ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Департамент программной инженерии

Согласовано

Доцент департамента
программной инженерии
факультета компьютерных наук
канд. техн. наук

_____ Александров Д.В.

"____"_____ 2018 г

Утверждаю

Академический руководитель
образовательной программы
«Программная инженерия»
профессор департамента программной
инженерии канд. техн. наук

_____ Шилов В. В.

"____"_____ 2018 г

Клиент-Серверное Веб-Приложение для Управления Скидками в Розничных Сетях

Пояснительная записка

ЛИСТ УТВЕРЖДЕНИЯ

RU.17701729.506900 81 01-1

Студент группы БПИ 151 НИУ ВШЭ
_____ Суровцев М.А.
"____"_____ 2018 г

2018

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

УТВЕРЖДЕНО
RU.17701729.506900 81 01-1

Клиент-Серверное Веб-Приложение для Управления Скидками в Розничных Сетях

Пояснительная записка

RU.17701729.506900 81 01-1

Листов 27

2018

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

1. Аннотация

В данном программном документе приведена пояснительная записка к программе «Клиент-Серверное Веб-Приложение для Управления Скидками в Розничных Сетях». В данном программном документе, в разделе «Введение» указано наименование программы, краткое наименование программы и документы, на основании которых ведется разработка. В разделе «Назначение и область применения» указано функциональное назначение программы и краткая характеристика области применения программы. В данном программном документе, в разделе «Технические характеристики» содержатся следующие подразделы:

- постановка задачи на разработку программы
- описание алгоритма и функционирования программы с обоснованием выбора схемы алгоритма решения задачи и возможные взаимодействия программы с другими программами
- описание и обоснование выбора состава технических и программных средств

Так же к документу прикреплены приложения. Настоящий документ разработан в соответствии с требованиями:

1. ГОСТ 19.101-77 Виды программ и программных документов [4];
2. ГОСТ 19.102-77 Стадии разработки [1];
3. ГОСТ 19.103-77 Обозначения программ и программных документов [4];
4. ГОСТ 19.104-78 Основные надписи [6];
5. ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом [7];
6. ГОСТ 19.404-79 Пояснительная записка. Требования к содержанию и оформлению [3].

Изменения к данному Техническому заданию оформляются согласно ГОСТ 19.603-78 [8]

Содержание

2. Введение

2.1. Наименование программы

Наименование программы на русском: «Клиент-Серверное Веб-Приложение для Управления Скидками в Розничных Сетях».

Наименование на английском: «The Client-Server Web Application for Managing the Products Discounts in Retail Networks».

2.2. Краткая характеристика

Клиент-серверное Web приложение для отслеживания скидок и акций в продуктовых магазинах. Основные функциональные возможности приложения: представление каталога акций в удобном и доступном виде, работа со списком покупок, отображение ближайших магазинов на карте.

3. Назначение разработки

3.1. Функциональное назначение

Продукт является приложением “на стыке онлайна и оффлайна”, позволяющим планировать покупки в магазине с учетом актуальных скидок. К функциональным возможностям программы относятся: просмотр списка актуальных акций, составление и редактирование списка покупок, поиск ближайшего магазина на карте, регистрация пользователей.

3.2. Эксплутационное назначение

Программа предназначена для запуска на персональных компьютерах (веб-версия) и мобильных устройствах (веб-версия). Программа является приложением, позволяющим пользователю искать и подбирать интересующие его товары по сниженной цене, и, таким образом, экономить средства.

4. Технические характеристики

4.1. Постановка задачи на разработку программы

Программа должна соответствовать требованиям, представленным в Техническом Задании.

Задачи работы (серверная часть приложения):

1. Реализовать REST API:
   (a) Обработка запросов для работы со списками покупок пользователей
   (b) Обработка запросов для кроулера
   (c) Обработка запросов для получения списка актуальных акционных товаров
   (d) Обработка запросов для авторизации пользователей
2. Спроектировать базу данных для хранения:
   (a) Акционных товаров
   (b) Аккаунтов пользователей
   (c) Списков покупок пользователей

Задачи работы (клиентская часть приложения):

4.1.1. Состав выполняемых функций. Клиентская часть.

1. Реализовать возможность просмотра списка доступных магазинов с акционными товарами
2. Реализовать представление текущих акций для конкретного магазина:
   (a) В виде общего списка
   (b) По категориям
3. Реализовать возможность регистрации и авторизации пользователей в системе для работы со списком покупок
4. Реализовать создание и удаление списка покупок
5. Реализовать работу со списком покупок:
   (a) Добавление и удаление товаров магазинов
   (b) Добавление и удаление пользовательских позиций
   (c) Рекомендация товаров магазинов на основе пользовательских позиций

4.2. Описание алгоритма и функционирования программы

4.2.1. Схема базы данных

Пожалуй, главной сущностью в схеме является «item». Именно в эту таблицу записываются товары, собранные кроулером, и из этой таблицы извлекаются данные для отобра-

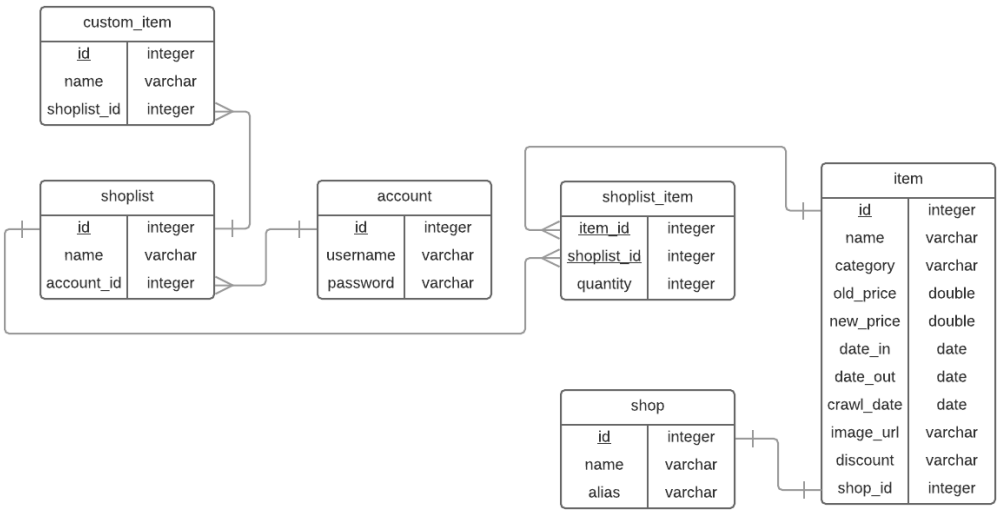

Рис. 1: Схема базы данных

жения в клиентских приложениях. Каждый товар также связан с магазином, которому он принадлежит. (Один магазин может иметь много товаров).

