

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
«Вычислительная техника»
(наименование)

Кафедра
«Системы автоматизированного проектирования»
(наименование)

Направление подготовки

09.03.01 «Информатика и вычислительная техника»
(код и наименование)

Профиль

«Системы автоматизированного проектирования»
(наименование)

БАКАЛАВРСКАЯ РАБОТА

на тему

Разработка системы контроля персонала
средствами Unity

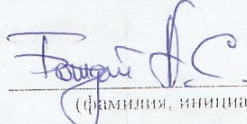
Студент

 19.05.2017 Вьюр С.А.
(подпись, дата) (фамилия, инициалы)

Руководитель

 29.05.2017 Энн В.В.
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр

Иванов С.А. ПТУ  30.05.17 
(должность, место работы) (подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____

№ _____

Заведующий кафедрой

(подпись)

(фамилия, инициалы)

Работа защищена с отметкой _____

(протокол заседания ГЭК от _____ № _____)

Секретарь ГЭК

(подпись)

(фамилия, инициалы)

Пенза 2017

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ.....	6
ВВЕДЕНИЕ.....	7
1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ.....	8
1.1 Обзор существующих средств контроля персонажа.....	8
1.2 Обзор аналогичных игр	11
1.3 Программные продукты для разработки игр	13
1.4 Выбор языка программирования.....	18
1.5 Программные продукты для 3D моделирования.....	19
1.6 Методы 3D моделирования.....	19
2 ПРОЕКТИРОВАНИЕ ПЕРСОНАЖА.....	24
2.1 Оптимизация полигонов.....	25
2.2 Использование четырехугольных полигонов	26
2.3 Создание анимаций.....	26
2.4 Создание текстур.....	27
2.5 Привязка управляющих объектов к модели.....	28
3 РАЗРАБОТКА СИСТЕМЫ КОНТРОЛЯ УПРАВЛЕНИЯ	
ПЕРСОНАЖЕМ	30
3.1 Структура файла скрипта	31
3.2 Анимация	33
3.3 Описание ключевых классов	34
3.4 Варианты работы с инвентарем.....	38
ЗАКЛЮЧЕНИЕ	41
СПИСОК ЛИТЕРАТУРЫ	43
ПРИЛОЖЕНИЕ А.....	48
А1 Используемые термины.....	48
А2 Игры в анкете.....	51
А3 Сравнительный анализ продуктов для 3D моделирования	52
А4 Упрощенная схема сравнения продуктов для 3D моделирования ..	60

A5 Схема 3D модели главного персонажа	61
A6 Набор анимаций в среде 3D Studio Max	62
A7 Система управления персонажем	63
A8 Диаграмма классов.....	64
A9 Взаимодействие скрипта с моделью	65
ПРИЛОЖЕНИЕ Б	66
B1 CharacterPanel.cs.....	66
B2 Item.cs	68
B3 ItemContainer.cs	69
B4 ItemManager.cs.....	70
B5 ItemScript.cs	72
B6 WorldInteraction.cs.....	72

ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ

1. FPS - Кадровая частота (Frames per Second) — количество сменяемых кадров за единицу времени в компьютерных играх, телевидении и кинематографе [30].
2. RPG - (англ. Computer Role-Playing Game, обозначается аббревиатурой CRPG или RPG) — жанр компьютерных игр, основанный на элементах игрового процесса традиционных настольных ролевых игр [21].
3. БД – база данных.
4. ЛКП – левая кнопка мышки.
5. ММО - глобальные мультиплеерные игры [21].
6. Моделирование – процесс создания 3D модели.
7. Модель – 3D объект, который может состоять как из одного объекта, так и из множества, связанных между собой.
8. ОЗУ – оперативно запоминающее устройство.
9. ПКМ – правая кнопка мышки.
10. ПО - программное обеспечение.

ВВЕДЕНИЕ

С развитием вычислительной техники широкую популярность обрели компьютерные игры. Они разрабатываются для различных платформ: персональные компьютеры, планшетные компьютеры, игровые консоли и мобильные устройства.

Разработка компьютерных игр состоит из нескольких частей. Для каждой из них используются различные технологии. В частности, для разработки персонажей используются различные системы трехмерного моделирования, а для создания приложений используются различные средства для разработки игр.

Задачи ВКР:

1. Обзор предметной области.
2. Обзор существующих средств контроля персонажа.
3. Обзор аналогичных игр и рассмотрение их недостатков.
4. Анализ ПО для разработки игр и 3D моделирования.
5. Создание 3D модели персонажа.
 - 5.1.оптимизация модели;
 - 5.2.создание различных групп анимации для персонажа;
 - 5.3.интеграция анимированного персонажа в среду разработки игр;
6. Разработка системы контроля управления персонажем.
 - 6.1.написание функций вызова анимаций персонажа.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

Изометрическое представление игр жанра RPG дает много преимуществ по графической составляющей. Отсутствие множества проблем позволяет более удобно и быстро реализовать необходимые функции, не исправляя неудобства, которые возникают при появлении артефактов изображения и уменьшении производительности, что повышает FPS по сравнению с играми других жанров.

Плюсы выбранного игрового жанра:

1. Отсутствие артефактов изображения.
 - 1.1. избегание трилинейной фильтрации;
 - 1.2. меньшее влияние антиалиасинга (сглаживание);
 - 1.3. отсутствие размытия (motion blur);
2. Отсутствие параметризации.
3. Низкополигональное моделирование.

1.1 Обзор существующих средств контроля персонажа

Существует несколько систем управления игрового персонажа:

1. На компьютере [14]:
 - 1.1. С помощью клавиатуры. Различными клавишами можно управлять передвижением персонажа и вызывать другие анимации.
 - 1.2. С помощью кнопок компьютерной мышки. ЛКП / ПКМ – передвижение. Остальные кнопки (если присутствуют) настраиваются самим пользователем по его предпочтениям.
 - 1.3. С помощью перемещения компьютерной мышки. Управляет перемещением камеры и направлением движения персонажа.
 - 1.4. Направленное (Target) и не направленное (Non-Target) управление. При таргет управлении объект выделяется пользователем. При нон-таргет выделение объекта происходит автоматически.

2. На планшетах:
 - 2.1. Сенсорный экран.
3. На игровых приставках:
 - 3.1. Джойстик.
4. Мобильные телефоны:
 - 4.1. Кнопки.

Пример общей схемы системы управления представлен на рисунке 1.1.
Пример более подробной схемы представлен в приложении А7.

Обще игровая система обеспечивает выполнение следующих функций [15]:

1. Визуализация игрового мира и персонажа.
3. Обработка пользовательского ввода.
4. Обработка действий персонажей.
5. Определение факта возникновения событий, требующих вмешательства игрока.
6. Проверка возникшего игрового события на необходимость изменения состояния системы.
7. Проверка на устранение причины события (например, персонажу задали действия).
8. Визуализация игровых событий.



Рисунок 1.1 – Упрощенная схема управления персонажем

Так же пользователь может управлять инвентарем персонажа. Диаграмма компонентов представлена на рисунке 1.2. Диаграмма компонентов — статическая структурная диаграмма, которая показывает разбиение программной системы на структурные компоненты и связи (зависимости) между компонентами.

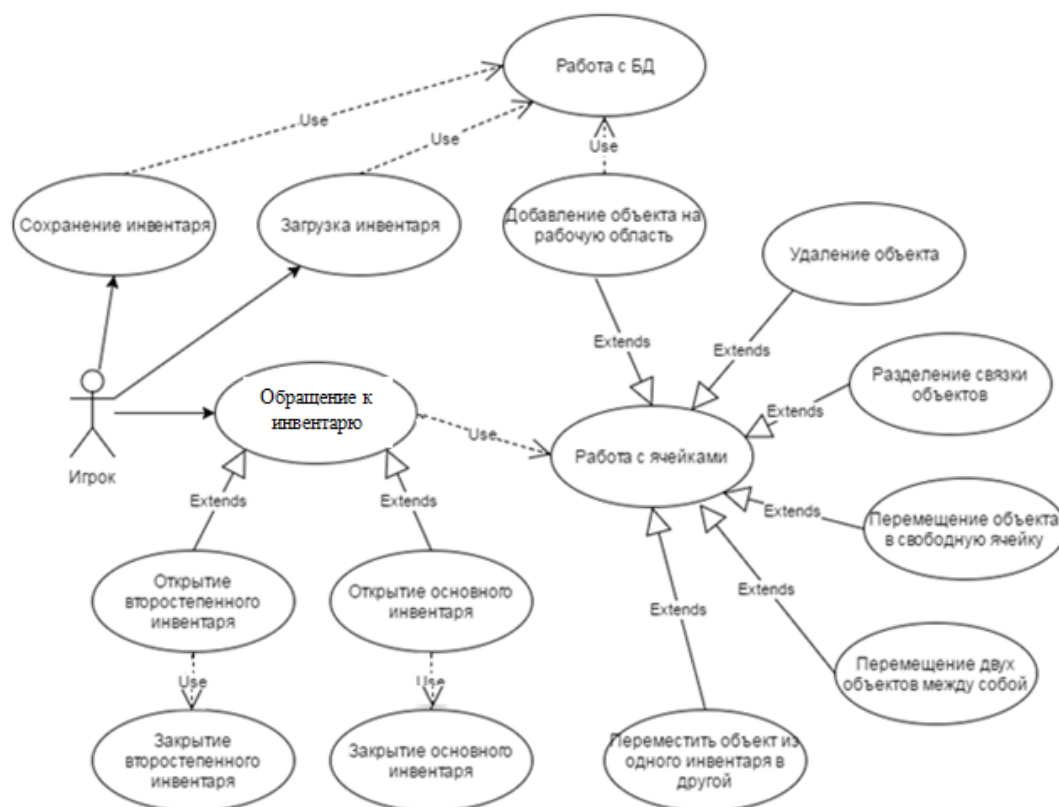


Рисунок 1.2 – Диаграмма компонентов

1.2 Обзор аналогичных игр

Наиболее похожими играми, которые были рассмотрены для данной работы, являются: «Blade and soul», «Diablo 2», «World of Warcraft», «Dota 2» и серии игр «Dragon Age».

Опрос среди школьников 13 и 14 лет в МОУ СОШ №221 г. Заречного показал, что среди 82 аналогичных игр (приложение А2) они играли в 20 из них, которые представлены на рисунке 1.3.

По их мнению, самыми популярными играми являются:

1. World of Warcraft (17,4%)
2. Dota 2 (14,1%)
3. Diablo II (7%)

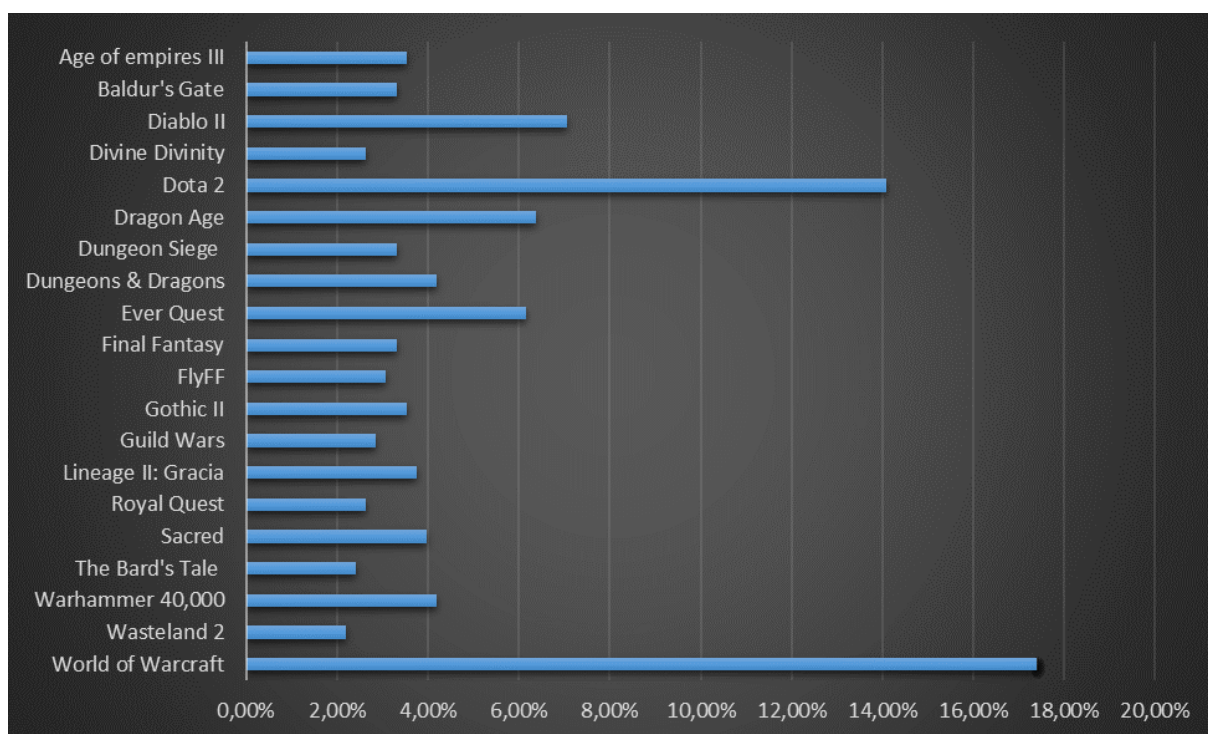


Рисунок 1.3 – Результаты анкетирования

В рассмотренных аналогах, например в игре «World of Warcraft», высокие требования к аппаратной составляющей компьютера (таблица 1), что делает игру доступной для меньшего числа игроков.

Таблица 1 - Сравнение технических средств

Технические средства	World Of Warcraft	Разрабатываемая система
Процессор	Intel Core 2 Duo E8500	Intel core i3-2500
Видеокарта	NVIDIA GeForce GT 440	NVIDIA GeForce GTS 250
ОЗУ	2 Гб	1 Гб

Графическая составляющая некоторых аналогов, например », «Diablo 2», несовершенна и данные проблемы нужно учитывать при создании ВКР.

В руководстве среды Unity для настольных платформ идеальный диапазон полигонов - примерно от 1500 до 4000.

Для героев игры «Blade And Soul» (рисунок 1.4) выделяется в среднем 8 000 полигонов. У нашего главного героя (мужчины) используется около

1780 полигонов (приложение А5), что позволяет оставить большее пространство для шейдеров (визуальных эффектов).

Количество полигонов в игре «Blade And Soul» [31]:

1. Female, Custom character — 10,026;
2. Female, Yuran — 7,904;
3. Female, Custom character 2 — 8,815;
4. young male, Custom character 3 — 7,386;
5. Young male, Custom character — 7,931;
6. Young male, Custom character 2 — 8,118;
7. Young male, Custom character 4 - 8,244;



Рисунок 1.4 – Игра «Blade and Soul»

1.3 Программные продукты для разработки игр

Программные продукты для разработки игр [19]:

1. Unity 5;
2. Torque 2D/3D;
3. CryEngine 3;
4. UDK (Unreal Development Kit) ;

Unity 5

Unity — это инструмент для разработки двух- и трёхмерных приложений и игр, работающий под операционными системами Windows, Linux и OS X [17].

Unity 5 [33] поддерживает два языка: C# (наиболее используемый) и Javascript. Разработчику необходимо знать один из языков в совершенстве, а другой на среднем уровне, так как некоторые моменты Unity 5 делает только на одном из двух языков.

Объекты в Unity могут быть пустыми, (для объединения несколько объектов в единую группу, т.е. сделать их дочерними (GameObject), содержать компоненты, с которыми взаимодействуют скрипты, им могут быть названы одним и тем же именем, им могут быть присвоены теги, служащие для того, чтобы скрипт нашел нужный объект).

