бал и ScoreMin убавляющее бал, а также текстовое поле Scores, отображающее набранное пользователем значение. При нажатии на кнопку “Scan” запускается метод OnClickSent(), который отправляет оценку запросом в сеть интернет по ссылке и возвращает пользователя в на сцену YummyBookSaved. Листинг кода скрипта SetScoresMenuConttroller.cs представлен в приложении Г. За реализацию работы кнопок ScoreMin и ScorePlus, в объекте Item, отвечает вложенный игровой объект ItemController со встроенным в него скриптом SetScoreMenuItemController.cs. Листинг кода скрипта SetScoreMenuItemController.cs представлен в приложении Д.

На рисунке 10 изображена модель пользовательского интерфейса и иерархия вложенных элементов сцены SetScoresMenu.

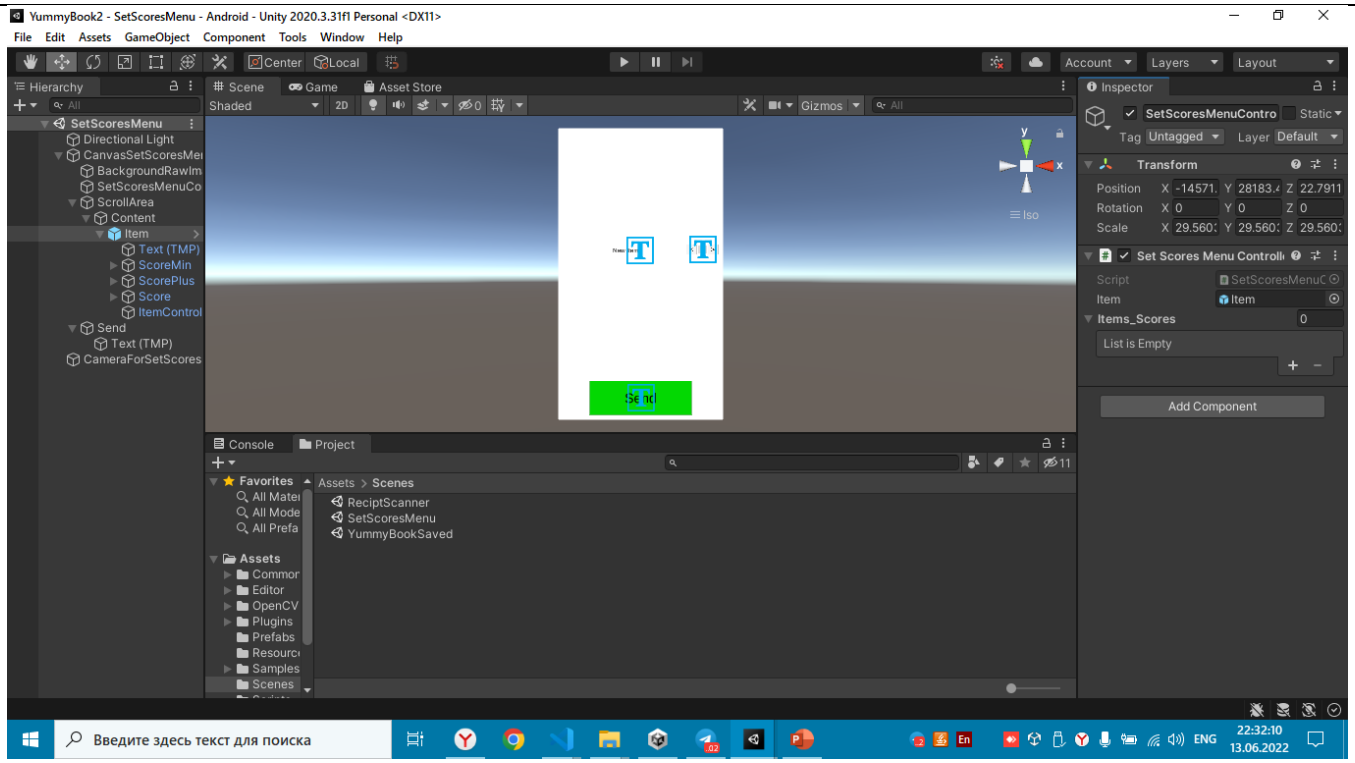

Рисунок 10 – Модель пользовательского интерфейса и иерархия вложенных элементов сцены SetScoresMenu

5 ТЕСТИРОВАНИЕ

На данный момент автором было проведено тестирование модуля обнаружения и декодирования QR-кодов, а также модуль отображения AR объектов. Тестирование показало, что модули выполняют свои функции. Модуль расшифровки успешно обнаруживает QR-коды на изображении с камеры и считывают с них информацию, а модуль отображения AR успешно отображает 3D таблички с расшифрованной информацией поверх QR-кодов, как маркеров.

