Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Космических и информационных технологий
институт

Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой

_____ О. В. Непомнящий
подпись инициалы, фамилия

«____»_____2020 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 Информатика и вычислительная техника
код и наименование специальности

Мобильное приложение с дополненной реальностью для изучения астрономии
тема

Руководитель	_____	доцент, канд. техн. наук	Постников А.И.
	подпись, дата	должность, учёная степень	инициалы, фамилия
Консультант	_____	старший преподаватель	И. В. Матковский
	подпись, дата	должность, учёная степень	инициалы, фамилия
Выпускник	_____		В. С. Свиридов
	подпись, дата		инициалы, фамилия
Нормоконтролер	_____	старший преподаватель	И. В. Матковский
	подпись, дата	должность, учёная степень	инициалы, фамилия

Красноярск 2020

Студенту Свиридову Владиславу Сергеевичу
фамилия, имя, отчество

Группа КИ16-06Б Направление (специальность) 09.03.01
номер код

Информатика и вычислительная техника
наименование

Тема выпускной квалификационной работы Мобильное приложение с дополненной реальностью для изучения астрономии

Утверждена приказом по университету №_____ от _____

Руководитель ВКР Постников А.И., канд. техн. наук, доцент, доцент кафедры ВТ
инициалы, фамилия, должность, учёное звание и место работы

Исходные данные для ВКР: Провести анализ применения дополненной реальности, исследовать известные решения, используемые для изучения астрономии. Разработать программное обеспечение для изучения астрономии с применением дополненной реальности.

Перечень разделов ВКР: Анализ существующих платформ для работы с дополненной реальностью, анализ основных алгоритмов, используемых в астрономии, проектирование и разработка программного обеспечения.

Перечень графического материала: презентация в формате Power Point

Руководитель ВКР _____ А. И. Постников
подпись инициалы, фамилия

Задание принял к исполнению _____ В. С. Свиридов
подпись инициалы, фамилия

«__»_____ 2020 г.

РЕФЕРАТ

Выпускная квалификационная работа по теме «Мобильное приложение с дополненной реальностью для изучения астрономии» содержит 63 страницы текстового документа, 2 таблицы, 40 иллюстраций, 6 формул, 30 использованных источников.

ДОПОЛНЕННАЯ РЕАЛЬНОСТЬ, ОБУЧЕНИЕ, VUFORIA, МОБИЛЬНЫЕ ПРИЛОЖЕНИЯ, АСТРОНОМИЯ, ТРЕХМЕРНАЯ ГРАФИКА, UNITY, HYG, VSOP, ШЕЙДЕРЫ, ECS.

Была рассмотрена технология дополненной реальности, а также способы ее применения. Был проведен анализ существующих приложений для изучения астрономии. На основе анализа были выделены основные требования к программному обеспечению. Проведено исследование основных алгоритмов, используемых в астрономии – VSOP87 для вычисления положения, каталог звезд HYG, алгоритм Floating Origin. По результатам проведенного анализа было принято решение использовать в программном комплексе Unity и Vuforia. Спроектировано и реализовано приложение для изучения астрономии.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Дополненная реальность (AR) — огромный прорыв в обучении и усвоении образовательного материала школьниками и студентами. Эффективность ее использования подтверждается различными тестами и экспериментами, которые показывают эффективные результаты.

Сегодня практически все люди обладают смартфонами. Большинство из них - активные пользователи, которые используют свои гаджеты для обучения, игр, для доступа к социальным платформам. Потенциал объединения смартфонов и дополненной реальности для образования велик, хотя его еще предстоит полностью раскрыть.

Дополненная реальность уже влияет на обычный процесс обучения [1]. В настоящее время мы можем найти отличные примеры использования дополненной реальности в образовании по всему миру. Способность соединять реальность и цифровой контент постоянно совершенствуется, открывая новые возможности для преподавателей и студентов. В дополненной реальности не меняется человеческое видение окружающего мира, а также его восприятие, происходит лишь дополнение реального мира искусственными элементами и новой информацией.

Одной из областей, где дополненная реальность может применяться является астрономия. Астрономия включает в себя наблюдения за небесными телами, такие как звезды, планеты, туманности, звездные скопления и галактики, а также явлениями, происходящими за пределами Земли.

Способы изучения астрономии весьма разнообразны, например, с помощью книг или наблюдения непосредственно с телескопа. Однако оба способа имеют недостатки, например информация в книгах представлена только в виде сухого текста и рисунков. Наблюдение с помощью телескопа потребует довольно больших затрат на приобретение оборудования. Поэтому

одним из наиболее перспективных и доступных способов изучения является использование дополненной реальности.

Таким образом, возможности технологии дополненной реальности позволяют сделать занятия более привлекательными, а информацию - более простой и понятной. Для изучения астрономии есть огромный потенциал – при помощи AR можно визуально воспроизвести процессы, которые трудно или почти невозможно воссоздать средствами реального мира.

1 Анализ предметной области

1.1 Обзор поставленной задачи

Целью текущей бакалаврской работы является разработка астрономического приложения с дополненной реальностью для образовательных целей.

Основные задачи данной работы:

- изучить технологию дополненной реальности;
- провести обзор возможностей существующих методов отображения дополненной реальности;
- рассмотреть существующие приложения для изучения астрономии;
- описать основные алгоритмы, архитектуру, а также обосновать выбор среды разработки;
- разработать и представить на рассмотрение программный продукт.

1.2 Применение дополненной реальности

Дополненная реальность расширяет наш физический мир, добавляя в него слой цифровой информации. Основное отличие от виртуальной реальности (VR) заключается в том, что дополненная не создает целые искусственные среды для замены реального окружения виртуальным. Напротив, AR ориентирован на взаимодействие виртуальных объектов с реальным миром, иными словами – виртуальные, трехмерные объекты добавляются в трехмерное пространство реального мира и воспринимаются как его часть.

Дополненная реальности используется в авиационной и военной технике [2]. В современных самолетах и вертолетах часто используется индикация на лобовом стекле или на шлеме пилота. Она позволяет пилоту получать наиболее

важную информацию прямо на фоне наблюдаемой им обстановки, не отвлекаясь на основную приборную панель.

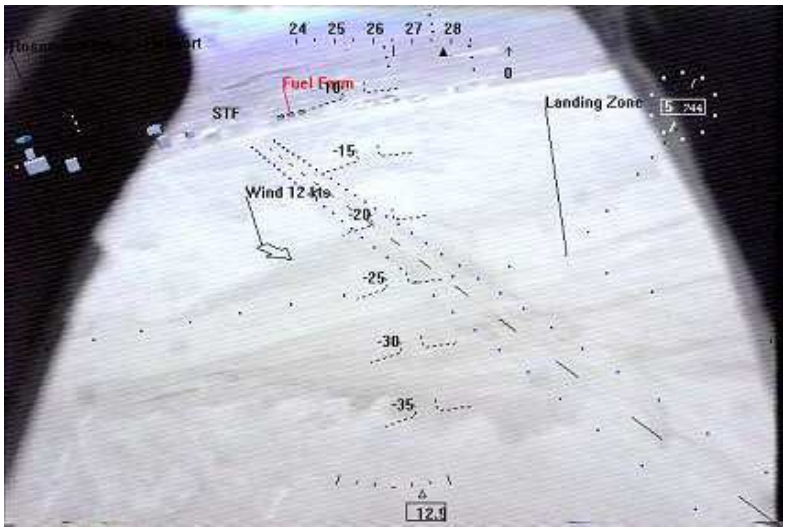


Рисунок 1.1 – отображение посадочной площадки на шлемах при помощи AR

Другим примером, где дополненная реальность оказывает влияние, является сфера медицинского образования [3]. При помощи дополненной реальности можно ускорить освоение материала, например изучать строение человеческого тела и протекающие в нем биологические процессы наглядным способом. Студенты изучают трехмерные модели системы кровообращения, нервной системы и т. д.

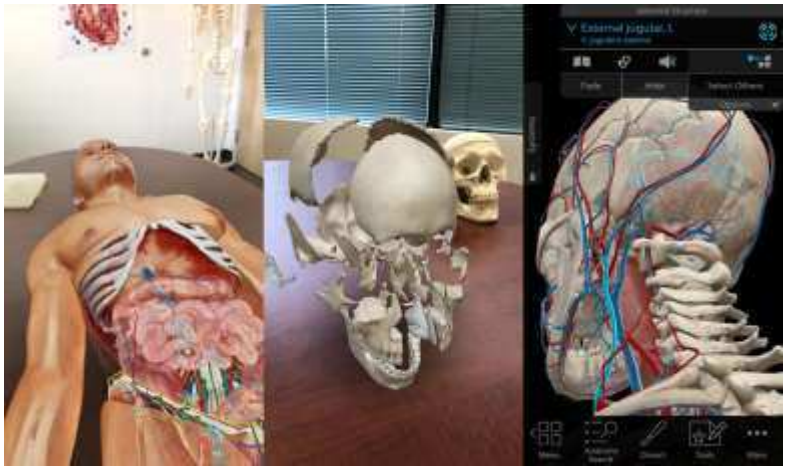


Рисунок 1.2 – изучение анатомии при помощи AR

Такие крупные компании как Google так же активно используют дополненную реальность. Приложение «Google Translate» в режиме реального времени переводит текст, на который указывает камера смартфона и показывает его поверх исходного изображения.

Рисунок 1.3 – дополненная реальность в приложении «Google Translate»

1.3 Обзор существующих приложений

1.3.1 Мобильное приложение «Star Walk 2»

«Star Walk 2» — приложение для изучения звезд, созвездий, планет и других небесных объектов [4]. Движение небесных тел рассчитывается в режиме реального времени, но можно отслеживать положение звезд, перемещение небесных объектов в прошлом и будущем. В приложении имеется справочник, содержащий информацию и интересные факты о различных небесных объектах, а также астрономические события и явления, актуальные для местоположения пользователя.

Рисунок 1.4 – карта звездного неба

Рисунок 1.5 – дополненная реальность в приложении «Star Walk 2»

Одна из основных преимуществ приложения – возможность просматривать звездное небо с помощью дополненной реальности [5]. Принцип работы заключается в том, что с помощью GPS и акселерометра, Star Walk определяет точное местоположение устройства, и соотносит то, что видит пользователь с картой звездного неба.

Так же в приложении имеется отдельная рубрика, где публикуются новости космоса, астрономии и космонавтики со всего мира.

В целом приложение отличается простым и удобным интерфейсом, наличием большого каталога небесных тел и возможностей визуализации с AR, однако является условно-бесплатным, имеет рекламу и встроенные покупки.

1.3.2 Мобильное приложение «Stellarium»

Рисунок 1.6 – приложение «Stellarium»

«Stellarium» — виртуальный планетарий с открытым исходным кодом, помимо мобильных платформ так же доступен для настольных ПК [6]. Определяет положения звезд, созвездий, планет, комет, спутник (в том числе МКС) и других объектов в реальном времени в небе.

Основные особенности:

- Просмотр модели ночного неба звезд и планет на любую дату, время и местоположение.
- Каталог из множества звезд, туманностей, галактик, звездных скоплений и других объектов глубокого неба.
- Возможность просмотра звездного неба в других регионах планеты
- Отслеживание искусственных спутников, в том числе Международной космической станции.