Как видно из Рис. 1, пользователь может иметь несколько списков покупок, которые в свою очередь могут содержать много товаров.

4.2.2. Список покупок

Список покупок имеет следующую структуру:

1. Имя списка покупок (Например, «Завтрак»)
2. Список товаров магазинов (Например, «Молочный коктейль Чудо детки шоколад; клубника 2,5%, 200 мл»)
3. Пользовательские позиции (Например, «Сок»)
4. Итоговая сумма покупок

При добавлении пользовательской позиции в список покупок, система предложит пользователю актуальные товары из магазинов.
Так, например, для позиции «Сок» будут предложены следующие товары, которые пользователь может добавить в список товаров магазинов:

1. Сок Сады Придонья Яблоко зеленое 125мл
2. Сок Добрый Яблочный 2л
3. Десерт Джелео многослойный с соком вишня-персик-яблоко 0,4%, 150 г

Список товаров магазинов и предложенные товары на основе пользовательских позиций для удобства группируются по магазинам. Добавлять товары в список покупок могут только авторизованные пользователи.

4.2.3. Переключение между VR Mode и Normal Mode

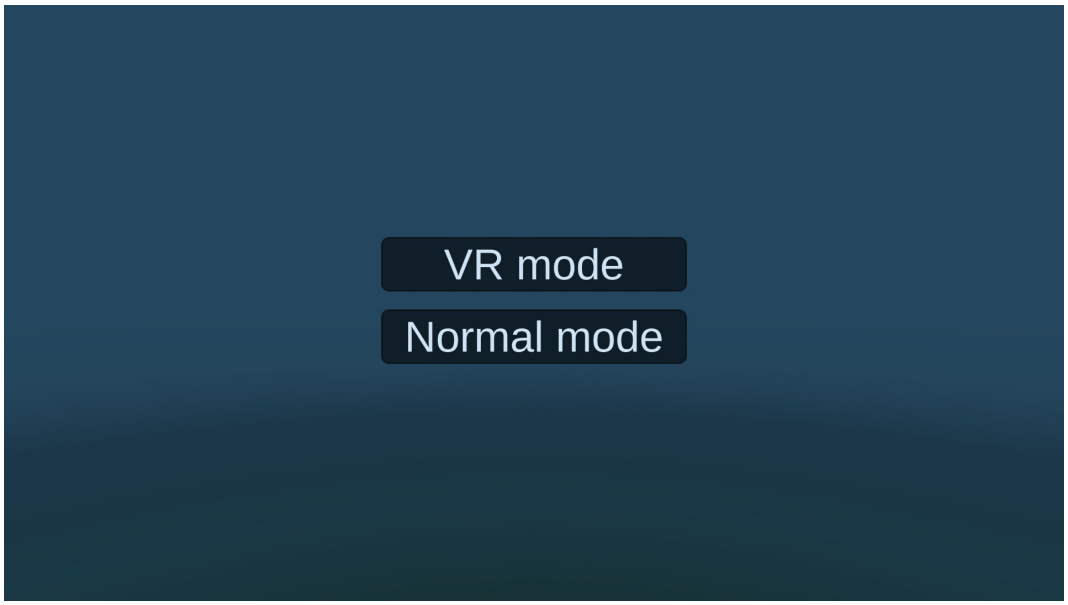

Рис. 2: меню переключения режима

В любой VR сцене есть объект GvrViewerMain. Этот prefab содержит скрипт GvrViewer, который контролирует параметры VR mode. При помощи изменения параметра VRModeEnabled на true или false, VR mode будет включен/ выключен соответственно. По скольку каждая сцена загружается по умолчанию в VR mode, надо запоминать выбор пользователя в начале и при каждой загрузке последующих сцен выставлять выбранный режим. Для этого был написан скрипт ModChanger.cs, который запоминает выбор игрока.

Рис. 3: разница между VR Mode (сверху) и Normal Mode (снизу)

4.2.4. Передвижение игрока без триггера

В VR играх передвижение игрока может происходить различными методами. Например, есть способ, в котором при нажатии триггера, начинается движение в сторону взгляда персонажа, при следующем нажатии на триггер, движение останавливается. Поскольку одной из главных задач проекта была разработать игру, в которой не задействуется триггер (для совместимости со всеми видами Cardboard), то была разработана система передвижения «взглядом».

