МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет	Кафедра
«Вычислительная техника»	«Системы автоматизированного проектирования»
(наименование)	(наименование)

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

Профиль «Системы автоматизированного проектирования»
(наименование)

БАКАЛАВРСКАЯ РАБОТА

на тему

Проектирование алгоритмов ветвлений игры жанра «квест»

Студент 25.05.17 Чудинов Ю.А.
(подпись, дата) (фамилия, инициалы)

Руководитель 25.05.17 Ган Д.Д
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр проф. каф. САПР ПГУ 26.05.17 Бондин Д.С.
(должность, место работы) (подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____ № _____)

Заведующий кафедрой _____ Бондин Д.С.
(подпись) (фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от_____№_____)

Секретарь ГЭК _____ _____
подпись) (фамилия, инициалы)

Пенза 2017

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ 5

1 **АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ** 7

1.1 Сравнение с аналогами 7

1.2 Рассмотрение игровых движков 12

1.3 Вывод по первой главе 15

2 **ПРОЕКТИРОВАНИЕ АЛГОРИТМОВ ВЕТВЛЕНИЙ ИГРЫ** 17

2.1 Проектирование алгоритма задания 17

2.2 Проектирование пользовательского интерфейса 18

2.3 Проектирование создания и редактирования заданий 19

2.4 Выбор языка программирования и среды разработки 20

2.5 Вывод по второй главе 23

3 **РАЗРАБОТКА СИСТЕМ ФУНКЦИОНАЛА ВЕТВЛЕНИЙ ИГРЫ** 24

3.1 Взаимодействие пользователя с системой 24

3.2 Описание ключевых классов 28

3.3 Описание алгоритмов 34

3.4 Контрольный пример 36

3.5 Вывод по третьей главе 41

ЗАКЛЮЧЕНИЕ 42

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 43

ПРИЛОЖЕНИЕ А 45

ПРИЛОЖЕНИЕ Б 46

ПРИЛОЖЕНИЕ В 52

ПРИЛОЖЕНИЕ Г 58

ПРИЛОЖЕНИЕ Д 59

ПРИЛОЖЕНИЕ Е 62

ВВЕДЕНИЕ

Тема выпускной квалификационной работы посвящена разработке необходимого функционала алгоритмов ветвлений игры жанра «квест» для создания сюжетных цепочек и побочных заданий. Весь функционал будет разрабатываться на игровом движке Unity 3d версии 5.4 + с использованием языка C# в среде разработки MonoDevelop.

Все игры делятся на бесчисленное количество жанров и их смесей, в выпускной квалификационной работе речь пойдет о таком жанре как «квест», представляющий собой интерактивную историю с главным героем, управляемым игроком с элементами индивидуального отыгрыша системы ролевой игры (RolePlayGame).

Неотъемлемой частью квеста являются поручения, которые игроку необходимо выполнять по мере прохождения игры. За выполнение поручений игрок получает различную награду в виде опыта, денег или различных предметов. Впервые термин квест использовался для обозначения заданий, за которые предполагается награда, на Game Developer Conference 2005г., после чего основательно закрепился и вышел за пределы искомого жанра, далее в работе будет употребляться термин квест именно в этом значении. Поджанр RolePlayGame(сокр. RPG или РПГ) указывает на то, что игроку предстоит выбрать индивидуальный путь развития персонажа и принятия решений в зависимости от его характера.

Цель работы: разработать функционал алгоритмов ветвлений игры жанра «квест».

В связи с поставленной целью решаются следующие задачи:

– Исследовать существующие аналоги.

– Выявить их достоинства и недостатки.

– Составить алгоритмы для реализации многовариантной системы с учетом достоинств и недостатков существующих аналогов.

– Реализовать составленные алгоритмы на языке программирования C# с использованием библиотек Unity.

– Обеспечить визуальное представление алгоритмов и варианты их прохождения.

Разработка необходимого функционала для создания квестов является одной из фундаментальных задач при создании игр такого жанра, ведь именно на нем основывается дальнейшая разработка сюжетной составляющей игры, а значит, чем удобнее и понятнее будет интерфейс создания новых квестов, а так же работы с квестами во время использования уже готовой игры для их непосредственного прохождения, тем выше будет цениться сам продукт среди разработчиков и пользователей.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

В данной главе будут описаны наиболее яркие представители жанра «квест». Поскольку большая их часть написана на собственных игровых движках игровых студий, на них не представляется возможной разработка, поэтому рассмотрения так же требует ассортимент доступных инструментариев.

