ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ДОПУСТИТЬ К ЗАЩИТЕ
Профессор с возложенными обязанностями заведующего Кафедрой информационных систем в искусстве и гуманитарных науках
_____(Борисов Н.В.)
"_21_"__05_____20__ г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Направление 09.03.03 «Прикладная информатика»
Уровень Бакалавриат
Основная образовательная программа
«Прикладная информатика в области искусств и гуманитарных наук»

На тему
«Создание мобильного приложения (AR) и моделей для иллюстрированной презентации учебного пособия»

Студентки *Усольцевой Ирины Игоревны*
_____
(подпись студента)

Руководитель (-ли): *канд. физ.-мат. наук, доцент Павел Петрович Щербаков*
_____
(подпись руководителя)

Консультант (-ты): *старший преподаватель Логдачева Елена Викторовна*
3D-художник Коротких Алексей Евгеньевич
_____

Санкт-Петербург
2021

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ИСКУССТВ
Кафедра информационных систем в искусстве и гуманитарных науках

УТВЕРЖДАЮ
Заведующий кафедрой

_____

ЗАДАНИЕ
по подготовке выпускной квалификационной работы студентки Усольцевой Ирины Игоревны

1. Тема работы: Создание мобильного приложения (AR) и моделей для иллюстрированной презентации учебного пособия
2. Срок сдачи студентом законченной работы: июнь 2021
3. Исходные данные к работе: Учебное пособие "*Gnome Innico - Colouring Fairy Tale in English* / ГНОМ ИННИКО - сказка-раскраска на английском языке".
4. План-график выполнения квалификационной работы:

Номера и содержание этапов работы	**Плановая дата сдачи**	**Фактическая дата сдачи**	**Дата выдачи рецензии**
1. Изучение программ и основ работы с дополненной реальностью (AR) и мобильными приложениями	09.2020 – 10.2020	01.10.2020	01.10.2020
2. Создание прототипа мобильного приложения	10.2020 – 11.2020	03.11.2020	03.11.2020
3. Отладка прототипа, перенос и сборка приложения с примитивами	10.2020 – 11.2020	09.11.2020	09.11.2020
4. Отладка приложения с настоящей/используемой моделью персонажа	11.2020 – 12.2020	05.12.2020	05.12.2020
5. Сборка приложения с персонажем	11.2020 – 01.2021	09.01.2021	09.01.2021
6. Создание и интеграция моделей персонажей в приложение	01.2021 – 03.2021	06.03.2021	06.03.2021
7. Финальная отладка приложения	03.2021 – 05.2021	02.05.2021	02.05.2021

Консультант по работе: старший преподаватель Логдачева Елена Викторовна
3D-художник Коротких Алексей Евгеньевич
Руководитель от кафедры: канд. физ.-мат. наук, доцент Павел Петрович Щербаков

_____
(должность, Фамилия Имя Отчество, подпись)

Задание принял к исполнению_____ Подпись студента_____
_____ (дата)

АННОТАЦИЯ

выпускной квалификационной работы

Усольцевой Ирины Игоревны

«Создание мобильного приложения (AR) и моделей для иллюстрированной презентации учебного пособия»

Целью данной работы является создание мобильного приложения дополненной реальности и моделей для иллюстрированной презентации учебного пособия "Gnome Innico - Colouring Fairy Tale in English / ГНОМ ИННИКО - сказка-раскраска на английском языке".

ВКР состоит из трёх глав, разбитых на параграфы. В первой главе проведен анализ ряда инструментов разработчика для взаимодействия с проектами дополненной реальности, а также описано создание и сравнение двух прототипов приложения при помощи разных инструментов разработчика с последующей доработкой наиболее оптимального из них до финальной версии приложения. Во второй главе освещен процесс разработки образа героя, создания трехмерной модели, адаптации трехмерной модели для мобильного приложения, текстурирования. В третьей главе описан процесс подготовки к анимации, анимации, а также интеграции моделей в мобильное приложение и последующей настройки анимации в приложении.

В процессе работы использовались программы Autodesk Maya, Pixologic ZBrush, Substance Painter, Marmoset Toolbag, Unity 3D, Adobe Photoshop, PureRef, а также плагины Advansed Skeleton, ngSkinTools, EasyAR, Vuforia

Автор работы _____ _____
подпись (фамилия, имя, отчество)

Руководитель работы _____ _____
подпись (фамилия, имя, отчество)

Содержание

Определения, обозначения и сокращения

Игровой движок – базовое программное обеспечение компьютерной игры.

SDK (software development kit) – комплект инструментов разработки, позволяющий специалистам по программному обеспечению создавать приложения для определённого пакета программ, программного обеспечения базовых средств разработки, аппаратной платформы, компьютерной системы, игровых консолей, операционных систем и прочих платформ.

Target (метка) – отслеживаемое изображение.

Дополненная реальность (augmented reality, AR) – среда, в реальном времени дополняющая физический мир цифровыми данными с помощью каких-либо устройств, когда виртуальные объекты проецируются на реальное окружение.

UE4 (Unreal Engine 4) - игровой движок, разрабатываемый и поддерживаемый компанией Epic Games.

Unity - межплатформенная среда разработки компьютерных игр, разработанная американской компанией Unity Technologies.

Плагин – программный модуль, подключаемый к основной программе, позволяющий расширить её возможности.

Эмулятор – компьютер или программа, имитирующая другой компьютер или программу.

Ассет – цифровой объект, преимущественно состоящий из однотипных данных, неделимая сущность, которая представляет часть игрового контента и обладает некими свойствами.

Введение

В современном мире технологии становятся всё более привычной частью нашей жизни. Появляются приложения, помогающие следить за здоровьем, smart-браслеты, появляются специальные мультфильмы для младенцев. Меняется и учебный процесс – многие учебные направления предполагают выполнение не только задач на бумаге, но и связанных с компьютером. В школах появляются «умные» доски, в музеях – специальные приложения, предоставляющие дополнительную информацию об экспонатах в формате простого текста или дополненной реальности.

Дополненная реальность, она же augmented reality, AR, является сферой, ещё не привычной большинству пользователей, но при этом активно развивающейся. При использовании средств AR обычная реальность дополняется новыми элементами, отображаемыми при помощи какого-либо устройства – это может быть текст, изображение, видео, 3D-объект и иные элементы. Подобные технологии позволяют ввести новые методы взаимодействия с пользователями, в том числе и детьми, в различных областях – медицине, маркетинге, образовании, искусстве, играх, организации и распорядке дня, учебе.

Целью ВКР было создание мобильного приложения дополненной реальности и моделей для иллюстрированной презентации учебного пособия "Gnome Innico - Colouring Fairy Tale in English / ГНОМ ИННИКО - сказка-раскраска на английском языке".

Данное учебное пособие представляет из себя книгу, разделенную на две части – текст сказки на английском языке с иллюстрациями для раскрашивания ребёнком и англо-русский словарь, позволяющий сказку перевести. Для использования в приложении была выбрана обложка и изображение героев на форзаце и эрзаце книги.

Процесс достижения поставленной цели был разбит на следующие задачи:

- изучение процесса и методов создания приложений дополненной реальности, анализ и выбор оптимального инструмента разработчика для работы с приложением;
- получение навыков разработки, создание приложения-основы;
- создание высоко- и низкополигональной трехмерной модели;
- текстурирование низкополигональной модели;
- подготовка модели к анимации;
- создание анимации модели;
- интеграция в приложение, тестирование.

Разработка приложения и создание моделей проводилась совместно с коллегой Дарьей Лактионовой. Работа была разделена следующим образом: создание приложения, модели героя-гнома, интеграция модели героя-гнома входило в мои обязанности, создание концептов героев, создание модели героя-гусенка, моделей домов входило в обязанности Дарьи.

Процесс работы включал в себя изучение и получение навыков работы с технологиями, используемыми в таких сферах, как: 3D-моделирование, 3D-текстурирование, риггинг, анимация, разработка мобильных игр, разработка компьютерных игр. Таким образом, полученные навыки могут быть применены при последующей работе в данных сферах или при работе с аналогичными проектами, а также проектами, использующими отдельные подобные технологии.

1. Создание приложения.

Целью работы, описанной в первой главе, было изучение возможностей и сравнительный анализ современных инструментов и платформ разработки для создания мобильного приложения дополненной реальности, выбор инструмента, позволяющего создать приложение, отвечающее всем поставленным в техническом задании требованиям, а также освоение навыков работы с выбранным инструментом.

Для достижения поставленной цели был составлен следующий порядок работы:

1. Поиск SDK и плагинов, позволяющих осуществлять работу с дополненной реальностью.

2. Изучение документации и опыта работы других пользователей с SDK и плагинами.

3. Проведение анализа каждого инструмента по следующим пунктам:

- количество, качество и актуальность документации;

- наличие или отсутствие водяного знака;

- возможность использовать инструмент без денежных вложений;

- количество распознаваний в месяц;

- возможность тестирования и работы с инструментом без использования дополнительных устройств;

- доступность собранного приложения для пользователей.

4. Составление таблицы по результатам проведённого анализа. Выбор оптимального инструмента.

5. Создание прототипа приложения при помощи ряда инструментов.

6. Загрузка в приложение модели с текстурой и анимацией, тестирование

1.1. Анализ существующих SDK для работы с AR

В настоящий момент существует более 20 SDK для работы с AR. Часть SDK являются универсальными, часть имеют область, для которой они подходят лучше, чем для других областей. Так, например, DeepAR, Xzimg в основном используются для отслеживания лица, эмоций и создания масок и эффектов, отображаемых на лице. ARKit, ARCore, ARFoundation являются универсальными SDK, при помощи которых можно реализовать практически любую реалистичную идею в виртуальной реальности. Vuforia, Wikitude, Kudan, MaxST, EasyAR, Amazon Sumerian, ARToolKit, Onirix, Pikkart AR SDK, BlippBuilder имеют меньший функционал и больше сосредоточены на отслеживании меток – как плоских, так и объемных, а также идентификации горизонтальных и вертикальных поверхностей. AR.js, 8th Wall – позволяют создать AR приложение в браузере, без установки его на какое-либо устройство.

В качестве основной платформы для разработки приложения было рассмотрено два варианта: Unity и Unreal Engine 4. Обе платформы являются достаточно распространёнными движками для разработки игр и приложений. Таким образом, в анализ плагинов и SDK был включён пункт проверки совместимости выбранного инструмента с одной из платформ.

Так как основной функцией приложения являлось отображение модели поверх метки, было принято решение не рассматривать SDK, направленные на работу с лицами. Всего тестированию и анализу подверглись 3 SDK и один плагин в связке с определёнными платформами:

1. ARKit и ARCore в Unreal Engine 4;

2. Vuforia в Unity;

3. EasyAR в Unity;

4. Augmented Reality Plugin для Unreal Engine 4.

1.2. ARKit (ARCore) в UE4

ARKit (ARCore) — два набора инструментов, которые разработчики Unreal Engine предлагают в своей документации для разработки проектов для мобильных устройств. ARKit — технология, разработанная компанией Apple, а ARCore, в свою очередь, технология, разработанная компанией Google. Обе технологии поддерживаются движком Unreal Engine и являются достаточно распространёнными среди уже созданных приложений в магазинах обеих компаний. Поскольку возможности проводить тестирование на устройстве с системой Apple не было ввиду отсутствия подобного, было принято решение продолжать разработку приложения только на системе Android.

Для того чтобы приложение, собранное на ARCore, работало на мобильном устройстве, необходимо, чтобы на это мобильное устройства, кроме приложения, был установлен и сам ARCore. В настоящий момент ARCore можно установить через Google Play, где данное приложение имеется под названием «Сервисы Google Play для AR», и, будучи частью сервиса Google Play, устанавливается и обновляется на поддерживаемых устройствах автоматически.

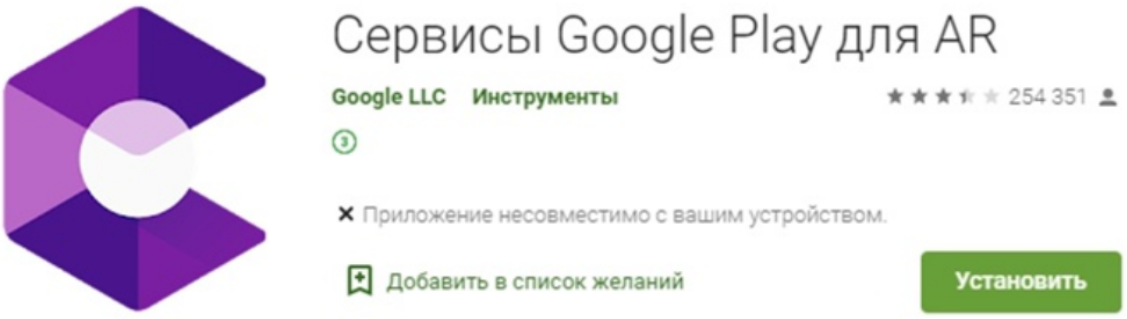


Рис. 1 – ARCore в магазине Google Play

У ARCore проработанная документация, в том числе и по работе в UE4, что обеспечивает простоту вхождения в разработку на данном SDK.

Отдельный достаточно большой блок, посвящённый работе в связке ARCore + UE4 есть и в документации самого Unreal. Документации и UE4, и ArCore проиллюстрированы, что также добавляет информативности и позволяет лучше понять работу SDK.