На рисунке 11 изображена демонстрация результат работы модулей отображения AR и расшифровки QR-кодов.

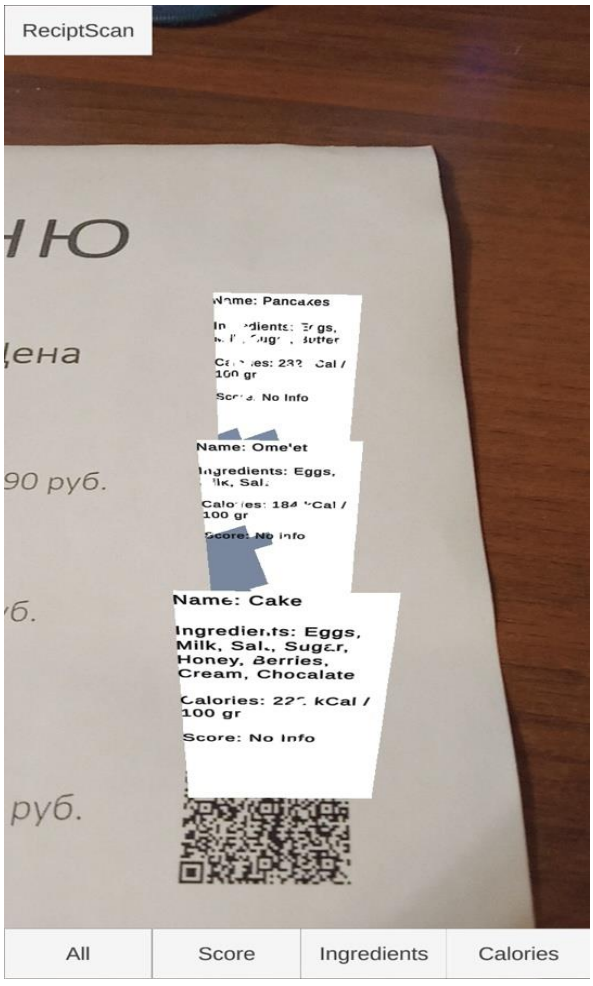

Рисунок 11 – Демонстрация работы модулей отображения AR объектов и расшифровки QR-кодов

Также было проведено тестирование функции сканирования QR-кода с чека заказа и задания оценки каждому блюду пользователем. На рисунках 12 и 13 изображены демонстрации работы этих функций в приложении.

Рисунок 12 – Демонстрация работы функции сканирования QR-кода чека

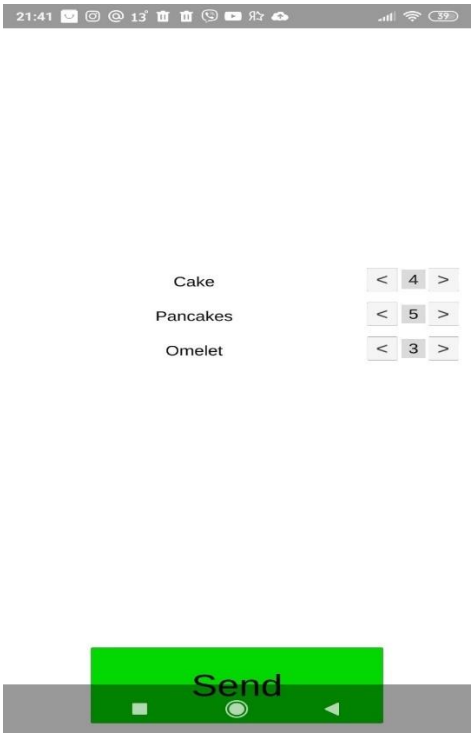

Рисунок 13 – Демонстрация работы функции задания оценки каждому блюду из чека

ЗАКЛЮЧЕНИЕ

В рамках выпускной квалификационной работы, было разработано приложение дополненной реальности на устройства с ОС Android 8 Nougat и выше, для оценки блюд по QR-коду в меню. Была разработана архитектура приложения, изучены технологии отображения объектов дополненной реальности в реальном времени, с помощью платформы Vuforia и платформы Unity, а также технологии обнаружения объектов и изображений в пространстве с помощью библиотеки OpenCV.