К объектам в Unity можно присвоить коллайдеры [34]:

1. Box Collider - куб.
2. Sphere Collider – сфера.
3. Character Collider - коллайдер, специально введенный в Unity 5 для персонажей.
4. Mesh Collider - коллайдер, созданный по мешу, то есть повторяющий геометрию объекта.
5. Wheel Collider - коллайдер для колес.
6. Terrain Collider - коллайдер для объекта «terrain».

Анимировать модели в Unity можно несколькими способами:

1. Создание анимаций в специальных программах, для 3D моделирования;
2. В среде Unity, так как здесь есть компонент для их создания.

Достоинства Unity [17]:

1. Поддержка множества передовых форматов для трехмерных объектов (.FBX, .OBJ).

2. IDE: сочетание редактора сцен (в комплексе общего редактора) с редактором игровых объектов и редакторов скриптов. Дополнительно прилагаются генераторы деревьев и «terrain».

3. Улучшенные возможности скриптинга (в Unity доступны три языка: JavaScript, C# и разновидность Python's Boo).

4. Кроссплатформенность – как уже упоминалось выше, поддерживаются Windows, MacOS, Wii, iPhone, iPod, iPad, Android, PS3 и Xbox 360, не все из которых, доступны в бесплатной версии.

5. Современный уровень графики, способный конкурировать с иными движками. Unity обладает такими возможностями, как deferred освещение, стандартный набор постпроцессинговых эффектов, SSAO, ускоренная проработка лайтмапов (специальные текстуры).

6. Проработанный физический движок.

7. Масштабируемость и производительность. Большую часть простых процессов движок обрабатывает на превосходном уровне.

8. Запуск любого приложения на Unity в веб-плагине.

9. Невысокая цена за полную лицензионную версию для крупного веб-разработчика.

Недостатки Unity [18]:

1. Закрытость кода. Невозможность получения исходных кодов движка даже у платной версии.

2. Невозможность дополнения физики движка сторонними возможностями. Нельзя добавить в движок стороннюю физику.

Torque 2D/3D

Достоинства Torque 2D/3D:

1. Бесплатность.

2. Открытый исходный код.

3. Для воспроизведения звуков T2D использует библиотеку -penAL.

4. Симуляция физики осуществляется посредством движка B-x2D, ставшего стандартом в двумерных физических исчислениях.

Недостатки Torque 2D/3D:

1. Доступны не все платформы.

CryEngine 3

Достоинства CryEngine 3:

1. Кроссплатформенность.
2. Поддержка MMO игр.
3. Множество технологий визуализации.
4. Встроенная система анимации:
 - 4.1. индивидуализация персонажей;
 - 4.2. параметрическая скелетная анимация;
 - 4.3. процедурное деформирование движения;
5. Встроенная система ИИ, которая позволяет обрабатывать поведение не только персонажей, но и транспортных средств:
 - 5.1. умные объекты;
 - 5.2. алгоритмы динамического обнаружения пути;
 - 5.3. система, управляемая сценариями;
6. Присутствует поддержка глобальных мультиплеерных игр.

Недостатки CryEngine 3:

1. Бесплатная версия предназначена только для некоммерческого использования.
2. Лицензировать движок может только коммерческое лицо.
3. Закрытый исходный код.

UDK

Достоинства UDK:

1. Бесплатность.
2. Гибкая система анимации позволяет контролировать каждую деталь анимируемого объекта.
3. Анимационная модель контролируется системой AnimTree, которая включает следующие механизмы:
 - 3.1 контроллер смешения (Blend);

- 3.2 контроллер, управляемый данными;
- 3.3 физические, процедурно-скелетные контроллеры;
4. Высоко оптимизированная сетевая архитектура.

Недостатки UDK:

1. Закрытый исходный код.
2. Не кроссплатформенный.
3. Система древовидных классов - дочерние и родительские.

Поскольку у первого рассматриваемого движка (Unity 5) было больше всего достоинств, поэтому был выбран именно он.

MonoDevelop

MonoDevelop [29] - это интегрированная среда разработки (IDE), поставляемая вместе с Unity.

IDE сочетает в себе функции текстового редактора с дополнительными возможностями для отладки и выполнения других задач по управлению проектами. Данная среда разработки, предназначенная для создания приложений C#, Java, Boo, Nemerle, Visual Basic .NET, Vala, CIL, C и C++.

MonoDevelop является частью проекта Mono. Встроен в дистрибутив Unity3D как средство написания скриптов.

Возможности:

1. подсветка синтаксиса;
2. сворачивание кода;
3. автодополнение кода;
4. браузер классов;
5. поддержка плагинов;
6. встроенный отладчик;
7. визуальный конструктор форм (GTK#);
8. модульное тестирование;

Поддерживаемые языки программирования:

1. C#;
2. Visual Basic.Net;

3. C;
4. C++ (до версии 4.0 в Windows);
5. Vala;

Microsoft Visual Studio

Microsoft Visual Studio — линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данный продукт позволяет разрабатывать как консольные приложения, так и приложения с графическим интерфейсом [32].

Microsoft Visual Studio позволяет разрабатывать программы с использованием следующих языков программирования: C++, C#, Visual Basic, F#, JavaScript, TypeScript, Python.

Продукты Microsoft для разработчиков давно входят в список наиболее востребованного программного обеспечения для программистов разного уровня. За восемь лет существования на рынке среда разработки Microsoft Visual Studio стала де-факто стандартом создания .NET-приложений.

1.4 Выбор языка программирования

Unity изначально поддерживает два языка программирования [23]:

1. C# стандартный в отрасли язык подобный Java или C++.
2. UnityScript, язык, разработанный специально для использования в Unity по образцу JavaScript.

Был выбран язык C#, поскольку он обладает некоторыми преимуществами перед языком JavaScript:

1. В Unity не чистый JavaScript.
2. Обучающей литературы по C# больше.
3. Поддержка и сопровождение C# будет в первую очередь, как заявляют создатели движка Unity.

4. С# находится в пятерке самых востребованных языков программирования по версии GitHub [24] (на 2016 год), что облегчает программирование в команде.

5. Работа программы более стабильная (при смене версий движка Unity).

1.5 Программные продукты для 3D моделирования

В приложении А3 представлены критерии основных продуктов для 3D моделирования:

1. 3D Studio Max: 52 преимущества и 8 недостатков.
2. Maya: 27 преимуществ и 19 недостатков.
3. Blender: 26 преимуществ и 24 недостатка.

Все преимущества и недостатки в данной приложении А4 имеют одинаковый вес и для каждого пользователя имеют разную ценность. Для одного пользователя недостаток продукта является преимуществом в другом продукте и наоборот.

Были использованы материалы лекций [20] на конференции «CG EVENT» от студии 3DMASTER, статьи пользователей и личный опыт работы.

В 3D Studio Max есть встроенная система костей Biped и CAT. Это позволяет обмениваться файлами между аниматорами. Можно воспользоваться огромным множеством готовых наборов качественных анимаций, созданных при помощи Motion Capture. Для иерархии костей не нужно настраивать инверсную кинематику. Так же 3D Studio Max идеально подходит для простых игровых анимаций.

1.6 Методы 3D моделирования

1. Сплайновое [7] моделирование (моделирование на основе кривых Безье).

2. NURBS моделирование (поверхностное моделирование на основе кривых NURBS).

3. Полигональное [28] моделирование (таблица 2).

Полигональное моделирование

Практически любой объект в игровом пространстве это набор точек (минимум трех), не лежащих на одной прямой и их образы однозначно задают аффинное преобразование плоскости.

Матричное представление используется, в частности, для записи аффинных преобразований в компьютерной графике. Матричное представление используется в OpenGL и DirectX (где координаты представляются в виде матриц 1×4), где она транспонирована.

Другими словами, каждый полигон состоит из набора точек и приходится хранить информацию о координатах каждой точки. Пример 3D модели и ее представление в виде точек показано на рисунке 1.6. И чем больше будет точек (следовательно, и полигонов), тем ниже будет производительность.

Преимущества полигонального моделирования:

1. Является универсальным.
2. Решает любые топологические [2] задачи.
3. Полнота метода и инструментария.
4. Интеграция с другими пакетами и технологиями (OBJ, FBX).
5. Обеспечивает любую требуемую степень точности.
6. Наиболее полно реализовано во всех современных пакетах.
7. Занимает меньше времени на освоение и получение качественного результата.
8. Простота работы, что снижает время на построение сложного объекта.
9. В отличие от других методов позволяет вносить более детальные изменения в объекте.
10. Естественный переход от низкополигональной к высокополигональной модели [3].

- 11.Переносимость техник моделирования между пакетами.
- 12.Перспективы развития.
- 13.Наличие инструментов скульптинга.
- 14.Наличие инструментов ретопологии.

При опросе пользователей сайта 3dmaster.ru выяснилось, что 3D Studio Max лучше всего подходит для полигонального моделирования.

Результаты представлены ниже на рисунке 1.5.

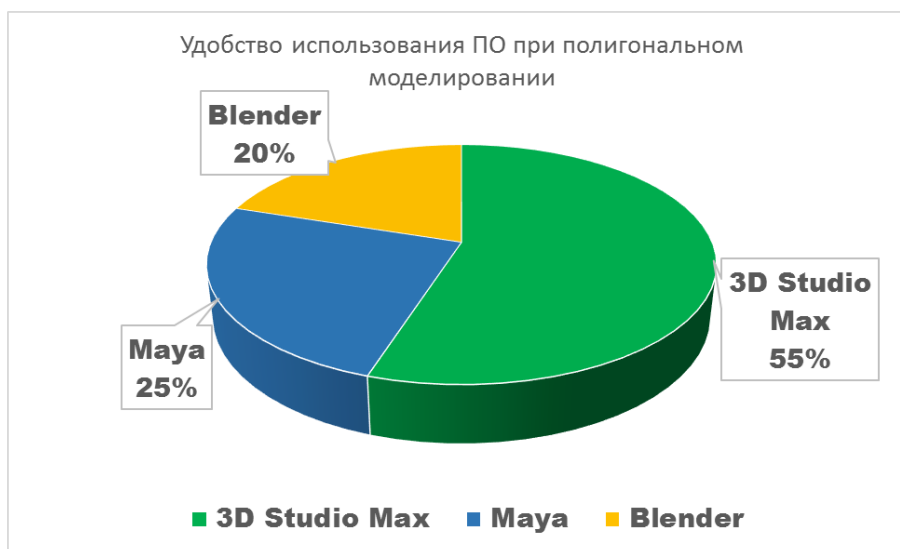


Рисунок 1.5 – Результаты голосования пользователей

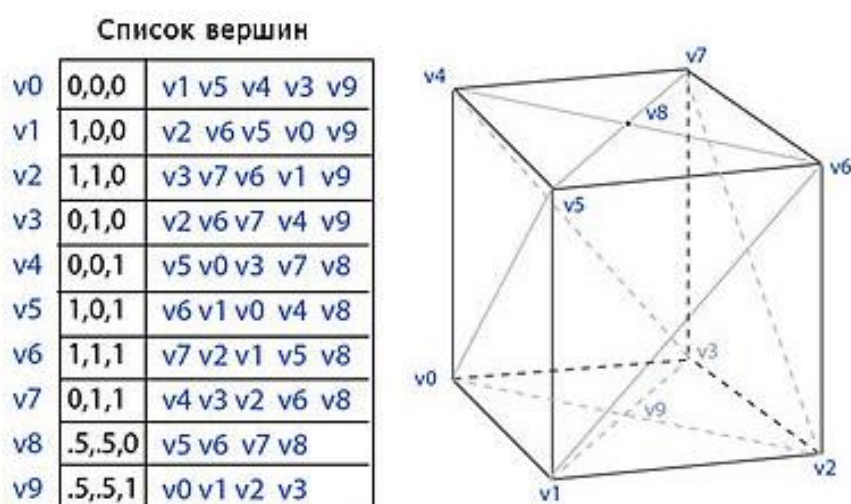


Рисунок 1.6 – Вершинное представление трехмерного объекта

Таблица 2 - Сравнение высокополигонального и низкополигонального моделирования

Низкополигональное моделирование (рисунок 1.7)	Высокополигональное моделирование (рисунок 1.8)
<p>Полигоны плоские.</p> <p>Есть сгиб — нужно ребро.</p> <p>3 или 4 угла в полигоне.</p> <p>Общий объем не нужен.</p> <p>Упрощение объекта.</p>	<p>Полигоны выпуклые.</p> <p>4-х угольники всегда.</p> <p>3-угольники на крайний случай.</p> <p>5-угольники иногда.</p> <p>Есть особые полосы полигонов.</p> <p>Избегание «звездочек» в полигонах.</p> <p>Предъявляет более высокие требования к модели.</p>