1.1 Сравнение с аналогами

Среди современных игр в настоящее время существует три популярных аналога:

1. Серия игр Diablo;
2. The Elder Scrolls: Arena
3. Neverwinter Nights
4. Dark souls
5. Diablo 2

Разработанная в 2001 году компанией Blizzard Entertainment Diablo 2[4] стала одной из лучших игр всех времен жанра РПГ, и является такой и на сегодняшний день. Игра представляет собой отличное сочетание хорошо выполненного основного сюжета, атмосферы и боевой системы. Однако в данной игре почти полностью отсутствует система побочных квестов. Лишь несколько раз за всю игру предстоит выполнить задания, не относящиеся к основной линии, что довольно сильно сказывается на разнообразии всей вселенной в целом. В игре полностью отсутствует нелинейность сюжета, игроку не приходится делать выбор между сюжетными развитиями игры, что так же делает её достаточно однообразной и уже на второе прохождение элементы жанра «квест» почти полностью перестают быть интересными. Ниже представлена диалоговая панель игры Diablo 2 (Рисунок 1).

Demons have spread throughout the desert wasteland and threaten to engulf our quiet port-city.
OLD-GAMES.RU

2. поимка преступника, получить данный квест можно от короля или королевы одного из крупных городов, главному герою предстоит отправиться на поиски опасного криминального элемента, обитающего в одноименной пещере, с целью его убийства;

3. доставка или возврат ценностей, бармен в городе предложит перевезти ценный груз из одного города в другой, во время путешествия на героя могут нападать разбойники и охотники, которые хотят грабить караваны.

Список видов побочных квестов на этом не заканчивается, однако нет смысла перечислять их дальше.

В игре каждое подземелье и пространство между городами генерируются процедурно и, к примеру, получить одинаковую планировку одного и того же города практически невозможно.

Релиз игры Neverwinter Nights[6] состоялся в 2002 году студией разработчиков BioWare. Игра с линейным сюжетом и небольшим количеством побочных зданий смогла заинтересовать уже к тому времени достаточно привередливую публику следующей новой особенностью жанра ролевых игр: перед игроком постоянно представали дилеммы морального выбора - прохождение квестов за злодея или за добряка.

Игроку по ходу по мере развития задания могли предложить несколько вариантов того что он сделает дальше, к примеру, отнять у бедных граждан последние деньги и отдать их бандиту, откупиться ли от этого бандита самостоятельно или же силой прогнать последнего. И хоть в конечном итоге это ни к каким существенным последствиям в развитии основного сюжета не приводило, элемент отыгрыша задуманного персонажа был на высоте, что позволяло полностью окунуться в атмосферу игры. Ниже представлено диалоговое окно игры Neverwinter Nights (рисунок 2).

Dark Souls[7] – самая молодая игра из всех сравниваемых аналогов, выпущенная в 2011 году. Для рядового игрока, желающего пройти её на раз.

Pavel
My name's Pavel, I'm one of the new recruits. I just arrived here at the Academy this morning. You're Daevaorn, aren't you?
1 Yes, I am Daevaorn.
2 No, I think you have me confused with someone else.
Journal
Date
Name
Priority
Academy Training
Day 01, Month 06, 1372 [1300]
You have trained long at the Academy, and today is your final day. To begin your adventures in Neverwinter, speak with each of the instructors in the complex. All will be able to give you tutorial information about basic skills. You will need to complete one last lesson depending on whether your main focus is combat, rogue skills, divine magic, or arcane sorcery. Bards will be able to complete any one of these tests, while most other classes must report to the expert in their chosen field.
Character Record, Daevaorn
Elf, Lawful Neutral
Sorcerer (1)
Strength 10 0
Dexterity 16 3
Constitution 12 1
Intelligence 12 1
Wisdom 10 0
Charisma 16 3
AC 13
HP 5/5
Experience 0
Next Level 1000
Main Weapon Unarmed
Attack Bonus 0
Damage 1-3 (Critical 20 x2)
Offhand Weapon Not Applicable
Fortitude 1
Reflex 3
Will 2
Malas
work you pretty hard here at the Academy.
Pavel: My name's Pavel. I'm one of the new recruits. I just arrived here at the Academy this morning. You're Daevaorn, aren't you?
Talk

глубокая и огромная история, хотя и многие её аспекты для фанатов до сих пор являются спорными.