Получены практические знания в области разработки программного продукта для мобильных устройств, а также его тестирования и отладки.

Был проведён анализ рынка и существующих аналогов данного приложения. Было установлено, что приложений данной направленности ещё не было разработано, и что наша разработка может быть востребованной. Этот программный продукт будет востребован, прежде всего, для посетителей ресторанов, которые хотят найти дополнительную информацию о блюдах, для выбора заказа, но не хотят искать о них информацию самостоятельно на сайтах ресторанов. Также это приложение будет полезно и для самих ресторанов, так как внедрение данной технологии повысит их инновационность и позволит сэкономить место на бумаге в меню, и более того повысит привлекательность таких ресторанов для клиентов за счёт информативности.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Черников, В. Разработка мобильных приложений на C# для iOS и Android : учебное пособие / В. Черников. – Москва : ДМК Пресс, 2020. – 188 с.

2. Day1: Augmented Reality (AR) Tutorial - Marker based AR with Vuforia And Unity : [видеозапись] / Joystick Lab. – 13.03.2020. – Изображение (движущееся) : электронное // Joystick Lab : [канал на YouTube]. – URL: https://www.youtube.com/watch?v=zpiFZtPxo2w&list=PLb1h4A0yB97_TeFuf9TAEah3b-VyIYzF9 (дата обращения: 15.06.2022).

3. Гущина, О. М. Разработка AR-приложений : учебно-методическое пособие / О. М. Гущина, А. В. Очеповский. – Тольятти : ТГУ, 2021. – 57 с.

4. Системы виртуальной, дополненной и смешанной реальности: учебное пособие / А. А. Смолин, Д. Д. Жданов, И. С. Потемин [и др.]. – Санкт-Петербург : НИУ ИТМО, 2018. – 59 с.

5. Соколова, В. В. Разработка мобильных приложений : учебное пособие / В. В. Соколова. – Томск : ТПУ, 2014. – 176 с.

6. Вольф, Д. OpenGL 4. Язык шейдеров. Книга рецептов / Д. Вольф ; перевод с английского А. Н. Киселева. – Москва : ДМК Пресс, 2015. – 368 с.

7. Гинсбург, Д. OpenGL ES 3.0. Руководство разработчика : руководство / Д. Гинсбург, Б. Пурномо ; перевод с английского А. Борескова. – Москва : ДМК Пресс, 2015. – 448 с.

8. Джонатан, Л. Виртуальная реальность в Unity / Л. Джонатан ; перевод с английского Р. Н. Рагимова. – Москва : ДМК Пресс, 2016. – 316 с.

9. Торн, А. Искусство создания сценариев в Unity : руководство / А. Торн ; перевод с английского Р. Н. Рагимова. – Москва : ДМК Пресс, 2016. – 360 с.

10. Unity AR Foundation Tutorial : Make a tile e-Commerce App - Part 1 : [видеозапись] / Joystick Lab. – 13.03.2020. – Изображение (движущееся) : электронное // Joystick Lab : [канал на YouTube]. – URL:

https://www.youtube.com/watch?v=WZgOYxRanPk&list=PLb1h4A0yB978SQuAeBsxup--7ITPCashH (дата обращения: 15.06.2022).

11. Getting Started With ARFoundation in Unity (ARKit, ARCore) : [видеозапись] / Joystick Lab. – 13.03.2020. – Изображение (движущееся) : электронное // Joystick Lab : [канал на YouTube]. – URL: https://www.youtube.com/watch?v=Ml2UakwRxjk&t=910s (дата обращения: 15.06.2022).