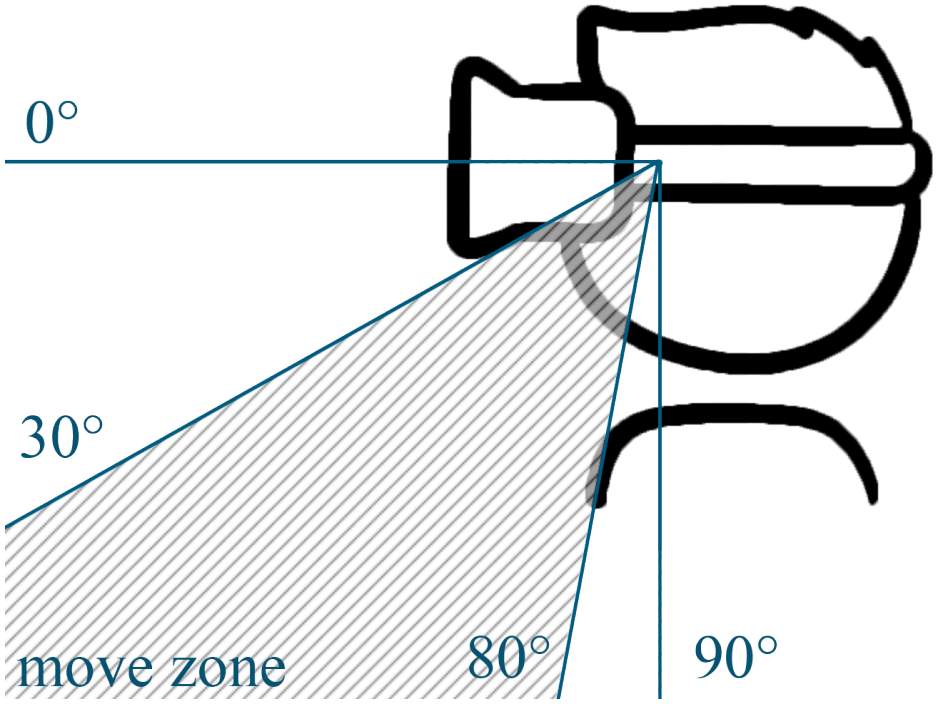

Рис. 4: передвижение при помощи угла взгляда

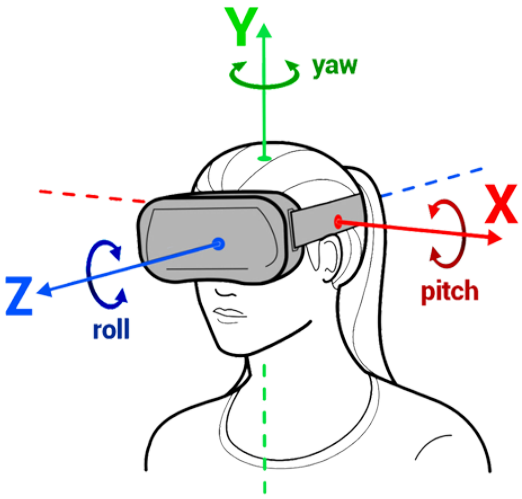

Рис. 5: расположение осей в виртуальной реальности

На схеме(рис. 4) видно, что при достижении угла от 30°до 80°по оси X (см. рис.

5) происходит движение игрока. Движение происходит в сторону направления взгляда (вперед).

Ниже приведен Update метод скрипта WalkByLook.cs. Любой Update метод в юнити вызывается раз за фрейм.

```
void Update()
{
    shouldMove = false;
    mustLookToTheObject = GazeInputModule.pointingAt != null;
    if (mustLookToTheObject && GazeInputModule.pointingAt.Count > 0)
        lookingToObject = (((bool)GazeInputModule.pointingAt[1] &&  (float)GazeInputModule.pointingAt[2] >= 4F)
        || !((bool)GazeInputModule.pointingAt[1]));

    moveForward = (vrCam.eulerAngles.x >= angle1 && vrCam.eulerAngles.x <= angle2);

    if (moveForward && (!mustLookToTheObject || (mustLookToTheObject && lookingToObject)))
    {
        shouldMove = true;
        Vector3 forward = vrCam.TransformDirection(Vector3.forward);
        cc.SimpleMove(forward * speed);
        if (!playing)
            StartCoroutine(PlaySteps());
    }
}
```

4.2.5. Взаимодействие предметов без триггера

В VR играх взаимодействие с предметами происходит различными методами. Например, при нажатии на триггер, предмет перемещается в руку персонажа или перемещается в инвентарь. Поскольку одной из главных задач проекта была разработать игру, в которой не задействуется триггер (для совместимости со всеми видами Cardboard), то была разработана система взаимодействия с объектами «взглядом».

При наведении reticle (прицела) на объект, над которым может быть совершено действие, полоса загрузки внизу экрана активируется и по ее заполнению произойдет действие над объектом. Это может быть, например, взятие его в руку (см рис. 6, рис. 7).