– Имитация ландшафта и атмосферы с реалистичным восходом, закатом и преломлением атмосферы.

1.3.3 Мобильное приложение «Star Chart»

«Start Chart» — это виртуальный планетарий с трехмерной графикой. Позволяет рассмотреть все видимые звёзды (более 6 000) и все созвездия [7]. Для каждого и каждой есть собственная информационная карта с детальными данными. В программе имеется голосовой поиск на английском языке.

Рисунок 1.7 – мобильное приложение «Star Chart»

Приложение Star Chart использует данные компаса, GPS, акселерометра и гироскопа и на их основе в режиме реального времени показывает вам расположение видимых звезд и планет на ночном небе.

В Star Chart входят:

– Более 120000 звезд
– Солнечная система в трехмерном виде

- Все 88 созвездий
- Каталог Мессье
- Метеоритные дожди
- Кометы
- Спутники

1.4 Сравнение приложений

Таблица 1.1 – сравнение существующих приложений

	Stellarium	Star Walk	Star Chart
Трехмерная модель планетария	-	-	+
Каталог звезд	+	+	+
Отображение небесных тел в реальном времени	+	+	+
Справочная информация	+	Ограничено в бесплатной версии	
Дополненная реальность	Только звездное небо		
Локализация на русский язык	-	Частично	
Модель распространения ПО	Платное	Условно-бесплатное	

1.5 Вывод

Существует множество приложений для изучения астрономии. Каждое из них, по сути, превращает смартфон в портативный планетарий. Функции дополненной реальности могут накладывать звездное небо на изображении с камеры. Независимо от того, является ли пользователь астрономом или просто заинтересован в том, что происходит "там", наши цифровые помощники открывают космос для индивидуального исследования.

Из этого следует, что разрабатываемый продукт должен обладать следующим функционалом:

- Наличие трехмерной модели солнечной системы (миниатюрная и реалистичная)
- Отображение положения планет в реальном времени
- Справочник с информацией о небесных телах
- Дополненная реальность

  Проецирование трехмерной модели небесного тела, что позволит рассматривать объект в реальности под разными углами
- «Путешествие во времени»: временная шкала на несколько тысяч лет, благодаря которой можно посмотреть на изменение расположения небесных тел в прошлом и будущем;
- Должно иметь локализацию на русском языке

Примечание [М. И. В.1]: Маркерные списки оформляются не так, проверьте СТО.

2. Проектирование

2.1 Проектирование архитектуры приложения

2.1.1 Архитектурный паттерн «Entity-Component-System»

Примечание [М. И. В.2]: Копипаста. Больше одного абзаца объяснения технологии не нужно. Дальше вставляете описание того, как это у вас

Паттерн «Entity-Component-System» — это шаблон проектирования, в основном используемый в играх, и обеспечивающий огромную гибкость в проектировании общей архитектуры программного обеспечения [8]. Паттерн позволяет уменьшить связность между различными частями проекта, что дает возможность добавлять, убирать, изменять логику независимо от других модулей и без потерь в производительности. Он базируется на следующих определениях:

1. **Сущность** — это любой объект в игре. Например, это может быть планета, событие с данными от одной системы к другой и т.п. Сущности сами по себе не имеют свойств и выступают контейнерами для компонентов. Иногда сущность называют «GameObject».
2. **Компоненты** — содержат только чистые данные, это является ключевой особенностью по расширению функционала в дальнейшем. Основная идея ECS - переход от стандартной ООП-идеи, где наследование заменяется композицией. Вместо иерархии наследования мы разделяем данные на изолированные блоки (Компоненты) и набираем из них нужную нам информацию об игровом объекте, добавляя компоненты на сущности.
3. **Системы** — блоки кода, который каким-либо образом обрабатывает требуемый набор компонентов на сущностях. Системы не содержат никаких локальных данных или ссылок на сущности или компоненты, они служат только для обработки потока отфильтрованных по их условиям сущностей и компонентов, висящих на этих сущностях.

Основное назначение ECS в приложении – логика работы и взаимодействия трехмерных объектов между собой. ECS ответственен за создание звезд, планет в сцене; вычисление положения, вращения; управление камерой и распознавание жестов и т.д.

В результате разделение логики и данных мы можем комбинировать любое сложное поведение набором простых систем, каждая из которых будет обрабатывать только нужные данные на сущностях. В результате мы можем добавлять и удалять системы обработки данных с минимальными изменениями в других системах, либо вообще без них.

Другим немаловажным аспектом использования ECS является возможность использования многопоточности в проекте, что особенно важно при вычислении положения небесных тел, т.к. алгоритмы требовательны к ресурсам [9].

2.1.2 Архитектурный паттерн MVVM

Model-View-ViewModel (MVVM) — это архитектурный шаблон проектирования, основная задача которого заключается в разделении визуализации пользовательского интерфейса (представление) от его логики или модели представления [10]. Итогом применения паттерна MVVM является функциональное разделение приложения на несколько слоев, которые проще разрабатывать и тестировать, а также в дальнейшем модифицировать и поддерживать.

Обычно приложение в игровых движках состоит из сложной иерархии с многочисленными скриптами (сценариями), прикрепленными и разбросанными по всему проекту. Вся эта логика должна взаимодействовать и работать вместе. Можно сказать, что MVVM помогает связывать и организовывать всю разрозненную логику, поэтому это способ формализации и структурирования

связей между классами, которые могут легко развиться случайным и непоследовательным образом.

В проекте MVVM планируется использовать для связывания логики работы модуля ECS и модуля данных, а также управление UI элементами, обработка нажатий в интерфейсе пользователя.

Основные экраны (и как вследствие ViewModel) в приложении:

1. Главный экран – основной экран, появляющийся при загрузке приложения. В нем происходит переход к другим экранам
2. Настройки – экран с настройками приложениям (звук, графика и т.д.)
3. Планетарий – экран с отображением трехмерной модели Солнечной системы
4. Звездное небо – экран, отображающий звезды с позиции наблюдателя с Земли.
5. AR – на этом экране отображается превью с камеры устройства, а также показывается каталог астрономических объектов, которых можно посмотреть через дополненную реальность

2.1.3 Паттерн «Репозиторий» для работы с данными

Примечание [М. И. В.3]: Копипаста. Больше одного абзаца объяснения технологии не нужно. Дальше вставляете описание того, как это у вас

Одним из наиболее часто используемых паттернов при работе с данными является паттерн «Репозиторий». Репозиторий позволяет абстрагироваться от конкретной базой данных, с которой работает программа, и является промежуточным звеном между классами, непосредственно взаимодействующими с данными, и остальными модулями программы [11].

Основные преимущества паттерна:

- Независимость бизнес-логики от способа хранения. При использовании репозитория, логика приложения связана только с коллекциями объектов.
- Работая через интерфейсы, можно создать несколько реализаций репозитория. Это помогает при тестировании (можно передавать

заглушку репозитория при тестировании логики) и при изменении способа хранения данных.

В приложении данный паттерн потребуется для обертки над SQL подключением к базе данных. Это позволит делать произвольные выборки из каталога звезд, а также репозиторий необходим чтобы выгружать данные о планетах – их описание, характеристики и т.д.

2.2 Описание предложенных методов и алгоритмов

2.2.1 Каталог звезд HYG

Примечание [М. И. В.4]: Слишком подробно. Сократите раза в два.

Одним из наиболее полных и точных каталогов звезд является HYG [13], но де-факто является сборником данных из трех других каталогов:

1. Каталог **Hipparcos** — это самая большая коллекция высокоточных данных о положении, расстояниях и собственных движениях звёзд.
2. Каталог **Yale Bright Star** (Йельских ярких звезд) содержит базовые данные практически обо всех звездах, видных невооруженным глазом
3. Каталог ближайших звезд **Gliese** (3-е издание) является наиболее полным каталогом ближайших звезд, находящихся в пределах 75 световых лет от Солнца.

Каталог содержит практически все данные о каждой звезде: идентификаторы в других каталогах, название, координаты (в экваториальной системе координат), расстояние в парсеках, видимая звездная величина, цвет, созвездие, спектр [14]. Данные положения астрономических объектов хранятся в экваториальной системе координат [15]. Преобразование в прямоугольные координаты происходит по следующим формулам:

R — расстояние до звезды (парсек),

D — склонение звезды (эпоха J2000),

A — прямое восхождение звезды (эпоха J2000)

$$\mathrm{X} = R * \cos\left(\frac{\pi}{180} * D\right) * \cos(\frac{\pi}{180} * (A * 15))$$

$$\mathrm{Y} = R * \cos\left(\frac{\pi}{180} * D\right) * sin(\frac{\pi}{180} * (A * 15))$$

$$\mathrm{Z} = R * sin\left(\frac{\pi}{180} * D\right)$$

В заключение можно отметить, что HYG является наиболее удобным и полным источником для разработки приложения. Каталог распространяется по свободной лицензии и представляет собой CSV файл, который можно импортировать в большинство программ баз данных и электронных таблиц.

2.2.2 Вычисление положения планет при помощи VSOP87

Для того чтобы точно рассчитывать положения небесных тел, нужно учитывать как можно больше возмущающих факторов [16]. Аналитического решения для системы более двух нет (исключение — частные решения Лагранжа), поэтому уравнения движения тел решают численно, но даже с учётом относительно новых методов численного интегрирования (таких, как метод Эверхарта) процедура эта очень затратна, и если достаточно точное решение на небольшой промежуток времени под силу среднестатистическому компьютеру, то интегрирование на глобальных временных диапазонах — сложная и трудоёмкая задача, поэтому астрофизики проблему решили следующим образом: нашли положения небесных тел при помощи интегрирования и аппроксимировали эти положения функцией, а на выходе

получили коэффициенты для этой функции. Именно набор этих коэффициентов и называют, как правило, эфемеридной теорией.

Теория VSOP87, разработанная французскими астрономами Пьером Бретаньоном и Жераром Франком, вероятно, является одной из наиболее используемых и точных алгоритмов, доступных сегодня, для определения положения планет без использования интерполяции [17]. Теория состоит из большого числа периодических слагаемых, которые затем специальным образом складываются вместе для создания трехмерных гелиоцентрических координат любой планеты в любой момент времени на протяжении тысячелетий в будущее и прошлое [18]. Эти координаты затем можно преобразовать в геоцентрические координаты, которые можно использовать для отображения их положения с точки зрения Земли.

Существуют разные версии теории VSOP87, большинство используют Солнце как точку отсчета, кроме VSOP87E, который сосредоточен на центре масс Солнечной системы. Для определения положения планет в небе наиболее удобно использовать VSOP87C с центром относительно Солнца, который создает прямоугольные координаты, которые легко преобразовать.

Данные для каждого небесного тела разделены на три секции, обозначенные тремя разными буквами: X, Y и Z. Каждая координата из этих секций вычисляется индивидуально. В каждом разделе данные дополнительно подразделяются на отдельные серии рядов, называемых X_0, X_1, X_2 и т.д. Каждый из этих рядов состоит из списка наборов из 3 коэффициентов (3 крайних справа числа в данных), обычно называемые A, B и C. Каждый набор коэффициентов используется для вычисления одного члена ряда с использованием определенной формулы. Все ряды затем суммируются вместе. Эти суммы ряда затем помещаются в другую формулу для получения фактического значения координаты. Затем это повторяется для каждой координаты планеты, чтобы получить гелиоцентрическое положение данных.