Проходя игру с надлежащей внимательностью, игрок может повторно встречать тех персонажей, что уже видел ранее в самых необычных для них местах: кто-то может застрять в колодце, кто-то попался в клетку, кто-то же просто сидит у костра. И если все делать в надлежащем ключе – их сюжетные линии будут развиваться, и в конечном итоге главный герой узнает историю каждого персонажа, и на развитие истории персонажа герой может повлиять, выполняя различные условия.

Проанализируем основные достоинства и недостатки данных проектов и сведем их в таблицу (Таблица 1):

Таблица 1 - Достоинства и недостатки

Критерии	Diablo 2	The elder scrolls: Arena	Neverwinter Nights	Dark Souls
Основной сюжет	5	4	3	3
Побочные задания	3	5	4	5
Нелинейность сюжета	2	2	5	5

Расшифровка знаков:

– «5» Данный пункт удовлетворяет полностью;

– «4» Данный пункт удовлетворяет полностью, но с небольшими нюансами;

– «3» Пункт выполняет минимальные требования оценки продукта, однако нуждается в доработке

– «2» Пункт не удовлетворяет требованиям.

1.2 Рассмотрение игровых движков

Инструментарий, называемый игровыми движками[12], создан для упрощения и ускорения разработки игр, чтобы не писать всё «с нуля». Игровой движок — это тот модуль игры, который включает в себя игровую логику.

Далее приведен список игровых движков, наиболее подходящих для создания игр жанра «квест».

1.2.1 Unity

Unity[11] — это инструмент для разработки двух и трёхмерных приложений и игр, работающий под операционными системами Windows, LinuxиOS X. Созданные с помощью Unity приложения работают под операционными системами Windows, OS X, WindowsPhone, Android, AppleiOS, Linux, а также на игровых приставках Wii, PlayStation 3, PlayStation 4, Xbox 360, XboxOne и MotionParallax3D дисплеях (устройства для воспроизведения виртуальных голограмм), например, Nettlebox. Есть возможность создавать приложения для запуска в браузерах с помощью специального подключаемого модуля Unity (UnityWebPlayer), а также с помощью реализации технологии WebGL.

Приложения, созданные с помощью Unity[13], поддерживают DirectX и OpenGL. Активно движок используется как крупными разработчиками (Blizzard, EA, QuartSoft, Ubisoft), так и разработчиками Indie-игр (например, ремейк Мор. Утопия (Pathologic), KerbalSpaceProgram, Slender: TheEightPages, Slender: TheArrival, SurgeonSimulator 2013, BaeklyseApps: Guesstheactor и т. п.) в силу наличия бесплатной версии, удобного интерфейса и простоты работы с движком.

Редактор Unity имеет простой Drag&Drop интерфейс, который легко настраивать, состоящий из различных окон, благодаря чему можно производить отладку игры прямо в редакторе. Движок поддерживает три сценарных языка: C#, JavaScript (модификация), Boo (диалект Python).

Boo убран в 5-ой версии. Расчёты физики производит физический движок PhysX от NVIDIA.

Проект в Unity делится на сцены (уровни) — отдельные файлы, содержащие свои игровые миры со своим набором объектов, сценариев, и настроек. Сцены могут содержать в себе как, собственно, объекты (модели), так и пустые игровые объекты — объекты, которые не имеют модели («пустышки»). Объекты, в свою очередь содержат наборы компонентов, с которыми и взаимодействуют скрипты. Также у объектов есть название (в Unity допускается наличие двух и более объектов с одинаковыми названиями), может быть тег (метка) и слой, на котором он должен отображаться. Так, у любого объекта на сцене обязательно присутствует компонент Transform — он хранит в себе координаты местоположения, поворота и размеров объекта по всем трём осям. У объектов с видимой геометрией также по умолчанию присутствует компонент MeshRenderer, делающий модель объекта видимой.