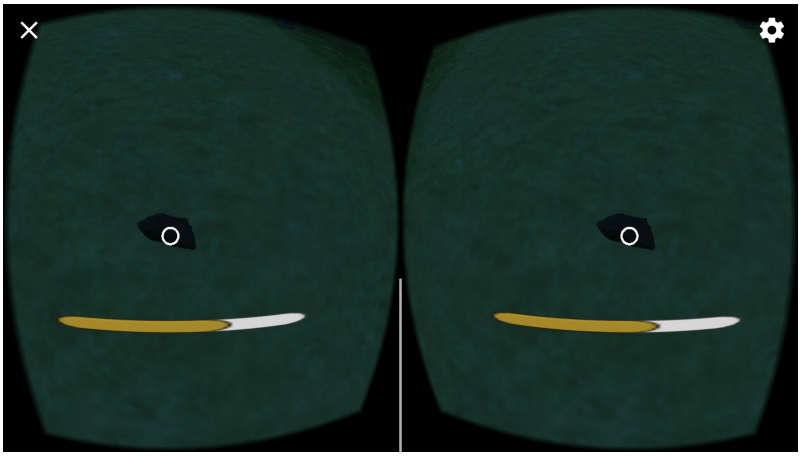

Рис. 6: ожидание выполнения действия над объектом камень

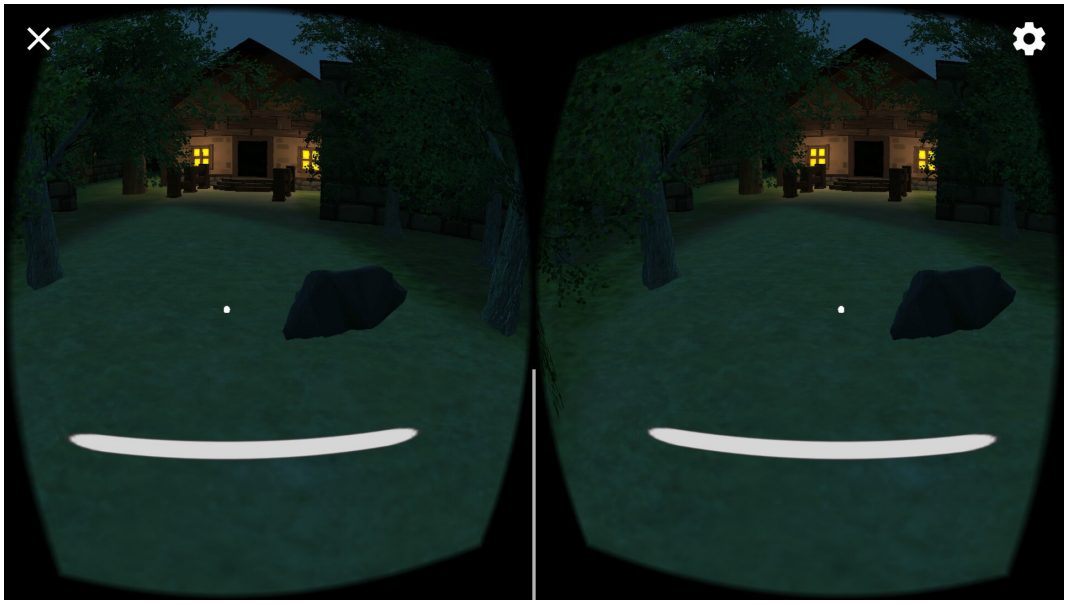

Рис. 7: результат выполнения действия над объектом - камень в руке

Поскольку при взгляде на объект часто бывает, что угол наклона камеры попадает в приемлемую для передвижения зону(от 30°до 80°), то в скрипте WalkByLook.cs предусмотрена остановка передвижения при взгляде на объект, над которым можно совершить действие. Это позволяет брать предметы «на ходу».

4.2.6. Перемещение предметов в руку

Поднятый предмет перемещается в руку путем изменения его положения на равное положению руке. Рука - невидимый игровой объект, расположенный относительно персонажа (подсвечен фиолетовым) как на рис. 8. Относительно игрока рука расположена как на рисунке 7.

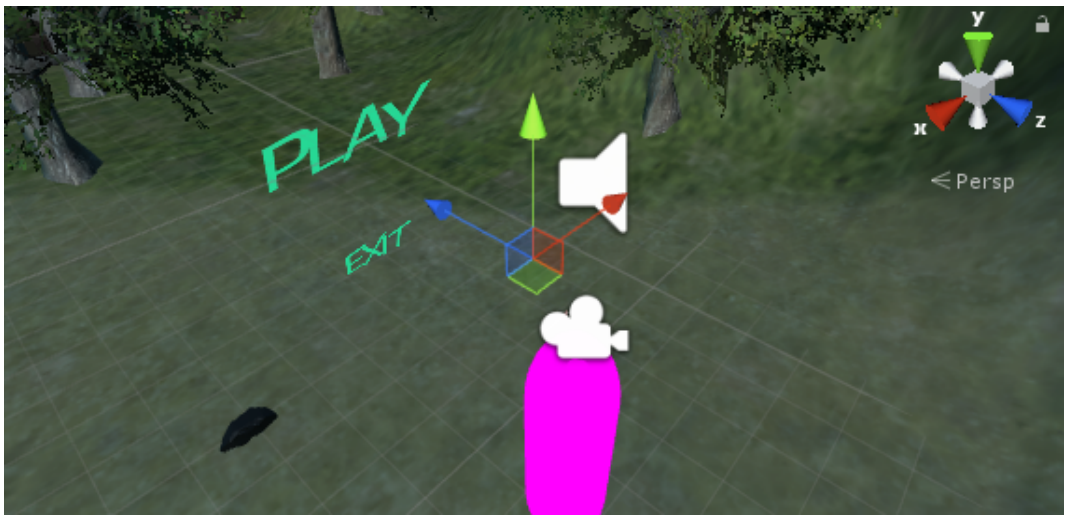

Рис. 8: расположение руки относительно персонажа (подсвечен фиолетовым)

За процесс поднятия предмета отвечает скрипт PickUpObject. При вызове метода MoveToPlayersHand() предмет, к которому прикреплен скрипт переместится в руку персонажа. Так же здесь сохраняются исходные пропорции объекта, поскольку при изменении родителя (а тут объект переходит от одного родителя к handMountingPosition) меняются пропорции объекта (особенность Unnity). Пока предмет находится в руке у него отключаются RigidBody и Collider для предотвращения столкновения с окружающей средой. В последствии при броске предмета они включаются обратно. В дополнение к этому у каждого предмета в этом скрипте задана позиция и вращение руки, которое необходимо ей придать при поднятии объекта (handPosition и handRotation). Это нужно для того, чтобы предмет, например, ключ, не был направлен обратной стороной к двери при попытки открыть ее им. Ниже приведен метод MoveToPlayersHand(). Этот скрипт также отвечает за проигрывание звука взятия предмета.

```
public void MoveToPlayersHand()
{
    if (!pickedUp)
    {
        vrCam.parent.GetComponents<AudioSource>()[0].Play();
        oldScale = gameObject.transform.localScale;
        gameObject.transform.parent = handMountingPosition;
        gameObject.GetComponent<Rigidbody>().useGravity = false;
        gameObject.GetComponent<Rigidbody>().isKinematic = true;
        gameObject.GetComponent<Collider>().enabled = false;
        gameObject.GetComponent<Rigidbody>().constraints = new RigidbodyConstraints();
        gameObject.transform.localScale = oldScale;

        gameObject.transform.localPosition = new Vector3(0F, 0F, 0F);
        gameObject.transform.localRotation = Quaternion.Euler(0F, 0F, 0F);
        handMountingPosition.localRotation = Quaternion.Euler(handRotation);
        handMountingPosition.localPosition = handPosition;
        handMountingPosition.localScale = new Vector3(1F, 1F, 1F);
    }
}
```

Этот же скрипт отвечает за бросание предметов. При положении камеры (головы игрока) от 275°до 303°вызывается метод ThrowObject(), его код приведен ниже. В нем проигрывается аудио бросания предмета, включаются RigidBody и Collider у объекта и родителем становится null.

```
public void ThrowObject()
{
    if (pickedUp && handMountingPosition.GetChild(0).name.CompareTo(gameObject.name) == 0)
    {
        vrCam.parent.GetComponents<AudioSource>()[0].Play();
        gameObject.GetComponent<Rigidbody>().useGravity = true;
        gameObject.GetComponent<Rigidbody>().isKinematic = false;
        gameObject.GetComponent<Collider>().enabled = true;
        gameObject.transform.parent = null;
        gameObject.transform.localScale = oldScale;
    }
}
```

4.2.7. Модификация исходников Google VR SDK

В процессе написания программы были выявлены проблемы при использовании скрипта GazeInputModule, предоставляемым в SDK от Google для VR приложений. Проблема заключалась в том, когда камера светит лучем вперед для того чтобы получить первый объект, который он ударит, он упирался в самого персонажа и видел дальше него лишь иногда (непредсказуемое поведение). См. рис. 9.