Математически это просто очень большой набор параметрических уравнений, являющихся функциями от времени, хотя обычно он так не записывается, из-за проблем с читаемостью, связанных с огромными размерами.

Например, вычисление X координаты любого небесного тела выглядит следующим образом:

$$X(T) = X_1(T) + X_2(T) + X_3(T) + \cdots = \sum_{i=1}^{n} X_i(T)$$

$$X_i(T) = \sum_{j=1}^{k} A_j * \cos(B_j + C_j * T)$$

Здесь A, B и C - коэффициенты выше, а T – число юлианского тысячелетия после эпохи J2000. Его можно найти по следующей формуле:

$$T = (JD - 2451545) / 365250$$

Под JD подразумевается текущая юлианская дата, или, иными словами, количество суток, прошедших после полудня 1 января 4713 года до нашей эры по григорианскому календарю.

VSOP87 чисто эфемеридная теория и как следствие, занимает крайне малые объемы, чего не скажешь, например о DE, которая является полуаналитической теорией, для которой нужно хранить подборки коэффициентов приличного размера. К тому же для настоящего времени оба метода дают достаточно приличной точности результаты, чтобы фактор размера вышел на первое место. К тому же VSOP не стоит на месте и есть более поздние и более точные реализации, например VSOP2013.

2.2.3 Рендеринг на больших масштабах. Алгоритм Floating Origin

Большинство 3D движков хранят координаты объектов используя трехмерный вектор, представленный числами с одинарной точностью (float), где вектор (0,0,0) называется началом сцены [19]. Эта система прекрасно работает для большинства приложений или игр, которые работают на небольших масштабах, но космические симуляторы часто должны работать на протяжении тысяч или миллионов километров, что числа с одинарной точностью не способны сделать.

Проблема заключается в том, что значения с плавающей точкой теряют десятичную точность по мере того, как они становятся больше (или меньше для отрицательных значений), что означает, что если вы находитесь на расстоянии 1 млн. метров от центра, точность координат и камеры будут очень низкими, создавая графические артефакты, проблемы физики и так далее [20]. Таким образом, значения позиции объектов и камеры должны быть как можно ближе к исходной точке (0,0,0), чтобы точность чисел с плавающей запятой оставалась высокой.

Одно из решений состоит в том, чтобы вместо перемещения удерживать камеру неподвижной в исходном положении в начале координат и перемещать только объекты в сцене на расстояние, на которое обычно перемещалась бы камера. Это прекрасно работает, однако, это может привести к проблемам с производительностью, потому что движущиеся объекты иногда могут быть вычислительно дорогими. Это также может усложнить написание кода и может вызвать проблемы с существующими компонентами и кодом, которые ожидают, что камера может двигаться.

Решением является алгоритм под названием «Floating Origin». Вместо того, чтобы постоянно держать камеру в начале сцены, она возвращается к точке (0,0,0), когда уходит слишком далеко [21]. Этот подход дает хорошую производительность, потому что объектам нужно только время от времени

сдвигать свои позиции. Алгоритм также поддерживает хорошую точность рендеринга, потому что значения всегда находятся около начала координат.

Дополнением к алгоритму служит сохранение реальных координат небесных тел в числах с более высокой точностью и разбиение всего пространства на полупространства: для этого используются числа с двойной точностью (double) для хранения локальных координат в текущем полупространстве и 64-битных целочисленных переменных (long) для хранения глобальных координат, это позволит позиционировать объекты в любом месте с высокой точностью.

Локальные значения эквивалентны мировым единицам (метрам), а глобальные значения (одного полупространства) эквивалентны 50 миллионам метров каждая. Каждый раз, когда локальное значение превышает этот порог, оно сбрасывается в 0 и увеличивает глобальное значение.

При использовании такого подхода, можно размещать объекты на протяжении десятков миллиардов световых лет. Точности хватит на сохранение координат объекта на $\pm 461168601842738790400000000$ метров по каждой оси. Это эквивалентно ± 48745559677 световых лет, или 48,7 млрд световых лет, что превышает радиус наблюдаемой вселенной (составляет 46,6 млрд световых лет) на ~2.1 млрд световых лет.

2.3 Выбор среды разработки

Для создания и воспроизведения учебного контента необходима программная платформа, обладающая возможностью создания визуальных и интерактивных сцен. Особенно это необходимо в данном проекте, так как AR дополняет реальность, которая является сама по себе трёхмерной, поэтому нужно использовать инструмент, который позволяет обрабатывать и работать с трёхмерными объектами. Для этой цели может послужить любой трёхмерный

или игровой движок, который позволяет создавать подобную функциональность.

Игровой движок — это программное обеспечение, предоставляющее набор необходимых функций для быстрого и эффективного создания трехмерных приложений [22]. Эти программные инструменты создают слои абстракции поверх самых распространенных задач при разработке видеоигр.

Игровые движки обеспечивают огромные преимущества в эффективности, уменьшая порог вхождения и время, необходимые для создания различных приложений. Они могут быть минимальными по своей предустановленной функциональности или полнофункциональными, что позволяет разработчикам полностью сосредоточиться на создании продукта.

В целом, трехмерные движки поддерживают и сводят вместе ряд ключевых аспектов:

- освещение, аудио, спецэффекты
- симуляция физики; анимация, интерактивность
- логика взаимодействия объектов;
- создания визуальных сцен посредством взаимодействия с трехмерными объектами;
- поддержку одного или нескольких языков программирования для реализации взаимодействия с объектами сцены;
- отлаживать и оптимизировать свои приложения для выбранных платформ.

2.3.1 Интегрированная среда разработки «Unreal Engine 4»

Unreal Engine — игровой движок от компании Epic Games. Предоставляет профессиональный набор инструментов и технологий для проектов на различных платформах [24]. Активно используется различными компаниями для высокобюджетных проектов, где нужно добиться максимального качества.

Из основных особенностей Unreal Engine 4 можно отметить: высококачественный рендеринг, визуальные эффекты, визуальное программирование, анимацию, встроенные инструменты для создания ландшафта, горячая перезагрузка при изменении кода, искусственный интеллект, аудио, сетевые функции. Имеет огромное сообщество и отличную документацию.

Рисунок 2.1 – среда разработки «Unreal Engine 4»

Разработка сценариев взаимодействия интерактивных объектов происходит при помощи использования языка программирования C++, так же стоит отметить наличие визуального языка программирования, благодаря чему возможно создание визуальных схем, при помощи которых программируются сценарии, задаются различные события и их взаимосвязи.

Основные плюсы:

- Богатый инструментарий: графика, визуальное программирование
- Открытый исходный код
- Гибкое лицензирование

Однако данный среда имеет свои недостатки:

- Более высокий порог вхождения, требует подготовки
- Требовательность среды разработки к ресурсам

2.3.2 Интегрированная среда разработки «Unity»

Unity – это кроссплатформенный игровой движок, в основном используемый для разработки видеоигр и симуляций для ПК, консолей, мобильных устройств. Используется разработчиками во всем мире, поэтому имеет активное сообщество. Предоставляет мощный визуальный интерфейс для создания трехмерный приложений. Главным его достоинством является возможность создания программ, одинаково работающих на Microsoft Windows, OS X, Linux, iOS, Android и прочих ОС [25].

Проект в Unity делится на сцены – отдельные файлы, содержащие наборы объектов, сценариев и настроек. Объекты в свою очередь содержат ряд компонентов, с которыми взаимодействуют сценарии. Среда поддерживает физику твердых тел и ткани, присутствует поддержка наследования объектов: дочерние объекты повторяют параметры позиции, вращения и масштаба родительского объекта.

Рисунок 2.2 – главное окно редактора

Разработка сценариев взаимодействия объектов возможна при помощи языков программирования C# и Javascript. Среда разработки Unity поддерживает более двадцати программных платформ, среди которых значатся: Linux, Microsoft Windows, OSX, iOS, Anrdoid и др. Все файлы проекта могут быть упакованы и переданы другим разработчикам.

Unity предлагает встроенные инструменты для разработки AR-приложений с взаимодействием интерфейса и реального мира. Поддерживается большое количество устройств и библиотек для работы с дополненной реальности- Vuforia, ARCore, ARKit, HoloLens и т.д.

2.4 Выбор инструментов для работы с дополненной реальностью

На данный момент рынок AR имеет довольно много платформ для создания приложений с AR для телефонов и планшетов на операционных системах iOS и Android. В данной части рассмотрим библиотеки двух самых популярных решений.

2.4.1 Qualcomm Vuforia

Vuforia – это платформа дополненной реальности, включающая в себя библиотеки и комплект средств разработки (SDK) для создания приложений дополненной реальности [26]. Vuforia использует технологию компьютерного зрения для распознавания и отслеживания изображений (называемых метками) в режиме реального времени. Возможность отслеживания меток позволяет разработчикам позиционировать виртуальные объекты относительно объектов реального мира, когда они просматриваются через камеру мобильного устройства.

Для того чтобы виртуальный объект появился в реальности, необходимо зацепиться за какой-либо предмет (метку) в реальном мире. Для этой цели в Vuforia используются метки. Метка – это некий реальный объект, зная который приложение может расставить виртуальные объекты в нужных местах и соответствующих пропорциях.

Vuforia предоставляет несколько типов меток:

- **Image targets**. Первый вид мишеней, представляющий собой обычную картинку или фотографию. Картинка выполняет роль двумерного штрих-кода, только без черно-белых регионов. По ней мы можем определить, какая именно картинка попала в объектив камеры, а также её расположение в пространстве и масштаб.
- **3D targets**. Это мишени в виде прямоугольных параллелепипедов (включая куб). Такой мишенью может служить любой предмет из реального мира, например спичечный коробок.
- **Text Targets**. В библиотеку встроено распознание текста, поэтому любое слово или их сочетание может являться мишенью.
- **Облачные метки**. Метки хранятся и распознаются в облаке. Одна облачная база данных может содержать до 1 миллиона целевых изображений.

Vuforia предоставляет SDK для приложений на языках C++, Java, и .NET через интеграцию с игровым движком Unity. Таким образом SDK поддерживает разработку нативных AR-приложений для iOS и Android, в то же время предполагая разработку в Unity, результаты которой могут быть легко перенесены на обе платформы.

2.4.2 Google ARCore

ARCore — это инструментарий от Google для создания приложений дополненной реальности. Библиотека поддерживает разработку для платформ

Android и iOS. Кроме того, эта библиотека не требует лицензирования, может быть внедрена свободно в любое приложение. ARCore использует три ключевые технологии для работы с дополненной реальностью:

- **Отслеживание положения**. Позволяет смартфону при помощи встроенных датчиков понять положение в реальном мире.
- **Анализ окружающей среды**. Позволяет смартфону определять размер и местоположение всех типов поверхностей (вертикальных, горизонтальных и угловых)
- **Оценка освещённости**. Позволяет оценить текущие условия освещения окружающей среды.

Принцип работы приложений, использующих ARCore, основан на таких технологиях, как GPS, акселерометр, гироскоп и более сложные алгоритмы обработки изображений, для размещения виртуальных объектов или информации в окружающей среде [27]. ARCore также использует технологию отслеживания движения для определения того, как некоторые объекты движутся, учитывая движения камеры.