1.2.2 UnrealEngine

UnrealEngine — игровой движок, разрабатываемый и поддерживаемый компанией Epic Games.

Первая игра, созданная на этом движке — Unreal — появилась в 1998 году. С тех пор различные версии движка были использованы в более чем сотне игр и других проектов.

Написанный на языке C++, движок позволяет создавать игры для большинства операционных систем и платформ: Microsoft Windows, Linux, MacOS и MacOSX; консолей Xbox, Xbox 360, XboxOne, PlayStation 2, PlayStation 3, PlayStation 4, PSP, PSVita, Wii, Dreamcast, GameCube и др., а также на различных портативных устройствах, например, устройствах Apple (iPad, iPhone), управляемых системой iOS и прочих.

Для игры по сети поддерживаются технологии WindowsLive, XboxLive, GameSpy и прочие, включая до 64 игроков (клиентов) одновременно. Таким

образом, движок адаптировали и для применения в играх жанра MMORPG (один из примеров: Lineage II).

Все элементы игрового движка представлены в виде объектов, имеющих набор характеристик, и класса, который определяет доступные характеристики. В свою очередь, любой класс является «дочерним» классом object. Среди основных классов и объектов можно выделить следующие:

Актёр (actor) — родительский класс, содержащий все объекты, которые имеют отношение к игровому процессу и имеют пространственные координаты.

Пешка (pawn) — физическая модель игрока или объекта, управляемого искусственным интеллектом. Название происходит от англ. pawn — тот, кем манипулируют (или пешка, поэтому такой объект без какой-либо модели выглядит как пешка).

Метод управления описан специальным объектом, такой объект называется контроллером. Контроллер искусственного интеллекта описывает лишь общее поведение пешки во время игрового процесса, а такие параметры как «здоровье» (количество повреждений, после которых пешка перестаёт функционировать) или, например, расстояние, на котором пешка обращает внимание на звуки, задаются для каждого объекта отдельно.

Мир, уровень (world, gamelevel) — объект, характеризующий общие свойства «пространства», например, силу тяжести и туман, в котором располагаются все актёры. Также может содержать в себе параметры игрового процесса, как, например, игровой режим, для которого предназначен уровень.

Для работы с простыми и, как правило, неподвижными элементами игрового пространства (например, стены) используется двоичное разбиение пространства — всё пространство делится на «заполненное» и «пустое». В «пустой» части пространства располагаются все объекты, а также только в ней может находиться «точка наблюдения» при отрисовке сцены. Возможность полного или частичного помещения объектов в «заполненную»

часть пространства не исключается, однако может привести к неправильной обработке таких объектов (например, расчёт физического взаимодействия) или неправильной отрисовки в случае помещения туда «точки наблюдения» (например, эффект «зала зеркал»). Все пешки, попадающие в «заполненную» часть пространства, сразу «погибают».

1.2.3 CryEngine

CryEngine 3 — игровой движок, разработанный немецкой компанией с ограниченной ответственностью Crytek и являющийся преемником движка CryEngine 2. CryEngine 3 изначально является кроссплатформенным движком — он изначально ориентирован на IBM PC-совместимые компьютеры и игровые консоли Microsoft Xbox 360 и Sony PlayStation 3, а также на их последующие версии. Кроме того, движок ориентирован на разработку массовых многопользовательских онлайновых игр (англ. Massively Multiplayer Online Game — MMOG). Движок CryEngine 3 является полностью коммерческим, но есть и Free SDK версия.

CryEngine один из самых популярных игровых движков, о чем свидетельствует количество «больших игр» созданных на нем: Giant, Sniper II: Ghost Warrior, Cabal II, Far Cry, Ryse: Sonof Rome. Он привлек к себе внимание разработчиков после выхода Crysis. Crysis продемонстрировал новый уровень графики благодаря CryEngine. Позже, были выпущены несколько версий CryEngine с поддержкой новых функциональных возможностей.

1.3 Вывод по первой главе

Подходить к решению вопроса создания квестов следует с особой ответственностью, поскольку эта задача невероятно сложна.

Перед проектированием функционала следует выбрать игровой движок.

Главным критерием выбора среды является удобный интерфейс и легкий порог вхождения в разработку на данном движке. Из всех

перечисленных в первой главе движков лучше всего по критериям подходит Unity 3D.