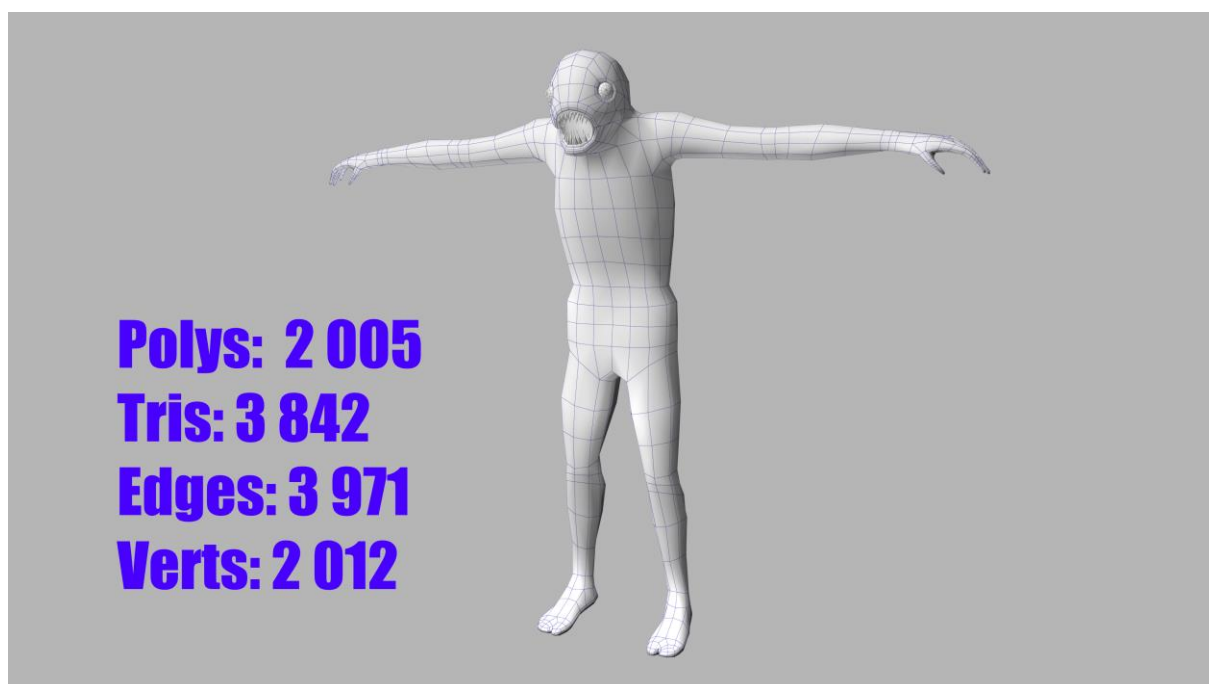


Рисунок 1.7 – Низкополигональная модель, 3842 треугольников

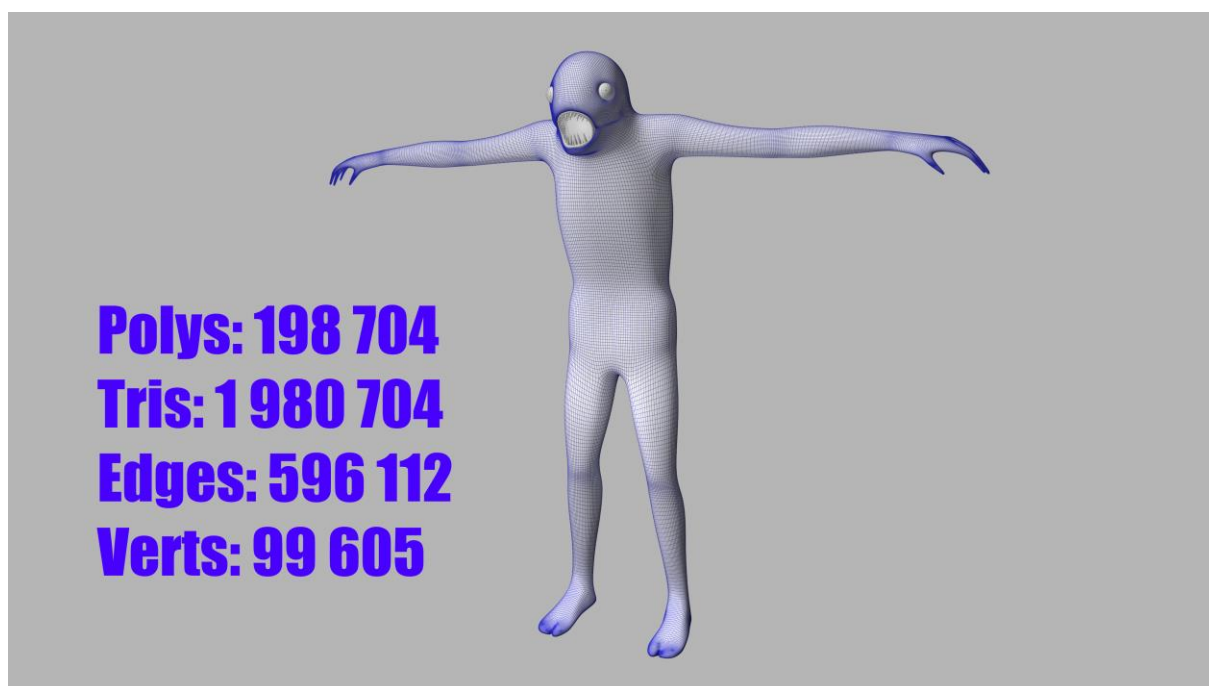


Рисунок 1.8 – Высокополигональная модель, 1 980 704 треугольника

2 ПРОЕКТИРОВАНИЕ ПЕРСОНАЖА

Этапы создания персонажа [12]:

1. Концепт [2].
2. Создание низкополигональной модели [10].
3. Создание развертки модели [5].
4. Создание текстур [9].
5. Создание набора управляющих объектов [1].
6. Привязка управляющих объектов к модели [13].
7. Создание групп анимаций [6].

В продукте 3D Studio Max есть множество удобных инструментов, необходимых для полигонального моделирования:

1. Изобилие параметрических примитивов; .
2. Полноценный стек модификаторов.
3. Изобилие параметризованных модификаторов.
4. Числовые координаты всегда видны и доступны даже на уровне подобъектов.
5. Полное отсутствие не логичных инструментов для полигонального моделирование (функционал).
6. Наличие уникальных инструментов (мелкая детализация и удобство работы).
7. Превосходный инструмент для ретопологии.
8. Есть пред просмотр будущей высокополигональной модели во время моделирования низкополигональной модели.

Данный метод в первую очередь выбран потому, что время строго ограничено, а полигональное моделирование позволяет справиться с задачей наиболее быстро по сравнению с другими методами 3D моделирования.

Были использованы следующие инструменты для полигонального моделирования (модификатор Edit Poly):

1. Inset (Для детализации элементов тела, пальцев, ботинок).

2. Extrude (Вдавливание полигонов для детализации носа и глаз. Выдавливание полигонов для рук и ботинок).

3. Chamfer (Создание фаски для плеч).

4. Slice plane - создать сечение, образующее новое ребро. Часто используется при создании конечностей.

5. Cut (Создание разрезов для придания элементу более сглаженный вид).

Это не полный список используемых функций, так же использовались группы сглаживания, проверки объектов на наличие ошибок и т.д.

Главный персонаж состоит из следующего набора объектов (приложение А5):

1. голова;
2. тело;
3. руки;
4. наплечники;
5. ноги;
6. ботинки;

2.1 Оптимизация полигонов

В руководстве игрового движка Unity для настольных платформ идеальный диапазон полигонов - примерно от 1500 до 4000.

Рассмотрим аналогичные игры и количество полигонов в каждой из них.

Blade And Soul:

1. Female, Custom character — 10,026;
2. Female, Yuran — 7,904;
3. Female, Custom character 2 — 8,815;
4. Young male, Custom character 3 — 7,386;
5. Young male, Custom character — 7,931;
6. Young male, Custom character 2 — 8,118;

7. Young male, Custom character 4 - 8,244 ;

Dragon Age 2:

1. Arcane Horror — 9,630;

2. Leliana Chantry — 13,870;

3. Hawke Warrior (Female) — 16,690;

4. Hawke Warrior (Male) - 19,364 ;

Видно, что для главных героев рассматриваемых игр выделяется в среднем 12 000 полигонов. У нашего главного героя (мужчины) используется около 1780 полигонов, что позволяет оставить большее пространство для шейдеров (визуальных эффектов).

2.2 Использование четырехугольных полигонов

Четырехугольные полигоны избавляют нас от множества проблем [9]:

1. Проблемы с деформацией сгиба при создании анимации;
2. Проблемы с визуализацией;
3. Проблемы со сглаживанием.

2.3 Создание анимаций

Существуют различные группы анимаций [4], которые видит игрок:

1. Анимации движения в пространстве.
2. Анимации атаки.
3. Прочие виды анимаций (получения урона, смерть, общение с NPC и тд).

Рассмотрим масштаб всех игровых моделей, включая статику и персонажей. В данном проекте все объекты не подвержены влиянию масштаба, который является учитывающим фактором. Поскольку был задан масштаб пропорциональный реальному миру - игровой движок не проводит лишних вычислений, что влияет на производительность.

Для лучшего представления персонажа во время игры было предложено:

1. Сделать анимацию более детальной.

2. Избавится от недочетов в анимации, как в аналогичных играх.

Был создан следующий набор анимаций (приложение А6):

1. стойка;
2. ходьба;
3. бег;
4. удар мечом;
5. получение урона;
6. смерть;

Данные анимации делались по отдельности и сохранялись в файл формата .VIP, которые в дальнейшем с помощью инструмента «Mixer» были сохранены в формат .MIX. При экспорте модели анимации сохранились вместе с моделью в файле .FBX, который в дальнейшем был импортирован в среду Unity.

2.4 Создание текстур

Стоит обратить внимание на размер текстуры. Были созданы текстуры для персонажей размером 4096x4096 пикселей, которые затем уменьшены до оптимального размера, когда пользователю не видно разницы в текстуре разных размеров [8].

Э. Вард в своей книге «Разработка игрового персонажа: Kila» пишет: «Как правило, будет хорошей практикой работать с текстурами высокого разрешения, чем с целевым разрешением. Если ваши целевые карты должны быть 1k, (1024x1024), работайте в 2k, т.е. удвойте размер. Мало того, что это соответствует требованиям завтрашнего дня к вашей работе, но и уменьшение карты более высокой детализации даст намного лучшую карту низкого разрешения по сравнению с работой в первоначальном размере» [3, с. 82].

Не стоит упускать и то, что текстуры высокого разрешения могут пригодиться в будущем, поскольку игровая индустрия быстро развивается, как и графическая составляющая игр.

3D редакторы позволяют создавать сложные текстуры с множеством каналов — свойств текстуры для различных целей — цвета, выпуклостей, прозрачности и т.д.

В руководстве игрового движка Unity рекомендуется использовать минимальное количество материалов для одного объекта.

Было предложено не создавать 2 изображения для канала цвета и канала прозрачности, а сделать одно изображение с альфа каналом, что позволяет увеличить производительность за счет хранения одного изображения вместо двух.

UV карты (рисунок 2.1) были созданы с помощью модификатора UVW.

2.5 Привязка управляющих объектов к модели

Скининг (skinning) — процесс привязки игрового персонажа к системе связанных между собой объектов (рисунок 2.2), с помощью которых можно управлять динамикой персонажа.

При привязке к костям не использовались никакие плагины и скрипты, поскольку они не дают точного результата. Вместо этого абсолютно каждой точке (в модели монстра их было 1533) вручную настраивался ее вес (процент влияния кости на точку).

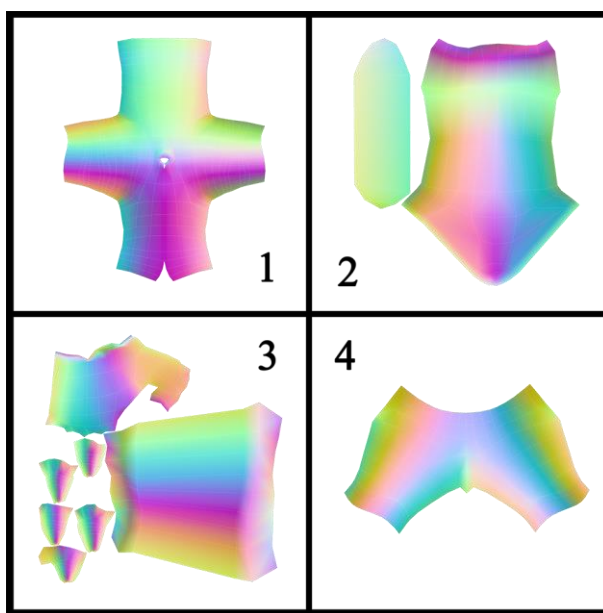


Рисунок 2.1 – развертки для тела (1), ботинок (2), рук (3) и штанов (4)

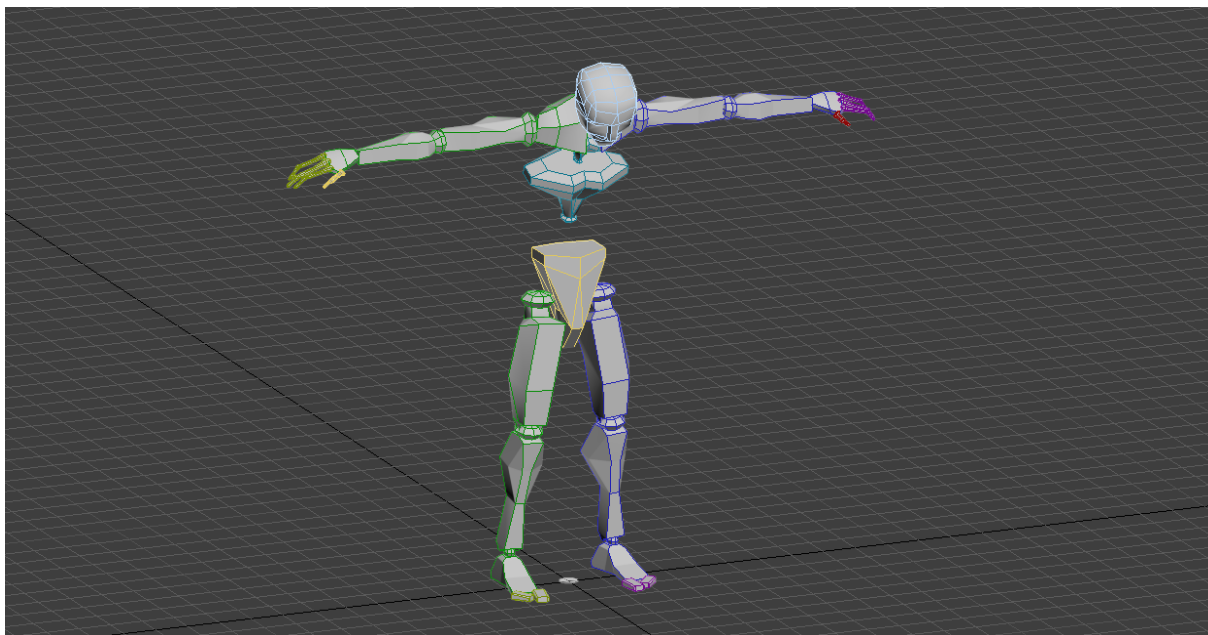


Рисунок 2.2 - Внешний вид костей для управления монстром

В руководстве среды Unity рекомендуется использовать до 60 костей у персонажа. Здесь же используются 40 костей — только те, которые необходимы.

Для оптимизации модели, например, при создании кисти рук учитывался тот фактор, что благодаря изометрическому представлению игры пользователь не будет видеть некоторые детали, такие как количество фаланг пальцев. Поэтому было решено убрать среднюю фалангу пальцев (рисунок 2.3), что уменьшает количество костей на 10.

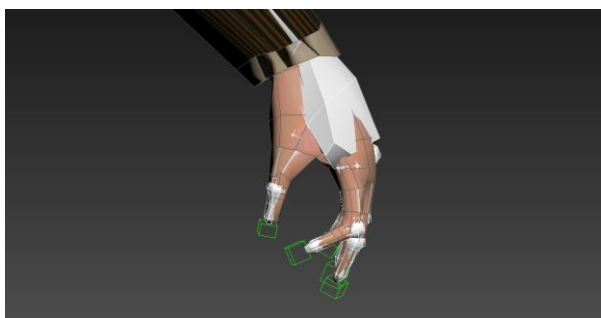


Рисунок 2.3 - Использование только двух фаланг пальцев, поскольку результат будет незаметен и не прибавится 10 лишних костей

3 РАЗРАБОТКА СИСТЕМЫ КОНТРОЛЯ УПРАВЛЕНИЯ ПЕРСОНАЖЕМ

Система контроля управления персонажем состоит из нескольких составляющих:

1. 3D модель (импортируется в среду Unity)
 - 1.1. объекты
 - 1.2. управляющие элементы
 - 1.3. набор анимаций
 - 1.4. текстуры
 - 1.5. UV-карты
2. Составляющие системы
 - 2.1. скрипты
 - 2.1.1. контроллеры
 - 2.1.2. коллайдеры

В приложении А7 представлена подробная схема системы контроля управления персонажем и ее подсистемы.

В приложении А8 показана диаграмма классов. Диаграмма классов — диаграмма, демонстрирующая классы системы, их атрибуты, методы и взаимосвязи между ними.

Прежде чем управлять моделью, необходимо импортировать ее в среду Unity. Сама модель была сохранена в одном из самых популярных форматов - .FBX, который сохраняет не только модель и ее свойства, но и набор анимаций.

Импортировать модель можно двумя способами:

1. Разместить файлы в папке проекта (например, «Assets»).
2. В среде разработки Unity выбрать меню «Assets > Import New Asset».

3.1 Структура файла скрипта

После двойного щелчка на скрипте в Unity, он будет открыт в текстовом редакторе [23]. По умолчанию Unity будет использовать MonoDevelop, но можно выбрать любой редактор из панели External Tools в настройках Unity.