ARCore основан на алгоритме SLAM, который используется в автономных системах для построения карты в неизвестном пространстве или для обновления карты в заранее известном пространстве с одновременным контролем текущего местоположения и пройденного пути.

Google предоставляет ARCore SDK для большинства платформ, а так же легко интегрируется с фреймворками и игровыми движками.

2.4.3 Сравнение технологий

Основным различием между Vuforia и ARCore является принцип работы с дополненной реальностью. В Vuforia работа с дополненной реальностью осуществляется с помощью технологии отслеживания предметов. Так, чтобы отобразить какой-либо 3D объект нужно навести камеру на target метку, при

этом, если упустить эту метку из поля зрения камеры, то объект пропадет. Технология отслеживания предметов позволяет создавать живые target метки, которые можно крутить под разным углом при этом 3D объект, прикрепленный к такой метке, также меняет свое положение в пространстве.

В ARCore основным принципом является анализ и построение виртуальной карты пространства. Эта технология, основанная на алгоритме SLAM, который строит виртуальную модель пространства и связывает ее с реальным миром, что позволяет устанавливать 3D объекты на произвольный участок виртуальной карты, при этом объект можно потерять из виду и вернуться к нему позже. На данный момент ARCore частично поддерживает технологию живых виртуальных меток.

2.5 Вывод

В данной главе проведено проектирование разрабатываемого продукта. Был определен перечень технологий для разработки программы. Также были выбраны и описаны алгоритмы для решения поставленной задачи.

Среди рассмотренных интегрированных систем разработки выбран Unity из-за возможности расширения функционала среды посредством применения разработанных скриптов. Большое сообщество Unity предоставляет актуальные инструменты для разработки, а поддержка языка программирования C# гарантирует совместимость с разрабатываемыми приложениями на разных платформах. В Unity интегрирована возможность работы с Vuforia, поэтому Vuforia выбрана в качестве библиотеки для работы с дополненной реальностью.

3. Разработка мобильного приложения

3.1 Основные модули приложения

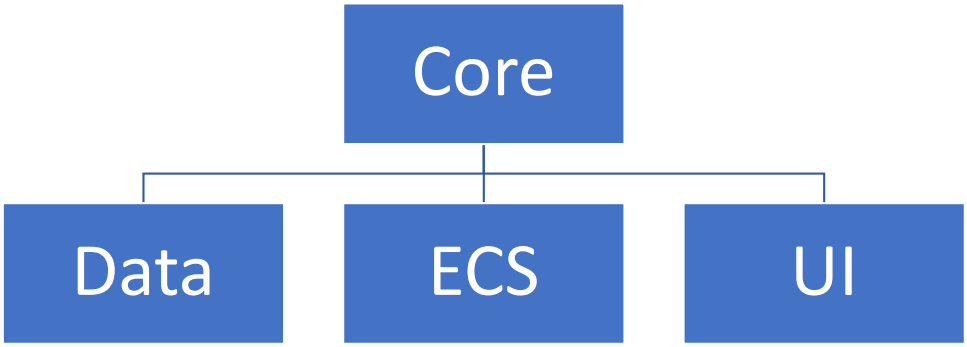


Рисунок 3.1 – основные модули приложения

Модули приложения включают в себя:

- Core – ядро приложения. Содержит в себе базовые классы, утилиты, константы и т.д.
- Data – модуль для работы с данными. Необходим преимущественно для погрузки данных из БД.
- ECS – модуль, содержащий основную логику работы всех объектов на сцене.
- UI – в этом модуле находится описание интерфейса и логика его работы. Данный модуль инкапсулирует логику работу остальных модулей программы

3.2 Разработка ядра приложения

В ядре приложения содержатся основные вспомогательные классы; константы для перевода из разных метрических систем. Так же представлены инструменты для астрономических вычислений, например преобразования индекса цвета в RGB цвет, перевод даты и времени григорианского календаря в юлианский календарь, конвертация между разными системами координат и т.д. Дополнительно ядро реализует логику загрузки и сохранения настроек через статический класс Settings, используя файлы в основе.

Одно из больших частей ядра приложения – вычисления, связанные с VSOP. Для каждой планеты имеется отдельный класс, где хранятся функции и константы, позволяющие получить ее местоположение, наклон и т.д. Все это может вызываться по отдельности, но для удобства совмещено в классе VSOP87, где можно получать данные требуемой планеты по индексу.

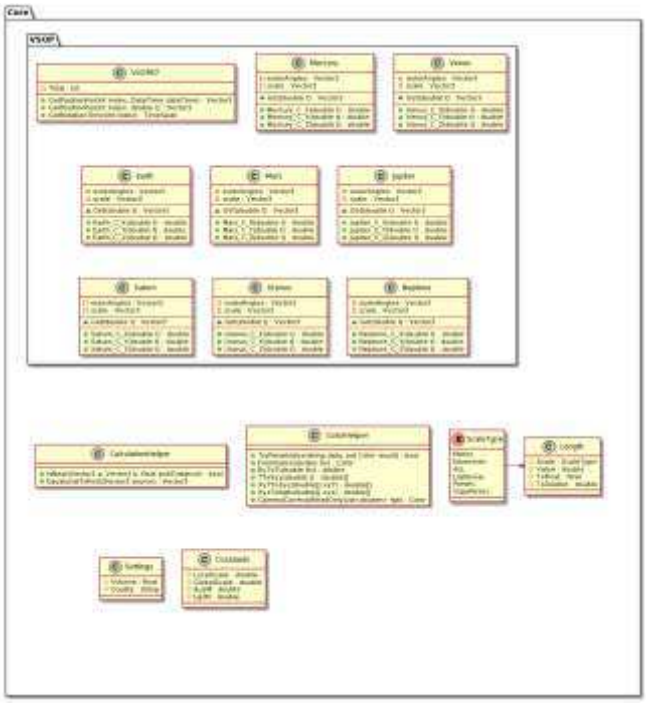


Рисунок 3.2 – диаграмма классов ядра приложения

3.3 Разработка модуля данных

Для хранения каталога астрономических объектов, справочной информации, кэширования необходима полноценная, но в тоже время компактная база данных. Обычно для мобильных приложений используется БД под названием SQLite.

SQLite — это встраиваемая кроссплатформенная БД, которая поддерживает достаточно полный набор команд SQL и доступна в исходных кодах на языке C, но также доступны обертки для других языков [28]. SQLite не использует парадигму клиент-сервер, то есть движок SQLite не является отдельно работающим процессом, с которым взаимодействует программа, а представляет собой библиотеку, с которой программа компонуется, и движок становится составной частью программы.

Поскольку разработка приложения происходит в Unity, то для работы с базой данных понадобится обертка на языке C#. В менеджере пакетов Nuget существует библиотека под названием sqlite-net-pcl, которая используется в проекте.

Рисунок 3.3 – SQLite в Nuget

Работа с базой данных происходит через ранее описанный паттерн репозиторий. Его реализация в общем виде будет более полезной, т.к. создание класса репозитория для каждого типа сущности может привести к большому

количеству избыточного кода, поэтому правильной минимизацией избыточного кода является использование общего (generic класс) репозитория.

Однако прежде, чем приступать к реализации репозитория, необходимо создать интерфейс IEntity, который будет базовым для всех остальных сущностей. В данном интерфейс присутствует лишь одно поле – универсальный идентификатор (ID) сущности. Поле ID помечено атрибутами, поэтому является первичным ключом в таблице и его значения генерируется автоматически.

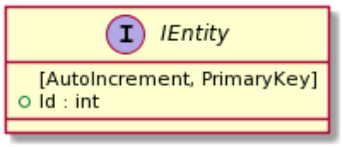


Рисунок 3.4 – базовый интерфейс для всех сущностей

В интерфейсе репозитория определены основные методы для CRUD (Create, Read, Update, Delete) операций. Интерфейс, как и его реализация, так же являются generic классами.

Существует так же абстрактная реализация репозитория (класс BaseRepository) для базы данных SQLite. Реализация инкапсулирует собой класс SQLiteConnection, который предоставляет собой подключение к конкретному файлу БД.

Таблица 3.1 – методы репозитория

Метод	**Использование**
IQueryable<T> Query()	Представляет запрос к таблице, благодаря которому можно выбирать и выгружать элементы без буферизации
IEnumerable<T> GetAll()	Возвращает все сущности из определенной таблице
void Add(T item)	Добавление элемента
void Remove(T item)	Удаление элемента

void Remove (Expression<Func<T, bool>> predicate)	Удаление по предикату
bool Exists(T item)	Проверка наличия элемента
bool Exists(Expression<Func<T, bool>> predicate)	Проверка наличия элемента по предикату

В основном базе данных имеется две таблицы – в одной хранятся объекты каталога HYG с информацией о звездах, в другой информация о планетах Солнечной системы. На рисунке 3.5 указана общая структура модуля работы с данными.

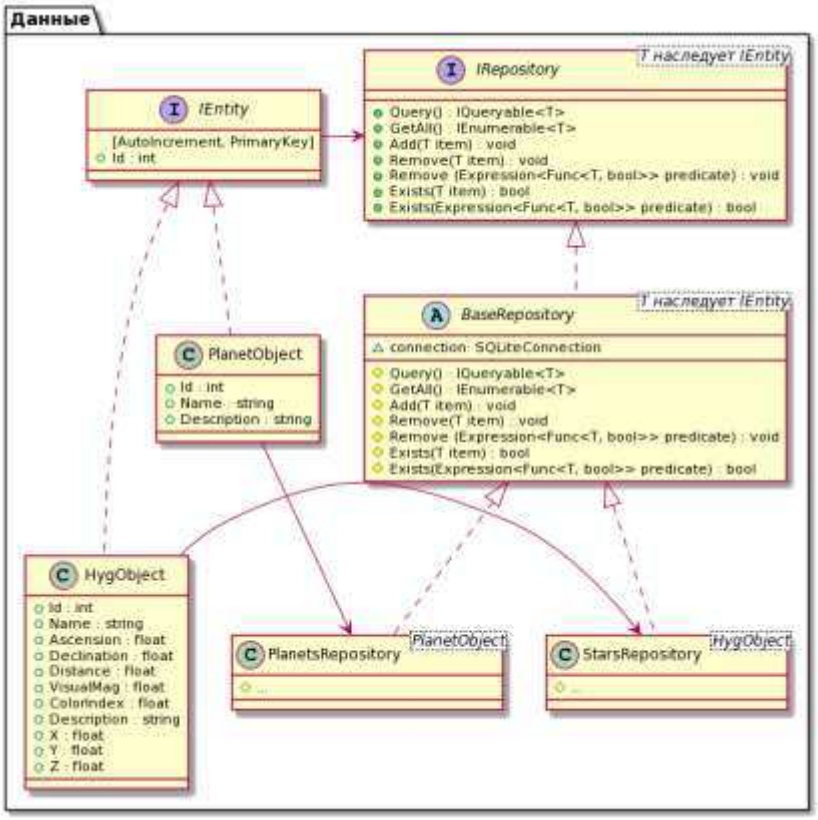


Рисунок 3.5 – модуль для работы с данными

3.4 Разработка модуля ECS

3.4.1 Общее описание модуля

Основные задачи модуля ECS:

- Обработка пользовательского ввода
- Создание планет, звезд и т.д. в сцене
- Расчет положения планет, звезд
- Загрузка уровней

Разобьем задачу на несколько подзадач, разделив обязанности между каждой системой:

- UserInputSystem — ввод пользователя (мышка, тач-скрин и т.д.).
- SpawnSystem — размещение трехмерных моделей небесных тел
- MoveSystem — перемещение небесных тел.
- ChangeLevelSystem — загрузка сцену по требованию.
- CameraMoveSystem – управление камерой

Определяем следующие компоненты:

- UserInputEvent — событие о наличии пользовательского ввода с данными о нем, т.к. события тоже компоненты
- PlanetSystemSpawnEvent — событие о том что необходимо создание планетной системы в сцене. Хранятся данные о звезде, планетах, данные об их орбитах и т.д.
- CelestialBodyData — данные о конкретном небесном теле.
- LoadChangelEvent — событие о необходимости загрузки новой сцены.