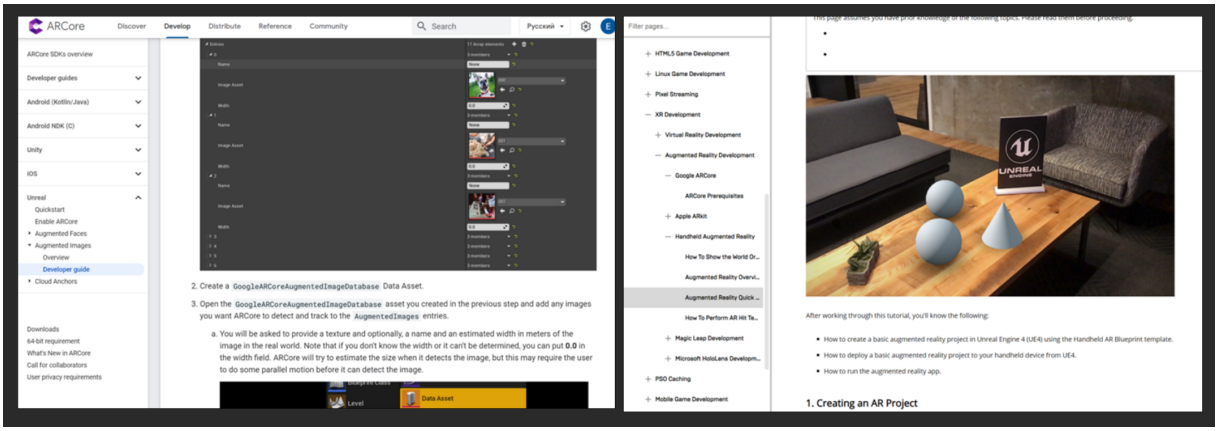


Рис. 2 – Документация ARCore. Рис. 3 – Документация UE4 по ARCore

Приложения, собранные при помощи ARCore, не имеют водяного знака, сам ARCore бесплатен, количество распознаваний метки не ограничено.

Из основных недостатков — большое количество устройств, на которых это приложение не поддерживается: в момент разработки количество моделей подобных устройств составляло примерно 60% от всех устройств, работающих на Android. Также разработку осложнял тот факт, что устройство, на котором планировалось проводить тестирование, не входило в список поддерживаемых, а также невозможность запустить собранное приложение в UE4 на компьютере, только на Android-устройстве. Таким образом, для осуществления тестирования нужно было либо находить доступ к устройству, поддерживающему ARCore, либо решать проблему путём установки ARCore или неофициальных «исправленных» версий через Root-права на устройстве.

Исследование ряда форумов показало, что проблему с тестированием можно решить установкой на мобильное устройство двух неофициальных приложений: TangoCore service — сервис, работающий в фоновом режиме,

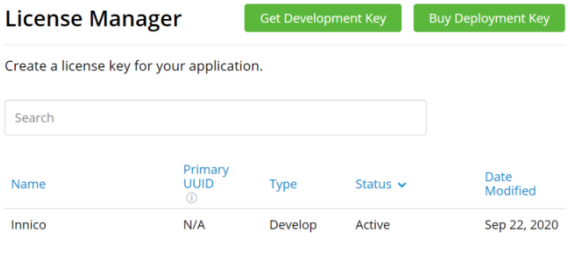
позволяющий вычислить расположение телефона в пространстве, основываясь на информации, полученной с камеры и внутренних датчиков и HelloAR — приложение, основанное на ARCore. Оба приложения изменены так, чтобы создать возможность запуска ARCore на любом Android-устройстве.

Несмотря на имеющееся решение проблемы, было принято решение не рисковать и не устанавливать неизвестные приложения на мобильное устройство, и потому вариант разработки приложения в связке ARCore+UE4 не был рассмотрен в дальнейшем.

1.3. Vuforia в Unity

Vuforia – популярный SDK, используемый для разработки приложений дополненной реальности. При помощи Vuforia можно создавать приложения как для Android, так и для iOS.

Vuforia является облачным приложением, при этом для связи проекта в облаке и в Unity используется License key, генерируемый в личном кабинете Vuforia для каждого отдельного проекта и вводимый в Unity в специальное поле.



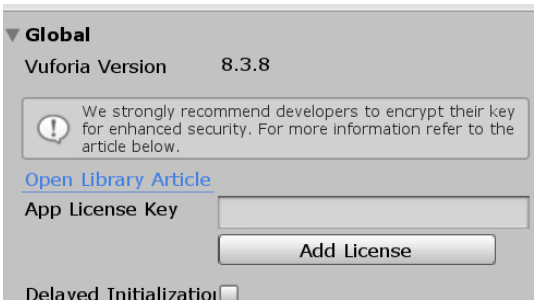
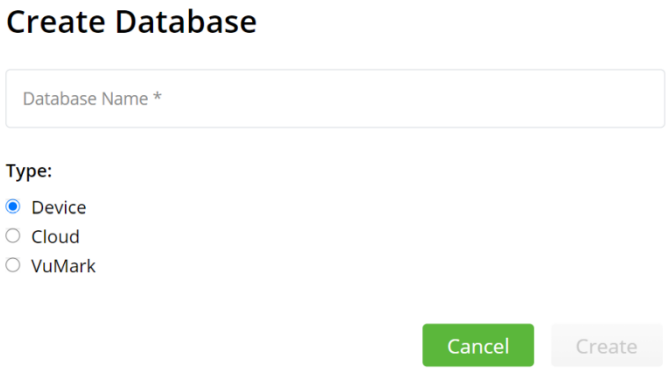


Рис. 4 – License Manager в Vuforia. Рис.5 – область для ввода ключа в Unity

Распознавание меток в Vuforia осуществляется при помощи предварительной обработки их при загрузке в Target Manager в личном кабинете Vuforia. Для этого в личном кабинете Vuforia создаётся база данных, при этом выбирается тип хранения базы – на устройстве или в

облаке, в которую загружаются метки, которые должно будет отслеживать приложение. Также при загрузке каждой метке присваивается уникальное имя и выбирается тип: простое изображение, параллелепипед, цилиндр или 3D-объект и вносятся дополнительные параметры, такие как, например, ширина для изображения, ширина, высота и длина для параллелепипеда, верхний и нижний диаметры и длина для цилиндра.



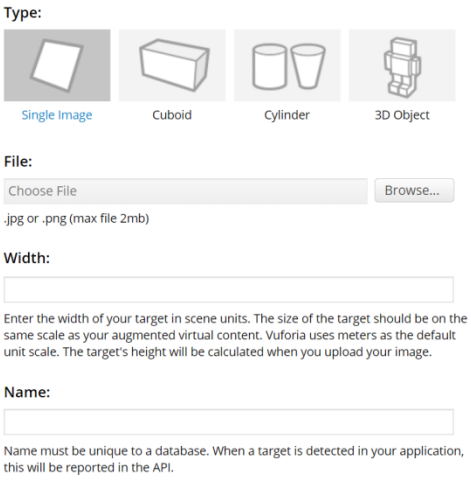


Рис. 6 – создание базы данных. Рис. 7 – добавление метки в базу данных

В базе данных происходит обработка полученных метки, в процессе которых на них добавляются точки, по которым будет проводиться распознавание изображения.

Рис. 8 – изображение до обработки Рис. 9 – изображение после обработки

Полученную базу данных можно скачать и загрузить в Unity, где впоследствии и настраиваются последующие действия приложения при обнаружении метки.

Основное преимущество использования Vuforia для проекта – поддержка модуля на всех устройствах Android ввиду отсутствия особых встроенных требований для запуска приложения, в отличие от ARCore, за исключением функций, напрямую необходимых для работы приложения: наличия камеры, возможности отслеживать местоположение телефона в пространстве и так далее, что даёт возможность работать с SDK и проводить тестирование на имеющемся устройстве, а также делает приложение доступным большему количеству пользователей. Дополнительным преимуществом стали и встроенные в модуль инструменты, благодаря которым можно было создать приложение без написания кода с нуля, а также наличие у Vuforia на момент разработки бесплатной версии. Однако бесплатная версия поддерживала только производить 1000 распознаваний в месяц, что было недостаточно для работы приложения. Также у приложений, собранных при помощи Vuforia

имеется большой водяной знак, закрывающий нижнюю левую часть экрана. Документация у Vuforia подробная, информативности добавляют иллюстрации и проекты-примеры.

1.4. EasyAR

EasyAR – SDK, часто используемое наравне с Vuforia ввиду схожего инструментария и методов работы. EasyAR не предполагает создание базы данных вне платформы разработки, метки загружаются сразу в Unity и после чего настраиваются самим пользователем вместе с действиями приложения.

EasyAR имеет встроенные инструменты, позволяющие создать необходимое приложение с минимальным взаимодействием с кодом. В отличие от Vuforia, инструменты которой после установки плагина отображаются в интерфейсе Unity наравне с изначальными, инструменты EasyAR находятся в отдельной директории Prefabs, которая появляется после импорта плагина. Приложения, собранные на EasyAR возможно запустить практически на всех устройствах Android, имеющих камеру, а также на встроенном эмуляторе в Unity, благодаря чему тестировать приложение можно и с компьютера, и с мобильного устройства, а также приложение будет доступно большему количеству пользователей.

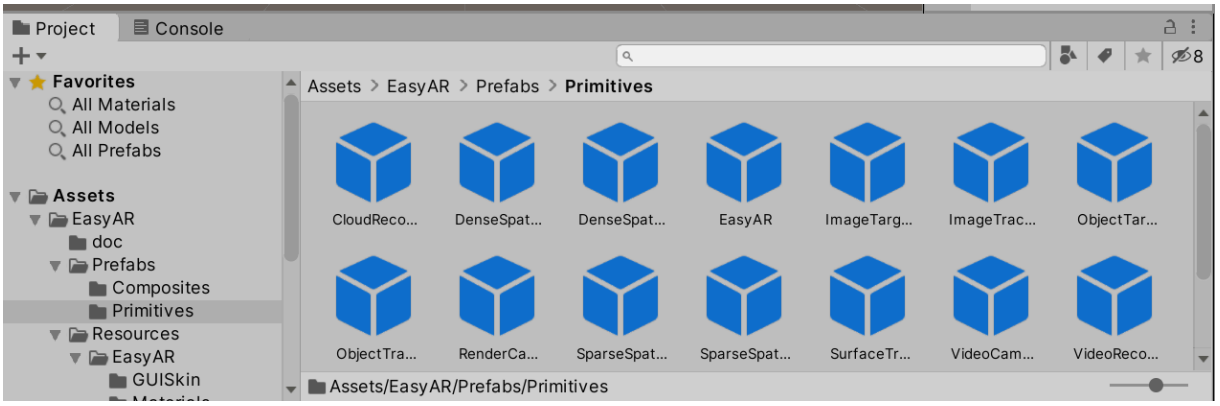


Рис. 10 – инструменты EasyAR

У EasyAR, как и у Vuforia, имеется бесплатная версия, но, в отличие от Vuforia, количество распознаваний, поддерживаемых бесплатной версией, примерно в три раза больше - по 100 распознаваний в день, что даёт 3000 распознаваний в месяц. Водяной знак у EasyAR небольшой и не отвлекает от происходящего на экране. Документация EasyAR практически отсутствует и состоит в основном из проектов-примеров и инструкций по их запуску, поэтому информацию по работе с SDK необходимо искать на сторонних ресурсах, где её тоже мало.

1.5. Итоги сравнения

В результате проведения сравнительного анализа вышеупомянутых инструментов разработчика, была составлена таблица с основными отличающимися характеристиками, повлиявшими на выбор. Также в таблицу был добавлен плагин Augmented Reality Plugin для Unreal Engine 4, который мне не удалось изучить подробнее из-за отсутствия бесплатной демо-версии, но который также рассматривался в качестве варианта.

Для создания приложения был выбран SDK EasyAR из-за наибольшего количества распознаваний в месяц и возможности тестирования без привлечения дополнительных устройств.

Таблица №1 – сравнение характеристик SDK

SDK/характеристики	простота и возможность тестирования	документация	водяной знак	количество распознаваний в месяц	стоимость
ARCore	тестирование оказалось невозможным ввиду отсутствия подходящего устройства	очень подробная	отсутствует	не ограничено	бесплатно

Vuforia	возможно, запускается и на эмуляторе, и на устройстве	большая часть документации устарела	присутствует, достаточно большой	1000 распознаваний	бесплатно с ограничениями
EasyAR	возможно, запускается и на эмуляторе, и на устройстве	практически отсутствует	присутствует, маленький	3000 распознаваний	бесплатно с ограничениями
Augmented Reality Plugin	невозможно ввиду невозможности покупки	очень подробная	отсутствует	не ограничено	100 евро

В результате проведённого сравнительного анализа были изучены, описаны и сравнены три SDK и один плагин для создания приложений дополненной реальности, а также выбран наиболее отвечающий условиям разработки планируемого приложения SDK и платформа разработки. Был сделан вывод, что оптимальным SDK является EasyAR, а оптимальной платформой – Unity. На выбор EasyAR во многом повлияло отсутствие искусственных ограничителей внутри SDK, и, следовательно, возможность тестировать и использовать приложение на большом количестве устройств. Дополнительным преимуществом в выборе также стало большое количество распознаваний метки, то есть, большее время работы приложения даже при использовании приложения с нескольких устройств одновременно.

1.6. Изучение инструментов и создание прототипа при помощи SDK Vuforia, проведение тестирования

Первый работающий прототип приложения был создан в Unity с использованием инструментов SDK Vuforia. В приложении производилось распознавание одной метки – изображения гнома, поверх которой на экране отображался куб.

Для создания приложения сначала была создана база данных в личном кабинете Vuforia, состоящая из одной метки – изображения гнома. Полученная база данных была скачана в формате .unitypackage и импортирована в проект.

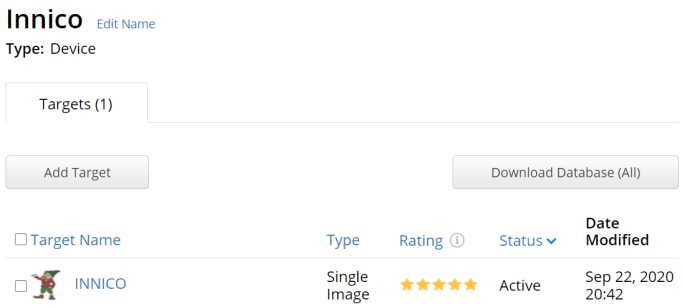


Рис. 11 – база данных

Также имеющийся изначально в проекте объект «Main Camera» был заменен на объект, предлагаемый SDK - «ARCamera», необходимый для работы с дополненной реальностью.