Простейшая структура файла скрипта [38]:

```
using UnityEngine;
using System.Collections;
public class MainPlayer : MonoBehaviour {
    // [ в н в ф т 2 н т ф г т н т 6 т т ц х ls ls в / ь ts 2 ц 2 в ф 2 q в ц :
    void Start () {
    }
    // 7 8 ts т ф г ф х ts 2 х 4 в j т в j ф y 2 н т ф г т н т т 8 r х 4 z в
    void Update () {
    }
}
```

Скрипт взаимодействует с внутренними механизмами Unity за счет создания класса, наследованного от встроенного класса, называемого MonoBehaviour. Можно сказать, что это класс для создания нового типа компонента, который может быть прикреплен к игровому объекту.

Каждый раз, когда скриптовый компонент присоединяется к игровому объекту, создается новый экземпляр объекта, определенный планом. Имя класса и имя файла должны быть одинаковыми, для того, чтобы скриптовый компонент мог быть присоединен к игровому объекту. Взаимодействие скрипта с моделью представлено в приложении А9.

Функция Update - это место для размещения кода, который будет обрабатывать обновление кадра для игрового объекта. Это может быть движение, срабатывание действий и ответная реакция на ввод пользователя, в основном всё, что должно быть обработано с течением времени во игровом процессе.

Чтобы позволить функции Update выполнять свою работу, часто бывает полезно инициализировать переменные, считать свойства и осуществить связь с другими игровыми объектами до того, как будут совершены какие-либо действия.

Функция Start будет вызвана Unity до начала игрового процесса (т.е. до первого вызова функции Update), и это идеальное место для выполнения инициализации.

Управление игровым объектом

Скрипт определяет только план компонента и, таким образом, никакой его код не будет активирован до тех пор, пока экземпляр скрипта не будет присоединен к игровому объекту. Скрипт можно прикрепить путем перетаскивания ассета скрипта, на игровой объект в панели «Hierarchy» или через окно «Inspector» выбранного игрового объекта.

Обращение к компонентам

В редакторе Unity свойства компонента можно изменить, используя окно Inspector. Так, например, изменения позиции компонента Transform приведет к изменению позиции игрового объекта. По большей части скрипты также изменяют свойства компонентов для управления игровыми объектами. Разница, в том, что скрипт может изменять значение свойства постепенно со временем или по получению ввода от пользователя. За счет изменения, создания и уничтожения объектов в заданное время может быть реализован любой игровой процесс [41].

Наиболее простым и распространенным является случай, когда скрипту необходимо обратиться к другим компонентам, присоединенных к тому же игровому объекту (GameObject) [26]. Это делается с помощью функции «GetComponent». Объект компонента сохраняют в переменную, это делается в C# посредством следующего синтаксиса:

```
void Start () {  
    Rigidbody rb = GetComponent<Rigidbody>();  
}
```

Если есть ссылка на экземпляр компонента, то можно устанавливать значения его свойств, тех же, которые можно изменить в окне Inspector:

```
void Start () {  
    Rigidbody rb = GetComponent<Rigidbody>();  
    rb.mass = 10f;  
}
```

3.2 Анимация

Animator Controllers (контроллеры аниматоров)

Контроллер аниматора позволяет настраивать и управлять набором анимаций персонажа или иного анимированного игрового объекта.

Контроллер имеет ссылки на используемые им анимационные клипы, и управляет различными анимационными состояниями и переходами между ними, используя инструмент, который может быть представлен в виде блок-схемы, или простой программы, написанной на языке визуального программирования в Unity.

Аниматор и контроллер аниматора

Контроллер аниматора (Animator Controller) требуется для применения анимаций к скелету. Контроллер аниматора создается Unity и позволяет руководить набором анимаций для персонажа и переключаться между ними когда выполняется некоторое условие.

Набор анимаций и сам контроллер представлен на рисунке 3.1. Например, можно переключиться от анимации походки к прыжку путем нажатия клавиши пробела. Контроллер управляет переходами между анимациями используя машину состояний [26].

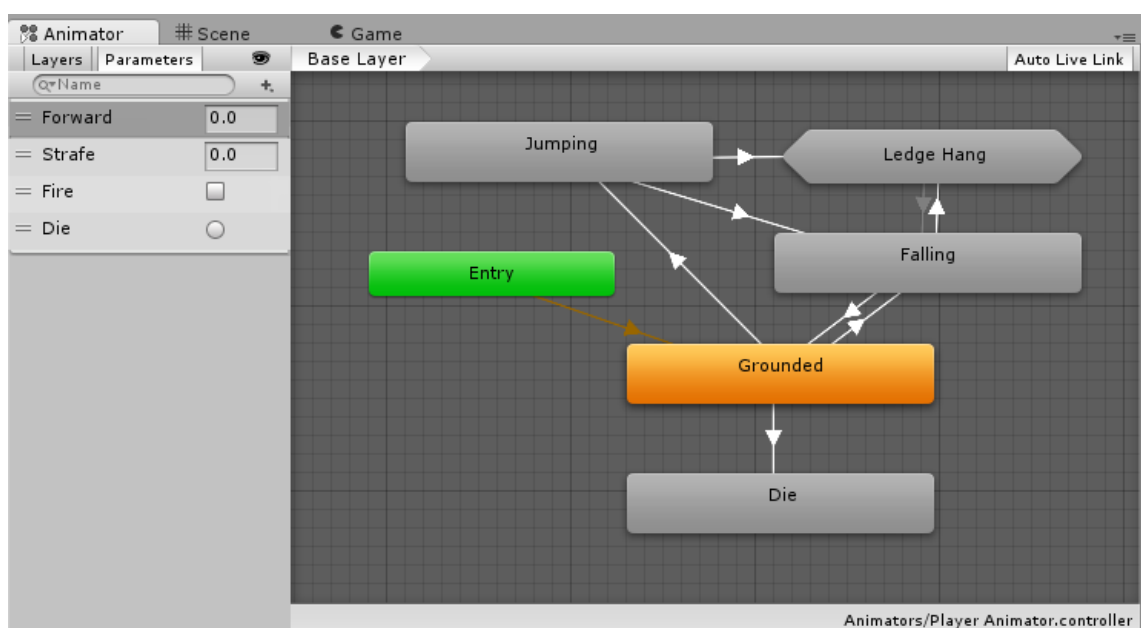


Рисунок 3.1 – Простой контроллер аниматора

Скриптинг анимации

Unity имеет множество инструментов для управления анимацией:

1. смешивание анимации;
2. слои анимации;
3. аддитивные анимации;
4. процедурная анимация.

3.3 Описание ключевых классов

Класс `CharacterPanel.cs`

Исходный код скрипта представлен в приложении Б1.

Класс панели. Показывает информацию о снаряжении персонажа и отображении статистических данных (рисунок 3.2).

Массив из слотов: `public Slot[] equipmentSlots;`

В функции `EquipItem` происходит смена положения предмета в инвентаре с тем предметом в той ячейке (возможно пустой ячейке), которая может содержать предмет данного типа.

В функции `ShowToolTip` происходит обработка положения курсора мыши и вывод информации о слоте, на который направлен курсор.

В функции `CalcStats` происходит обновление статистических данных при изменении состояния объекта (при снятии предмета или его надевании).



Рисунок 3.2 - Отображение статистических данных

Класс Item.cs

Исходный код скрипта представлен в приложении Б2.

Создается конструктор для создания объектов. При создании можно указать его характеристики (имя, описание и т.д.):

```
public Item(string itemName, string description, ItemType
itemType, Quality quality, string spriteNeutral, string
spritehighlighted, int maxSize)
```

Данный конструктор вызывается, когда мы создаем новый экземпляр объекта.

При помощи функции GetToolTip можно вывести подсказку о предмете, на который наведен курсор мыши.

Класс ItemContainer.cs

Исходный код скрипта представлен в приложении Б3.

Содержит списки типов объектов:

1.Оружие:

```
public List<Item> Weapons
```

2.Экипировка:

```
public List<Item> Equipment
```

3.Расходные материалы:

```
public List<Item> Consumeables
```

Заполняем поля при помощи аксессоров get и set. Эти поля можно читать и в эти поля можно записывать данные.

Класс ItemManager.cs

В классе ItemManager (приложение Б4) можно указать свойства (рисунок 3.3) для генерируемого объекта.

В функции CreateItem происходит чтение XML документа и занесение полученных объектов по контейнерам (EQUIPMENT, WEAPON, CONSUMEABLE).

Property	Value
Script	ItemManager
Item Type	NECK
Quality	RARE
Category	EQUIPMENT
Sprite Neutral	Equipment/NeckNormal
Sprite Highlighted	Equipment/NeckNormal
Item Name	Rare Neck
Description	Nice neck
Max Size	1
Intellect	13
Agility	15
Strength	16
Attack Speed	0
Health	0
Mana	0

Add Component

Рисунок 3.3 - Свойства для генерируемого объекта в классе

ItemManager

Класс ItemScript.cs

Данный класс (приложение Б5) содержит информацию о типах предметов и их качестве.

```
public enum ItemType {CONSUMEABLE, MAINHAND, TWOHAND, OFFHAND,  
HEAD, NECK, CHEST, RING, LEGS, BRACERS, BOOTS, TRINKET,  
SHOULDERS, BELT, GENERIC, GENERICWEAPON};  
public enum Quality {COMMON, UNCOMMON, RARE, EPIC};
```

В функции «Use» обрабатывается предмет в конкретном слоте при клике мышки.

В функции «GetTooltip» выводится информации о предмете при наведении на него мышки.

Класс WorldInteraction.cs

Исходный код скрипта представлен в приложении Б6.

В функции Start инициализируются все основные параметры игрового мира.

В функции Update обрабатывается обновление кадра для игрового объекта.

В функции AddItemsToInventory мы генерируем случайное число от 0 до 2 и в зависимости от него выполняем алгоритм, находящийся внутри оператора «case». Происходит добавление скрипта на объект и генерация случайного предмета.

Возможные варианты:

0: генерация расходного материала:

```
inventory.Additem (newConsumeable);
```

1: генерация оружия:

```
inventory.Additem (newWeapon);
```

2: генерация снаряжения:

```
inventory.Additem (newEquipment);
```

3.4 Варианты работы с инвентарем

Пользователь может совершать множество манипуляций с инвентарем.

На рисунке 3.4 пользователь выбирает определенный предмет, при наведении на который отображаются его характеристики.



Рисунок 3.4 – Показ статических данных о предмете

При нажатии ПКП нужный предмет автоматически перенесется в нужный слот, если слот доступный для этого предмета доступен. Аналогичные действия можно совершить, используя ЛКМ и перенос предмета в нужный слот.

При успешном переносе он займет слот в панели характеристик персонажа. Данные о характеристиках персонажа обновляются. При надевании предмета все показатели прибавляются (рисунок 3.5).



Рисунок 3.5 – Вставка выбранного предмета в слот панели характеристик персонажа

При попытке переместить предмет (рисунок 3.6) в слот, который предназначен для предмета другого типа он автоматически вернется в инвентарь (рисунок 3.7).

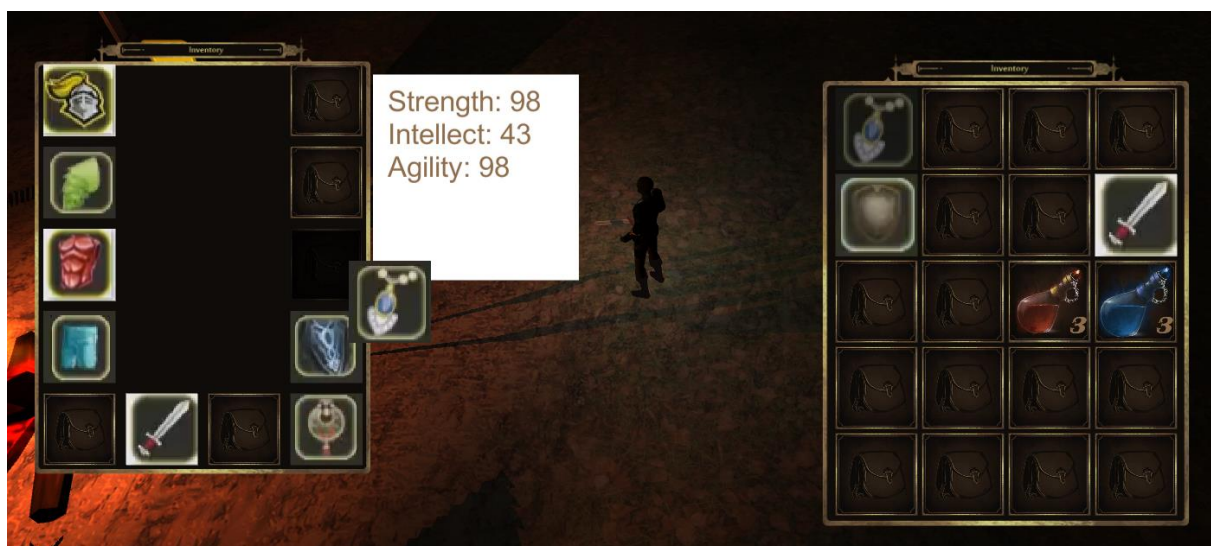


Рисунок 3.6 – Попытка переместить предмет в слот

При удалении предмета (рисунок 3.8) из панели персонажа характеристики этого предмета вычитаются из статистики (рисунок 3.9).