ПРИЛОЖЕНИЕ А
Исходный код скрипта CreatingItems.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using OpenCVForUnity;
using OpenCVForUnity.CoreModule;
using OpenCVForUnity.UnityUtils;
using UnityEngine;
using UnityEngine.UI;
using System.Text.RegularExpressions;
using Vuforia;
using TMPro;

public class CreatingItem : MonoBehaviour
{
    public static CreatingItem instance{get; private set;}
    public static Texture2D textureFile;
    public static float printedTargetSize;
    public static string targetName;
    public static string targetInfo;
    [SerializeField]
    public List<ImageTargetBehaviour> ListmTarget;

    void Awake()
    {
        if(instance==null){
            instance = this;
            DontDestroyOnLoad(this.gameObject);
            return;
        }else
            Destroy(this.gameObject);
    }
    void Start()
    {
       VuforiaApplication.Instance.OnVuforiaStarted +=
CreateImageTargetFromSideloadedTexture;
    }
    void CreateImageTargetFromSideloadedTexture()
    {
        Texture2D NewtextureFile=textureFile;
```

```
        float NewprintedTargetSize=printedTargetSize;
        string NewtargetName=targetName;

        ListmTarget.Add(VuforiaBehaviour.Instance.ObserverFactory.CreateImageTarge
t(NewtextureFile, NewprintedTargetSize, NewtargetName));
        (ListmTarget[ListmTarget.Count-
1]).gameObject.AddComponent<DefaultObserverEventHandler>();
(ListmTarget[ListmTarget.Count-
1]).gameObject.transform.Find("Item").gameObject.transform.Find("Plane").GetCompon
ent<TextMeshPro>().text=targetInfo;
        Instantiate((ListmTarget[ListmTarget.Count-1]));
    }

    public void AddItem(QRCodeScanner.Item unit){
        Mat image=unit.Img;
        Utils.matToTexture2D(image, textureFile, false);
        targetName=unit.GetName();
        targetInfo=unit.GetInfo();
        CreateImageTargetFromSideloadedTexture();
    }
}
```

ПРИЛОЖЕНИЕ Б
Исходный код скрипта QRCodeScanner.cs

```
using UnityEditorInternal;
using System;
using System.Collections;
using System.Collections.Generic;
using System.Net.Http;
using OpenCVForUnity;
using OpenCVForUnity.CoreModule;
using OpenCVForUnity.UnityUtils;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;
using System.Text.RegularExpressions;
using TMPro;
public class QRCodeScanner : MonoBehaviour
{

public static QRCodeScanner instance{get; private set;}
    void Awake()
    {
        if(instance==null){
            instance = this;
            return;
        }else if(instance==this){
            Destroy(this.gameObject);
        }
        DontDestroyOnLoad(this.gameObject);
    }

public class Item{
    public static int maxScore=5;
    public static int minScore=0;
    public Item(string qr_info, Mat img){
        string unit="";
        Regex regex = new Regex(@"Name: ((\,)|(\s)|(\w*))*;");
        MatchCollection matches = regex.Matches(qr_info);
        if(qr_info!=""){
            unit=matches[0].ToString();
            regex = new Regex(@"Name: |;");
```

```
        Name=regex.Replace(unit, "");

        unit="";
        regex = new Regex(@"Ingridients: ((\,)|(\s)|(\w*))*;");
        matches = regex.Matches(qr_info);
        unit=matches[0].ToString();
        regex = new Regex(@"Ingridients: |;");
        Ingridients=regex.Replace(unit, "");

        unit="";
        regex = new Regex(@"Calories: ((\,)|(\.)|(\s)|(\w*))*;");
        matches = regex.Matches(qr_info);
        unit=matches[0].ToString();
        regex = new Regex(@"Calories: |;");
        Ingridients=regex.Replace(unit, "");
        Img=img;
    }else{
        Name="";
        Ingridients="";
        Callories="";
    }
}
public Mat Img{get;}
private string Name;
private string Callories;
private string Ingridients;
private int Score;

public string GetName(){
    return this.Name;
}
public string GetCallories(){
    return this.Callories;
}
public string GetIngridients(){
    return this.Ingridients;
}
public string GetScore(){
    if(Score<=maxScore&&Score>=minScore)
        return (this.Score).ToString();
```