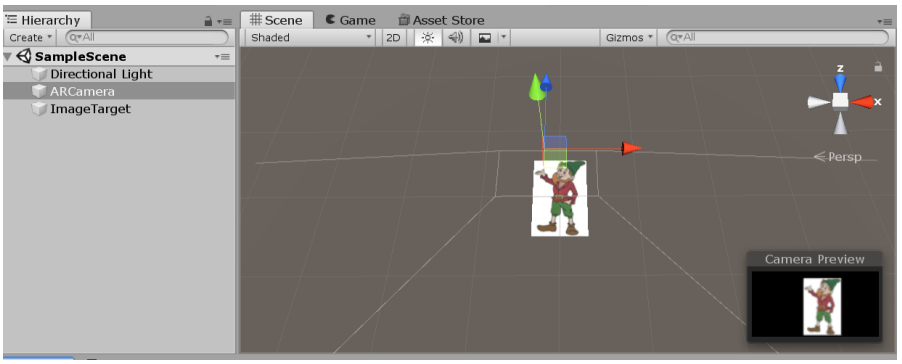
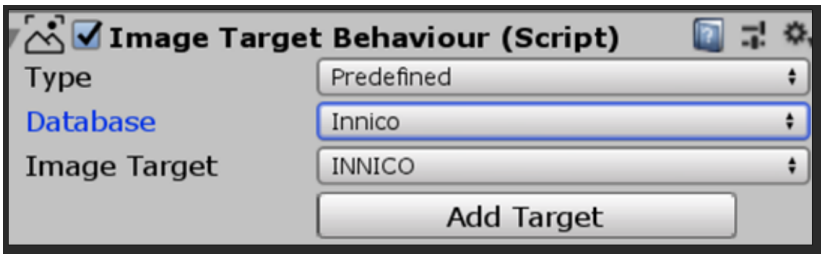


Рис. 12 – настройка ARCamera в проекте

При этом был автоматически проведен импорт необходимых недостающих инструментов в Vuforia, в том числе и необходимого ассета VuforiaConfiguration.asset, являющегося основным файлом настроек проекта, в котором используется Vuforia. В этом же ассете производится загрузка ключа лицензии, при помощи которого Vuforia распознает проект. При помощи инструмента SDK Image был создан объект ImageTarget, являющийся меткой, которую будет искать приложение. Также была

проведена настройка ARCamera таким образом, чтобы изображение занимало большую часть окна предварительного просмотра. В настройках ImageTarget необходимая база данных была прикреплена к метке. Для последующей работы был создан кубический контейнер, подключенный в иерархии к ImageTarget и расположенный над меткой в месте появления модели.



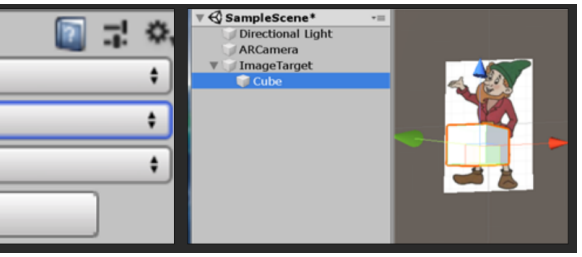


Рис. 13 – подключение базы данных. Рис. 14 – подключение контейнера

Над полученным приложением был проведён ряд тестов в эмуляторе Unity, чтобы удостовериться, что метка распознается, остается на месте (на гноме) при перемещении и изменении освещения, а также при вращении камеры вокруг метки.

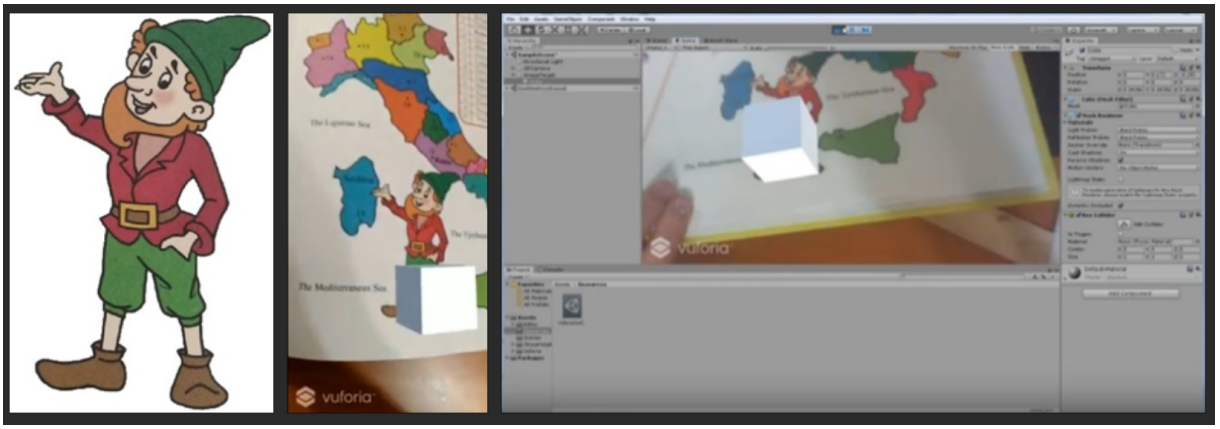


Рис. 15 – метка. Тестирование на мобильном устройстве и в эмуляторе.

Полученное приложение было собрано и скомпилировано в .apk-файл, который был загружен и установлен на мобильном устройстве. После чего был проведен ряд тестов, аналогичных тестам на эмуляторе в Unity для проверки корректности работы приложения.

1.7. Изучение инструментов и создание прототипа при помощи SDK EasyAR, проведение тестирования

Второй прототип приложения был создан на платформе Unity с использованием плагина EasyAR. EasyAR, как и Vuforia, распознает приложение при помощи ключа лицензии, генерируемого на сайте разработчиков EasyAR для каждого из проектов и добавляемого в поле для ключа лицензии в объект настроек в Unity. Скачанный плагин был импортирован в Unity вместе со всеми включёнными в него объектами.

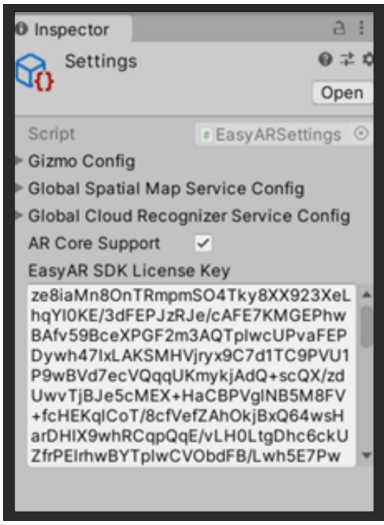


Рис. 16 – поле для ключа. Рис. 17 – генерация ключа лицензии на сайте

Для корректной работы приложения была произведена настройка сцены и действующих ассетов в соответствии с имеющимся примером настройки для создания AR приложения. В отличие от SDK Vuforia, где новые инструменты встраиваются в интерфейс Unity, инструменты EasyAR загружаются как префабы – особый тип ассетов, позволяющий хранить весь объект со всеми компонентами и значениями свойств, выступающий в роли шаблона для создания экземпляров хранимого объекта в сцене.

В EasyAR метки загружаются не в формате базы данных, а через загрузку в директорию Streaming Assets, после чего путь до метки прописывается в объекте ImageTarget. К объекту Image Target с определенной меткой также

прикрепляется соответствующая метке модель. Для неодновременного отслеживания двух и более меток создаётся несколько объектов ImageTarget, каждый со своей меткой и объектом, срабатывающих при появлении необходимого изображения в поле видимости камеры. Полученное приложение было скомпилировано и протестировано на эмуляторе и мобильном устройстве.

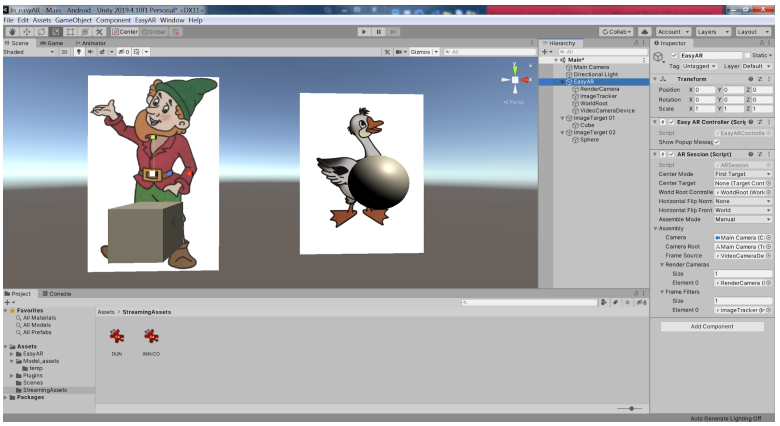


Рис. 18 – Настройка сцены и взаимодействия инструментов EasyAR.

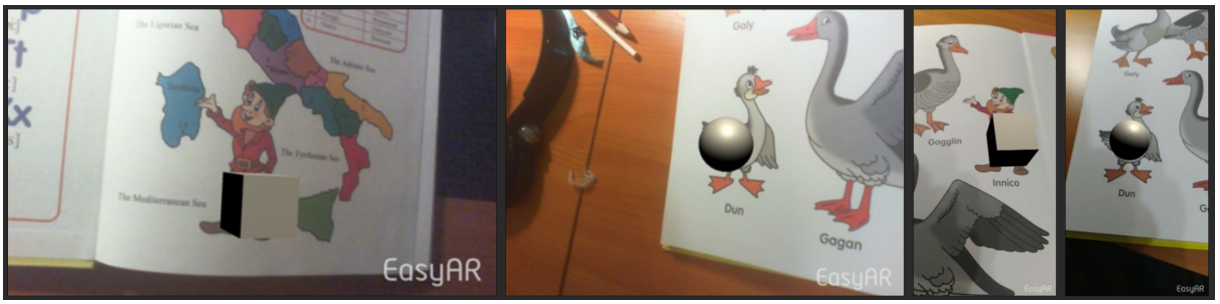


Рис. 19 – Тестирование срабатывания двух меток на эмуляторе и мобильном устройстве.

1.8. Тестирование модели с анимацией и текстурой

Следующий этап тестирования и создания приложения включал в себя обеспечение правильного отображения текстур и анимации на моделях. Для упрощения работы на данном этапе были использованы не финальные модели, а специальная модель для тестирования. В программе 3D-моделирования Maya была создана низкополигональная модель колеса со смещёнными спицами для лучшей читаемости силуэта при

тестировании. Для колеса также была создана простая анимация перемещения в пространстве одновременно с вращением и шахматная цветная текстура для лучшей видимости отображения в приложении. Модель вместе с текстурой и анимацией была импортирована в Unity, подключена к соответствующему ImageTarget. Для работы анимации был создан и настроен отдельный управляющий объект – Animation controller, к которому была подключена анимация колеса, после чего компонент был подключен к самой модели колеса.

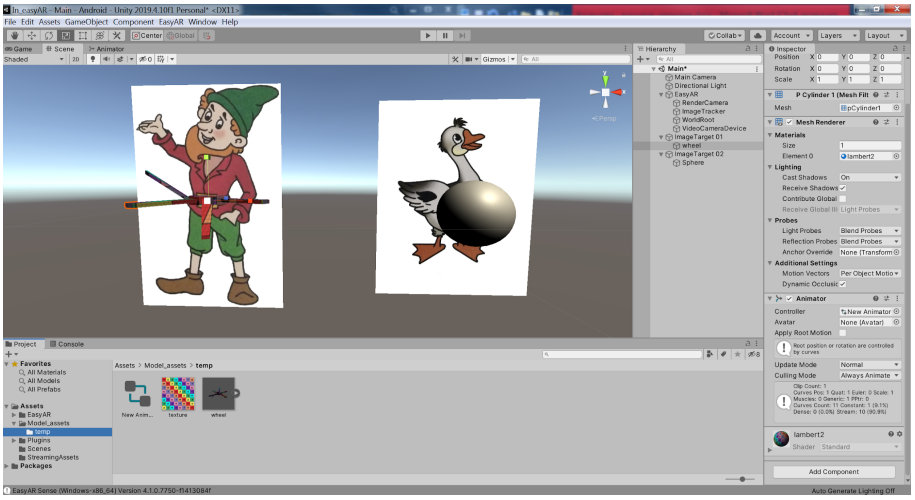


Рис. 20 – настройка текстуры и анимации

Дополненное приложение было протестировано в эмуляторе Unity и, после компиляции в .apk-файл, установлено и протестировано на мобильном устройстве. Тестирование показало, что приложение работает корректно: на всех устройствах обе метки правильно распознаются, над ними отображаются установленные модели, проигрывается анимация колеса, а также правильно отрисовывается текстура. Приложение работает даже при плохом освещении и быстром перемещением меток.

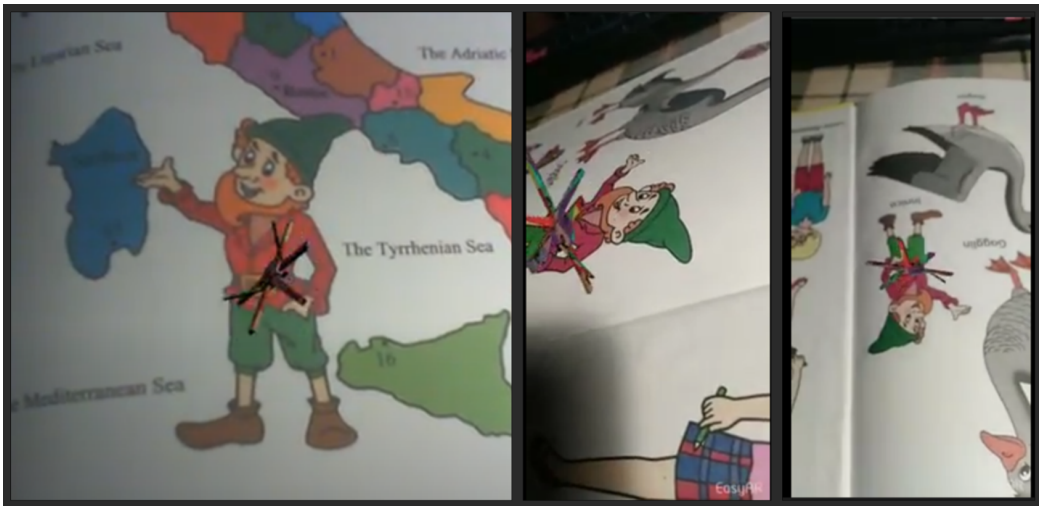


Рис. 21 – тестирование текстур и анимации на эмуляторе и мобильном устройстве

1.9. Тестирование различных типов меток

Полученное в результате проведенной работы приложение отвечало изначальным требованиям, поэтому было принято решение внести новые типы меток и провести дополнительное тестирование.