Примерный алгоритм работы:

- Загружается сцена и инициализируются все системы в указанной выше последовательности. Порядок обработки систем можно контролировать без сложных манипуляций, что является огромным плюсом
- Создается PlanetSystemSpawnEvent
- Запускается основной цикл обработки систем в методе MonoBehaviour.Update.
- UserInputSystem проверяет пользовательский ввод через стандартное Unity API и создает новую сущность с компонентом UserInputEvent и данными о вводе
- SpawnSystem – проверяет наличие PlanetSystemSpawnEvent, после чего создает сущности с CelestialBodyData
- MoveSystem проверяет — есть ли сущности с компонентом CelestialBodyData. Если есть – то для них высчитывается положение, угол наклона и т.д. для указанного временного промежутка.
- CameraMoveSystem проверяет — есть ли сущности с компонентом UserInputEvent. Если пользовательский ввод есть, обрабатываем его, перемещаем камеру
- LevelChangeSystem проверяет — есть ли сущности с компонентом LoadChangelEvent и выполняет загрузку новой сцены при наличии. Все сущности с таким компонентом удаляются перед этим.
- Повтор основного цикла обработки систем.

Данный подход может выглядеть как чрезмерное усложнение кода по сравнению со стандартным «MonoBehaviour» классом, где данные и логика объединены, но ECS имеет больше плюсов:

- Работает очень быстро, т.к. не вызывает работы Garbage Collector, все объекты уходят во внутренний пул для последующего переиспользования [29].

– Позволяет разделить ввод с устройства и его обработку. Например, мы можем менять не только ввод с клавиатуры на мышь, а также с тачскрина, при этом остальной код не ломается.
– Дает возможность расширять поведение новыми способами без серьезного рефакторинга текущего кода

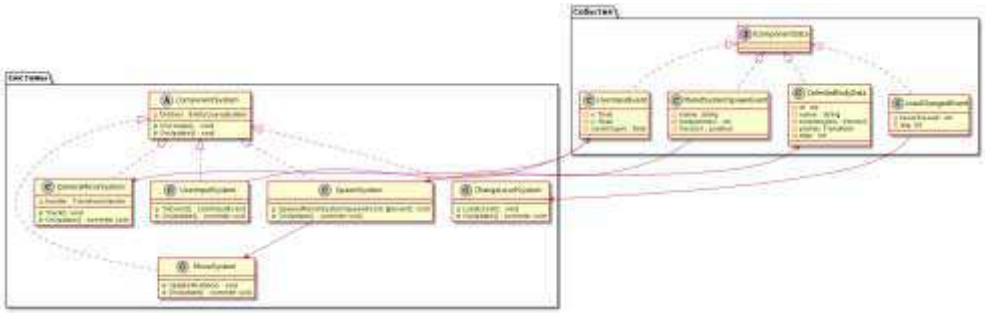


Рисунок 3.6 – диаграмма основных классов модуля ECS

3.4.2 Отрисовка орбит

Для отрисовки линий используется компонент LineRenderer. Он берет массив из двух или более точек в трехмерном пространстве и рисует прямую линию между каждой из них. Используя LineRenderer, можно нарисовать что угодно, от простой прямой линии до сложной спирали. Стоить отметить, что линия всегда является непрерывной, поэтому для создания отдельных линий необходимо каждый раз создавать новый LineRenderer.

Для того чтобы отрисовать орбиту для конкретной планеты, необходимо получить массив точек (Vector3) используя при этом VSOP87, т.е. просчитать точки орбиты небесного тела в трехмерном пространстве за один сидерический год. Зная период оборота планет вокруг Солнца, разделим его на некоторое количество частей N (например, 50), и посчитаем для каждого части положение небесного тела в определенной период времени, сформировав окончательный

результат в массив. Чем больше число N, тем более детализированной получается траектория движения планет. Для того чтобы сделать линию траектории движения планет более компактной и круглой, необходимо нормализовать каждый полученный Vector3.

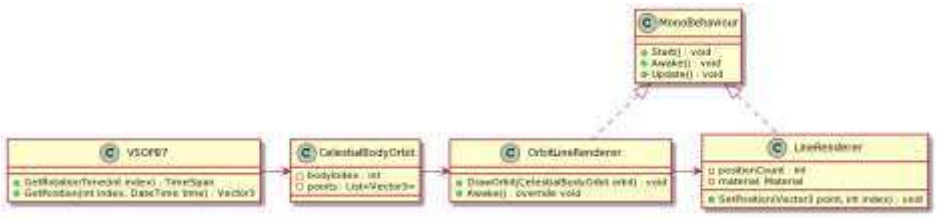


Рисунок 3.7 – диаграмма классов для отрисовки орбит

3.4.3 Реализация Floating Origin

Как и описывалось ранее, координаты в Unity имеют тип данных с плавающей запятой, поэтому пространство, в котором симуляции происходит без ошибок ограничено. Алгоритм Floating Origin всегда отслеживает положение камеры в виртуальной сцене, поэтому, когда расстояние между началом сцены и камерой переходит определенный порог, то все объекты в сцене смещаются. Порог определяет расстояние, на котором происходит массовое перемещение объектов. Это не легкая операция, поэтому чем реже это происходит, тем лучше. По умолчанию порог имеет значение около 5000 единиц, что примерно равно 5 км.

В ECS реализация выглядит следующим образом: когда камера переходит порог, класс FloatingOriginTracker создает событие FloatingOriginEvent, которое наследуется от IComponentData и содержит в себе Vector3 со смещением, которое необходимо применить к объектам. Это событие обрабатывается классом FloatingOriginSystem. FloatingOriginSystem содержит логику смещения, при помощи свойства Entities мы можем получить список сущностей,

содержащих компонент Translation, т.е. координаты объекта. Получая этот массив компонентов, мы применяем смещение ко всем объектам.

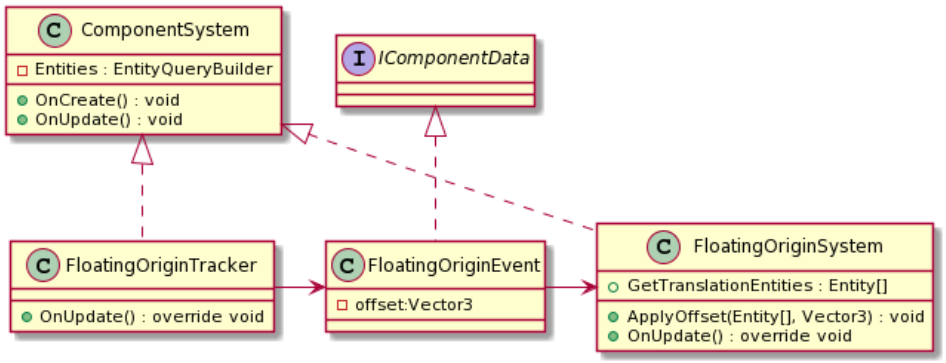


Рисунок 3.8 – диаграмма классов для Floating Origin

3.5 Разработка модуля UI

Модуль интерфейса является одним из основных модулей в программе. В Unity существует встроенная система для создания UI, позволяющая легко создавать интерфейсы, которые позиционируются среди других 2D или 3D объектов.

Одним из ключевых объектов системы UI является Canvas [30]. Canvas – область, где находятся все элементы пользовательского интерфейса. Сам по себе Canvas это объект с компонентом, и все элементы пользовательского интерфейса должны быть дочерними. Элементы пользовательского интерфейса в Canvas отображаются в том же порядке, в котором они находятся в иерархии объектов.

На рисунке 3.9 изображено приблизительное устройство иерархии элементов интерфейса в приложении.

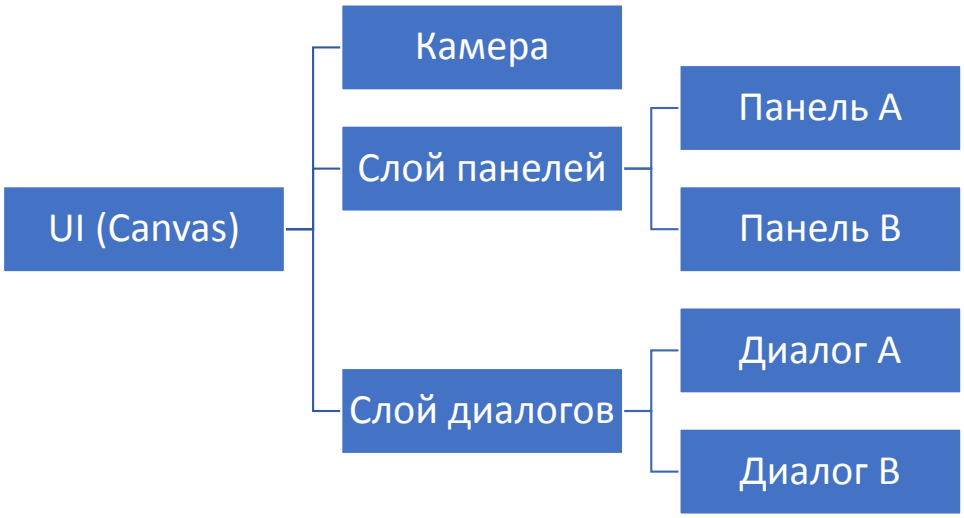
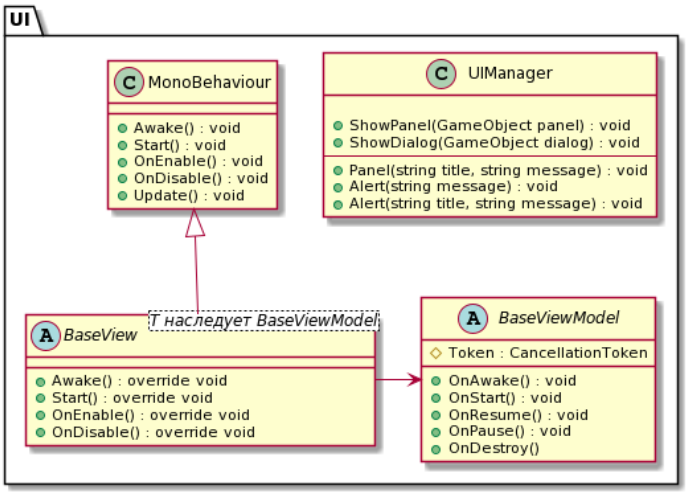


Рисунок 3.9 – иерархия UI элементов

Модуль интерфейса разделён на две части: базовая часть и ViewModel конкретных экранов.