```
 else
        return "No info";
}
public void SetCallories(string input){
    if(input!=""||input!=null){
        Callories=input;
    }else
        Callories="No info";
}
public void SetIngridients(string input){
    if(input!=""||input!=null){
        Ingridients=input;
    }else
        Ingridients="No info";

}
public void SetScore(int input){
    if(input<=maxScore&&input>=minScore){
        Score=input;
    }else
        Score=-1;
}
public void SetName(string input){
    if(input!=""||input!=null){
        Name=input;
    }else
        Name="";
}
public static int GetMinScoreValue(){
    int val=minScore;
    return val;
}
public static int GetMaxScoreValue(){
    int val=maxScore;
    return val;
}
public string GetInfo(){
    string result="";
    result+="Name: "+GetName()+"\n";
    if(pressedIngridients){
```

```
             result+="\nIngridients: "+(GetIngridients()+"\n");
        }
        if(pressedCallories){
            result+= "\nCalories: "+(GetCallories()+"\n");
        }
        if(pressedScore){
            result+="\nScore: "+(GetScore()+"\n");
        }
        return result;
    }
};
    [SerializeField]
    private RawImage _rawImageBackground;
    [SerializeField]
    private AspectRatioFitter _aspectRatioFitter;
    Mat img, points;
    int a;
    [SerializeField]
    private RectTransform _scanZone;
    private bool _isCamAvailable;
    public static bool pressedScore=true, pressedIngridients=false,
pressedCallories=false;
    private WebCamTexture _cameraTexture;
    public static string messHttp;
    private WebCamDevice[] devices;
    OpenCVForUnity.ObjdetectModule.QRCodeDetector decoder;
    List<string> message=new List<string>();
    public List<Item> items;
    List<Mat> imges_out;
    string res;
    void Start()
    {
        SetUpCamera();
    }
    void Update()
    {
        UpdateCameraRender();
        Scan();
    }
    private void UpdateCameraRender(){
```

```
        if(_isCamAvailable==false){
            return;
        }
        float ratio = (float)_cameraTexture.width/(float)_cameraTexture.height;
        _aspectRatioFitter.aspectRatio=ratio;
        int orientation = -_cameraTexture.videoRotationAngle;
        _rawImageBackground.rectTransform.localEulerAngles=new Vector3(0, 0,
orientation);
    }
    public void SetUpCamera()
    {
        devices = WebCamTexture.devices;
        if (devices.Length == 0)
        {
            _isCamAvailable=false;
            return;
        }
        for(int i=0;i<devices.Length;i++){
            _cameraTexture = new WebCamTexture(devices[i].name,
(int)_scanZone.rect.width, (int)_scanZone.rect.height);
        }
        _cameraTexture.Play();
        _rawImageBackground.texture = _cameraTexture;
        _isCamAvailable=true;
    }
    private void Scan(){
        decoder=new OpenCVForUnity.ObjdetectModule.QRCodeDetector();
        Color32[] colors =new Color32[_cameraTexture.width *
_cameraTexture.height];
        points = new Mat(_cameraTexture.height, _cameraTexture.width,
CvType.CV_8UC4);
        img = new Mat(_cameraTexture.height, _cameraTexture.width,
CvType.CV_8UC4);
        Utils.webCamTextureToMat(_cameraTexture, img, colors, false, 0);
        decoder.detectAndDecodeMulti(img, message, points, imges_out);
        Regex regex = new Regex(@"Score:
((https://)|(http://))((\,)|(\/)|(\.)|(\:)|(\s)|(\w*))*;");
        Regex regexDel = new Regex(@"Score: |;");
        if(message.Count>0){
            for(int i=0;i<message.Count;i++){
```