Учебное пособие, для которого создавалось приложение, является книгой-раскраской, где примерно 40% страницы занимает чёрно-белая иллюстрация с заголовком и ещё примерно 40% - небольшой рассказ о происходящем на иллюстрации. Ранее рассмотренные метки – изображения гланых героев, находящиеся на форзаце книги. Новые типы меток включали в себя: чёрно-белое изображение с надписью, две похожих версии чёрно-белого изображения, раскрашенную чёрно-белую метку и отдельную надпись.

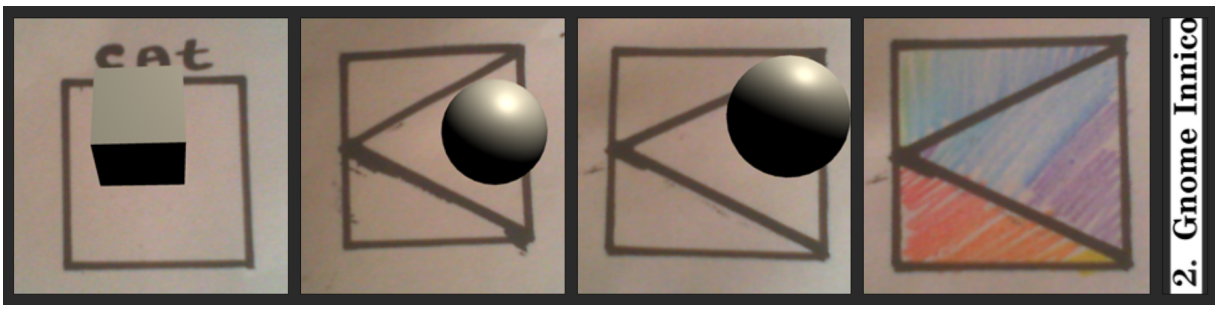


Рис. 22 – новые типы меток

По результатам тестирования простая метка распознавалась в чёрно-белом состоянии, но после её раскрашивания или изображения чего-то внутри программа переставала её воспринимать. Изображение, похожее на метку, но не дублирующее её, распознавалось, и поверх изображения появлялась модель. Не распознавалась, как метка, и надпись, что было связано с её малым размером и невозможностью приложения выделить опорные области и точки. Из-за выявленных особенностей работы меток было принято решение сфокусировать приложение на изображениях на форзацах и обложке, а также предложить издательнице добавить к книге наклейки, которые могли бы выполнить роли меток на необходимых страницах. Для проверки возможности такого решения был проведен тест с созданием сетки из моделей, размещённых так, чтобы покрывать зону гораздо большего размера, чем метка, а именно, гном. Подобные модели отображались корректно даже в условиях скудного освещения, что подтверждало работоспособность идеи.

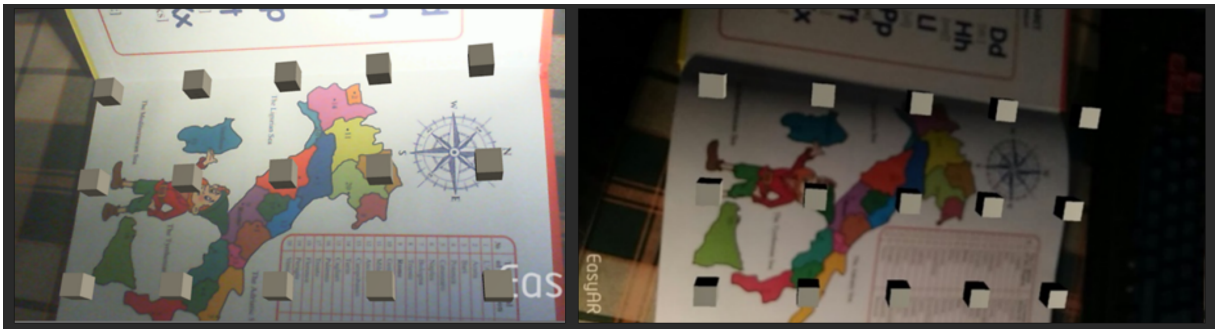


Рис. 23 – проверка отображения моделей на расстоянии от метки в эмуляторе и на мобильном устройстве в условиях скудного освещения.

2. Создание модели гнома.

2.1. Сбор концептов и референсов

Создание любой 3D-модели начинается с поиска необходимых референсов – изображений, на детали которых опирается моделлер при работе. Также часто для работы необходимы концепт-арты – изображения, передающие основную идею того, что будет создаваться. Стоит также учитывать, что зачастую невозможно полностью перенести в объемный формат плоское изображение – какие-то красиво выглядящие на изображении детали могут выглядеть некрасиво в реалистичной перспективе, или вовсе быть невозможными. В качестве основных референсов было использовано изображение гнома из самой книги, а также иллюстрации моей коллеги Дарьи, в которых гном был поставлен в Т-позу для более наглядного изображения пропорций, особенностей, расположения типа одежды.

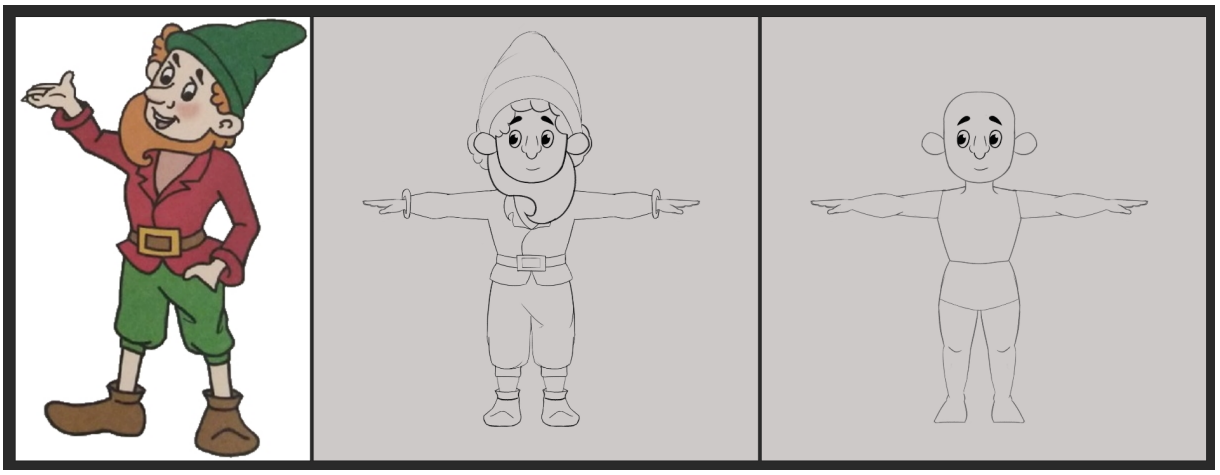


Рис. 24 – основные и опорные концепты

Кроме концепт-артов, выступающих в качестве основы, был также использован ряд дополнительных иллюстраций и изображений моделей других художников. Референсы были разделены на подгруппы для более удобной работы:

1. Образ – набор изображений со схожим образом и стилевым решением подобной 3D модели – достаточно стилизованные существа с «детскими»

пропорциями, с ярко выраженными, будто нарисованными, складками на одежде и общим образом радости и открытости.

2. Особенности подобных моделей – набор изображений с моделями, при создании которых были использованы техники работы, схожие с планируемыми для использования.

3. Одежда и волосы – две группы, включающих в себя изображения с 3D-моделями, волосы и детали одежды которых имеют схожую стилизацию с выбранной для работы над гномом.

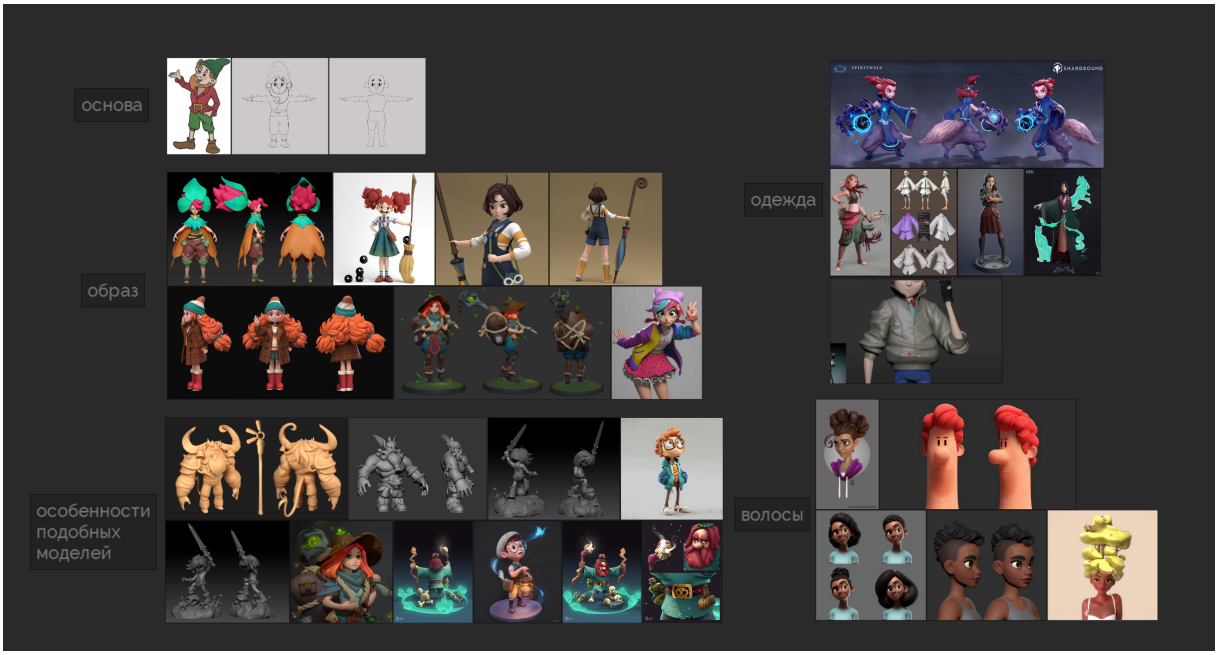


Рис. 25 – вспомогательные референсы

Третьим типом референсов стали иллюстрации, связанные с анатомией человеческого тела. Для создания аккуратной и анатомически правильной стилизации необходимо было проанализировать обрубовки головы и тела и соотнести с имеющимися концепт-артами гнома. Большое количество информации для последующей работы было взято из книги Улдиса Заринса и Сандиса Кондратса «Анатомия для скульпторов. Понимание структуры тела человека».

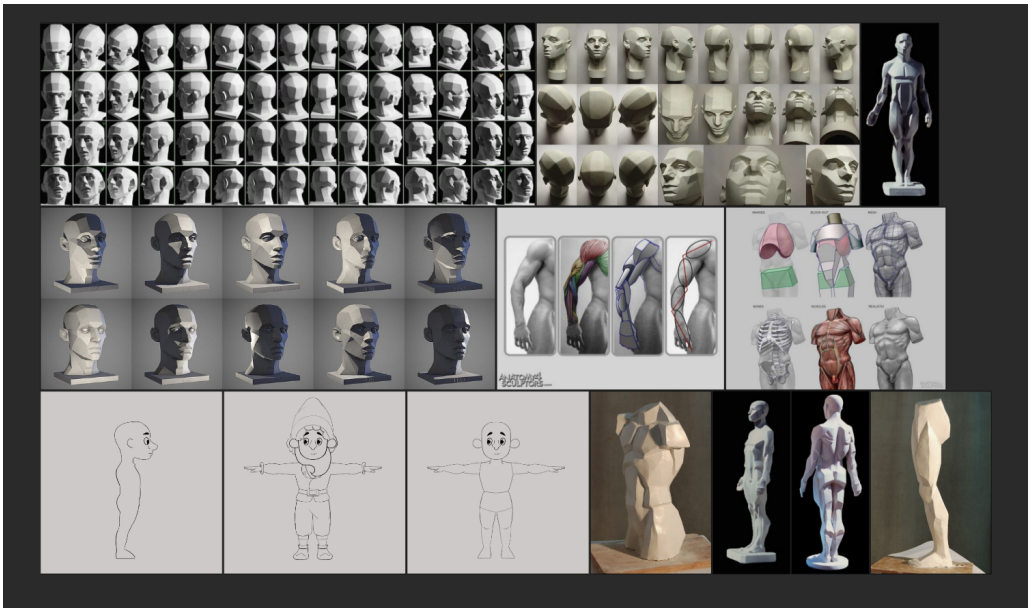


Рис. 26 – референсы анатомии тела и головы.

2.2. Создание базовой модели гнома

Создание базовой модели, или basemesh – этап, на котором закладываются основные пропорции, размеры модели и отдельных ее составляющих. Для работы на данном этапе использовалась программа Maya. Концепт-арты с изображением тела гнома в профиль и анфас были размещены в Maya под углом 90°, тем самым становясь частью виртуальной студии – набора плоскостей с проекциями модели. При помощи инструмента Extrude, вытягивающего полигоны, Cut, создающего новые ребра, Move и Rotate, перемещающих и поворачивающих полигоны, а также Mirror и Symmetry, отвечающих за симметрию модели было создано низкополигональное тело. После сетка была уплотнена и сглажена при помощи инструмента Smooth.