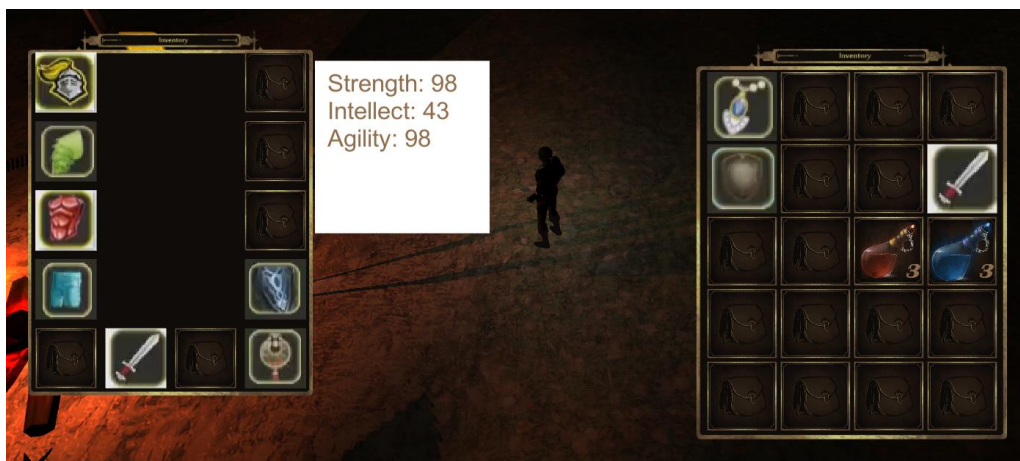


Рисунок 3.7 – Предмет автоматически вернулся в инвентарь

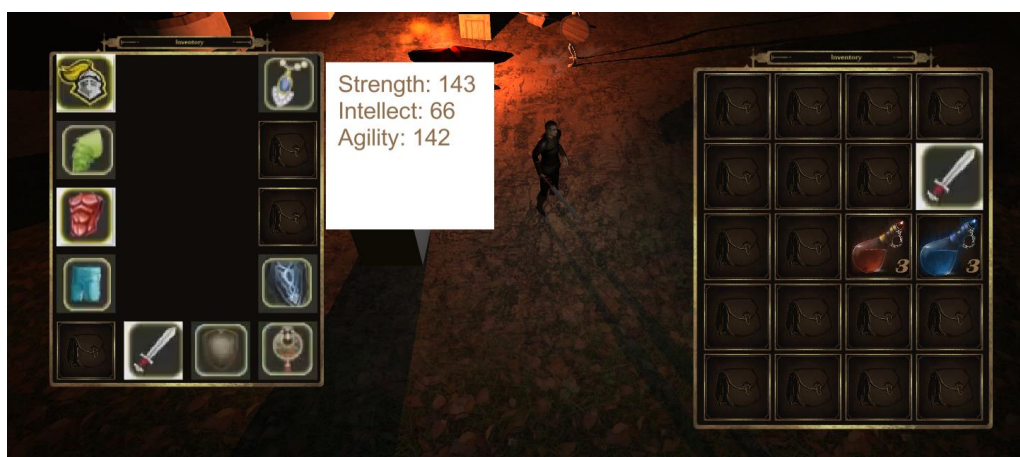


Рисунок 3.8 – Текущие показатели персонажа

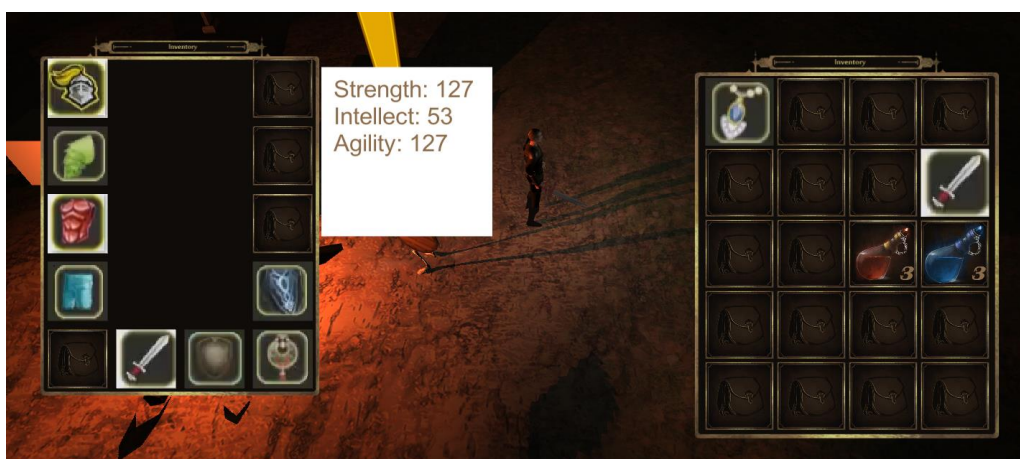


Рисунок 3.9– Вычитание показателей после снятия предмета

ЗАКЛЮЧЕНИЕ

В ходе выполнения ВКР был проведен анализ предметной области, сред для создания игр и программных продуктов для 3D моделирования. Изучена среда для разработки игр Unity 5, включая среду MonoDevelop, а также среда для моделирования трехмерных моделей 3D Studio Max.

При помощи детального анализа программных средств для реализации проекта было выбрано наиболее понятное и простое ПО, что позволяет избежать трудностей при создании игры и разработать её в ограниченный срок.

Создана 3D модель главного персонажа игры, состоящая из нескольких объектов:

1. голова;
2. тело;
3. руки;
4. наплечники;
5. ноги;
6. ботинки;

Разработаны необходимые группы анимаций:

1. стойка;
2. ходьба;
3. бег;
4. удар мечом;
5. получение урона;
6. смерть;

Благодаря изучению аналогичных игр были найдены их недостатки, которые были учтены при написании ВКР. Оптимизация 3D модели и ее текстур дало большую производительность.

Исходя из всего вышеперечисленного, была разработана собственная система контроля управления персонажа на примере компьютерной игры жанра «RPG».

Рынок компьютерных игр является большим сегментом мирового рынка цифрового контента, в котором ежегодно генерируются многомиллиардные доходы. Пользователи ПК-игр составляют ядро всей аудитории. Поэтому развитие данного проекта будет актуально еще долгое время.

Требования технического задания были выполнены в полном объёме и в установленные сроки.

СПИСОК ЛИТЕРАТУРЫ

1. Билл Ф. Создание трехмерных персонажей: М.: ДМК, 1999 – 445 с.
2. Бриллиант, К. Цифровая модель человека: М.: КУДИЦ - ОБРАЗ, 2004 – 140 с.
3. Вард Э. Разработка игрового персонажа: Kila. М.: Game Character Creation Series, 2017 – 319 с.
4. Рис С. Анимация персонажей в 3D Studio Max: М.: СОМ, 2002 – 267 с.
5. Сыркин Ю.И. Краткое учебное пособие по курсу 3ds Max. СПб.: Международная школа дизайна, 2016 – 29 с.
6. Тозик. В., Меженин А., Звягин К. 3ds Max. Трехмерное моделирование и анимация на примерах. СПб.: БВХ, 2008 – 18 с.
7. Харьковский А.В. Лучший самоучитель 3ds Max 2013: Учебный курс. Изд. 4-е. М.: Компьютер-House iBoox.Ru, 2012 – 477 с.
8. Render Magazine. 2016. Т. 3, вып. 42. С. 55-60.
9. Render Magazine. 2016. Т. 12, вып. 51. С. 54-62.
10. Render Magazine. 2016. Т. 11, вып. 50. С. 55-63.
11. Создание и настройка персонажа в 3ds Max. Часть 1: Моделирование. Бесшовное текстурирование [Электронный ресурс] // Render.ru: Компьютерная графика и анимация, 2007. 2 ноября. — Режим доступа: <http://render.ru/book/594>, свободный — Загл. с экрана. (Дата обращения: 14.02.2017).
12. Создание персонажа для игры - полный цикл [Электронный ресурс] // 3dmir.ru: Вся компьютерная графика, 2007. 27 февраля. — Режим доступа: http://www.3dmir.ru/s_tutor/tutor/452.html, свободный — Загл. с экрана. (Дата обращения: 14.02.2017).
13. Игровой персонаж - Википедия [Электронный ресурс] // <https://ru.wikipedia.org>: Википедия – свободная энциклопедия, 2013. 8 августа. — Режим доступа: https://ru.wikipedia.org/wiki/Игровой_персонаж, свободный — Загл. с экрана. (Дата обращения: 20.05.2017).

14. [Гайд] Управление персонажем // GoHa.ru [Электронный ресурс] // <http://forums.goha.ru>: GoHa.ru, 2016. 30 июня. — Режим доступа: http://forums.goha.ru/showthread_82_0_t1134854/, свободный — Загл. с экрана. (Дата обращения: 20.05.2017).
15. Способ управления игрой с помощью ввода пауз spm // GoHa.ru [Электронный ресурс] // <http://www.findpatent.ru/>: Патентный поиск, Поиск патентов и изобретений РФ и СССР, 2012. — Режим доступа: <http://www.findpatent.ru/patent/230/2302893.html/>, свободный — Загл. с экрана. (Дата обращения: 20.05.2017).
16. Джойстик - Википедия [Электронный ресурс] // <https://ru.wikipedia.org>: Википедия – свободная энциклопедия, 2015. 29 декабря. — Режим доступа: <https://ru.wikipedia.org/wiki/Джойстик>, свободный — Загл. с экрана. (Дата обращения: 20.05.2017).
17. Unity (игровой движок) - Википедия [Электронный ресурс] // <https://ru.wikipedia.org>: Википедия – свободная энциклопедия, 2017. 13 мая. — Режим доступа: [https://ru.wikipedia.org/wiki/Unity_\(игровой_движок\)](https://ru.wikipedia.org/wiki/Unity_(игровой_движок)), свободный — Загл. с экрана. (Дата обращения: 20.05.2017).
18. Особенности бесплатного кроссплатформенного движка Unity3d [Электронный ресурс] // <http://gamesmaker.ru>: Создание игр и игровых движков, 2013. 27 декабря. — Режим доступа: <http://gamesmaker.ru/3d-game-engines/unity3d/osobennosti-besplatnogo-krossplatformennogo-dvizhka-unity3d/>, свободный — Загл. с экрана. (Дата обращения: 20.05.2017).
19. Обзор самых популярных движков для разработки игр [Электронный ресурс] // <https://хакер.ru/>: «Хакер» - Безопасность, разработка, DevOps, 2014. 5 сентября. — Режим доступа: <https://хакер.ru/2014/09/05/game-development-engines-review/>, свободный — Загл. с экрана. (Дата обращения: 20.05.2017).
20. Противостояние 3D-титанов: 3ds Max, Maya, Blender — кто лучше? (часть 1) – YouTube [Электронный ресурс] // <https://www.youtube.com/>: YouTube, 2014. 19 октября. — Режим доступа:

<https://www.youtube.com/watch?v=ruIcEgPvl8Y>, свободный — Загл. с экрана. (Дата обращения: 20.05.2017).

21. Компьютерная ролевая игра - Википедия [Электронный ресурс] // <https://ru.wikipedia.org>: Википедия – свободная энциклопедия, 2017. 27 марта. — Режим доступа: https://ru.wikipedia.org/wiki/Компьютерная_ролевая_игра, свободный — Загл. с экрана. (Дата обращения: 20.05.2017).

22. UV-преобразование - Википедия [Электронный ресурс] // <https://ru.wikipedia.org>: Википедия – свободная энциклопедия, 2017. 5 марта. — Режим доступа: <https://ru.wikipedia.org/wiki/UV-преобразование>, свободный — Загл. с экрана. (Дата обращения: 21.05.2017).

23. Unity – Руководство: Создание и Использование Скриптов [Электронный ресурс] // <https://unity3d.com>: Unity – Game Engine, 2015. — Режим доступа: <https://docs.unity3d.com/ru/530/Manual/CreatingAndUsingScripts.html>, свободный — Загл. с экрана. (Дата обращения: 21.05.2017).

24. 15 самых популярных языков программирования по версии GitHub [Электронный ресурс] // <https://habrahabr.ru>: Хабрахабр, 2016. 26 декабря. — Режим доступа: <https://habrahabr.ru/post/310262/>, свободный — Загл. с экрана. (Дата обращения: 21.05.2017).

25. Unity – Руководство: Аниматор и контроллер аниматора [Электронный ресурс] // <https://unity3d.com>: Unity – Game Engine, 2015. — Режим доступа: <https://docs.unity3d.com/ru/530/Manual/Animator.html>, свободный — Загл. с экрана. (Дата обращения: 21.05.2017).

26. Управление игровыми объектами (GameObjects) с помощью компонентов [Электронный ресурс] // <https://unity3d.com>: Unity – Game Engine, 2015. — Режим доступа: <https://docs.unity3d.com/ru/530/Manual/ControllingGameObjectsComponents.html>, свободный — Загл. с экрана. (Дата обращения: 21.05.2017).

27. Полигональное моделирование - Википедия [Электронный ресурс] // <https://ru.wikipedia.org>: Википедия – свободная энциклопедия, 2016. 31 марта. — Режим доступа: https://ru.wikipedia.org/wiki/Полигональное_моделирование, свободный — Загл. с экрана. (Дата обращения: 21.05.2017).
28. Полигональная сетка - Википедия [Электронный ресурс] // <https://ru.wikipedia.org>: Википедия – свободная энциклопедия, 2017. 8 мая. — Режим доступа: https://ru.wikipedia.org/wiki/Полигональная_сетка, свободный — Загл. с экрана. (Дата обращения: 21.05.2017).
29. MonoDevelop - Википедия [Электронный ресурс] // <https://ru.wikipedia.org>: Википедия – свободная энциклопедия, 2017. 14 января. — Режим доступа: <https://ru.wikipedia.org/wiki/MonoDevelop>, свободный — Загл. с экрана. (Дата обращения: 21.05.2017).
30. Кадровая частота - Википедия [Электронный ресурс] // <https://ru.wikipedia.org>: Википедия – свободная энциклопедия, 2017. 17 апреля. — Режим доступа: https://ru.wikipedia.org/wiki/Кадровая_частота, свободный — Загл. с экрана. (Дата обращения: 22.05.2017).
31. "Yes, but how many polygons?" An artist blog entry with interesting numbers [Электронный ресурс] // <https://forum.beyond3d.com/>: Beyond3d Forum, 2008. 2 ноября. — Режим доступа: <https://forum.beyond3d.com/threads/yes-but-how-many-polygons-an-artist-blog-entry-with-interesting-numbers.39321/page-60/>, свободный — Загл. с экрана. (Дата обращения: 22.05.2017).
32. Microsoft Visual Studio - Википедия [Электронный ресурс] // <https://ru.wikipedia.org>: Википедия – свободная энциклопедия, 2017. 7 мая. — Режим доступа: https://ru.wikipedia.org/wiki/Microsoft_Visual_Studio, свободный — Загл. с экрана. (Дата обращения: 22.05.2017).
33. Полный обзор Unity 5 [Электронный ресурс] // <http://devgam.com/>: Информационный портал об игровой индустрии, 2017. 7 мая. — Режим

доступа: <http://devgam.com/polnyj-obzor-unity-5>, свободный — Загл. с экрана. (Дата обращения: 22.05.2017).

34. Unity – Руководство: Коллайдеры (Colliders) [Электронный ресурс] // <https://unity3d.com>: Unity – Game Engine, 2015. — Режим доступа: <https://docs.unity3d.com/ru/530/Manual/CollidersOverview.html>, свободный — Загл. с экрана. (Дата обращения: 23.05.2017).

35. Инверсная кинематика- Википедия [Электронный ресурс] // <https://ru.wikipedia.org>: Википедия – свободная энциклопедия, 2013. 21 октября. — Режим доступа: https://ru.wikipedia.org/wiki/Инверсная_кинематика, свободный — Загл. с экрана. (Дата обращения: 04.05.2017).

ПРИЛОЖЕНИЕ А

А1 Используемые термины

1. Захват движения (Motion Capture) – процесс фиксирования движений человека и дальнейшего переноса движений на модель.
2. NURBS (Non-Uniform Rational B-Spline) - неоднородные рациональные B-сплайны.
3. Terrain - площадка, которую используют для отображения «земли» в среде Unity.
4. Unity— инструмент для разработки двух- и трёхмерных приложений и игр, работающий под операционными системами Windows, Linux и OS X [17].
5. UV-преобразование или развёртка в трёхмерной графике (англ. UV map) — соответствие между координатами на поверхности трёхмерного объекта (X, Y, Z) и координатами на текстуре (U, V) [23].
6. Аксессор – вывод неявного метода. Get – для получения данных. Set – для записи.
7. Ассет – компонент проекта в Unity.
8. Высокополигональная модель (Hi-poly) - 3D модель с большим количеством полигонов (от 500 000 до нескольких миллионов). Пример показан на рисунке 6.2.
9. Джойстик - устройство ввода информации в персональный компьютер, которое представляет собой качающуюся в двух плоскостях вертикальную ручку [16].
10. Игровой движок – среда для разработки компьютерных игр.
11. Инвентарь – набор игровых предметов, которые есть в наличии у персонажа.
12. Инверсная кинематика – тип планирования движения, который активно используется в робототехнике, трёхмерной компьютерной анимации и в разработке компьютерных игр, алгоритм которого противоположен алгоритму прямой кинематики.