```
    items.Add(new Item(message[i],imges_out[i]));
            res=regex.Match(message[i]).ToString();
            if(res!=""){
                res=regexDel.Replace(res, "");
                GetRequest(res);
                if(Int32.TryParse(messHttp,out a)){
                    items[i].SetScore(Int32.Parse(messHttp));
                }
            }
        }
    }
    for(int i=0;i<items.Count;i++){
        CreatingItem.instance.AddItem(items[i]);
    }
}
async static void GetRequest(string url){
    using (HttpClient client = new HttpClient()){
        using ( HttpResponseMessage response = await client.GetAsync(url)){
            using (HttpContent content = response.Content){
                string mycontent =  await content.ReadAsStringAsync();
                messHttp= mycontent;
            }
        }
    }
}
public void OnClickCallories(){
    pressedScore=false;
    pressedIngridients=false;
    pressedCallories=true;
}
public void OnClickIngridients(){
    pressedScore=false;
    pressedIngridients=true;
    pressedCallories=false;
}
public void OnClickScore(){
    pressedScore=true;
    pressedIngridients=false;
    pressedCallories=false;
}
```

```
    public void OnClickAll(){
        pressedScore=true;
        pressedIngridients=true;
        pressedCallories=true;
    }
    public void OnClickReciptScanner(){
        SceneManager.LoadSceneAsync("ReciptScanner");
    }
}
```

ПРИЛОЖЕНИЕ В
Исходный код скрипта ReciptScanner.cs

```
using UnityEditorInternal;
using System;
using System.Collections;
using System.Collections.Generic;
using System.Net.Http;
using OpenCVForUnity;
using OpenCVForUnity.CoreModule;
using OpenCVForUnity.UnityUtils;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;
using System.Text.RegularExpressions;
using TMPro;

public class ReciptScanner : MonoBehaviour
{
    public static ReciptScanner instance{get; private set;}
    void Awake()
    {
        if(instance==null){
            instance = this;
            return;
        }else if(instance==this){
            Destroy(this.gameObject);
        }
        DontDestroyOnLoad(this.gameObject);
    }
public class Item{
    private string name;
    private string url;
    public Item(string qr_info){
        name="";
        url="";
        string unit;
        Regex regexNames=new Regex(@"Name: ((\,)|(\s)|(\w*))*;");
        Regex regexNamesDel=new Regex(@"Name: |;");
        Regex regexScores=new Regex(@"Score:
((https://)|(http://))((\,)|(\/)|(\.)|(\:)|(\s)|(\w*))*;");
```

```
        Regex regexScoresDel=new Regex(@"Score: |;");
        if(qr_info!=""){
            unit="";
            unit = (regexNames.Match(qr_info).ToString());
            name=regexNamesDel.Replace(unit, "");

            unit="";
            unit = (regexScores.Matches(qr_info).ToString());
            url=regexScoresDel.Replace(unit, "");
        }
    }
    public string GetName(){
        if(name!=""){
            return name;
        }else{
            return "No Info";
        }
    }
    public string GetUrl(){
        if(url!=""){
            return url;
        }else{
            return "No Info";
        }
    }
};
    [SerializeField]
    private RawImage _rawImageBackground;
    [SerializeField]
    private AspectRatioFitter _aspectRatioFitter;

    [SerializeField]
    private RectTransform _scanZone;
    public Mat frameMat, rectifiedImage;
    private bool _isCamAvailable;
    private WebCamTexture _cameraTexture;
    public OpenCVForUnity.ObjdetectModule.QRCodeDetector decoder;
    public List<string> message;
    public  List<Item> items;
```

```
     void Start()
    {
       SetUpCamera();
    }
    void Update()
    {
        UpdateCameraRender();
    }
    public void SetUpCamera(){
        WebCamDevice[]  devices=WebCamTexture.devices;
        if(devices.Length==0){
            _isCamAvailable=false;
            return;
        }
        _cameraTexture=new WebCamTexture(devices[0].name,
(int)_scanZone.rect.width, (int)_scanZone.rect.height);
        if(_cameraTexture==null){
            _isCamAvailable=false;
            return;
        }
        _cameraTexture.Play();
        _rawImageBackground.texture=_cameraTexture;
        _isCamAvailable=true;
    }
    private void UpdateCameraRender(){
        if(_isCamAvailable==false){
            return;
        }
        float ratio=(float)_cameraTexture.width/(float)_cameraTexture.height;
        _aspectRatioFitter.aspectRatio=ratio;
        int orientation =-_cameraTexture.videoRotationAngle;
        _rawImageBackground.rectTransform.localEulerAngles = new
Vector3(0,0,orientation);
    }

    private void Scan(){
        try{
            decoder=new OpenCVForUnity.ObjdetectModule.QRCodeDetector();
            string res="";
            Mat img = new Mat(_cameraTexture.height, _cameraTexture.width,
CvType.CV_8UC4);
```