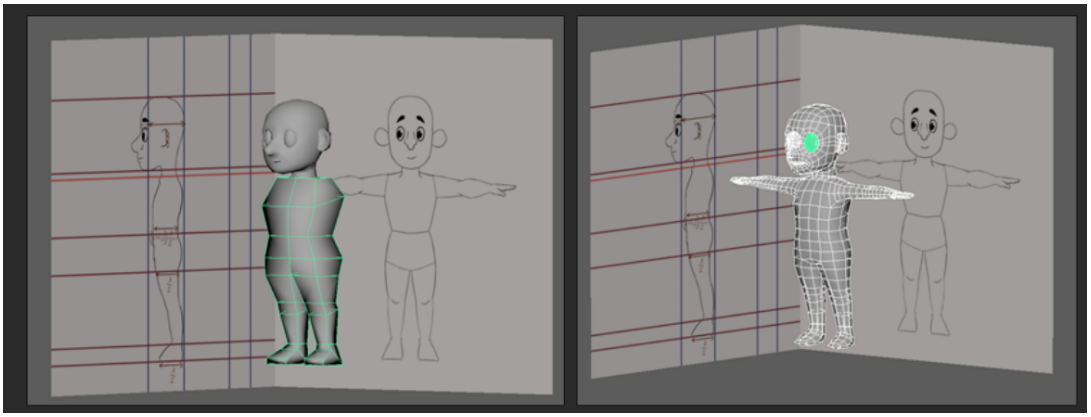


Рис. 27 – создание низкополигональной и сглаженной сетки

Для большего сходства с концепт-артами часть точек модели была передвинута при помощи инструмента Soft Selection, при помощи которого можно перемещать точки не по одной, а группами, с разным распределением весов смещения. Полученная модель была экспортирована в файл формата fbx для переноса в другую программу на следующем этапе.

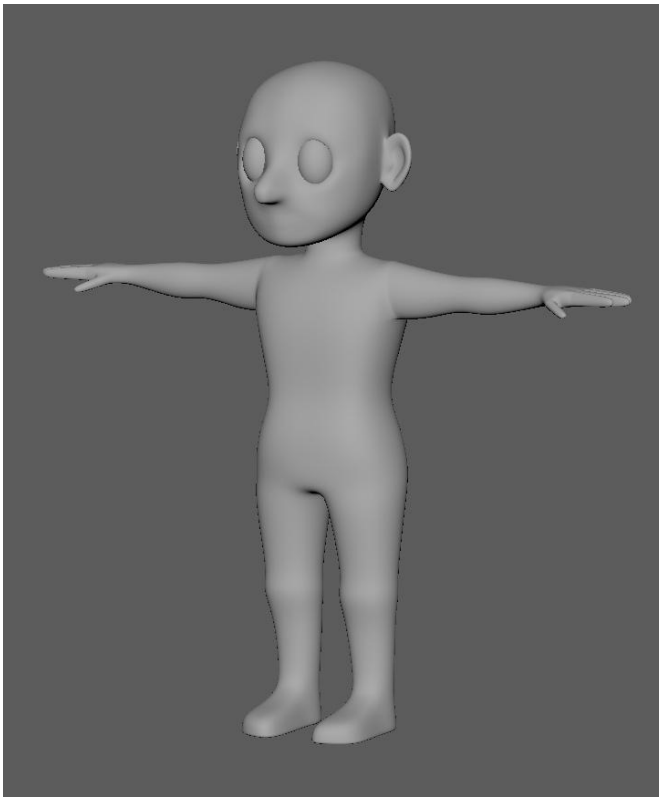


Рис. 28 – результат работы этапа базовой модели

2.3. Создание высокополигональной модели гнома

На предыдущем этапе проводилась с точками, полигонами и ребрами. На этапе высокополигонального моделирования использовалась программа Zbrush, техника работы в которой больше похожа на работу с глиной, чем с геометрией – подобный эффект достигается за счёт увеличения количества полигонов в модели до сотен тысяч и даже миллионов и наличия специальных инструментов для работы в таком формате. Все основные формы, а также детализация, создается именно на этом этапе. Иногда мелкая детализация, такая как швы, потертости, текстура ткани, маленькие царапины, поры кожи и так далее добавляется не на этапе моделирования, а на этапе текстурирования, но ввиду того, что модель создавалась для мобильного приложения, что приводило к её небольшому размеру и невидимости, а также ввиду выбранной стилизации, в которой подобные небольшие детали не создаются, было принято решение их не добавлять совсем.

Первоначальная подготовка модели к работе в программе Zbrush заключалась в экспорте, сглаживании и увеличения количества полигонов сетки инструментом Divide, создающим несколько уровней Subdivision с разным количеством полигонов, благодаря которому возможно создание крупной детализации на низких уровнях без разрушения средней детализации на верхних уровнях, и наоборот. На этапе подготовки также были внесены небольшие изменения пропорций, которые было бы тяжело добавить на этапе basemesh.

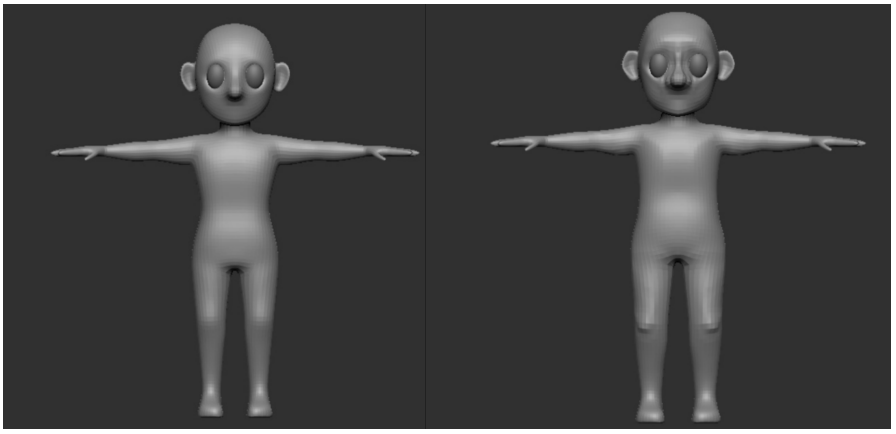
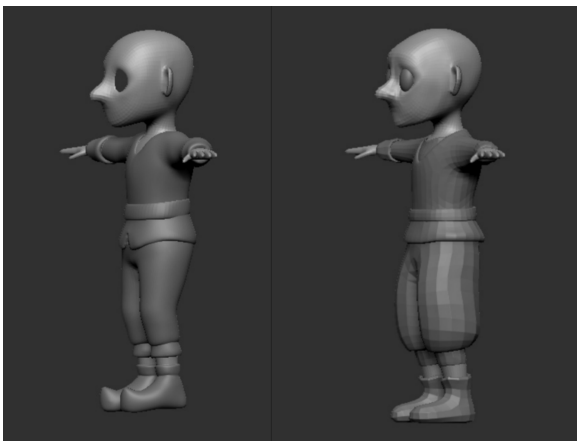


Рис. 29 – результат сглаживания и уточнения пропорций

Для того чтобы сделать финальную модель как можно более интересной и приближенной к изначальным концептам, было создано несколько вариантов одежды разными методами. Самый первый и самый простой включал в себя выделение части тела при помощи маски с применением инструмента Extract, добавляющего сетке толщины и выносящего её в новый объект. Полученные части одежды повторяли форму тела гнома, но были неровными и не читались на силуэте. Для повышения уровня аккуратности одежда была перемоделирована с применением улучшенной техники: после применения инструмента Extract в части одежды через ZRemesher уменьшалось количество полигонов, благодаря чему появлялись нижние слои сглаживания, на которых выстраивалась ровная сетка, которая при сглаживании давала более аккуратный результат.



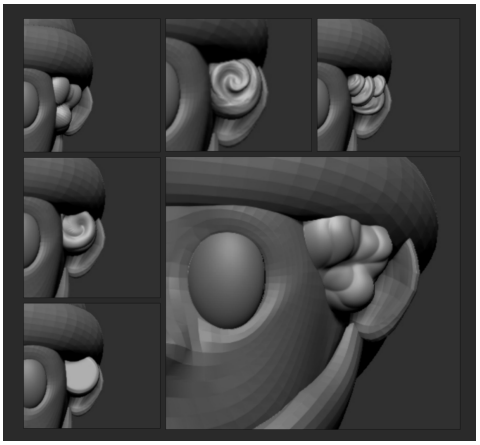


Рис. 30 – одежда гнома. Рис. 31 – волосы гнома

Кроме одежды, эксперименты над техниками и стилизацией проводились и при создании волос гнома. Всего было опробовано шесть различных вариантов изображения волос, из которых наиболее подошел метод создания общей массы из сферы при помощи ClayBuildup и разбиение на пряди при помощи узких линий DamStandart.

Во время работы были привлечены дополнительные источники информации – такие, как видеоуроки и статьи по созданию моделей и работе в программе Zbrush. В результате постоянного изучения и анализа новой информации уровень навыков работы повышался, из-за чего некоторые части гнома пришлось полностью переделывать два и более раза – чтобы общий уровень стилизации и детализации был одинаковым по всей модели. Одной из таких частей стало лицо гнома – изначальный basemesh был дополнен деталями, а полученное лицо через какое-то время работы было переделано на более анатомичное: добавлены брови, увеличена толщина век, проведены более четко плоскости носа и скул, изменена форма губ и глаз.

Рис. 32 – изменение пропорций лица персонажа

Подобные изменения коснулись и шляпы гнома, складки которой были исправлены на более аккуратные и физически правильные, а также кожаной куртки, ворот которой стал больше похож на прототип, добавилась толщина и складки вокруг пояса. При помощи Polypaint на модели появился цвет и стала отчётливей видна детализация.

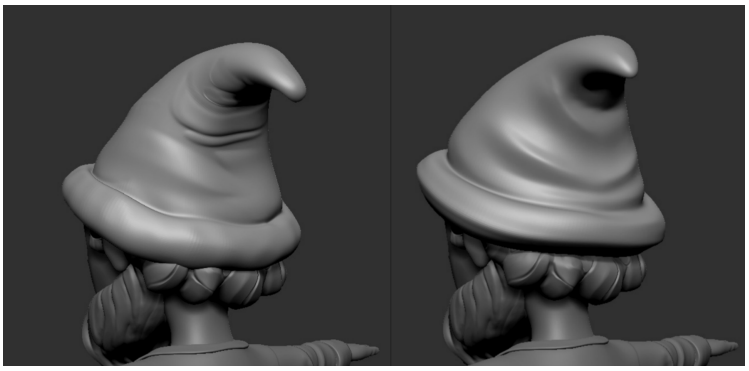


Рис. 33 – изменение шляпы гнома

Рис. 34 – изображения гнома в процессе (слева) и в финальной детализации (справа)

2.4. Создание низкополигональной модели гнома

Большая детализация модели в Zbrush несёт в себе и большой полигонаж, из-за которого модель имеет большой вес и не может быть импортирована напрямую в приложение. Даже с максимальным уменьшением веса модели через Zremesher при добавлении её в приложение могут возникнуть трудности при запуске даже на самых мощных устройствах. Для увеличения производительности модель проводят через этап ретопологии, во время которого создаётся низкополигональная модель, состоящая из минимально необходимого для передачи силуэта количества полигонов. Подобную модель можно создавать как в Zbrush, так и экспортировав высокополигональную модель в Maya. Для разработки был выбран второй вариант, с экспортом модели в Maya. Подготовка модели заключалась в использовании специального плагина в Zbrush – Decimation Master, при помощи которого производится пересчёт сетки и её упрощение с сохранением детализации.

Инструмент Quad Draw в Maya нужен для создания новой топологии поверх старой. В Quad Draw включены инструменты, реализующие такие действия, как: создание полигона между установленных точек, вытягивание ребра полигона в новый полигон поверх сетки выбранной модели, вытягивание края сетки или петли полигонов, выбор расстояния между новой и старой сеткой, разглаживание сетки.

Часто на этапе ретопологии используют изначальный basemesh, немного изменяя его, чтобы силуэты низко- и высокополигональной модели лучше совпадали. В случае гнома осуществлению этого метода мешала одежда, созданная на этапе работы в Zbrush и не отраженная в basemesh, поэтому новая сетка была выстроена с нуля.

При создании низкополигональной модели также важно учитывать её последующее использование: будет ли она частью иллюстрации или игры, будет отображаться на компьютере или мобильном устройстве, является ли частью окружающей среды, неким предметом или героем, будет ли модель анимирована. Предполагалось, что модель гнома будет главным героем в приложении мобильного устройства, а также, что гном будет совершать движения – перечисленные факторы повлияли на плотность сетки – достаточно малую, чтобы хорошо читаться с небольшого экрана и уплотнённую в районе лица и рук для лучшей анимации, а также её особенности, например, на преимущественное составление сетки из квадратных, а не треугольных, полигонов, а также на расположение колец рёбер вокруг глаз, носа, рта и сгибов тела.