В базовой части содержатся абстрактные классы, например BaseViewModel, где представлены методы для инициализации (например подгрузки данных), управления подписками и высвобождения ресурсов (OnResume/OnPause).

Абстрактный класс BaseView является «мостом» между ViewModel и Unity. BaseView создает экземпляр ViewModel и вызывает методы жизненного цикла на ViewModel. На рисунке 3.11 показана диаграмма базовых классов.



Рисунке 3.10 – базовые классы UI модуля

Класс UIManager отвечает за управление панелями и диалогами, логика его работы представлена на рисунке 3.12.

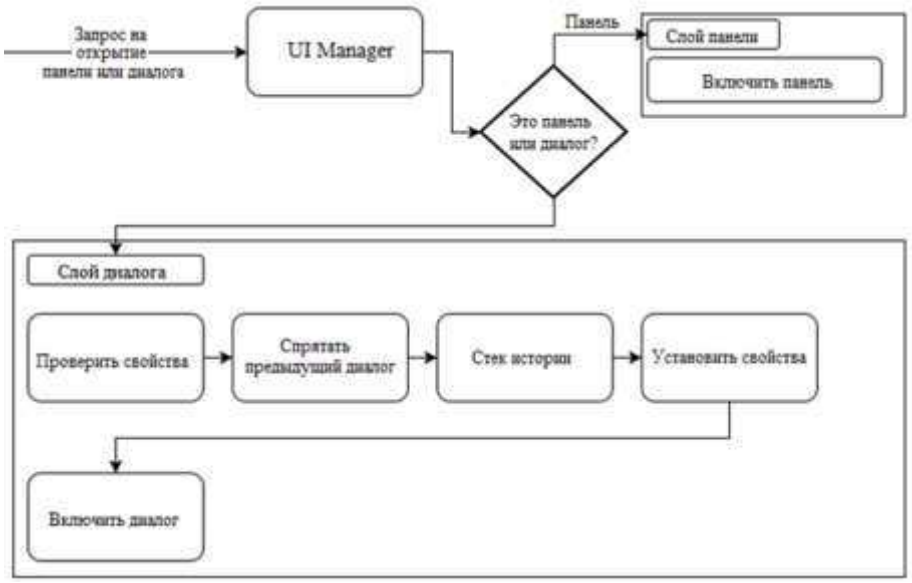


Рисунок 3.11 – схема работы UIManager

Основные ViewModel в приложении:

- MainViewModel отвечает за главный экран, содержит логику перехода к остальным экранам
- SettingsViewModel содержит собой логику работы экрана настроек. Сохраняет все настройки через класс Settings.
- PlanetaryViewModel – ViewModel экрана с отображением модели планетария. Связывает между собой слой данных и ECS. При инициализации делается небольшая выгрузка звезд HYG (около 10000) и планет через репозиторий, после чего передает на обработку модулю ECS. Еще обрабатывает логику нажатий на кнопки и частично трехмерные объекты.
- StarsViewModel – экран с отображением звездного неба.
- ARViewModel отвечает за экран работы с дополненной реальностью. Содержит каталог заранее заготовленных объектов, которые размещаются после того, как пользователь выбирает объект и Vuforia захватывает метку.

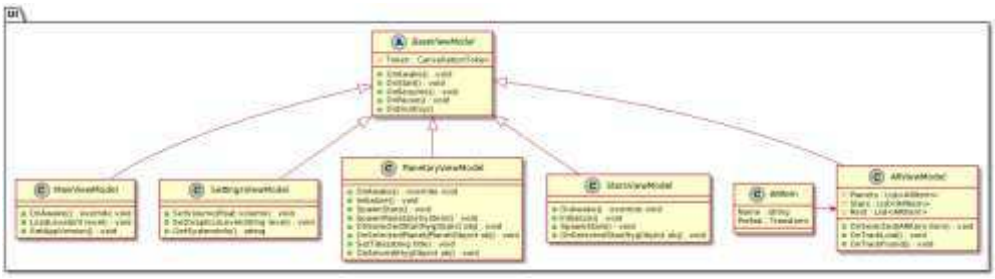


Рисунок 3.12 – диаграмма классов ViewModel

3.4 Графический рендеринг

Шейдер — это небольшая программа, содержащая инструкции для графического процессора. Эти инструкции описывают общий алгоритм, как должен выглядеть конкретный объект на экране. Исторически, написание

Примечание [М. И. В.5]: Перечитал раздел пару раз… мне кажется, или вы просто рассказываете, как рисовать шейдеры средствами Юнити? Во-первых, опять копипаста – причем откуда-то из середины учебника, потому что что за узлы Френеля, неясно вообще. Во-вторых, это настолько же интересно и важно, насколько рассказ о том, как рисовали градиентную текстуру для бэкграунда диалового окна в Paint. Не особо.
Смысл есть, если шейдеры в вашем приложении как-то особо хитро строятся, ввиду особенностей предметной области – астрономии ли, AR ли…

шейдеров требовало знания специального языка, такого как Cg или HLSL, с синтаксисом, отличающимся от обычных языков программирования.

Встроенное средство в Unity под названием Shader Graph позволяет с легкостью разрабатывать шейдеры в визуальном интерфейсе с отображением результатов в реальном времени. Вместо написания кода, программирование шейдеров происходит через визуальные узлы, которые выполняют простые задачи, такие как сложение двух векторов, выборка текстуры или интерполяция между двумя значениями и подключение выходов одного узла к входам других узлов.

3.4.1 Разработка шейдера атмосферы

Атмосферные эффекты трудно воссоздать в трехмерной графике, потому что атмосфера является полупрозрачным, а не твердым объектом. Традиционные методы рендеринга предполагают, что все графические вычисления происходят только на поверхностях материала, независимо от того, что внутри. Это значительное упрощение позволяет очень эффективно визуализировать твердые объекты. Особенности некоторых объектов, однако, определяются тем, что свет может проникать сквозь них. Окончательный вид полупрозрачных объектов является результатом взаимодействия света с их внутренней структурой.

Для начала необходимо создать два узла Френеля, которые будут являться оболочкой сферы. Один из узлов отвечает за внешнее свечение, второй – за внутреннее. Оба узла соединяются функцией Substract и умножаются на цвет атмосферы, который является переменной и можно позже менять в материале.

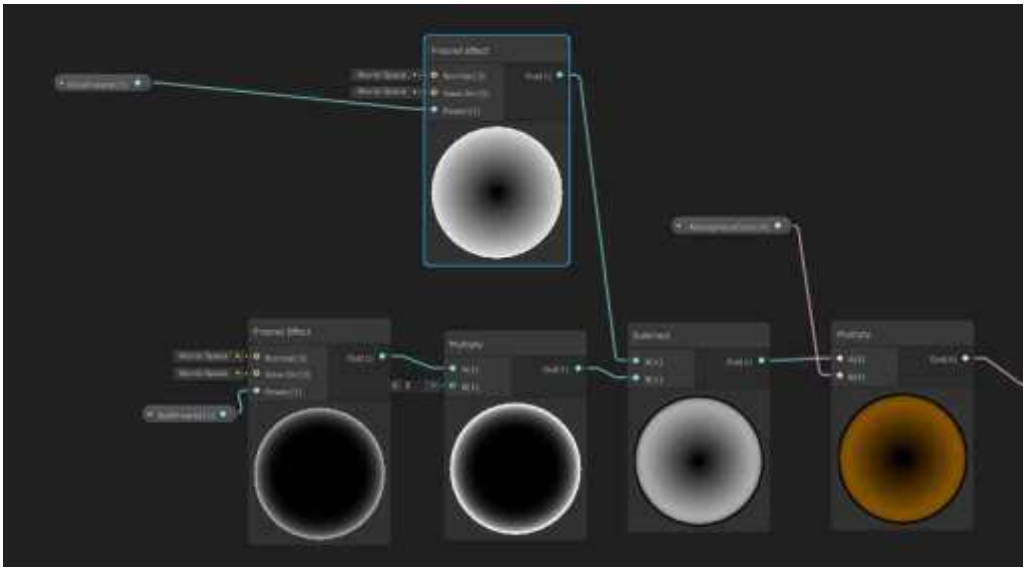


Рисунок 3.13 – узлы для формирования полупрозрачной сферы

Для того чтобы обратная сторона была затемнена, воспользуемся узлом Normal Vector и операцией Dot. Узел Dot — это значение с плавающей точкой, равное величинам двух векторов, умноженных вместе и затем умноженных на косинус угла между ними. Для векторов данный узел возвращает 1, если они указывают точно в одном направлении, -1, если они указывают в совершенно противоположных направлениях, и ноль, если векторы перпендикулярны.

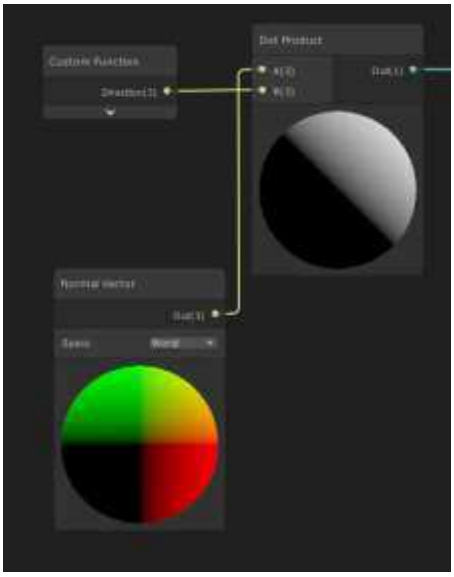


Рисунок 3.14 – затемнение обратной стороны

Функция для получения вектора направления источника света

Рисунок 3.15 – функция получения вектора источника освещения

Получившиеся узлы шейдера отображены на рисунке:

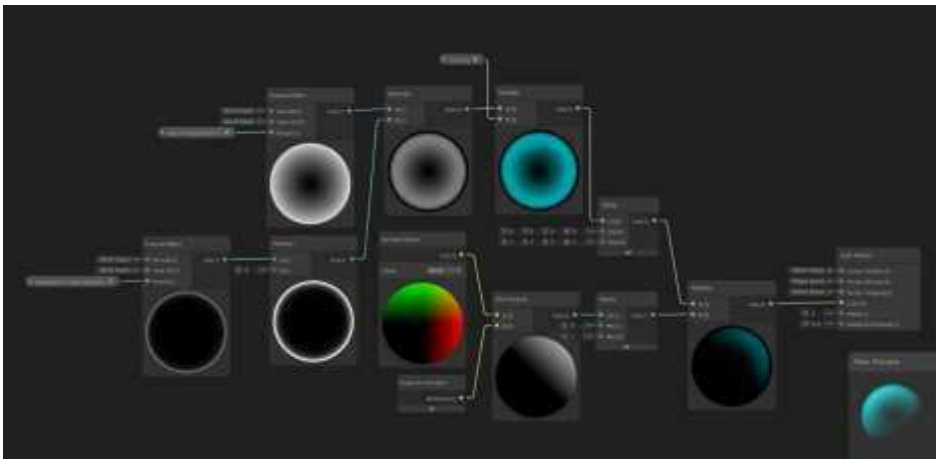


Рисунок 3.16 – узлы шейдера

Результат работы шейдера после настройки:

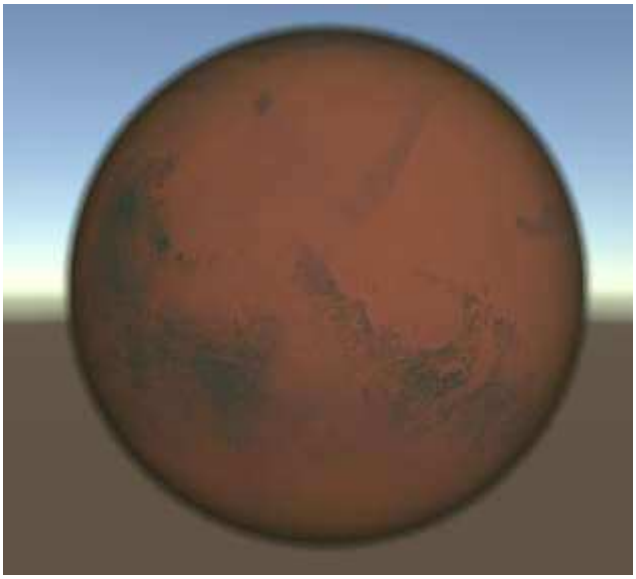


Рисунок 3.17 – примера работы шейдера на основе модели Марса

3.4.2 Разработка шейдера черной дыры

Шейдер черной дыры состоит из двух частей. Первая часть – это фоновое искажение (рефракция), а вторая часть – черная сфера посередине.