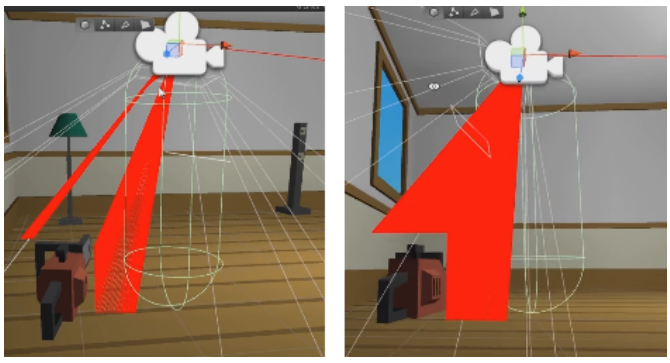

Рис. 9: Debug режим. Проблема при использовании оригинального скрипта от Google GazeInputModule.cs. Слева - до, справа - после внесения модификаций в скрипт.)

Изменения приняли следующий вид. В методе CastRayFromGaze(), который «стреляет лучем», чтобы получить объект, на который в данный момент указывает камера, был добавлен следующий код:

```
...
if (pointerData.pointerCurrentRaycast.gameObject != null)
    if (pointerData.pointerCurrentRaycast.gameObject.name.CompareTo("Character") == 0)
        if (m_RaycastResultCache.Count > 1)
            pointerData.pointerCurrentRaycast = m_RaycastResultCache[1];
        else
    pointerData.Reset();
...
```

После того, как был взят первый объект, с которым столкнулся луч, он проверяется на равенство null. Затем, если это сам персонаж, то берется следующий объект за персонажем, который ударила камера. На рисунке 9 видно, что после внесения изменений, камера стала видеть весь Collider формы прямоугольного параллелепипеда корректно.

Далее, при условии, что текущий GameObject не null, формируется статичесикй массив pointingAt из трех объектов для получения дополнительных данных из просвечивания объектов лучем:

0 - сам GameObject

1 - hasEventTrigger. Интерактивный ли это объект. В зависимости от этого параметра определяется будет ли происходить взаимодействие с объектом или нет

2 - distance. Расстояние до объекта

```
float distance = Vector3.Distance(pointerData.pointerCurrentRaycast.gameObject.transform.position, vrCam.transform.position);
 bool hasEventTrigger = pointerData.pointerCurrentRaycast.gameObject.GetComponent<EventTrigger>()s
  as EventTrigger != null;

 pointingAt.Add(pointerData.pointerCurrentRaycast.gameObject);
 pointingAt.Add(hasEventTrigger);
 pointingAt.Add(distance);
```

Далее, было решено сделать объекты интерактивными на определенном расстоянии. Поскольку неправильно и не логично было бы взаимодействовать с объектом, который находится в одном углу комнаты, находясь в другом. Поэтому было выбрано расстояние 4.35. Дальше него камера пусть и пускает лучи, но в результат ничего не записывает.

```
...
if (!(distance <= 4.35F && hasEventTrigger))
{
pointerData.pointerCurrentRaycast = new RaycastResult();
m_RaycastResultCache.Clear();
}
...
```

4.2.8. Описание головоломок внутри комнаты

Конечная цель игры - найти ключ от двери и выбраться из заперти. Ключ игроку открывается после решения головоломки с цифрами на правой книжной полке. Код от нее спрятан за картиной, которая висит справа от второй кровати и появляется после решения головоломки с колокольчиками. Чтобы решить головоломку с колокольчиками, необходимо чтобы на стене между двумя кроватями висели все 5 колокольчиков. Намек на это есть в записке на столе. Один колокольчик можно найти в котле справа от входа, второй висит над входной дверью, его необходимо открутить при помощи отвертки, которая находится на левой книжной полке.

Откручивание отверткой колокольчика реализовано в скрипте Uscrew.cs, его код:

```
bool unscrewed = false, hasScrewDriver = false;
public Transform handMountingPosition;

void Update()
{
    if (handMountingPosition.childCount > 0)
        hasScrewDriver = handMountingPosition.GetChild(0).name.Contains("ScrewDriver");
}

public void PointerClick()
{
    if (!unscrewed && hasScrewDriver)
    {
        gameObject.SetActive(false);
        gameObject.GetComponent<BoxCollider>().enabled = false;
        GameObject.Find("InnerBell").GetComponent<Rigidbody>().constraints = new RigidbodyConstraints();
        unscrewed = true;
```

```
    }
}
```

В Update() проверяется наличие отвертки в руке персонажа, а в PointerClick() у колокольчика отключаются constraints, что держат его на месте, и он падает (рис. 10).