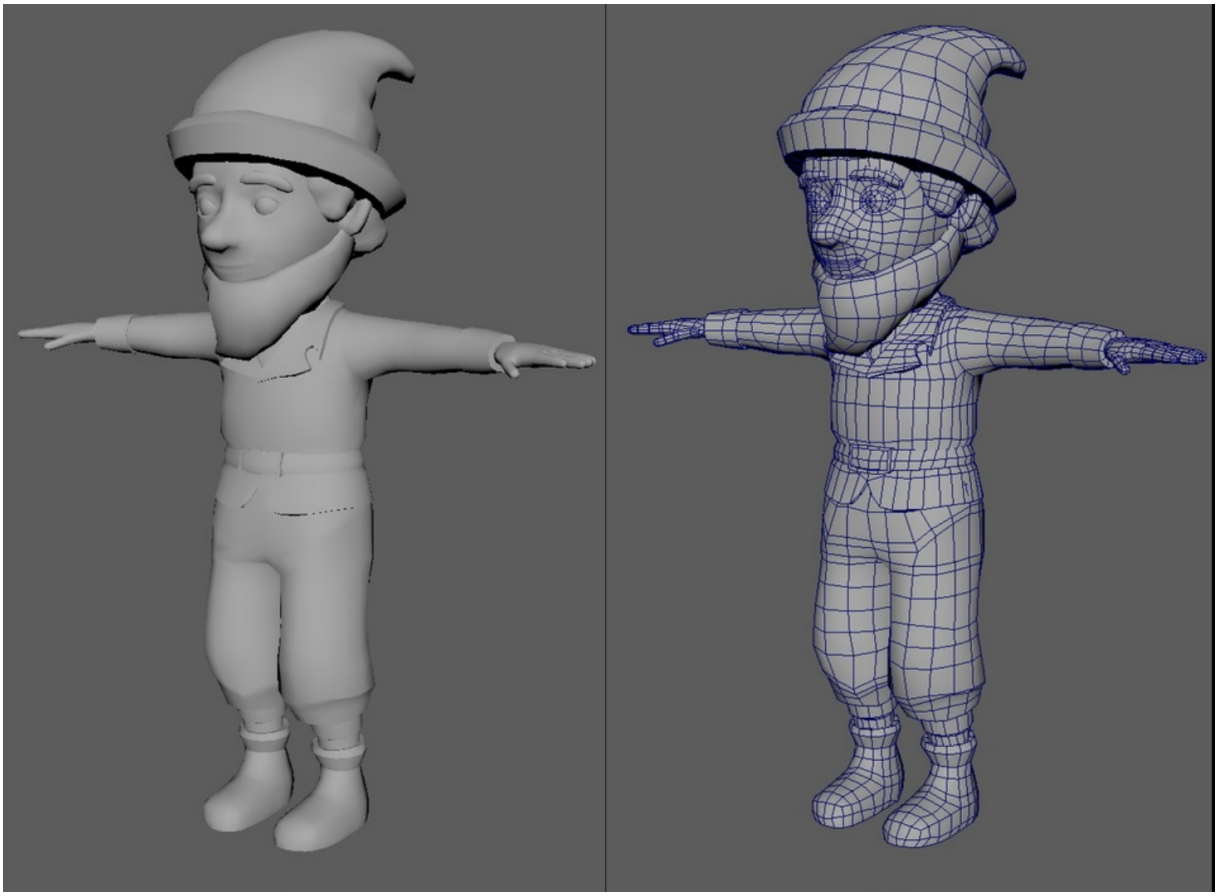


Рис. 35 – низкополигональная модель гнома и сетка модели

2.5. Запечка текстур

2.5.1. Создание UV

Для того чтобы модель можно было текстурировать - добавлять цвет и детализацию, производят процесс развертки: UV-преобразования, то есть, преобразования координат поверхности объемного объекта, находящегося в координатах X, Y, Z в координаты на текстуре – U, V.

Во время процесса развертки на модели выбираются рёбра-швы, по которым «разрезается» поверхность, создавая острова – отдельные участки развертки. Важно создавать швы в наименее заметных местах, а также располагать их так, что для уплощения острова необходимо было минимально растягивать точки на текстуре, чтобы впоследствии из-за большого растяжения не появлялись искажения. Те острова, которые можно растянуть в прямоугольники, квадраты или длинные узкие полосы, обычно растягивают до соответствующей фигуры, чтобы реализовать больше полезного пространства на текстуре.

Во время создания развертки гнома приходилось решать сложности, связанные с тем, что гном – существо, и, следовательно, органика, а развертку органических предметов практически невозможно сделать с идеальным выполнением всех важных условий развертки, что приводит к частым выборам между растяжением и швом, или между растяжением и аккуратным островом развертки. Анимация также влияла на выбор, потому что во время движения модели её части будут растягиваться и сжиматься, что означало что в некоторых местах, например, плечах, локтях, коленях искажение текстур неизбежно, и небольшая деформация на UV не повлияет не общую картину. Было создано два варианта UV – обычный и максимально оптимизированный так, чтобы занять как можно большее пространство текстуры. Часть островов, такая, как, например, руки и ботинки, была наложена друг на друга, чтобы сэкономить пространство и

упростить в будущем процесс текстурирования рисованием одной руки вместо двух.

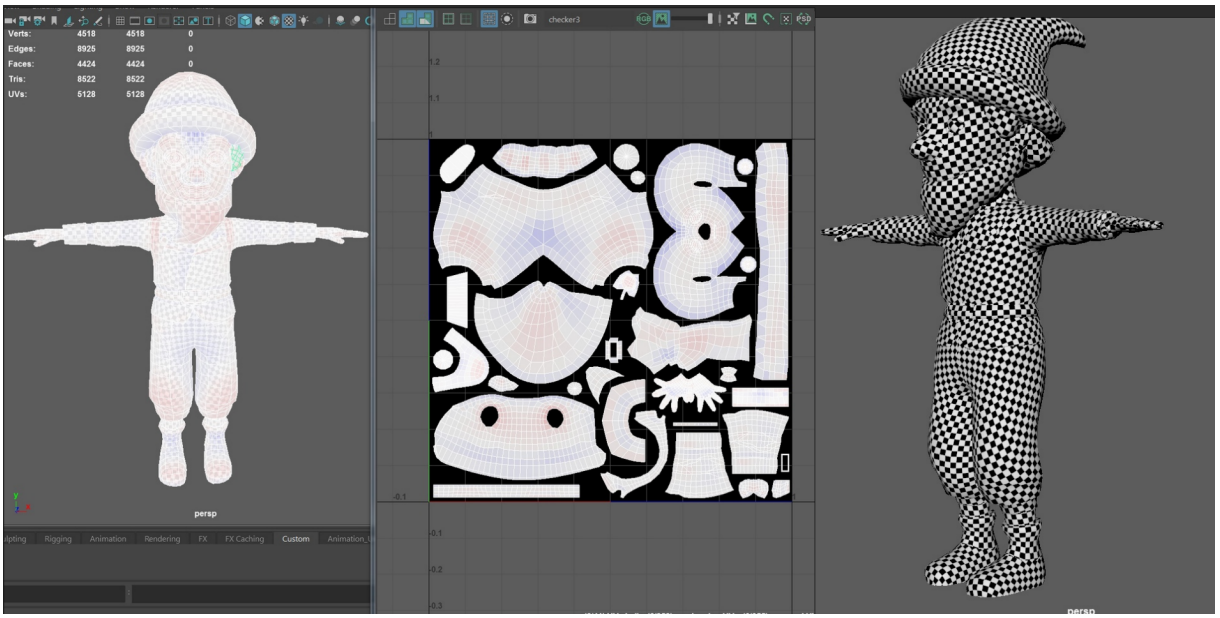


Рис. 36 – процесс создания развёртки

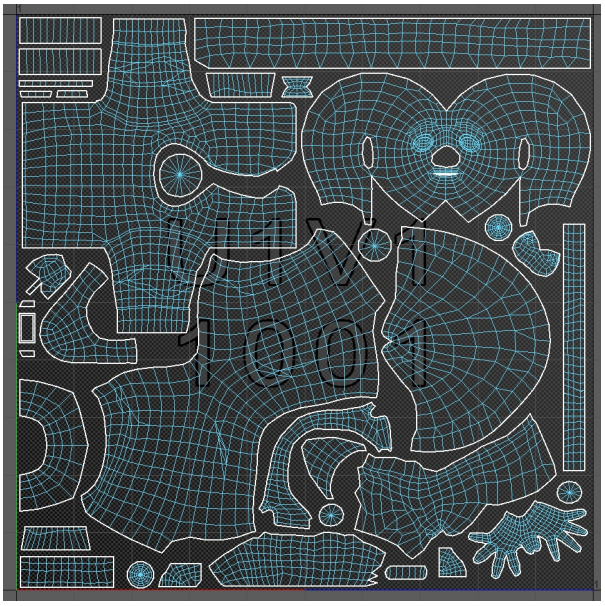


Рис. 37 – итоговая UV-развертка

2.5.2. Запечка карты нормалей

Карта нормалей – это специальная текстура, цвета которой обозначают вектор смещения точки модели в стороны, что приводит к изменению поведения блика таким образом, как если бы на низкополигональной

модели была детализация высокополигональной. Процесс переноса информации с высокополигональной модели на карту нормалей низкополигональной называется запечкой.

Для создания карты нормалей была выбрана программа Marmoset Toolbar, в которую были загружены высоко- и низкополигональные модели гнома, распределенные по разным частям в отдельные непересекающиеся группы запечки. Для ряда объектов – для рук, штанов и куртки гнома был дополнительно настроен Cage – геометрия, ограничивающая пространство, из которого берется информация для переноса.

Кроме карты нормалей, для последующей работы с текстурами также были запечены AO – карта с затемнениями мест, куда с трудом проникает свет и ID map – карта, при помощи которой разным частям модели даются собственные идентификаторы в виде цветов, что упрощает процесс текстурирования.

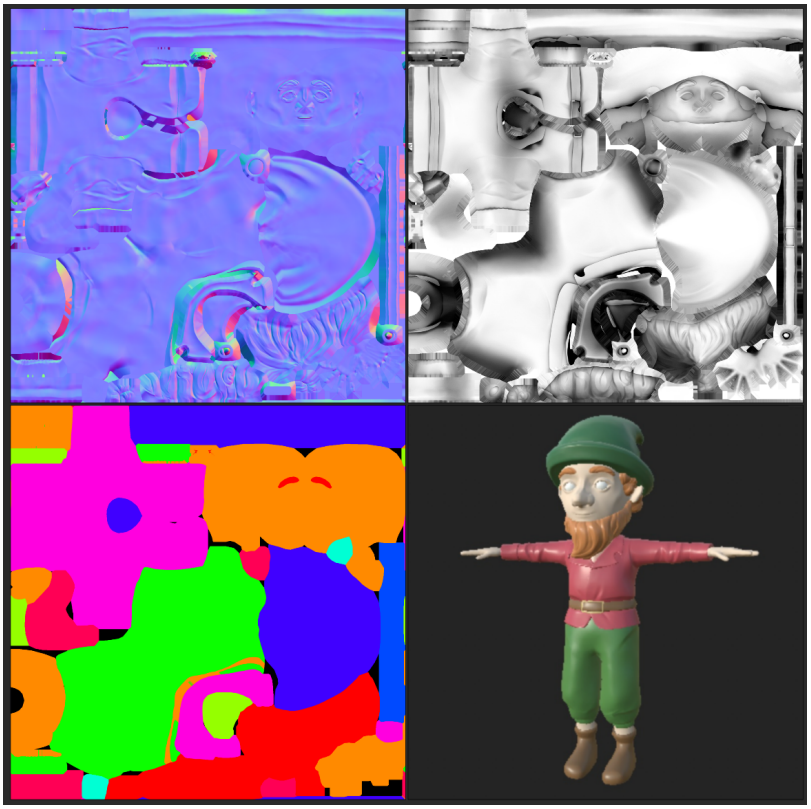


Рис. 38 – полученные карты Normal, AO, ID, изображение гнома с применёнными картами

2.6. Текстурирование

Для текстурирования была выбрана программа Substance Painter, в которую была загружена низкополигональная модель, а также запеченные на предыдущем этапе карты.

Для поддержания общего «сказочного» стиля книги-раскраски было решено создавать текстуры в стиле Handpaint – ручной рисовки. Подобная методика позволяет модели выглядеть более мягко и мультипликационно, что являлось хорошим дополнением для детской книги.

Основной метод создания текстур в Substance Painter – настройка процедурных слоёв для получения базы текстур с последующим дополнением базы нарисованными вручную деталями. Для поддержки единого вида всего гнома в качестве базы был создан «умный материал» - материал, использующий запеченные ранее, а также запеченные в программе карты в качестве масок, через которые добавляется цвет. При использовании чёрно-белого изображения как маски, черный цвет обозначает 0, не пропуская информацию, а белый – 1, передавая всю информацию дальше. Промежуточные значения варьируются в этом промежутке.

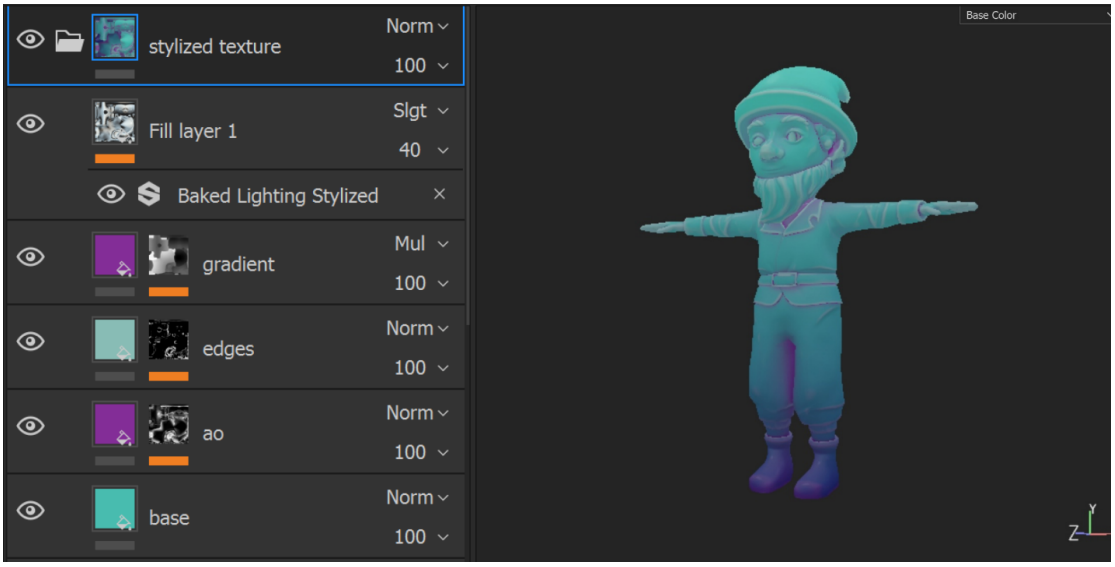


Рис. 39 – применение «умного материала» на модель

Созданный умный материал stylized texture работает следующим образом:

1. Вся модель заливается основным цветом из слоя «base»;

2. В слое «AO» цвет тени накладывается на текстуру, при этом в качестве маски используется AO карта – карта высокодетализированных теней;

3. В слое «Edges» на края модели наносится осветленный базовый цвет. Подобный эффект достигается при помощи применения карты кривизны в качестве маски, где белым обозначены выпуклые части модели.