Для того чтобы создать эффект рефракции или гравитационного линзирования, сначала необходимо создать узел Френеля. Узел Френеля будет играть роль «маски» для преломления и черной сферы внутри. Потребуется также инвертировать результат узла Френеля, чтобы получить эффект преломления по краям и умножить на значение узла Power, чтобы его усилить.

Фоновое изображение может быть искажено смещением координат сцены (UV) к центру нашей сферы. Чтобы сделать это, воспользуемся узлами Screen Position и Remap.

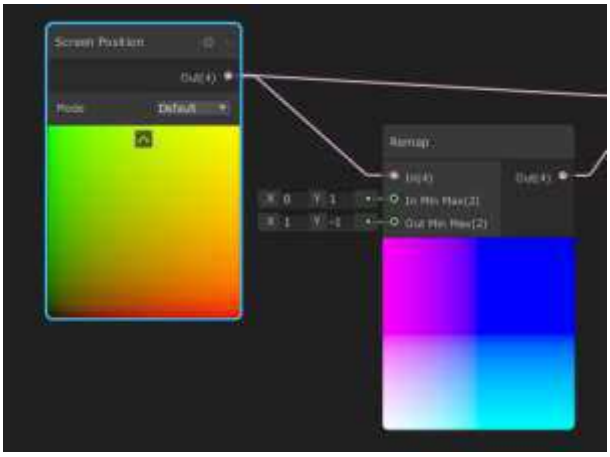


Рисунок 3.18 – узлы Screen Position и Remap

Создав узел Френеля и узлы для смещения UV сцены, соединим их между собой узлом Multiply, чтобы получить эффект гравитационной линзы.

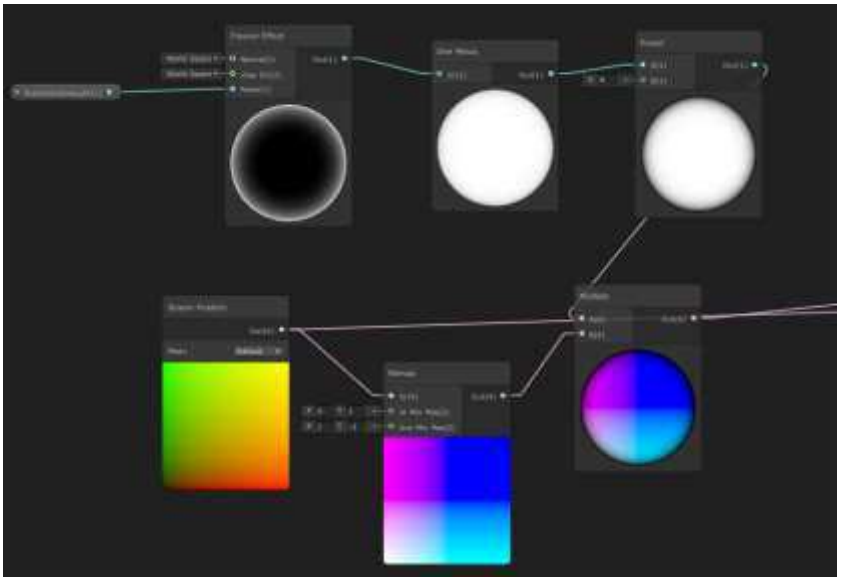


Рисунок 3.19 – узлы для создания линзы

Для того чтобы сделать черную сферу посередине нам так же понадобится узел Френеля, но для более плавного перехода между сферой и линзой так же воспользуемся узлами Remap и Smoothstep.

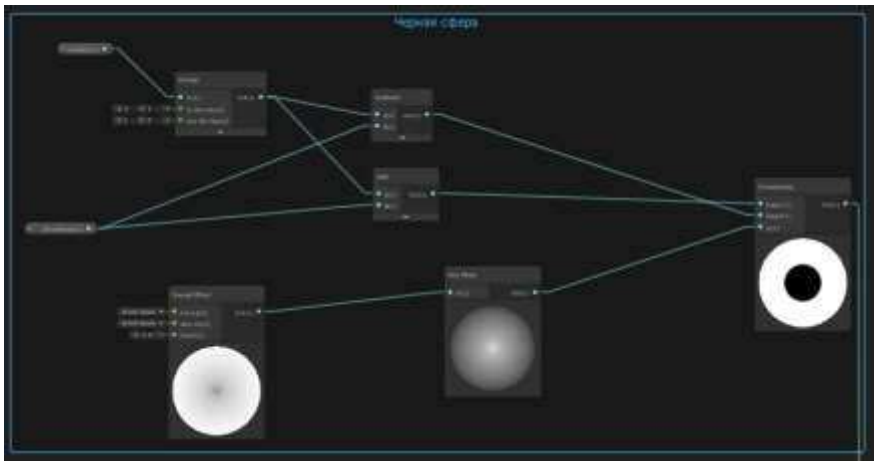


Рисунок 3.20 – создание черной сферы внутри посередине линзы

В результате мы получаем следующие узлы шейдера:

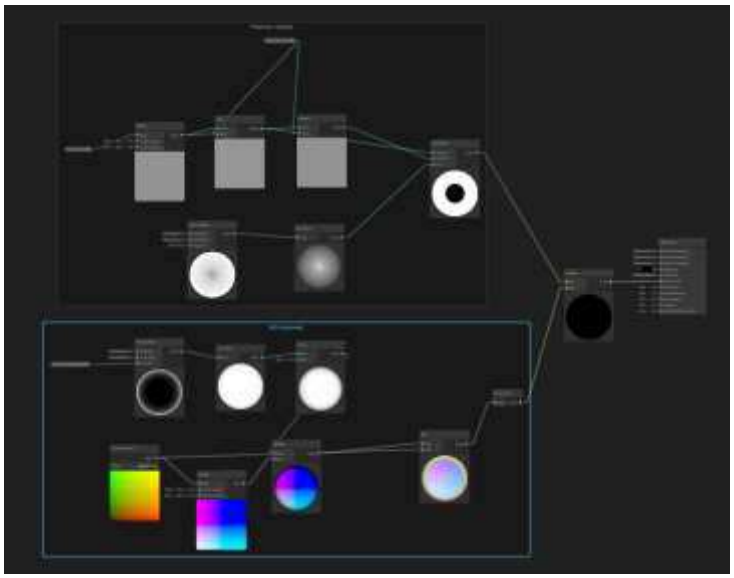


Рисунок 3.21 – шейдер черной дыры

После компиляции шейдера необходимо включить опцию «Opaque Texture» в настройках рендера для того, чтобы эффект линзы сработал, после чего создать материал с данным шейдером и добавить его на меш со сферой. Для тестирования была создана новая сцена, и добавлен куб, можно наблюдать его искажение левее сферы.

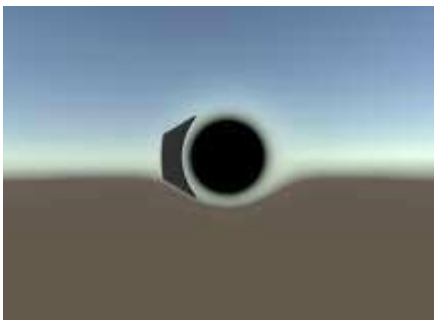


Рисунок 3.22 – сфера с разработанным шейдером

3.5 Работа с дополненной реальностью

3.5.1 Настройка Unity для работы с Vuforia

Для разработки приложений с использованием Vuforia необходимо добавить библиотеку через менеджера пакетов в окне редактора Unity. После этого в ассетах редактора, появятся ARCamera и ImageTargetBehaviour – они являются основной создания приложений с дополненной реальностью.

По умолчанию Vuforia работает с распознаванием изображений. Распознавание изображений или распознавание трекеров, представляет собой процесс, с помощью которого библиотека через камеру обнаруживает предварительно обработанное изображение и знает, что с ним делать, например, визуализирует некоторый контент поверх него. Поэтому помимо установки SDK, в проект потребуется импортировать базу заранее подготовленных изображений, используемых в качестве меток, которые прошли обработку на сайте Vuforia.

3.5.2 Создание виртуальной сцены для меток

Все объекты, используемые в Unity, имеют свое место в древовидной иерархической структуре. У каждого объекта может быть только один родитель (если объект находится в корне, то родитель отсутствует) и произвольное количество потомков. На рисунке 3.9 приведен скриншот окна редактора Unity, в левой части которого содержится панель с иерархией объектов. В центре части рисунка виден фрагмент виртуальной сцены с размещенной на ней меткой ImageTargetBehaviour. Объекты, которые будут использоваться для дополнения реальности, размещаются на виртуальной сцене и становятся потомками метки. На рисунке 3.24 приведен результат отрисовки виртуальной сцены, на которую добавлено трехмерная модель Солнца.

Рисунок 3.23 – Окно редактора Unity и метка ImageTargetBehaviour

При запуске такого приложения Unity на любой из доступных платформ произойдет следующее: до тех пор, пока метка не была обнаружена камерой устройства, все потомки ImageTarget «выключены», т.е. находятся в неактивном состоянии и не отрисовываются на экране. Как только происходит обнаружение метки, камера виртуальной сцены (объект ARCamera) перемещается в такую позицию, что ориентация и размер метки, наблюдаемой в поле зрения ARCamera, совпадают с тем, как видит эту метку пользователь приложения, т.е. виртуальная камера и камера устройства смотрят на метку с одинакового ракурса.

Объекты, являющиеся потомками обнаруженной метки, становятся активными и видимыми для камеры виртуальной сцены. После этого изображение предметов, попадающих в поле зрения ARCamera, накладывается на изображение реального мира, полученное с помощью камеры устройства, и, тем самым, «дополняет» его.