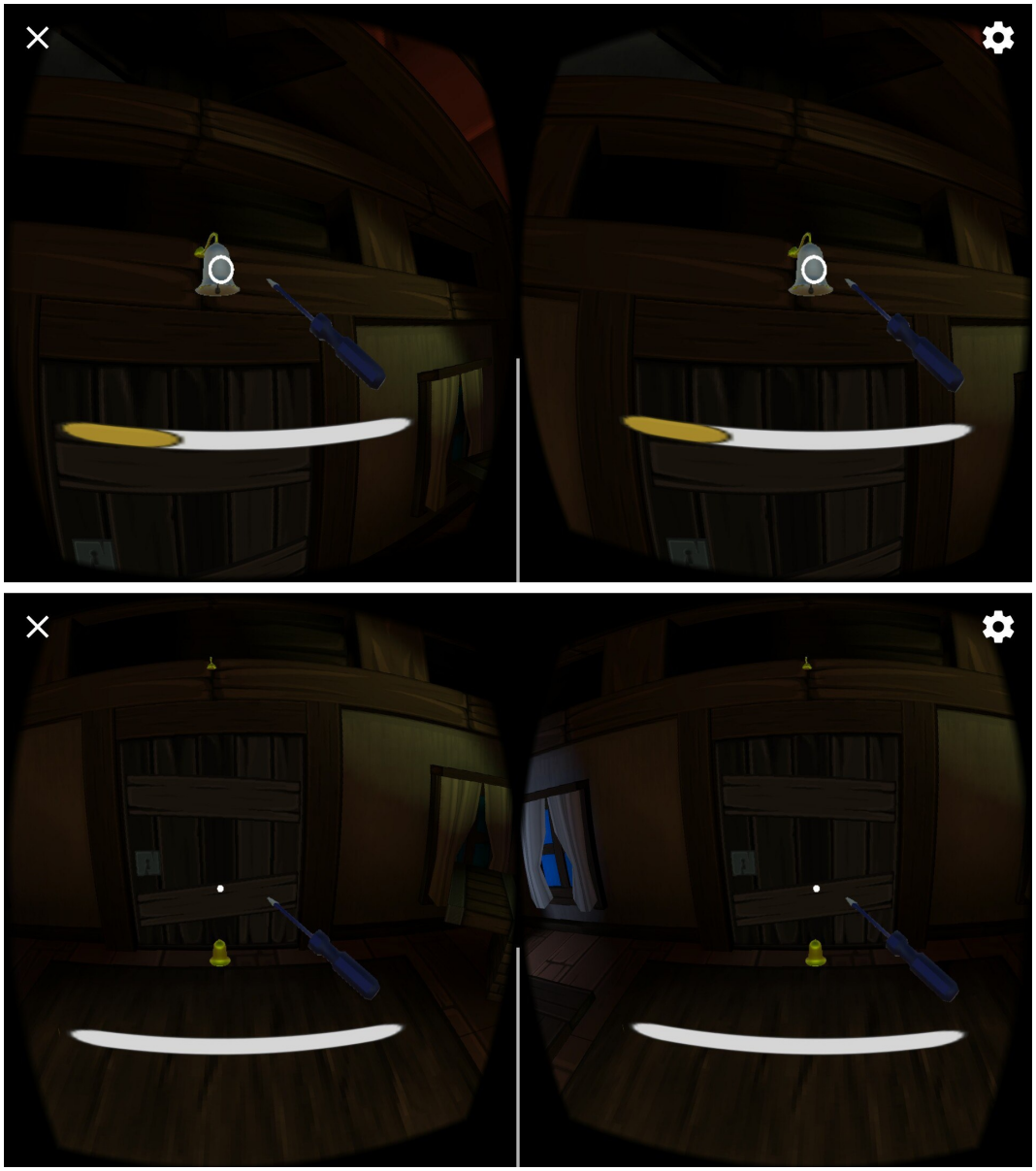

Рис. 10: откручивание колокольчика при помощи отвертки. Сверху - сам процесс, снизу - результат.

Когда все 5 колокольчиков висят на месте, сразу же начнется следующая головоломка. Основная и самая сложная. Заключается она в том, что сначала проигрывается последовательность (5 раз бьют по разным колокольчикам), а затем, игрок должен ее повторить, ударив по тем же колокольчикам, что были показаны.

За данную головоломку отвечает скрипт BellPuzzleLogic.cs. Сначала показывается паттерн сгенерированной последовательности, затем игрок может начинать вводить

свою. При очередном неверном выборе колокольчика, подается звуковой сигнал, значащий, что был сделан неверный выбор и произойдет показ паттерна снова.

При верном вводе всей последовательности подается особый звуковой сигнал, означающий, что задание пройдено. Производится визуальный переход ко второму. Появляется и подсвечивается фиолетовыми частицами картина справа (рис. 11).

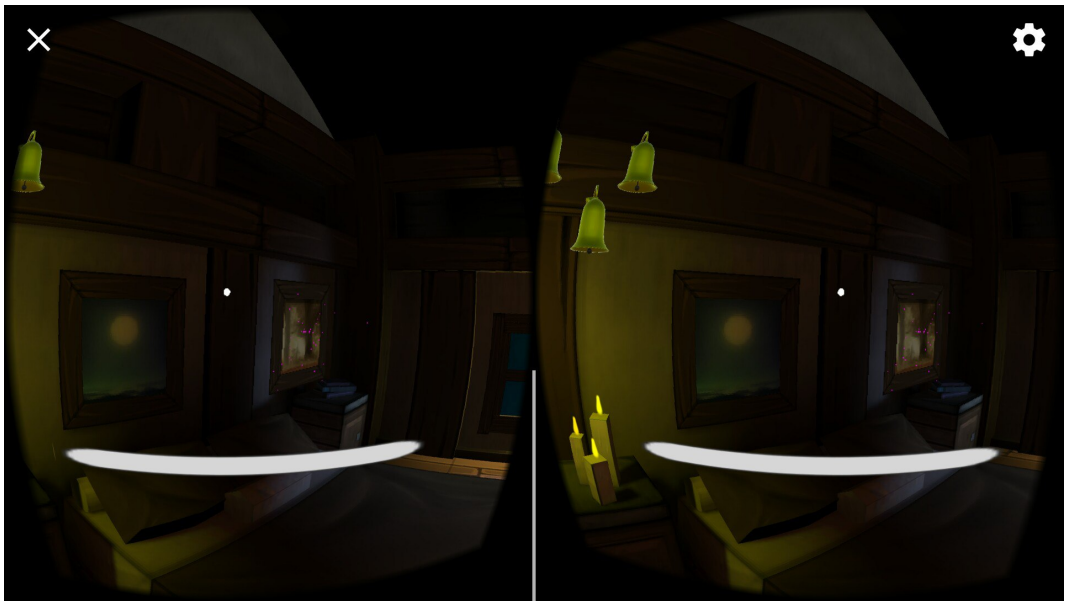

Рис. 11: появившаяся картина после завершения задания с колокольчиками.

За картиной находится код. Игрок должен запомнить его. Рис (12).