- 13.Коллайдер (Colliders) – инструмент, который определяет тип взаимодействия объекта с игровым миром.
- 14.Контроллеры аниматоров (Animator Controllers) – инструмент, позволяющий настраивать и управлять набором анимаций персонажа или иного анимированного объекта (Game Object).
- 15.Концепт арт – растровое изображение, которое выступает в качестве основы для создания 3D модели.
- 16.Кости (Bones) – набор управляющих элементов для анимирования 3D модели. Пример костей указан на рисунке 9.
- 17.Машина состояний (State Machine) - род некоторой программы написанной языком визуального программирования в Unity.
18. Модификатор – свойства или опции, применяемые к 3D модели, представляемые в виде стека.
- 19.Низкополигональная модель (Low-poly) – 3D модель с низким количеством полигонов (до 40 000). Пример показан на рисунке 6.1.
- 20.Персонаж - обозначение персонажа в компьютерных играх, который управляется человеком-игроком [13].
- 21.Полигон - если три или более точек координат заданы в качестве вершин и соединены ребрами, то они формируют многоугольник (полигон), который может иметь цвет и текстуру [27].
- 22.Развертка – растровое изображение отражающие UV-ординаты [22].
- 23.Размытие (motion blur) – один из видов артефактов изображения.
- 24.Ретопология – процесс оптимизации Hi-poly модели.
- 25.Риггинг (Rig) – процесс создания костей.
- 26.Скининг (Skinning) — процесс привязки игрового персонажа к системе связанных между собой объектов, с помощью которых можно управлять динамикой персонажа.
- 27.Скрипт — это программа или программный 1файл сценарий, которые автоматизируют некоторую задачу, которую пользователь делал бы вручную, используя интерфейс программы.

- 28.Скульптинг – процесс создания Hi-poly модели используя специальные инструменты для вдавливания и выдавливания.
- 29.Текстура – объект, который включает в себя множество свойств, для наложения на 3D модель.
- 30.Текстурирование – процесс создания текстурных карт.
- 31.Топология (меш) – сеточное [28] представление 3D модели.

A2 Игры в анкете

Diablo, Diablo II , Diablo III, World of Warcraft, Dota 2, Grim Down, Titan Quest, Nox, Torment: Tides of Numenera, Warhammer 40,000: Dawn of War II — Retribution, Gothic, Gothic II, Gothic 3, Arcania – Gothic 4, Dragon Age: Origins, Dragon Age II, Dragon Age: Inquisition, Aion, Grim Dawn, Torchlight II, Blade and Soul, TERA online, Neverwinter, Lineage II: Gracia, Age of Conan: Hyborian Adventures, Panzar: Forged by Chaos, Drakensang, Drakensang: The River Of Time, Drakensang Online, Karos online, ArcheAge, Revelation (Revelation online), Guild Wars, Guild Wars 2, MU Legend (MU2), Arcanum: Of Steamworks And Magick Obscura, Baldur's Gate / Baldur's Gate II: Shadows Of Amn, Divine Divinity, Beyond Divinity, Divinity II: Ego Draconis, Divinity: Original Sin, Dungeon Siege , Dungeon Siege II, Dungeon Siege III, Ever Quest, EverQuest II, Fable: The Lost Chapters, Fable II, Fable III, Hellgate: London, Icewind Dale, Icewind Dale II, Might & Magic I – X, Planescape: Torment, Risen, Risen 2, Risen 3: Titan Lords, Sacred, Sacred 2: Fallen Angel, Sacred Citadel, Sacred 3, Tales Of The Unknown: Volume I – The Bard's Tale, The Bard's Tale II: The Destiny Knight, The Bard's Tale III: The Thief of Fate, Final Fantasy XIII / XIII-2 / XV, Lightning Returns: Final Fantasy XIII, Kingdoms of Amalur: Reckoning, Ni no Kuni, The Banner Saga, Arche Age, Wasteland 2, Pillars of Eternity, Tyranny, Rohan: Blood Feud, Atlantica Online, Dungeons & Dragons, FlyFF, Ultima Online, Royal Quest, Age of empires III, Lineage II: Gracia, Divine Divinity.

А3 Сравнительный анализ продуктов для 3D моделирования

Таблица 3 - Анализ программных продуктов для 3D моделирования

Инструмент	3D Studio Max	Maya	Blender
<p>Â Интерфейс</p> <p>Â Настройки</p>	<p>Плюсы:</p> <p>Â Минимализм</p> <p>Â Дружелюбность</p> <p>Â Визуальная понятность</p> <p>Â Числовые координаты всегда видны и доступны даже для подобъектов</p> <p>Â Квад-меню</p> <p>Â Разнообразие горячих клавиш</p> <p>Â Гибкий стек модификаторов</p> <p>Â Пиктограммы инструментов</p> <p>Минусы:</p> <p>Â Неудобные текстовые поля</p>	<p>Плюсы:</p> <p>Â Функциональный интерфейс</p> <p>Â Наличие кнопок с инструментами</p> <p>Â Возможности создания собственных полок</p> <p>Â Отстегивающееся меню</p> <p>Â Наличие практически всех команд в меню, разные раскладки меню</p> <p>Â Настройка абсолютно всего под себя</p> <p>Минусы:</p> <p>Â Много настроек</p> <p>Â Неудобный большой интерфейс</p>	<p>Плюсы:</p> <p>Â Простой интерфейс</p> <p>Â Удобные горячие клавиши</p> <p>Â Разные раскладки интерфейса</p> <p>Â Фиксация вида при поворотах</p> <p>Â Удобная работа со студией</p> <p>Â Быстрый поиск команды</p> <p>Â Точный клавиатурный ввод большинства команд</p> <p>Минусы:</p> <p>Â Выделение правой кнопкой мыши</p> <p>Â Несоответствие стандартам управления видами</p> <p>Â Нестандартный интерфейс</p> <p>Â Неудобная настройка интерфейса по умолчанию (выделение, управление видами)</p> <p>Â Разбросанность команд, отсутствие логики группировки, сложность освоения</p>

Инструмент	3D Studio Max	Maya	Blender
			Â Практически полное отсутствие пиктограммы инструментов
Удобство видовых окон	<p>Плюсы:</p> <ul style="list-style-type: none"> Â Есть видовой куб Â Есть Инструменты для работы с видами 	<p>Плюсы:</p> <ul style="list-style-type: none"> Â Есть видовой куб <p>Минусы:</p> <ul style="list-style-type: none"> Â При выделении объекта появляется сетка 	<p>Минусы:</p> <ul style="list-style-type: none"> Â Отсутствие поворота окна
Простота создания объектов	<p>Плюсы:</p> <ul style="list-style-type: none"> Â Легко найти Â Параметры создания видны в момент создания 	<p>Плюсы:</p> <ul style="list-style-type: none"> Â Полнота инструментов 	<p>Плюсы:</p> <ul style="list-style-type: none"> Â Есть меню, легко доступно
Разнообразие объектов	<p>Плюсы:</p> <ul style="list-style-type: none"> Â Огромное Â Есть множество готовых решения для архитектурных проектов Â Есть отличная система создания волос с большим количеством инструментов 	<p>Достаточное</p> <p>Плюсы:</p> <ul style="list-style-type: none"> Â Модуль PaintEffects, который дает возможность рисовать виртуальной кистью такие трехмерные объекты, как цветы, трава, объемные узоры и прочее 	<p>Минусы:</p> <ul style="list-style-type: none"> Â Минимально достаточное
Свойства объекта	<p>Плюсы:</p> <ul style="list-style-type: none"> Â Наглядная параметризация 	<p>Минусы:</p> <ul style="list-style-type: none"> Â Нет свойств объекта 	<p>Плюсы:</p> <ul style="list-style-type: none"> Â Геометрически е объекты не параметризованы
Моделирование	<p>Плюсы:</p> <ul style="list-style-type: none"> Â Изобилие 	<p>Плюсы:</p> <ul style="list-style-type: none"> Â Возможность 	<p>Плюсы:</p> <ul style="list-style-type: none"> Â Числовые

Инструмент	3D Studio Max	Maya	Blender
	<p>параметрических примитивов</p> <p>Â Полноценный стек модификаторов</p> <p>Â Изобилие параметризованных модификаторов</p> <p>Â Числовые координаты всегда видны и доступны даже на уровне подобъектов</p> <p>Â Работа с кривыми безье</p> <p>Â Удобные сплайновые методы моделирования</p> <p>Â Мощное полигональное моделирование</p> <p>Â Полное отсутствие не логичных инструментов для полигонального моделирования (функционал)</p> <p>Â Наличие уникальных инструментов (мелкая детализация и удобство работы)</p> <p>Â Превосходный инструмент для ретопологии</p> <p>Â Есть пред просмотр будущего hi-poly объекта во время</p>	<p>настроить любые параметры через инструмент Channel Box или Attibute Editor</p> <p>Â Продвинутое полигональное моделирование</p> <p>Â Работа с кривыми безье</p> <p>Â Хорошая реализация NURBS</p> <p>Â Удобное управление гладкими и жесткими ребрами</p> <p>Â Любую задачу можно решить несколькими способами</p> <p>Â Более просто решаются сложные задачи</p> <p>Â Существует продукт MayaLT , который сосредоточен только на играх</p> <p>Минусы:</p> <p>Â Нелинейное представление объекта (Отсутствие модификаторов и стека)</p> <p>Â Необходимость обращения с историей объекта</p>	<p>координаты всегда видны и доступны даже на уровне подобъектов</p> <p>Â Простые и необходимые модификаторы</p> <p>Â Удобное полигональное моделирование</p> <p>Â Работа с кривыми безье</p> <p>Â Сплайновое моделирование (специфичное)</p> <p>Â Простая и ясная идеология объекта</p> <p>Â Скульптинг</p> <p>Â Самое быстрое моделирование простых объектов (горячие клавиши)</p> <p>Минусы:</p> <p>Â Отсутствие параметрических примитивов</p> <p>Â Проблема гладких и жестких ребер</p> <p>Â Практически отсутствует NURBS моделирование</p> <p>Â Есть не</p>

Инструмент	3D Studio Max	Maya	Blender
	<p>моделирования low-poly модели</p> <p>Â Возможность создавать скрипты</p> <p>Â Есть среда для архитекторов</p> <p>Минусы:</p> <p>Â Сложная реализация NURBS</p>	<p>Â Неудобный доступ к координатам подобъектов</p> <p>Â Есть не логичные инструменты для полигонального моделирования (функционал)</p> <p>Â Не удобный инструмент ретопологии, многие используют другие программы и скрипты</p>	<p>логичные инструменты для полигонального моделирования (функционал)</p> <p>Â Позволяет моделировать в 3-5 раз меньше полигонов, чем в аналогичных продуктах</p>
Полнота главного меню	<p>Плюсы:</p> <p>Â Логично</p> <p>Â Много общих команд</p> <p>Â Легко найти</p> <p>Â Параметры создания видны в момент создания</p>	<p>Плюсы:</p> <p>Â Меню полное</p> <p>Â Переключение интерфейса</p>	<p>Минусы:</p> <p>Â Нестандартное меню</p> <p>Â Непривычно для начинающего</p>
Контекстное меню	<p>Присутствует, но малоинформативно</p>	<p>Плюсы:</p> <p>Â Marking menu</p>	<p>Минусы:</p> <p>Â Не работает контекстное меню на объекте</p>
Удобство преобразований	<p>Плюсы:</p> <p>Â Отличное</p> <p>Â Всегда видны текстовые поля координат</p>	<p>Хорошее</p>	<p>Хорошее, но инструменты маловаты и координаты не совсем очевидны</p>
Стек модификаторов	<p>Плюсы:</p> <p>Â Все реализовано через стек</p> <p>Â Огромное</p>	<p>Плюсы:</p> <p>Â Есть история создания объекта</p>	<p>Плюсы:</p> <p>Â Стек есть, но меньше модификаторов</p>

Инструмент	3D Studio Max	Maya	Blender
	<p>количество модификаторов</p> <p>Â Удобен как для новичков, так и для профессионалов</p>	<p>Минусы:</p> <p>Â Нелинейное представление объекта (Отсутствие модификаторов и стека)</p>	
Булевские операции	<p>Специфическая реализация</p>	<p>Минусы:</p> <p>Â Не совсем просто сделать</p>	<p>Плюсы:</p> <p>Â Реализованы через модификатор</p>
Переход к подобъектам	<p>Плюсы:</p> <p>Â По горячим клавишам и модификаторам</p>	<p>Контекстное меню</p>	<p>Через список</p>
Настройки материалов	<p>Минусы:</p> <p>Â Неудобная реализация</p>	<p>Минусы:</p> <p>Â Трудная настройка</p>	<p>Плюсы:</p> <p>Â Легко настраиваются и редактируются</p>
Работа с развертками	<p>Плюсы:</p> <p>Â На простых объектах удобнее и быстрее выполняет задачу</p> <p>Минусы:</p> <p>Â Производительность</p>	<p>Плюсы:</p> <p>Â Наличие системы рисования и ретуширования текстур прямо на самом объекте</p> <p>Â Удобно заканчивать процесс сложной развертки путем сглаживания кластеров</p> <p>Минусы:</p> <p>Â Неудобство выполнения</p>	<p>Плюсы:</p> <p>Â Хорошо справляется с разверткой простых объектов</p> <p>Минусы:</p> <p>Â Мало инструментов</p>
Текстурирование и запекание (Baking)	<p>Плюсы:</p> <p>Â Легко сделать быструю развертку или</p>	<p>Плюсы:</p> <p>Â Наличие системы рисования и</p>	<p>Плюсы:</p> <p>Â Удобная и логичная система</p>

Инструмент	3D Studio Max	Maya	Blender
	<p>накинуть текстуру</p> <p>Минусы:</p> <ul style="list-style-type: none"> • Запекание 	<p>ретуширования текстур прямо на самом объекте</p> <ul style="list-style-type: none"> • Удобно заканчивать процесс сложной развертки путем сглаживания кластеров 	<p>рисования и ретуширования текстур прямо на самом объекте</p> <ul style="list-style-type: none"> • Удобно делать сложную развертку швами (для персонажей)
Источники света	<p>Плюсы:</p> <ul style="list-style-type: none"> • Есть нацеленные источники света • Фотореалистичны 	<p>Минусы:</p> <ul style="list-style-type: none"> • Источники нужно настраивать (по теням) 	<p>Наглядно настраиваются и создаются</p>
Визуализация	<p>Плюсы:</p> <ul style="list-style-type: none"> • Вызывается по одной кнопке • Множество различных альтернативных визуализаторов • Есть визуализатор Vray (преимущество над Maya) 	<p>Плюсы:</p> <ul style="list-style-type: none"> • Вызывается по одной кнопке <p>Минусы:</p> <ul style="list-style-type: none"> • Нет бесплатных лицензий для визуализатора Mental Ray 	<p>Минусы:</p> <ul style="list-style-type: none"> • Рендер только из камеры • Малое количество визуализаторов
Количество обучаемого материала	<p>Плюсы:</p> <ul style="list-style-type: none"> • Огромное множество 	Достаточно	Достаточно
Востребованность на данный момент	<p>Плюсы:</p> <ul style="list-style-type: none"> • Популярно как и среди новичков, так и профессионалов 	<p>Плюсы:</p> <ul style="list-style-type: none"> • Очень востребовано 	<p>Минусы:</p> <ul style="list-style-type: none"> • Низкая, большие студии не практически не используют • Большая трата на изучение интерфейса сужает круг потенциальных