4. Слой «Gradient» добавляет общий градиент от прозрачного цвета к цвету теней, при этом центр и края градиента можно настраивать.

5. Самый верхний слой имитирует тени и блики от освещения – таким образом производится экономия ресурсов отрисовки и просчёта освещения в приложении.

Полученный материал был продублирован в количестве 10 штук, равном количеству цветов, использованных в дизайне гнома. После чего на каждый из дубликатов была применена маска, сделанная при помощи ID Map, ограничивающая действие материала на определенной области, например, на шапке, штанах, рубахе. Внутри каждого подматериала была

произведена настройка цветов в соответствии с изначальным концептом, а также теорией цвета. Кроме цвета слоёв, дополнительно настроены были и внутренние свойства масок, такие как, например, размытость краёв осветления, контрастность затемнения, зашумленность теней и другие.

Особенность Substance Painter состоит в том, что наносить текстуры можно как напрямую на модель, так и на ее UV-проекцию, при этом нанесенная текстура будет параллельно отображаться и там, и там.

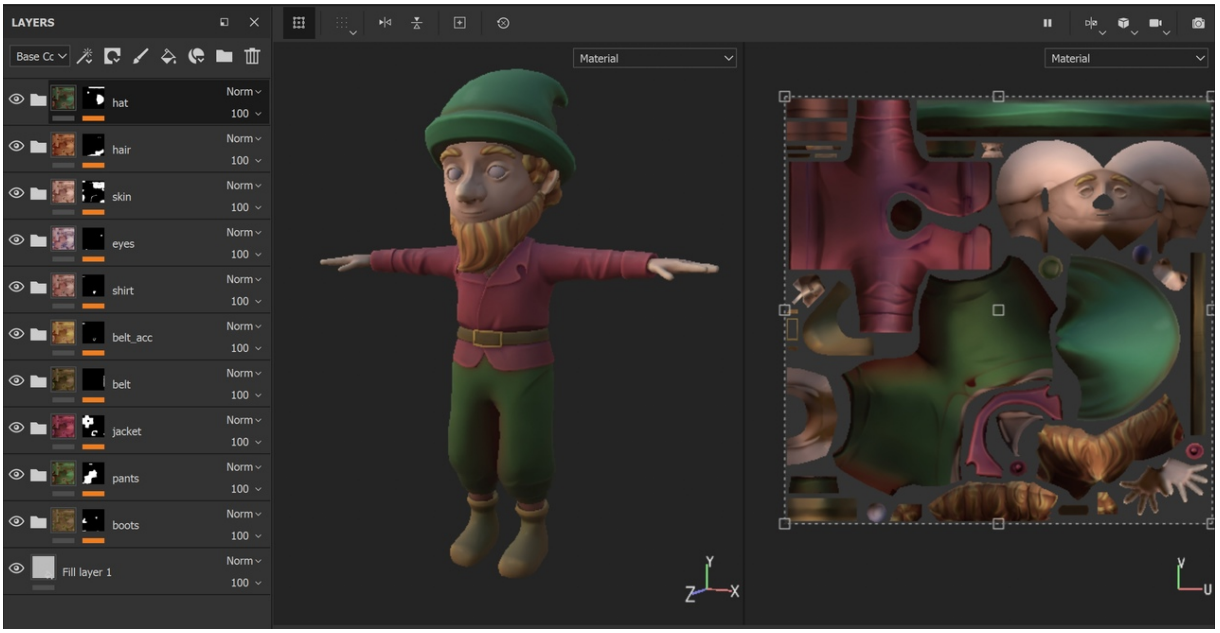


Рис. 40 – результат настройки материалов

Полученная база была доработана вручную: добавлены недостающие и убраны лишние тени, нарисована радужка и зрачки, дорисован плавный переход от волос к коже.

Рис. 41 – итоговая модель

Полученные текстуры были настроены для последующего импорта в unity и экспортированы.

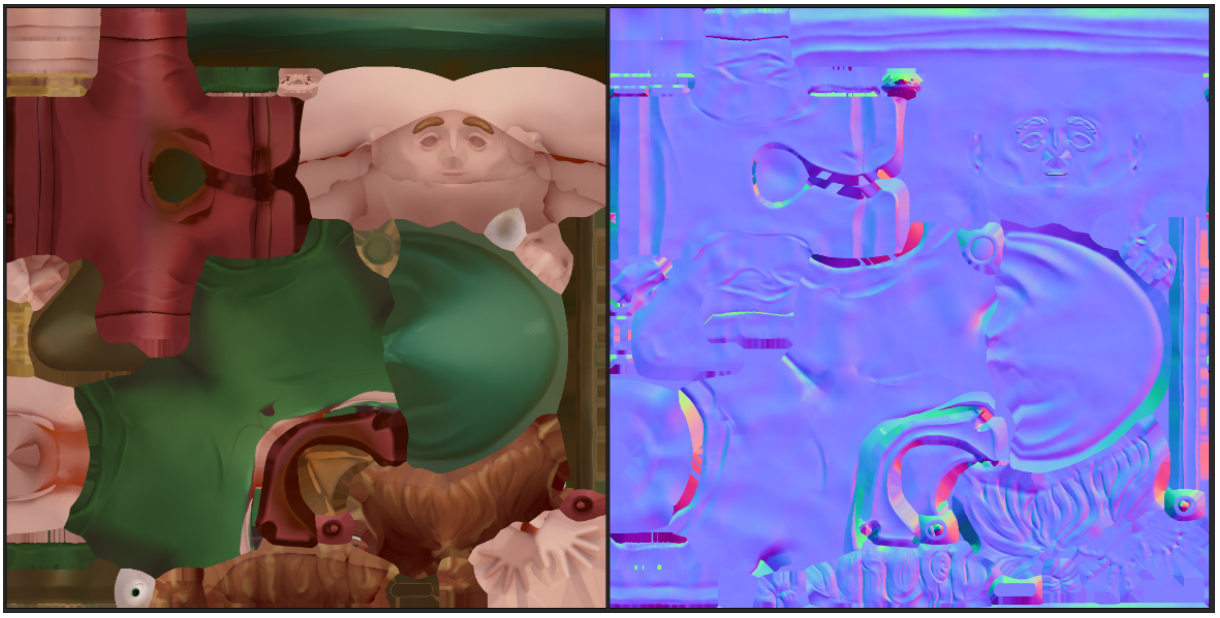


Рис. 42 – итоговые текстуры

3. Создание анимации

3.1. Создание рига.

3.1.1. Создание рига тела

Для того чтобы модель можно было анимировать, создаётся скелет – набор объектов-костей, выстроенных в определенную иерархию, влияющую на взаимодействие костей друг с другом, к которому привязываются точки модели. Анимация может создаваться как при помощи перемещения костей скелета вручную, так и при помощи управления скелетом через риг – набор сплайнов, привязанный к костям, при перемещении/вращении которых изменяется положение костей в пространстве.

Поскольку гном является двуногим человекоподобным существом, было принято решение не создавать скелет и риг с нуля, а использовать плагин advanced skeleton, в котором имеется ряд уже готовых скелетов, открытых для редактирования, в том числе и для двуногих существ. Кости встроенного в плагин скелета были перемещены таким образом, чтобы соответствовать пропорциям и правильному расположению костей рига персонажа для анимации относительно частей тела. Так, например, голова гнома гораздо больше, чем стандартная голова человека, из-за чего кость верхушки головы была перемещена выше.

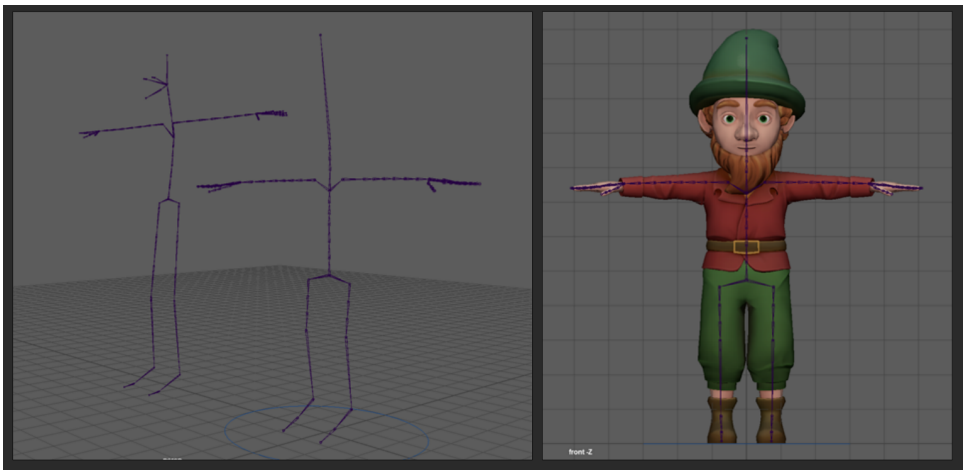


Рис. 43 – скелет гнома

Особенность рига в Advanced skeleton состоит в том, что на каждой кости можно настраивать значения атрибутов, основываясь на которых, плагин создает различные механики движения, например, IK и FK (прямая и обратная кинематики), плавное вращение, независимость движения одной кости от костей, стоящих выше по иерархии.

Прямая кинематика (IK) позволяет при перемещении/вращении одной кости автоматически передвигать все кости ниже по иерархии, так, как если бы вся цепочка костей была сделана из единого проволочного каркаса. Обратная кинематика (FK) действует обратным способом: при перемещении конечной кости из цепочки вся цепочка автоматически подстраивается под новое расположение кости: таким образом можно перемещать, например, кисть или ступню, и при этом локоть и колено модели будут сгибаться, как в реальности. Атрибуты скелета гнома были настроены таким образом, чтобы создать IK, FK контроллеры, а также переключатель между кинематиками на всех конечностях, а также спине героя.

Полученный риг был привязан к модели при помощи встроенной в плагин автоматической функции привязки и протестирован путём расположения модели в позах с максимальным изменением расположения костей, а также использованием всех имеющихся костей и кинематик.

Рис. 44 – процесс привязки скелета

3.1.2. Скиннинг

Несмотря на то, что результат, полученный при помощи встроенной в плагин функции привязки точек модели к скелету, был приемлемым, его всё ещё необходимо было доработать. Этот процесс называется скиннингом. При тестировании модели был выявлен ряд областей на модели, неправильно реагировавших на перемещение костей. Так, например, боковые точки бороды оставались на месте при перемещении головы, поскольку кости плеча влияли на них больше, чем кость головы, а точки, расположенные на внутренней стороне штанов находились под небольшим влиянием костей соседних ног, из-за чего при перемещении левой ноги вслед тянулась и штанина правой, и наоборот. Часть проблемных точек обнаружилась и на спине гнома – при создании анимации с большой амплитудой движения они оставались на месте.

На каждую из точек модели влияет одна или несколько костей, каждая – со своим коэффициентом силы, называемым весом. Для корректировки весов был выбран плагин ngtools, позволяющий изменять влияние костей, раскрашивая на самой модели области, на которые должна влиять та или иная кость. При этом, в отличие от встроенного инструмента программы Maya, у ngtools есть функция рисования весов на разных слоях с разными режимами взаимодействия друг с другом, благодаря чему можно исправлять разные части модели, не боясь испортить уже правильно настроенные области.

Полученная система управления позой модели была также протестирована перемещением и вращением костей при помощи направляющих рига.

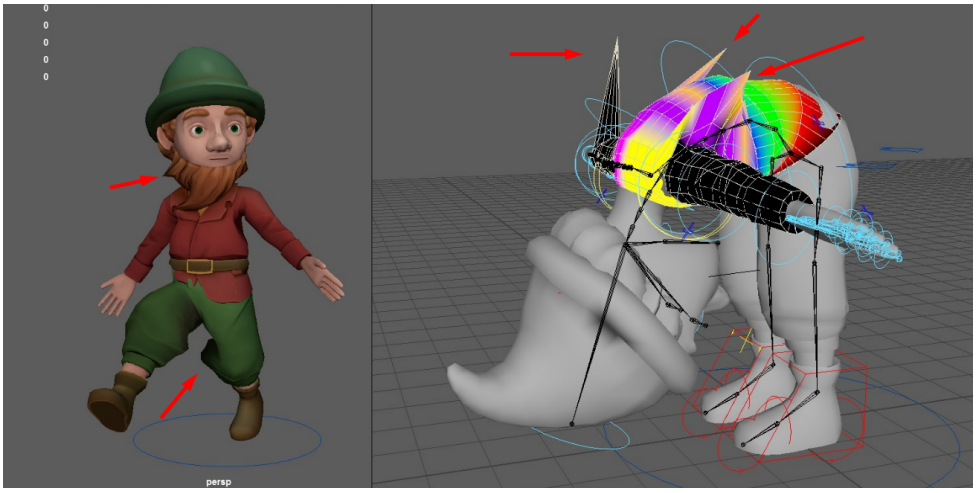


Рис. 45 – процесс исправления влияния костей на точки модели

3.1.3. Создание рига лица

Кроме основной анимации предполагалось, что гном будет также моргать и улыбаться шире, чем на изначально созданной базовой модели, что привело к созданию не только рига для тела, но и отдельного рига для лица. Для этого был также использован плагин advanced skeleton, так как кроме инструментов работы со скелетом в нём присутствуют и специальные инструменты для работы с лицом.

Инструментами FacePre и FaceFit была создана разметка лица модели – обозначены глаза, внутренние и внешние границы век, и рта, а также опорные точки на бровях, носе, щеках и шее. После чего был создан риг лица при помощи инструмента Build.

Рис. 46 – созданный риг лица

Получившаяся система позволила изменить черты лица модели и привести его к более дружелюбному выражению, а также добавить простую анимацию закрывания и открывания глаз, необходимую для последующей анимации.