```
            Utils.webCamTextureToMat(_cameraTexture, img);
            res=decoder.detectAndDecode(img);
            Regex regex = new
Regex(@"\[((\w*)|(\s*)|(\/*)|(\,*)|(\;*)|(\:*)|(\.*)|(\n*))*\]");
            Regex regesDel =new Regex(@"\[|\]");
            MatchCollection matches=regex.Matches(res);
            foreach(var un in matches){
                items.Add(new Item(regesDel.Replace(un.ToString(), "")));
            }
        }
        catch{
        }
    }

    public void OnClickCameraForSetScoresMenu(){
        Scan();
        SceneManager.LoadSceneAsync("SetScoresMenu");
    }
}
```

ПРИЛОЖЕНИЕ Г
Исходный код скрипта SetScoresMenuController.cs

```
using UnityEditorInternal;
using System;
using System.Collections;
using System.Collections.Generic;
using System.Net.Http;
using OpenCVForUnity;
using OpenCVForUnity.CoreModule;
using OpenCVForUnity.UnityUtils;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;
using System.Text.RegularExpressions;
using TMPro;
public class SetScoresMenuController : MonoBehaviour
{
    [SerializeField]
    public GameObject item;
    public Transform[] items_Scores;
    public static string messHttp;
    public static SetScoresMenuController instance{get; private set;}
    void Awake()
    {
        if(instance==null){
            instance = this;
            return;
        }else if(instance==this){
            Destroy(this.gameObject);
        }
        DontDestroyOnLoad(this.gameObject);
    }
    void Start()
    {
        foreach(var unit in ReciptScanner.instance.items){
            item.gameObject.GetComponent<TextMeshPro>().text=unit.GetName();
            Instantiate(item);
        }
    }
    void Update()
    {
```

```
    }
    async static void SendRequest(string url){
        using (HttpClient client = new HttpClient()){
            using ( HttpResponseMessage response = await client.GetAsync(url)){
                using (HttpContent content = response.Content){
                    string mycontent =  await content.ReadAsStringAsync();
                    messHttp= mycontent;
                }
            }
        }
    }
    IEnumerator Upload(string url, int score)
    {
        string myData = score.ToString();
        using (UnityWebRequest www = UnityWebRequest.Put(url, score))
        {
            yield return www.SendWebRequest();

            if (www.result != UnityWebRequest.Result.Success)
            {
                Debug.Log(www.error);
            }
            else
            {
                Debug.Log("Upload complete!");
            }
        }
    }
    void OnClickARCameraForQRCodeScanner(){
        SceneManager.LoadSceneAsync("YummyBookSaved");
    }
    public void OnClickSend()
    {
        items_Scores=new Transform[transform.childCount];
        foreach(Transform t in items_Scores){
            StartCoroutine(Upload(t.gameObject.GetUrl(),
t.gameObject.GetScore()));
        }
        OnClickARCameraForQRCodeScanner();
    }
}
```

ПРИЛОЖЕНИЕ Д
Исходный код скрипта SetScoresMenuItemController.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using TMPro;
public class ScoresMenuItemController : MonoBehaviour
{
    private int Scores;
    private int minScores;
    private int maxScores;
    private TextMeshProUGUI Name;
    public string url;
    public TextMeshProUGUI ScoreText;
    void Start(){
        Scores=3;
        minScores=QRCodeScanner.instance.GetMinScoreValue();
        maxScores=QRCodeScanner.instance.GetMaxScoreValue();
        ScoreText.text=Scores.ToString();
    }
    void Update()
    {
        ScoreText.text=Scores.ToString();
    }
    public void OnClickPlusScores(){
        if(Scores<maxScores){
            Scores++;
        }
    }
    public void OnClickMinusScores(){
        if(Scores>minScores){
            Scores--;
        }
    }
    public string GetUrl(){
        return url;
    }
    public int GetScore(){
        return Scores;
    }
}
```