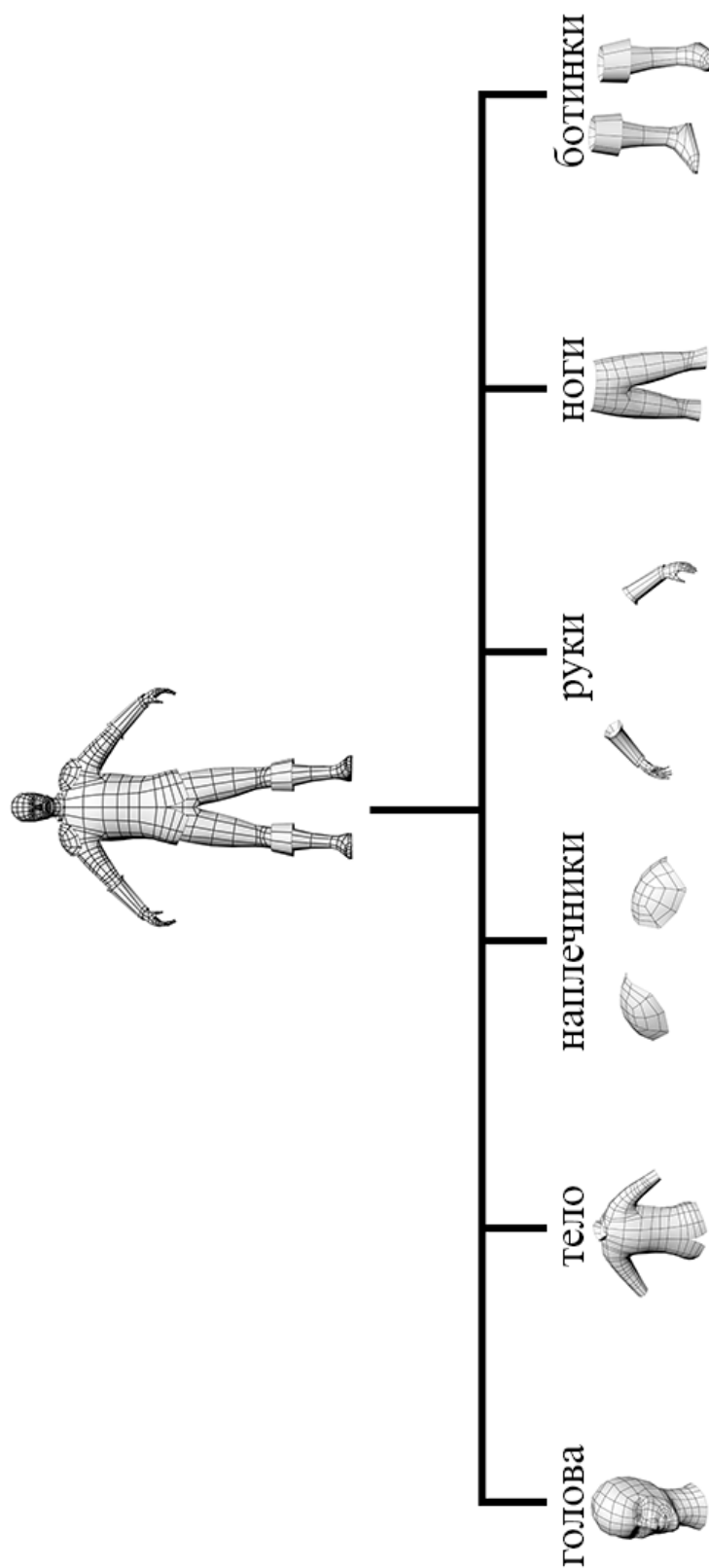
Инструмент	3D Studio Max	Maya	Blender
			пользователей
Работа с анимацией	<p>Плюсы:</p> <ul style="list-style-type: none"> • Имеется готовая иерархическая система костей Viped и CAT. • Все кости (данных систем) легко настраиваются и интегрируются в другие сцены (возможность работы с файлами) • Существует огромное множество готовых наборов качественных анимаций, созданных с помощью Motion Capture • Для иерархии костей не нужно настраивать инверсную кинематику • Идеально подходит для простых анимаций для игр • Возможность обмениваться файлами анимации <p>Минусы:</p> <ul style="list-style-type: none"> • Анимация не реалистична • Без настраиваемой системы костей сложно создавать и управлять обычными костями 	<p>Плюсы:</p> <ul style="list-style-type: none"> • Самая реалистичная анимация • Идеально подходит для профессиональных аниматоров (Используют крупнейшие анимационные студии (Disney, Pixar)) • Создаются в большей степени уникальные анимации • Превосходная система Motion Capture (Human IK) • Многопроходный рендеринг RenderMan <p>Минусы:</p> <ul style="list-style-type: none"> • Приходится анимировать с нуля • Нужны вспомогательные объекты • Специфическая работа с костями, долгая настройка • Очень сложно вносить изменения для готовой анимации, вплоть до 	<p>Плюсы:</p> <ul style="list-style-type: none"> • Инструменты для удобного скиннинга — встроенный скрипт. <p>Минусы:</p> <ul style="list-style-type: none"> • Не идеальная система инверсной кинематики • Приходится много исправлять последствия не точного скиннинга

Инструмент	3D Studio Max	Maya	Blender
		переделывания проекта	
Цена	<p>Минусы:</p> <ul style="list-style-type: none"> • Не многие студии могут позволить (Около 200 000 рублей за трехлетнюю лицензию на момент апреля 2017 года) 	<p>Минусы:</p> <ul style="list-style-type: none"> • Не многие студии могут позволить (Около 200 000 рублей за трехлетнюю лицензию на момент апреля 2017 года) 	<p>Плюсы:</p> <ul style="list-style-type: none"> • Бесплатность
Технические моменты	<p>Плюсы:</p> <ul style="list-style-type: none"> • Закрытость архитектуры (программное и пользовательское представление почти не связаны) • Существует огромное множество готовых скриптов или плагинов • MAX Script <p>Минусы:</p> <ul style="list-style-type: none"> • Не очень хорошая стабильность • Поддерживает только ОС Windows 	<p>Плюсы:</p> <ul style="list-style-type: none"> • язык программирования mel <p>Плюсы:</p> <ul style="list-style-type: none"> • Программистам удобно работать с объектами 	<p>Плюсы:</p> <ul style="list-style-type: none"> • Очень высокая производительность для малых проектов • Высокая стабильность • Низкие требования к ПК • Быстрая установка (Установочный файл занимает мало места) на диске <p>Минусы:</p> <ul style="list-style-type: none"> • Недоработка драйверов видеоадаптера • Нет поддержки OpenGL

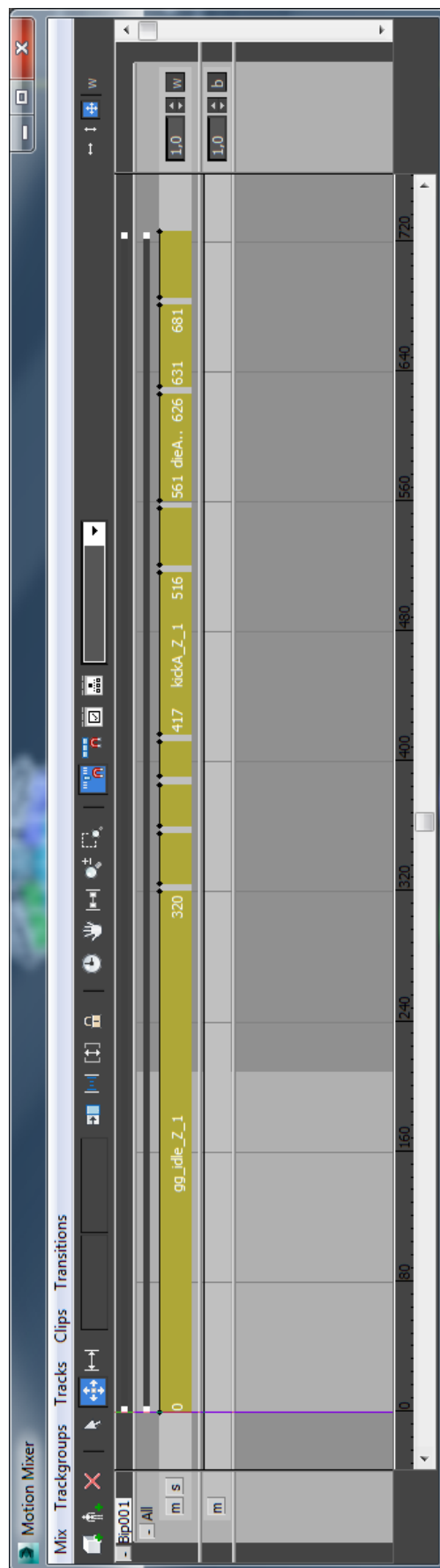
А4 Упрощенная схема сравнения продуктов для 3D моделирования

Инструмент	3D Studio Max	Maya	Blender
■ Интерфейс ■ Настройки	7	4	1
Удобство видовых окон	2	0	-1
Простота создания объектов	2	1	1
Разнообразие объектов	3	1	-1
Свойства объекта	1	-1	-1
Моделирование	12	3	3
Полнота главного меню	4	2	-2
Контекстное меню	0	1	-1
Удобство преобразований	2	0	0
Стек модификаторов	3	0	1
Булевские операции	-1	-1	1
Переход к подобъектам	1	0	0
Настройки материалов	-1	-1	1
Работа с развертками	0	1	0
Текстурирование и запекание (Baking)	0	2	2
Источники света	2	-1	1
Визуализация	3	0	0
Количество обучающего материала	1	0	0
Востребованность на данный момент	1	2	-1
Работа с анимацией	4	1	-1
Цена	-1	-1	1
Технические моменты	1	0	2