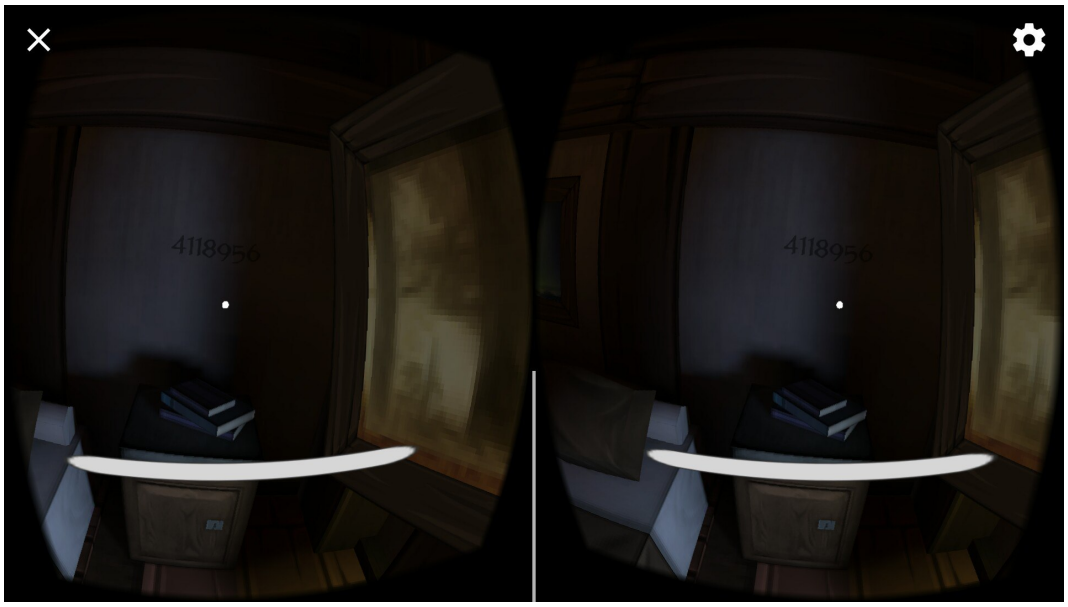

Рис. 12: код дл финальной головоломки за картиной.

Рис. 13: ввод кода для получения ключа.

Финальная задача состоит в том, чтобы ввести полученный код в поле, находящееся на правом книжном шкафу. В скрипте LastPuzzleLogic.cs расположен код для последней задачи. Здесь при вводе очередной цифры пользователем последовательность проверяется на соответствие правильному ключу:

```
void Start()
{
    solved = false;
    numbers = new char[7];
}

void Update()
{
    if (!solved)
    {
        int i = 0;
        foreach (Transform child in transform)
        {
            numbers[i] = child.GetComponent<TextMesh>().text[0];
            ++i;
        }

        if (new String(numbers).CompareTo(key) == 0)
        {
            solved = true;
            Key.SetActive(true);
            Key.GetComponent<Rigidbody>().constraints = new RigidbodyConstraints();
            gameObject.SetActive(false);
        }
    }
}
```

Когда игрок ввел верную последовательность, ему выдается ключ, при помощи ко-

торого он может отпереть дверь.

Рис. 14: получение ключа.

Рис. 15: открытие двери.

В скрипте KeyDoorOpener.cs содержится логика открытия двери и загрузки внешнего (снаружи дома) окружения. Изначально оно было отключено из соображений ресурсоэффективности и временных требований к программе. Объекты, которые игрок не наблюдает, не должны прорисовываться. Так же скрипт проигрывает звук отпирания замка, открытия двери.

```
public GameObject door, final;
private static bool opened = false;
```

```
private float timeToOpen = 1F;
public Transform handMountingPosition;

void Start()
{
    opened = false;
    gameObject.GetComponent<BoxCollider>().enabled = false;
}

void Update()
{
    if (handMountingPosition.childCount > 0)
  gameObject.GetComponent<BoxCollider>().enabled = (handMountingPosition.GetChild(0).name.CompareTo("Key") == 0);
}

public void PointerClick()
{
    StartCoroutine(LoadFinalScene());
    StartCoroutine(OpenDoor());
}

private IEnumerator LoadFinalScene()
{
    final.SetActive(true);
    yield return null;
}

private IEnumerator OpenDoor()
{
    if (!opened)
    {
        // play unlock sound
        gameObject.GetComponents<AudioSource>()[0].Play();
        // show input Key and disable current holding Key
        transform.GetChild(0).gameObject.SetActive(true);
        handMountingPosition.GetChild(0).gameObject.SetActive(false);
        yield return new WaitForSeconds(1F);
        transform.GetChild(0).gameObject.SetActive(false);
        transform.GetChild(1).gameObject.SetActive(true);
        gameObject.GetComponents<AudioSource>()[1].Play();
        door.GetComponent<Animation>().Play();
        OpenDoorAndLoadScene.opened = opened = true;
        yield return new WaitForSeconds(timeToOpen);
    }
}
```

По завершении игры, пользователю будет показано благодарственное сообщение с приглашением пройти игру заново. Происходит этот показ по столкновению персонажа с невидимым объектом, который и вызывает показ.

Рис. 16: финал игры.

5. Технико-экономические показатели

5.1. Предполагаемая потребность

«Игра — Эскейп Квест с Использованием Очков Виртуальной Реальности» может быть использована в игровой сфере. Программу могут использовать люди, нуждающиеся как в отдыхе в мире виртуальной реальности, так и желающие интеллектуально развиться в игровой форме.

5.2. Экономические преимущества разработки

На момент принятия решения о написании данного продукта во Всемирной сети Интерент был лишь 1 аналог: игра «Lost in the Kismet». Преимущества данной разработки перед конкурентом:

1. Совместимость со всеми девайсами Cardboard
2. VR Mode и Normal Mode
3. Оригинальный сюжет

6. Источники, используемые при разработке

6.1. Список используемой литературы

1. ГОСТ 19.102-77 Стадии разработки. //Единая система программной документации. -М.: ИПК Издательство стандартов, 2001.
2. ГОСТ 19.201-78 Техническое задание. Требования к содержанию и оформлению // Единая система программной документации. -М.:ИПК Издательство стандартов, 2001.
3. ГОСТ 19.404-79 Пояснительная записка. Требования к содержанию и оформлению. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001
4. ГОСТ 19.101-77 Виды программ и программных документов //Единая система программной документации. -М.: ИПК Издательство стандартов, 2.: 001.
5. ГОСТ 19.103-77 Обозначения программ и программных документов. //Единая система программной документации. -М.: ИПК Издательство стандартов, 2001.
6. ГОСТ 19.104-78 Основные надписи //Единая система программной документации. -М.: ИПК Издательство стандартов, 2001.
7. ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001
8. ГОСТ 19.603-78 Общие правила внесения изменений. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001
9. Oculus Documentation [Электронный ресурс]: Режим доступа: https:// developer3.oculus.com/documentation/
10. Oculus Developers Blog [Электронный ресурс]: chrispruett – Squeezing Performance out of your Unity Gear VR Game, 2015 - Режим доступа: https://developer3.oculus.com/blog/squeezing-performance-out-of-your-unity-gear-vr-game/
11. Uninty Scripting Reference [Электронный ресурс]: Режим доступа: https: //docs.unity3d.com/ScriptReference/

7. Приложение 1. Терминология

7.1. Терминология

VR - Virtual Reality (Виртуальная Реальность). Созданный программными и техническими средствами мир, передаваемый человеку при помощи взаимодействия специальных внешних устройств с его органами чувств.