Для того чтобы получившуюся модель можно было анимировать и в программе Maya, и в программе Unity, необходимо было преобразовать имеющийся риг лица в другую систему. Проблема связывания результатов работы Advanced skeleton с программой Unity состоит в том, что на плагин создает на лице большое количество костей, каждая из которых имеет несколько вариантов влияния на привязанные к ней точки. При этом большинство версий Unity не поддерживает отображение анимации точек, на которые влияют более четырёх костей. Преобразование было необходимо и для уменьшения веса модели и будущей анимации – поскольку в будущей работе предполагалось, что гном будет использоваться в мобильном приложении, необходимо было максимально облегчить все его составляющие.

В качестве новой системы была выбрана внутренняя функция программы Maya – Blendshape. Данная функция позволяет создать плавный переход между двумя и более положениями точек, «запекая» его в дополнительный атрибут модели, который можно изменять от 0 – первого положения набора точек до 1 – второго положения набора точек. При этом промежуточные значения могут быть как высчитаны автоматически, как среднее положение между 0 и 1, так и добавлены вручную для большей детализации.

Для лица гнома было создано два Blendshape – один для изменения улыбки, и второй для моргания. В обоих случаях в качестве 0 было выбрано изначальное расположение точек модели, а в качестве 1 было выбрано расположение, настроенное при помощи созданного ранее рига. Для глаз

было также добавлено промежуточное положение, настроенное как при помощи инструментов рига, так и перемещением областей вручную. Таким образом, из сложного рига были выделены и вынесены в отдельные шкалы две необходимые для последующей работы области.

Рис. 47 – подготовка модели к анимации моргания

3.2. Анимация

Для создания плавной анимации был использован сервис mixamo. Данный сервис позволяет добавить анимацию на модель при помощи внутренних систем автоматического рига и скиннинга. Модель гнома была загружена в систему, настроена для авторига, после чего среди имеющихся в сервисе анимаций были выбраны две наиболее отвечающие общему образу гнома – спокойное стояние на месте с небольшим покачиванием из стороны в сторону и дыханием, а также приветственное махание рукой. Модель с автоматическим ригом и анимациями была импортирована в программу Maya, где при помощи инструмента Copy Skin Weights и переименовывания костей анимация была перенесена на изначальную модель со сделанным вручную ригом. Также был исправлен ряд кадров

анимации, например, расположение рук при стоянии на месте, а также амплитуда перемещения кисти.

Анимация была дополнена морганием глаз, сделанным при помощи использования blendshape, при этом для усиления реалистичности было сделано несколько вариантов моргания на один цикл персонажной анимации.

3.3. Перенос в приложение

Полученная модель, а также модели гусёнка и окружения, созданные моей коллегой Дарьей, были импортированы в программу Unity вместе со скелетом, ригом, текстурами и анимациями. Модели были прикреплены к изображению метки, после чего с использованием импортированных текстур были созданы и настроены соответствующие материалы. Также в самой программе был настроен свет таким образом, чтобы добавить атмосферу жизнерадостности. Перед добавлением анимации получившееся приложение было собрано и протестировано на мобильном устройстве, тестирование показало адекватно отображающиеся текстуры и свет, а также правильное расположение моделей на изображении и удобный для рассматривания масштаб.

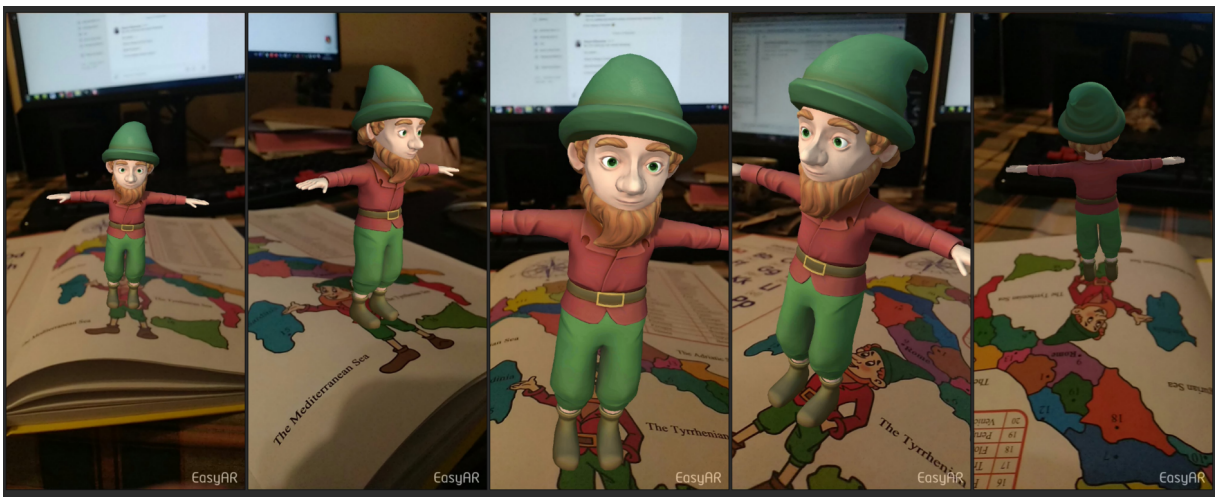


Рис. 48 – тестирование отображения модели гнома на мобильном устройстве

Для того чтобы добавить анимацию, был создан объект Animation controller, в котором были проведены связи между имеющимися анимациями, а также их связь с bool-переменной isclicked, используемой как переключатель между двумя состояниями.

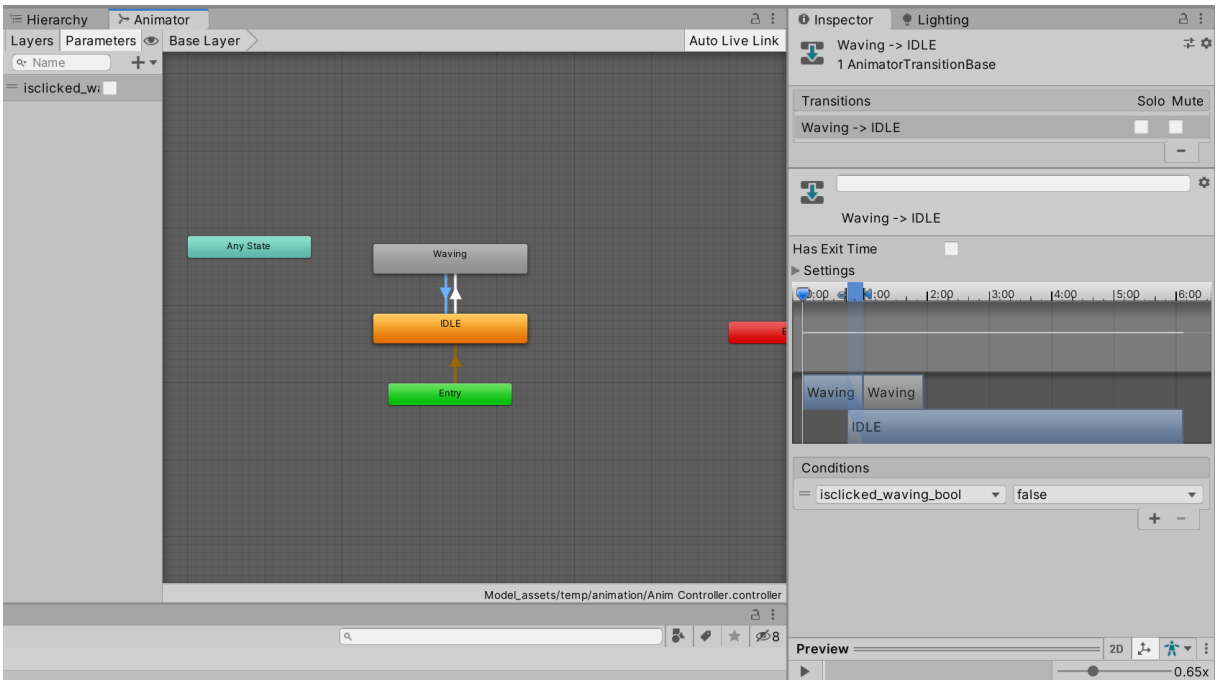


Рис.49 – объект Animation controller

Для связи переменной isclicked с нажатием любой клавиши на компьютере и нажатии на экран на мобильном устройстве был написан скрипт:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class animate : MonoBehaviour
{
    Animator animator;

    void Start()
    {
        animator = GetComponent<Animator>();
    }

    void Update()
```

```
    {
        if (Input.anyKey)
        {
            animator.SetBool("isclicked_waving_bool", true);
        }
        if (!Input.anyKey)
        {
            animator.SetBool("isclicked_waving_bool", false);
        }
    }
}
```

Полученное приложение было также собрано, установлено на мобильное приложение и успешно протестировано на работоспособность.

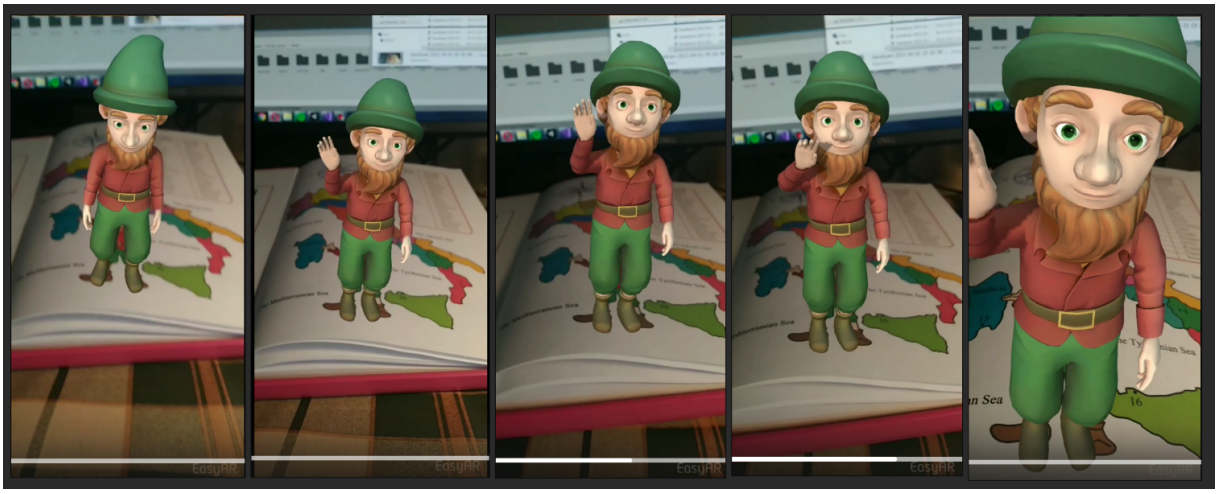


Рис. 50 – тестирование работы готового приложения с финальной версией модели гнома

Заключение

В результате данной работы была достигнута изначально поставленная основная цель: создано мобильное приложение дополненной реальности и модели для иллюстрированной презентации учебного пособия "Gnome Innico - Colouring Fairy Tale in English / ГНОМ ИННИКО - сказка-раскраска на английском языке".

Было проведено исследование и подробный анализ ряда инструментов разработчика для работы с проектами дополненной реальности. Описание основных характеристик, а также достоинств и недостатков этих инструментов было сведено в специальную таблицу. В результате тестов был выбран оптимальный SDK EasyAR для осуществления поставленных задач по созданию приложения.

В ходе выполнения работы были изучены технологии, используемые в таких сферах, как: 3D высоко- и низкополигональное моделирование, 3D-текстурирование, риггинг, анимация, разработка мобильных игр и приложений, разработка компьютерных игр. Полученные навыки могут быть применены при последующей работе в данных сферах или при работе с аналогичными проектами, а также проектами, использующими отдельные подобные технологии.

Были получены навыки работы в таких программах, как Autodesk Maya, Pixologic ZBrush, Substance Painter, Marmoset Toolbag, Unity 3D, Adobe Photoshop, PureRef.

Список использованных источников

1. https://docs.unrealengine.com/en-US/index.html - официальные обучающие материалы от разработчиков. [Электронный ресурс] Режим доступа: свободный

2. https://www.youtube.com/ - независимые обучающие материалы от пользователей. [Электронный ресурс] Режим доступа: свободный

3. https://knowledge.autodesk.com/ru/support/maya?sort=score - официальные обучающие материалы от разработчиков. [Электронный ресурс] Режим доступа: свободный

4. https://developers.google.com/ar/develop - официальные обучающие материалы от разработчиков. [Электронный ресурс] Режим доступа: свободный

5. https://developer.vuforia.com - официальные обучающие материалы от разработчиков. [Электронный ресурс] Режим доступа: свободный

6. https://www.easyar.com - официальные обучающие материалы от разработчиков. [Электронный ресурс] Режим доступа: свободный

7. http://ar.uplugins.com - официальные обучающие материалы от разработчиков. [Электронный ресурс] Режим доступа: свободный

8. https://forums.unrealengine.com/community/work-in-progress/58050-augmented-reality-for-ue4/page2?86525-Augmented-Reality-for-UE4= - официальные обучающие материалы от разработчиков. [Электронный ресурс] Режим доступа: свободный

9. https://unity.com/ru/unity/features/ar - официальные обучающие материалы от разработчиков. [Электронный ресурс] Режим доступа: свободный

10. https://knowledge.autodesk.com/ru/support/maya?sort=score - официальные обучающие материалы от разработчиков. [Электронный ресурс] Режим доступа: свободный

11. http://www.mir3d.ru/ - Мир 3D [Электронный ресурс] Режим доступа: свободный

12. http://3dpapa.ru/ - Непринужденный блог о 3D-графике, Анимации [Электронный ресурс] Режим доступа: свободный

13. https://support.allegorithmic.com/documentation/spdoc/substance-painter-20316164.html - официальные обучающие материалы от разработчиков Substance Painter [Электронный ресурс] Режим доступа: свободный.

14. https://www.pinterest.ru – Pinterest [Электронный ресурс] Режим доступа: свободный

15. https://www.artstation.com – Artstation [Электронный ресурс] Режим доступа: свободный