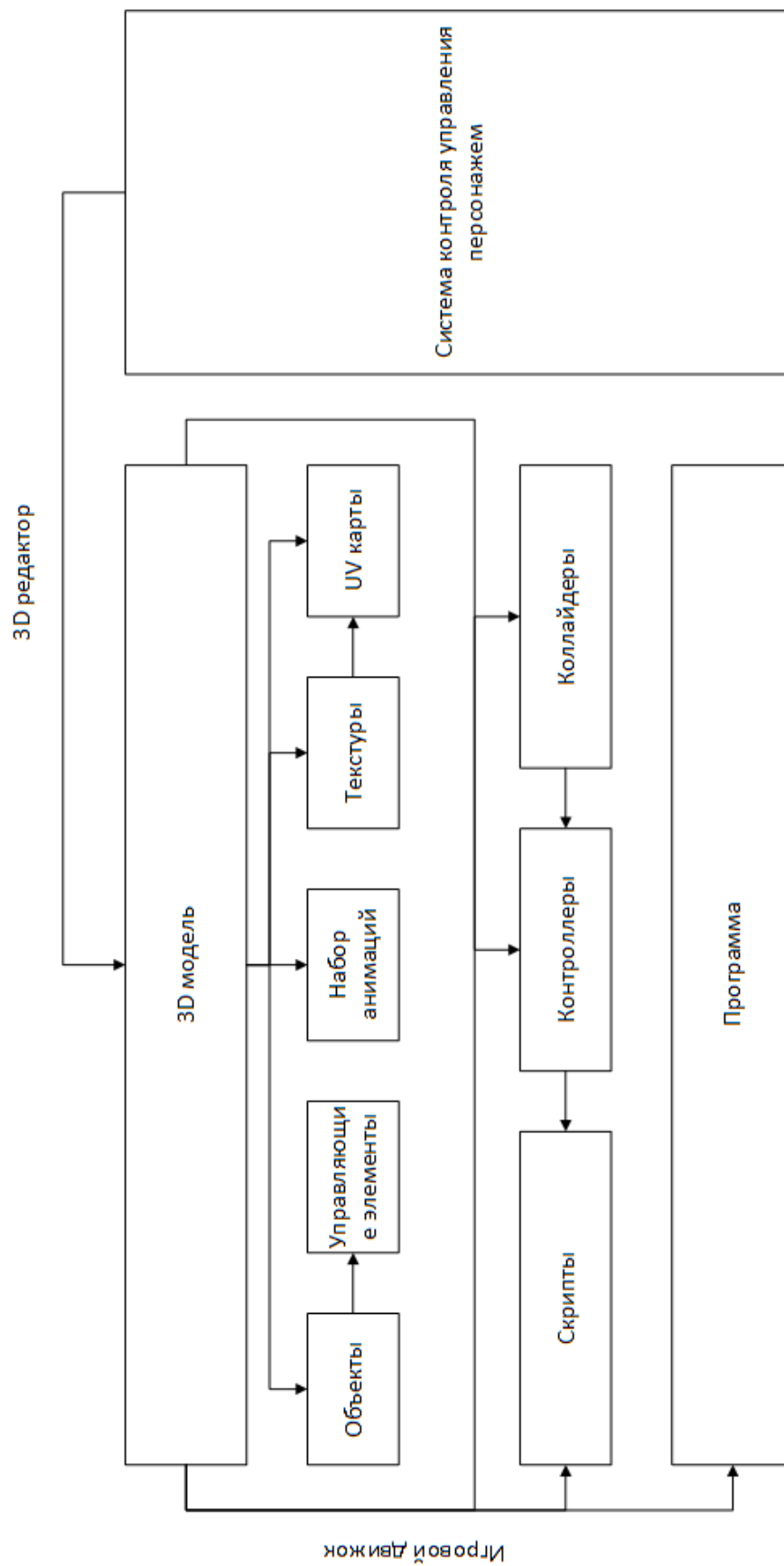
A5 Схема 3D модели главного персонажа



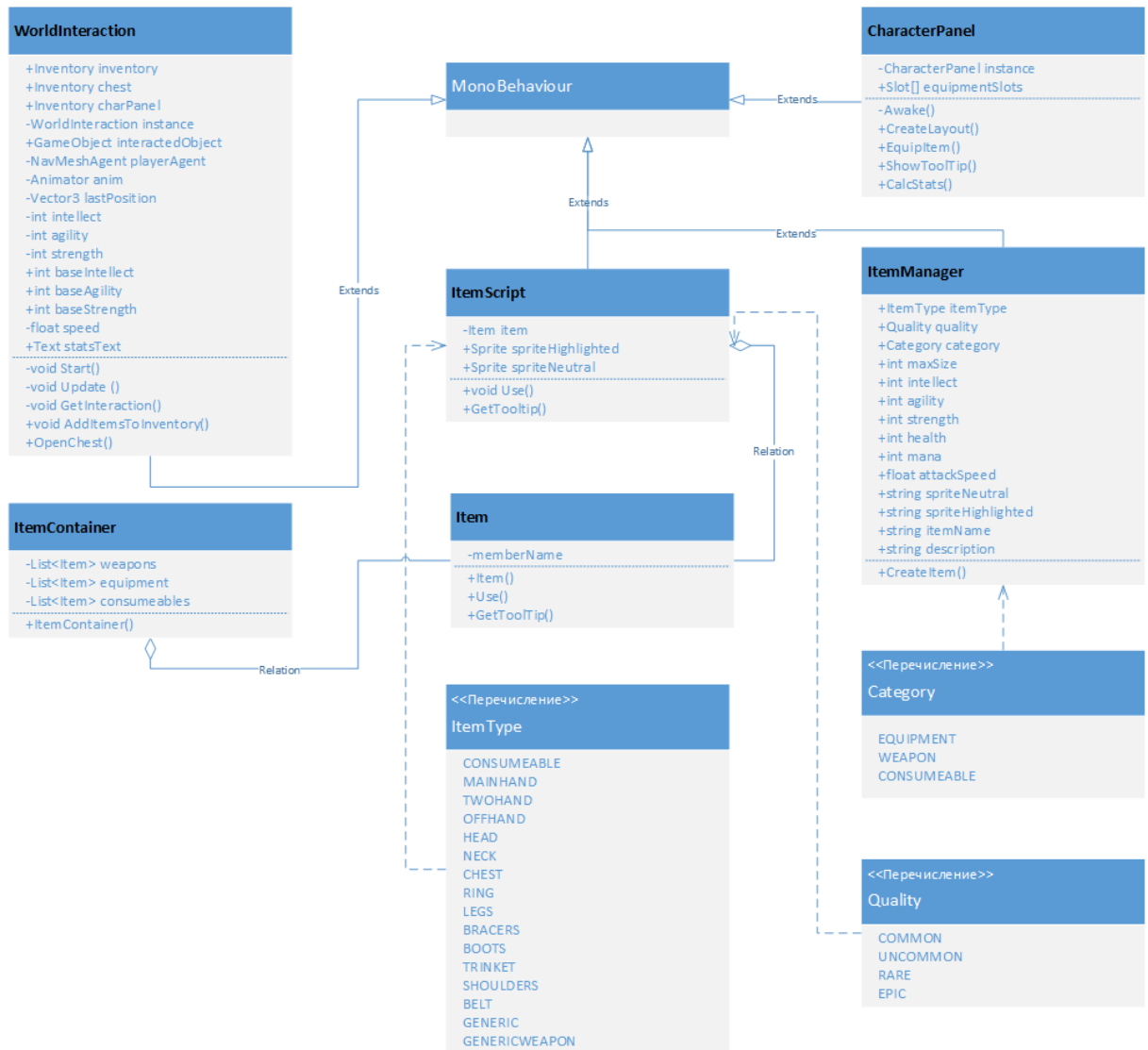
А6 Набор анимаций в среде 3D Studio Max



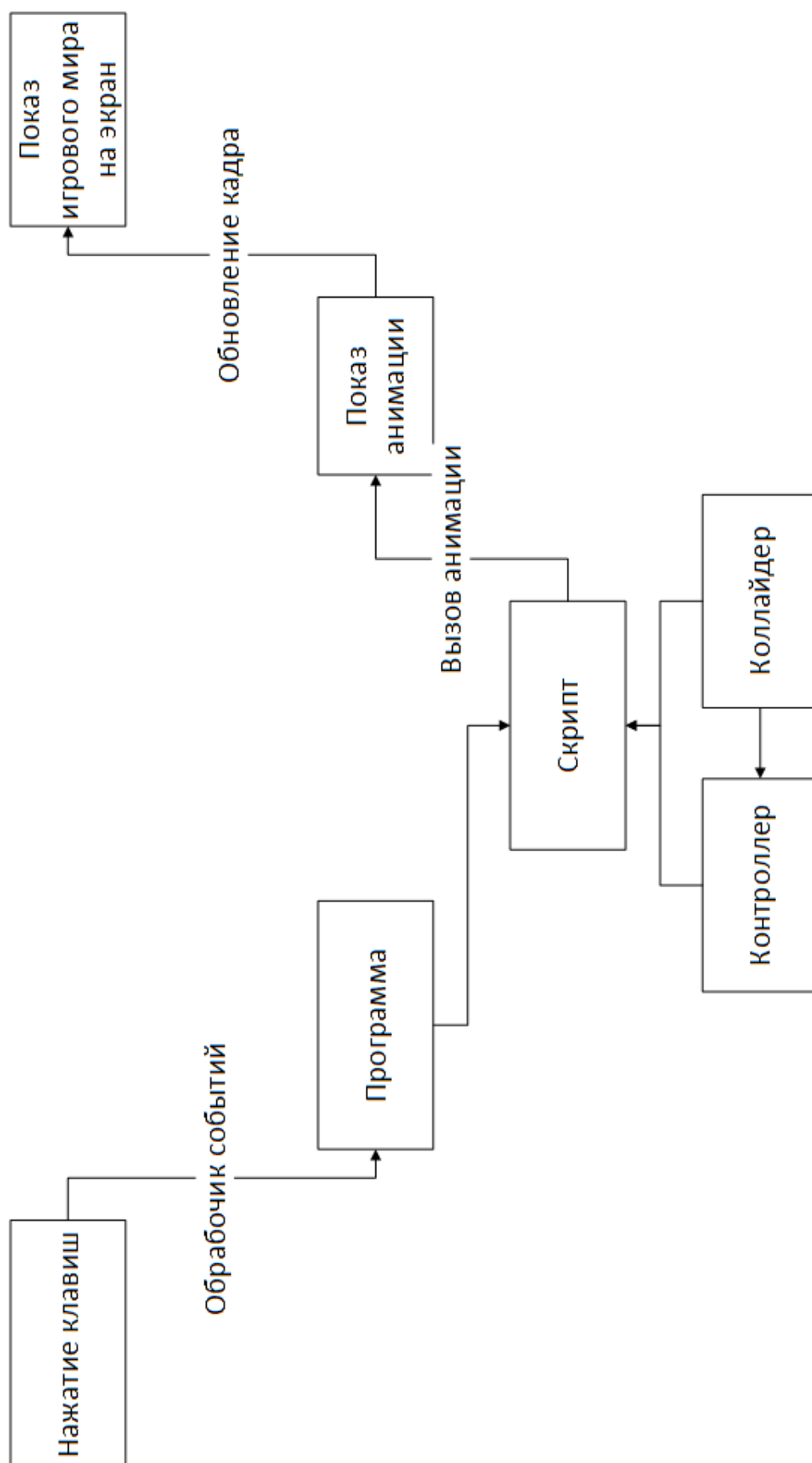
A7 Система управления персонажем



А8 Диаграмма классов



А9 Взаимодействие скрипта с моделью



ПРИЛОЖЕНИЕ Б

Б1 CharacterPanel.cs

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

public class CharacterPanel : Inventory
{
    // 1 8 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
    public Slot[] equipmentSlots;

    private static CharacterPanel instance;

    public static CharacterPanel Instance
    {
        get
        {
            if (instance == null)
            {
                instance =
                GameObject.FindObjectOfType<CharacterPanel> ();
            }
            return instance;
        }
    }

    public Slot WeaponSlot
    {
        get{ return equipmentSlots [9]; }
    }

    public Slot OffHandSlot
    {
        get{ return equipmentSlots [10]; }
    }

    void Awake()
    {
        equipmentSlots =
        transform.GetComponentsInChildren<Slot>();
    }

    public override void CreateLayout()
    {
    }
}

// 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
public void EquipItem(Slot slot, ItemScript item)
```



```

        int intellect = 0;

        foreach (Slot slot in equipmentSlots)
        {
            if (!slot.IsEmpty)
            {
                Equipment e =
(Equipment)slot.CurrentItem.Item;
                strength += e.Strength;
                agility += e.Agility;
                intellect += e.Intellect;
            }
        }

        WorldInteraction.Instance.SetStats (agility, strength,
intellect);
    }
}

```

B2 Item.cs

```

using UnityEngine;
using System.Collections;

public abstract class Item
{
    public ItemType ItemType { get; set; }
    public Quality Quality { get; set; }
    public string SpriteNeutral { get; set; }
    public string SpriteHighlighted { get; set; }
    public int MaxSize { get; set; }
    public string ItemName { get; set; }
    public string Description { get; set; }

    //  utf q j b x d   t w l s z t f c   d t f t s l s z f n w d z t f n ,   d t f z t f n
    //  l s x f n 2 b d 2 d x m u 3 2 d w d x t s z b
    public Item()
    {
    }

    //  f t f t s l s z f n w d z t f n   j d y   l s t f d j b t 2 y   t f o r x d z t f c .   j t
    //  t f o d x z b   d t f l s z f c d   x h t f ,   2 d t c 2 l s b t 2   j f n w h 2 x   l s g t f c l
    //  f t f t s l s z f n w d z t f n   g , d d t f n j z s y   d   l s t f d j b x d   t t f g j c
    //  t f o r x d z b
    public Item(string itemName, string description, ItemType
itemType, Quality quality, string spriteNeutral, string
spriteHighlighted, int maxSize)
    {
        this.ItemName = itemName;
        this.Description = description;
        this.ItemType = itemType;
        this.Quality = quality;
        this.SpriteNeutral = spriteNeutral;
    }
}

```

```

        this.SpriteHighlighted = spriteHighlighted;
        this.MaxSize = maxSize;
    }

    public abstract void Use (Slot slot, ItemScript item);

    // 1 ' 5 7 j 6 7 j 5 4 8 9 4 2
    public virtual string GetToolTip()
    {
        string stats = string.Empty;
        string color = string.Empty;
        string newLine = string.Empty;

        if(Description != string.Empty)
        {
            newLine = "\n";
        }

        switch (Quality)
        {
            case Quality.COMMON:
                color = "white";
                break;
            case Quality.UNCOMMON:
                color = "lime";
                break;
            case Quality.RARE:
                color = "navy";
                break;
            case Quality.EPIC:
                color = "magenta";
                break;
        }
        return string.Format (" <color=" + color +
"><size=16>{0}</size></color><size=14><i><color=lime>" + newLine
+ "\n {1}</color></i></size>", ItemName, Description);
    }
}

```

B3 ItemContainer.cs

```

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

// 5 6 2 5 4 2 4 2 6 7 5 7 8 ' 1 4 4 7 5
public class ItemContainer
{
    private List<Item> weapons = new List<Item>();
    private List<Item> equipment = new List<Item>();
    private List<Item> consumeables = new List<Item>();
    public List<Item> Weapons {

```

```

        get {
            return weapons;
        }
        set {
            weapons = value;
        }
    }

    public List<Item> Equipment {
        get {
            return equipment;
        }
        set {
            equipment = value;
        }
    }

    public List<Item> Consumeables {
        get {
            return consumeables;
        }
        set {
            consumeables = value;
        }
    }

    public ItemContainer()
    {

    }
}

```

B4 ItemManager.cs

```

using UnityEngine;
using System.Collections;
using System;
using System.Xml.Serialization;
using System.IO;

public enum Category {EQUIPMENT, WEAPON, CONSUMEABLE}

public class ItemManager : MonoBehaviour
{

    public ItemType itemType;
    public Quality quality;
    public Category category;
    public string spriteNeutral;
    public string spriteHighlighted;
    public string itemName;
    public string description;
    public int maxSize;
    public int intellect;
}

```

```

        public int agility;
        public int strength;
        public float attackSpeed;
        public int health;
        public int mana;

//  z k x t z x   X M L   j t f d w d x t z b   z   q b t x l s x t z x   t e t f d w h
d t t z x f t x f b d
    public void CreateItem()
    {

        //  z k x t z x   X M L   j t f d w d x t z b
        ItemContainer itemContainer = new ItemContainer ();
        Type[] itemTypes = { typeof(Equipment), typeof(Weapon),
typeof(Consumeable) };
        FileStream fs = new FileStream
(Path.Combine(Application.streamingAssetsPath, "Items.xml"),
        FileMode.Open);
        XmlSerializer serializer = new XmlSerializer
(typeof(ItemContainer), itemTypes);
        itemContainer = (ItemContainer)serializer.Deserialize
(fs);
        serializer.Serialize (fs, itemContainer);
        fs.Close ();
        //  j t t x u   h z x t z y

        //  u t f f z z f t f d b   t e t f d w f f x j f d x l t f f   d t t z x f t x f b d
( o f f t t w   f , o f f t t w L z x   f   t f f w L z x ,   z l s t e t f d w f f x j f d x f .
        switch (category)
        {
            case Category.EQUIPMENT:
                itemContainer.Equipment.Add (new
Equipment(itemName,description,itemType,quality,spriteNeutral,sp
riteHighlighted,maxSize,intellect,agility,strength));
                break;
            case Category.WEAPON:
                itemContainer.Weapons.Add (new
Weapon(itemName,description,itemType,quality,spriteNeutral,spr
iteHighlighted,maxSize,intellect,agility,strength,attackSpeed));
                break;
            case Category.CONSUMEABLE:
                itemContainer.Consumeables.Add (new
Consumeable(itemName,description,itemType,quality,spriteNeutral,
spriteHighlighted,maxSize,health,mana));
                break;
        }
        fs = new FileStream (Path.Combine
(Application.streamingAssetsPath, "Items.xml"),
        FileMode.Create);
        serializer.Serialize (fs, itemContainer);
        fs.Close ();
    }
}

```

B5 ItemScript.cs

```
using UnityEngine;
using System.Collections;

// 0 2 6 1 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99
public enum ItemType {CONSUMEABLE, MAINHAND, TWOHAND, OFFHAND,
HEAD, NECK, CHEST, RING, LEGS, BRACERS, BOOTS, TRINKET,
SHOULDERS, BELT, GENERIC, GENERICWEAPON};
public enum Quality {COMMON, UNCOMMON, RARE, EPIC}

public class ItemScript : MonoBehaviour
{
    public Sprite spriteNeutral; // 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99
    public Sprite spriteHighlighted;
    private Item item;
    public Item Item
    {
        get{ return item;}
        set
        {
            item = value;
        }
    }

    public Sprite spriteHighlighted = Resources.Load<Sprite>
(value.SpriteHighlighted);
    public Sprite spriteNeutral = Resources.Load<Sprite>
(value.SpriteNeutral);
}

}

// 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99
public void Use(Slot slot)
{
    item.Use (slot, this);
}

// 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99
public string GetTooltip()
{
    return item.GetToolTip ();
}

}
```

B6 WorldInteraction.cs

```
using UnityEngine;
using System.Collections;
using UnityEngine.UI;

public class WorldInteraction : MonoBehaviour
{
```

```

private static WorldInteraction instance;

public static WorldInteraction Instance
{
    get
    {
        if(instance == null)
        {
            instance =
GameObject.FindObjectOfType<WorldInteraction> ();
        }
        return instance;
    }
}

NavMeshAgent playerAgent;
Animator anim;
float speed;
Vector3 lastPosition;
public Inventory inventory;
public Inventory chest;
public Inventory charPanel;
public Text statsText;
public int baseIntellect;
public int baseAgility;
public int baseStrength;
private int intellect;
private int agility;
private int strength;
public GameObject interactedObject;

// 开始游戏时初始化
void Start()
{
    SetStats (0,0,0);
    playerAgent = GetComponent<NavMeshAgent> ();
    anim = GetComponent<Animator> ();
}

// 每帧更新
void Update ()
{
    if (Input.GetMouseButtonDown (0) &&
!UnityEngine.EventSystems.EventSystem.current.IsPointerOverGameO
bject ())
        GetInteraction ();
    speed = (transform.position - lastPosition).magnitude;
    lastPosition = transform.position;
    if (speed > 0.05)
        anim.SetBool ("IsWalking", true);
    else
        anim.SetBool ("IsWalking", false);
}

```

```

        if(Input.GetKeyDown(KeyCode.I))
        {
            inventory.Open ();
        }
        if (Input.GetKeyDown (KeyCode.E))
        {
            chest.Open ();
        }

        if(chest.IsOpen)
        {
            if
(!interactedObject.Equals(GameObject.Find("Chest")))
            {
                chest.Open ();
            }
        }

        if (Input.GetKeyDown (KeyCode.C))
        {
            if (charPanel != null)
            {
                charPanel.Open ();
            }
        }
    }

    void GetInteraction()
    {
        Ray interactionRay = Camera.main.ScreenPointToRay
(Input.mousePosition);
        RaycastHit interactionInfo;
        if (Physics.Raycast (interactionRay, out
interactionInfo, Mathf.Infinity))
        {
            interactedObject =
interactionInfo.collider.gameObject;
            if (interactedObject.tag == "Interactable Object")
            {
                playerAgent.stoppingDistance = 1.5f;

                interactedObject.GetComponent<Interactable>().MoveToInteract
ion(playerAgent);
            } else
            {
                playerAgent.stoppingDistance = 0;
                playerAgent.destination =
interactionInfo.point;
            }
        }
    }
}

```



```

        QuestManager.questManager.AddItem
(interactedObject.name, 1);
}

    public void OpenChest()
    {
        chest = interactedObject.GetComponent<ChestScript>
().chestInventory;
        chest.Open ();
    }

    public void SetStats(int agility, int strength, int
intellect)
    {
        this.agility = agility + baseAgility;
        this.strength = strength + baseStrength;
        this.intellect = intellect + baseIntellect;

        statsText.text = string.Format ("Strength:
{0}\nIntellect: {1}\nAgility: {2}", this.strength,
this.intellect, this.agility);
    }
}

```