Google Cardboard - Эксперимент компании Google в области виртуальной реальности, в основе которого лежит шлем, который, по замыслу разработчиков, можно собрать из подручных материалов. Состоит за картона, оптических линз, липучек-застёжек. Так же необходимо наличие смартфона с поддержкой технологии VR и установленным VR приложением. Он закрепляется непосредственно в шлеме, а шлем крепится к голове пользователя, что передает программе движения говолы.

VR Mode - Режим отображения картинки на экране мобильного устройства, при котором экран разделен на 2 части, на которые выводятся изображения для левого и правого глаза. Система линз в Google Cardboard (Cardboard) корректирует геометрию изображения, доставляя пользователю полное ощущение присутствия в виртуальном 3D мире.

Normal Mode - Режим отображения картинки на экране мобильного устройства, при котором экран не разделяется на 2 части. Не требует наличия Google Cardboard.

Reticle - Указатель (прицел) в центре экрана, меняющий свою позицию синхронно с главной камерой. Позволяет более точно прицеливаться для взаимодействия с объектами.

Draw Calls - Вызовы отрисовки. То, сколькло объектов отрисовываются на экране за один кадр (frame).

Escape the room - (рус. Выйти из комнаты; покинуть комнату) — жанр компьютерных игр, поджанр квестов, основная цель которого - найти выход из запертого помещения, используя любые подручные средства.

Rigidbody - Rigidbody дает игровым объектам физические свойства, такие как масса, влияние гравитации и constraints (ограничения, задержки) по всем осям для задержания объекта на месте.

Collider - Компонент коллайдер определяет фигуру объекта с целью установки физических столкновений.

8. Приложение 2. Описание классов

При программировании на Unity, каждый скрипт предсавляет собой отдельный класс, в котором, как правило, есть 2 метода: Start и Update. Start вызывается 1 раз при запуске скрипта, а Update вызывается 1 раз за фрейм.

8.1. WalkByLook

Поля:

```
public float angle1 = 30F, angle2 = 80F;
public float speed = 5F;
private bool moveForward;
private CharacterController cc;
public Transform vrCam;
private bool lookingToObject = false;
private bool mustLookToTheObject = false;
public AudioSource[] footSteps;
private bool shouldMove = false, playing = false;
```

Класс, созданный для передвижения игрока при помощи угла наклона головы по оси X.

8.2. BellBehaviour

Поля:

```
public Material bellHoverMaterial;
public Material defaultBellMaterial;
private AudioSource audioSrc;
```

Отвечает за поведение колокольчика при наведении на него reticle и нажатии (загрузки нижнего индикатора).

8.3. BellPuzzleLauncher

Поля:

```
public Transform handMountingPosition;
private bool hasBell = false;
private float angle;
public Transform vrCam;
public GameObject bellPuzzle;
public Transform firstToComplete, secondToComplete;
private bool firstPut = false, secondPut = false;
```

Прикрепляется к невидимому кубу с коллайдером, который при столкновении с персонажем и наведении им указателя на место головоломки вешает колокольчик из руки, и, если все 5 на месте, запускает головоломку.

8.4. KeyDoorOpener

Поля:

```
public GameObject door, final;
private static bool opened = false;
private float timeToOpen = 1F;
public Transform handMountingPosition;
```

Класс, отвечающий за открытие двери когда в руке у игрока находится ключ.

8.5. LastPuzzleLogic

Поля:

```
char[] numbers;
bool solved;
float step;
string key = "4118956";
public GameObject Key;
```

Класс, отвечающий за поведение последней головоломки с ключем на книжном шкафу.

8.6. MainMenu

Поля:

```
public GameObject character, story;
```

Класс главного меню, прикрепляется к двум лэйблам «PLAY» и «EXIT».

8.7. ModChanger

Поля:

```
public GameObject modeChooser;
public static bool vrModeEnabled = false;
private bool checked_ = false;ы
```

Класс, отвечающий за выбор режима (VR Mode или Normal Mode)

8.8. NumberChanger

Поля:

```
int n;
public float timeDelay = 1F;
float currentTime = 0F;
bool gazedAt = false;
```

Класс, отвечающий за изменение каждой цифры в последней головоломке.

8.9. OpenDoorAndLoadScene

Поля:

```
public GameObject door;
public static bool opened = false;
public float timeToOpen = 0.6F;
```

Класс, отвечающий за открытие двери с TransitScene для входа в дом.

8.10. PickUpObject

Поля:

```
public Vector3 handPosition;
public Vector3 handRotation;
Vector3 oldScale;
public float angle1 = 275F, angle2 = 303F;
public Transform handMountingPosition;
public Transform vrCam;
private bool tilted;
bool pickedUp = false;
```

Класс, отвечающий за перемещение объекта с места, где он лежал, в руку игроку.

8.11. StoryLogic

Поля:

```
public GameObject character, vrCam, guide, tryIt, glhf, stone, constraint;
private bool freezeMove = true, thrown = false, shown = false;
```

Класс, отвечающий за последовательность показа приветственных текстов и обучающего фрагмента вначале.

8.12. Unscrew

Поля:

```
bool unscrewed = false, hasScrewDriver = false;
public Transform handMountingPosition;
```

Класс, отвечающий за откручивание колокольчика со стены при помощи отвертки в руке игрока.

8.13. VRSlider

Поля:

```
public float fillTime = 2f;
public int Scene = 1;
private Slider mySlider;
private float timer;
private bool gazedAt;
private Coroutine fillBarRoutine;
```

Класс, отвечающий за анимацию слайдера - индикатора внизу экрана.

Лист регистрации изменений

Изм.	Номера листов (страниц)				Всего листов (страниц) в докум.	№ докум.	Входящий № сопроводительного докум. и дата	Подпись	Дата
	измененных	замененных	новых	аннулированных					