Рисунок 3.24 – результат наложения модели на метку

Необходимо отметить, что Vuforia может «потерять» метку, если она пропала из поля зрения камеры устройства, или пользователь отошел от метки слишком далеко. В таком случае, объекты виртуальной сцены, связанные с этой меткой, становятся неактивными и невидимыми для камеры виртуальной сцены, т.е. не выводятся на экран.

3.6 Выводы

Было разработано мобильное приложение, соответствующее сформированным требованиям и функционалу, а также соответствующее программной платформе реализации. Так же в данной главе описаны алгоритмы работы и реализации основных модулей системы.

Примечание [М. И. В.6]: Организация вашей БД? Типы данных, взаимодействия?
Два с половиной модуля из 4 заявленных в 2.1. есть, но слабо. Про Core вообще ничего не видно. Из чего эти модули состоят? Как взаимодействуют?

ГЛАВА 4. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

При открытии приложения пользователь увидит главное меню, где может выбрать в какой раздел перейти:

- Планетарий – просмотр модели солнечной системы
- Звездное небо – просмотр звездного неба
- Дополненная реальность (AR) – просмотр астрономических объектов из каталога в дополненной реальности
- Настройки – настройки приложения (звук, графика и т.д.)
- Выход – выход из приложения

Примечание [М. И. В.7]: Где это в инструкции?

Рисунок. 4.1 – главное меню приложения

Рисунок 4.2 – меню с настройками

В разделе «Планетарий» находится полная трехмерная модель Солнечной системы. Положение планет отображается в режиме реального времени, но можно посчитать положение для любой даты. Слайдер снизу позволяет переметывать время вперед и назад. Так же имеются настройки для отображения названия планет и колец орбит. На фоне отображаются звезды из каталога HYG, выборка звезд – 10000 ближайших к Солнцу.

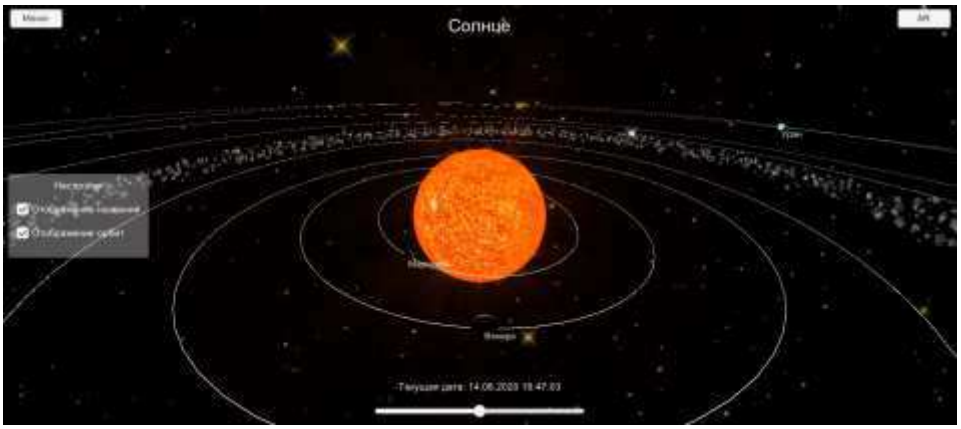


Рисунок 4.3 – планетарий

При нажатии на какую-либо из планет камера фокусируется на ней и можно посмотреть справочную информацию о небесном теле.

Рис. 4.4 – информация о планете

В режиме звездного неба можно посмотреть большинство звезд, наблюдаемых с Земли. Для некоторых звезд есть справочная информация.

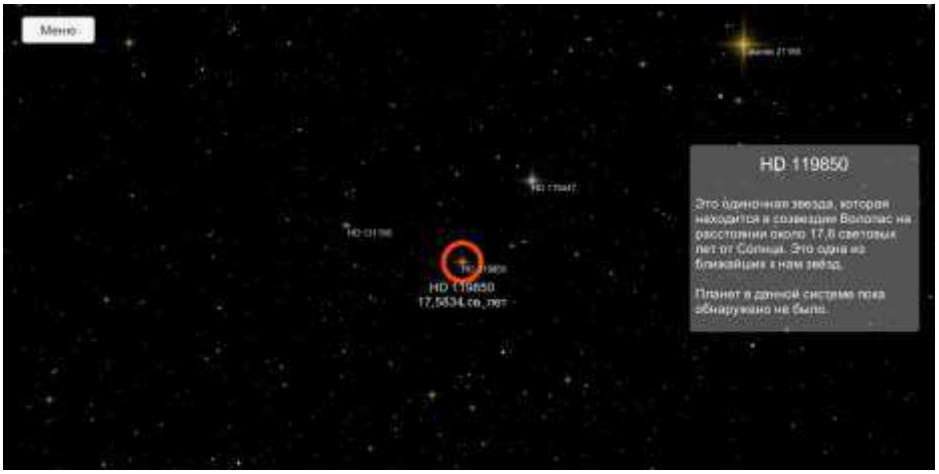


Рисунок 4.5 – звездное небо

В режиме дополненной реальности можно посмотреть, как выглядят астрономические объекты. Для этого необходимо выбрать объект из каталога и

навести его на изображение. Изображение можно распечатать или показать на экране другого устройства.

Рисунок 4.6 – дополненная реальность

ЗАКЛЮЧЕНИЕ

В ходе работы была спроектировано и реализовано приложение для изучения астрономии:

- Произведен анализ предметной области, в ходе которого рассмотрена технология дополненной реальности и существующие аналоги
- Произведен анализ технического стека использованного при разработке решения
- Реализован функционала приложения: трехмерная модель планетария, справочная информация, отображение модели небесных тел при помощи дополненной реальности, вычисление положения в зависимости от текущей даты
- Выполнено тестирование

Разработанное приложение обладает большим потенциалом, может быть легко расширено в виду модульной природы и может быть перенесено на многие операционные системы, благодаря архитектуре программы, возможностям языка программирования и среды разработки.

СПИСОК ИСТОЧНИКОВ

1. Хабрахабр [Электронный ресурс]: Технологии AR и VR в образовании / Блок компании Mail.Ru – Режим доступа: https://habr.com/ru/company/mailru/blog/435996/
2. The Telegraph [Электронный ресурс]: Augmented reality: making sci-fi come true for the modern military – Режим доступа: https://www.telegraph.co.uk/education/stem-awards/defence-technology/augmented-reality-in-the-military/
3. ThinkMobiles [Электронный ресурс]: Augmented Reality in Medicine and Healthcare – Режим доступа: https://thinkmobiles.com/blog/augmented-reality-medicine
4. GoodHouseKeeping [Электронный ресурс]: 15 Best Stargazing Apps – Режим доступа: https://www.goodhousekeeping.com/life/g26089673/best-stargazing-apps/
5. Tom’s Guide [Электронный ресурс]: 15 Best stargazing apps for looking at the night sky – Режим доступа: https://www.tomsguide.com/round-up/best-stargazing-apps
6. Stellarium [Электронный ресурс]: Официальный сайт – Режим доступа: https://stellarium.org/ru/
7. Sky&Telescope [Электронный ресурс]: Sky & telescope mobile apps – Режим доступа: https://skyandtelescope.org/sky-and-stargazing-apps/
8. Хабрахабр [Электронный ресурс]: Шаблон проектирования Entity-Component-System — реализация и пример – Режим доступа: https://habr.com/ru/post/343778/
9. Unity3D [Электронный ресурс]: Технологический стек DOTS – Режим доступа: https://unity.com/ru/dots
10. Metanit [Электронный ресурс]: Паттерн MVVM – Режим доступа: https://metanit.com/sharp/wpf/22.1.php

11. Metanit [Электронный ресурс]: Паттерн Repository – Режим доступа: https://metanit.com/sharp/articles/mvc/11.php
12. Wikipedia [Электронный ресурс]: Каталог звезд – Режим доступа: https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D1%82%D0%B0%D0%BB%D0%BE%D0%B3_%D0%B7%D0%B2%D1%91%D0%B7%D0%B4%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BD%D0%B5%D0%B1%D0%B0
13. AstroNexus [Электронный ресурс]: The HYG Database– Режим доступа: http://www.astronexus.com/hyg
14. Github [Электронный ресурс]: HYG Database – Режим доступа: https://github.com/astronexus/HYG-Database
15. FlargetWins [Электронный ресурс]: Creating a 3D Star Map – Режим доступа: https://www.flerlagetwins.com/2018/01/creating-3d-star-map-in-tableau_55.html
16. Хабрахабр [Электронный ресурс]: Расчёт положения небесных тел и эфемеридные теории – Режим доступа: https://habr.com/ru/post/204986/
17. Caglow [Электронный ресурс]: Planetary Positions with VSOP87 – Режим доступа: https://www.caglow.com/info/compute/vsop87
18. Totaleclipse [Электронный ресурс]: VSOP87 – Режим доступа: http://totaleclipse.eu/Astronomy/VSOP87.html
19. Google Developers [Электронный ресурс]: Floating Origin – Режим доступа:https://developers.google.com/maps/documentation/gaming/move_floating_origin
20. StackExchange [Электронный ресурс]: Game Development – Режим доступа: https://gamedev.stackexchange.com/questions/111000/why-dont-objects-far-away-from-the-camera-experience-floating-point-issues
21. Хабрахабр [Электронный ресурс]: Математика в Gamedev по-простому – Режим доступа: https://habr.com/ru/post/432544/

22. Журнал «Хакер» [Электронный ресурс]: Анатомия игровых движков – Режим доступа: https://xakep.ru/2014/09/05/game-development-engines-review/
23. Godot Engine [Электронный ресурс]: Официальный сайт – Режим доступа: https://godotengine.org/
24. Epic Games [Электронный ресурс]: What is Unreal Engine 4 — Режим доступа: https://www.unrealengine.com/what-is-unreal-engine-4.
25. Unity [Электронный ресурс]: Официальный сайт – Режим доступа: https://unity.com/ru
26. Vuforia Developer Portal [Электронный ресурс]: Документация для разработчиков – Режим доступа: https://developer.vuforia.com/
27. Хабрахабр [Электронный ресурс]: Что такое ARCore? – Режим доступа: https://habr.com/ru/post/437378/
28. Хабрахабр [Электронный ресурс]: SQLite как встраиваемая БД– Режим доступа: https://habr.com/ru/post/149356/
29. Хабрахабр [Электронный ресурс]: Unity, ECS – Режим доступа: https://habr.com/ru/post/358108/
30. Unity Documents [Электронный ресурс]: Unity UI: Unity User Interface – Режим доступа: https://docs.unity3d.com/Packages/com.unity.ugui@1.0 /manual/index.html

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Космических и информационных технологий
институт

Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О. В. Непомнящий
подпись инициалы, фамилия

«____»_____2020 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 Информатика и вычислительная техника
код и наименование специальности

Мобильное приложение с дополненной реальностью для изучения астрономии
тема

Руководитель	_____ подпись, дата	доцент, канд. техн. наук должность, учёная степень	Постников А.И. инициалы, фамилия
Консультант	_____ подпись, дата	старший преподаватель должность, учёная степень	И. В. Матковский инициалы, фамилия
Выпускник	_____ подпись, дата		В. С. Свиридов инициалы, фамилия
Нормоконтролер	_____ подпись, дата	старший преподаватель должность, учёная степень	И. В. Матковский инициалы, фамилия

Красноярск 2020