Требуется разработать инструментарий, подходящий для создания нелинейных ветвей заданий, разработать систему добавления и удаления заданий, таким образом, чтобы разработка самих ветвей алгоритмов игры была максимально комфортна разработчику. Так же требуется создать удобный для пользователя графический интерфейс работы с заданиями и их выполнением.

2 ПРОЕКТИРОВАНИЕ АЛГОРИТМОВ ВЕТВЛЕНИЙ ИГРЫ

Перед тем как реализовать систему, необходимо спроектировать прототип алгоритма задания и основной функционал.

Процесс проектирования можно разделить на 3 этапа:

1. Проектирование алгоритма задания.
2. Проектирование пользовательского интерфейса.
3. Проектирование редактирования и создания заданий.

2.1 Проектирование алгоритма задания

Первым этапом проектирования является создание прототипа алгоритма задания в игре.

Этапы выполнения задания:

1. Доступ к списку заданий у неигрового персонажа.
2. Получение задания.
3. Выполнение необходимых требований согласно заданию.
4. Завершение задания.
5. Разблокировка следующих заданий в квестовой цепи, если такие присутствуют.

Исходя из данного списка, можно судить о том, что у заданий в игре должны быть следующие состояния:

1. Доступное для получения.
2. Принятое.
3. Выполненное.
4. Завершенное.

Таким образом, помимо уже имеющегося доступного списка заданий, после того как игрок получил и выполнил задание, у него появляется возможность взять новые задания, которые стали доступны по завершению предыдущего.

После того, как все требования к заданию были сформированы, можно приступить к проектированию интерфейсов.

2.2 Проектирование пользовательского интерфейса

Основными требованиями пользовательского интерфейса в игре могут являться:

1. Возможность взаимодействия с неигровым персонажем посредством панелей диалогов или же систем повествований.
2. Получение заданий, доступных у конкретного неигрового персонажа.
3. Сдача заданий тем неигровым персонажам, которые связаны с этими заданиями.
4. В случае необходимости, отмена задания ранее полученного.
5. Получение подробного описания задания.
6. Просмотр взятых заданий в собственном журнале заданий.

После того, как основные требования были сформулированы, представим их в виде диаграммы. Для этого будем использовать диаграмму прецедентов.

Диаграмма прецедентов (диаграмма вариантов использования) в UML[1] — диаграмма, отражающая отношения между актерами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне.

Прецедент — возможность моделируемой системы (часть её функциональности), благодаря которой пользователь может получить конкретный, измеримый и нужный ему результат. Прецедент соответствует отдельному сервису системы, определяет один из вариантов её использования и описывает типичный способ взаимодействия пользователя с системой. Варианты использования обычно применяются для спецификации внешних требований к системе.

Диаграмма прецедентов пользовательского интерфейса (Actor - игрок) представлена на рисунке 3.

На диаграмме прецедентов игрока видно, что в данной системе игрок может взаимодействовать с журналом заданий, либо с неигровым персонажем.

При взаимодействии с журналом заданий у игрока есть возможность просмотра списка полученных заданий и получение подробной информации о каждом элементе этого списка.

При взаимодействии с неигровым персонажем у игрока есть возможность просмотра списка принимаемых у NPC заданий, просмотр списка доступных у NPC заданий.

После получения этих списков игрок может получить подробную информацию о задании, а также варианты взаимодействия с заданиями, такие как: сдать задание, принять задание и отменить задание.

2.3 Проектирование создания и редактирования заданий

Функционалом для создания заданий будет являться:

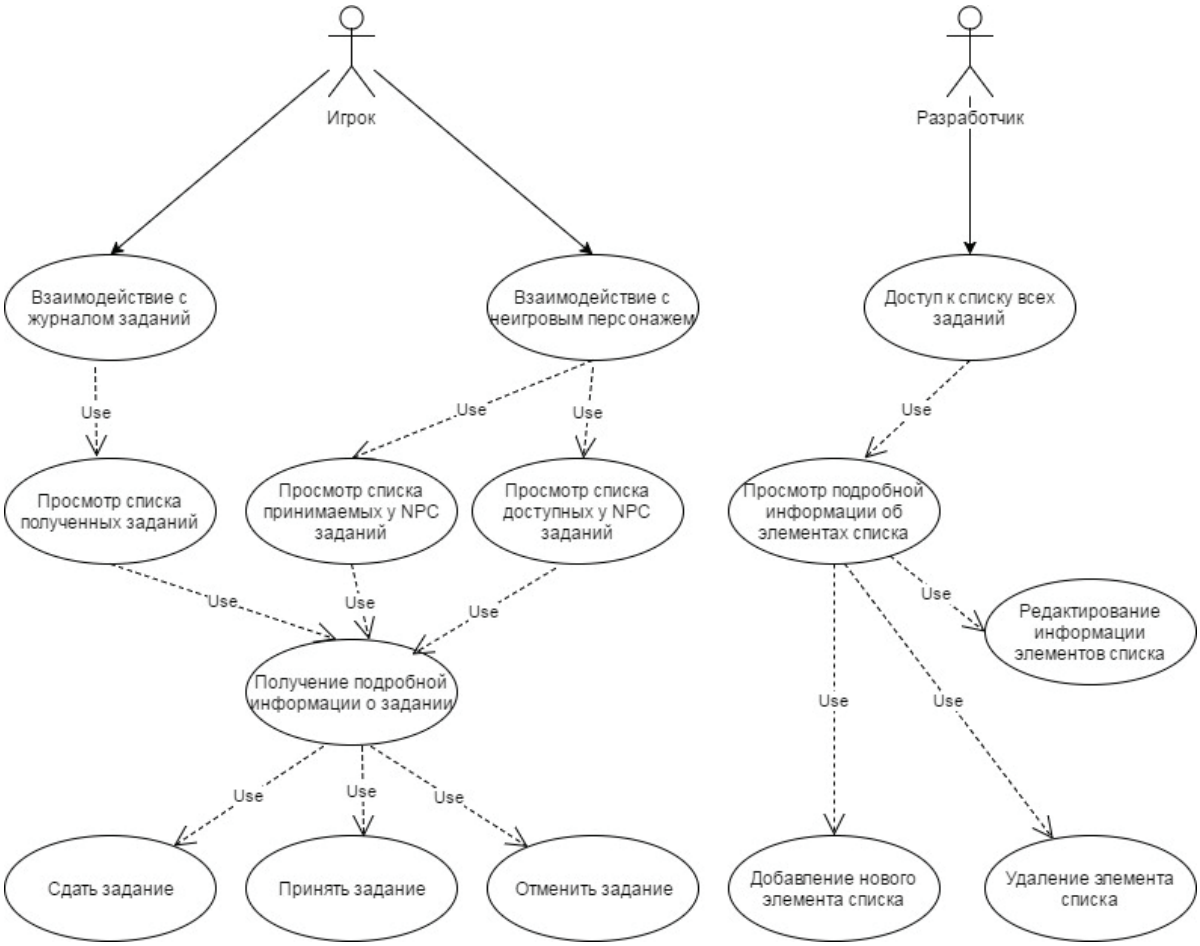
1. Работа с полным списком заданий.
2. Работа с отдельными элементами списка заданий.

Работа с полным списком представляет собой просмотр всего списка заданий и добавление в него новых элементов.

Работа с отдельными элементами списка - редактирование информации каждого элемента списка.

Как и в первом случае, для создания диаграмм, основанных на списке требований, будет разработана диаграмма прецедентов. Диаграмма прецедентов создания и редактирования заданий (Actor - Разработчик) представлена на рисунке 3.

У разработчика, согласно диаграмме, должен быть доступ к полному списку всех заданий.


Игрок
Разработчик
Взаимодействие с журналом заданий
Взаимодействие с неигровым персонажем
Доступ к списку всех заданий
Use
Просмотр списка полученных заданий
Просмотр списка принимаемых у NPC заданий
Просмотр списка доступных у NPC заданий
Просмотр подробной информации об элементах списка
Редактирование информации элементов списка
Получение подробной информации о задании
Сдать задание
Принять задание
Отменить задание
Добавление нового элемента списка
Удаление элемента списка

конструкциями. Хотя Lua не содержит понятия класса и объекта в явном виде, механизмы объектно-ориентированного программирования, включая множественное наследование, легко реализуются с использованием метатаблиц, которые также отвечают за перегрузку операций и т. п.

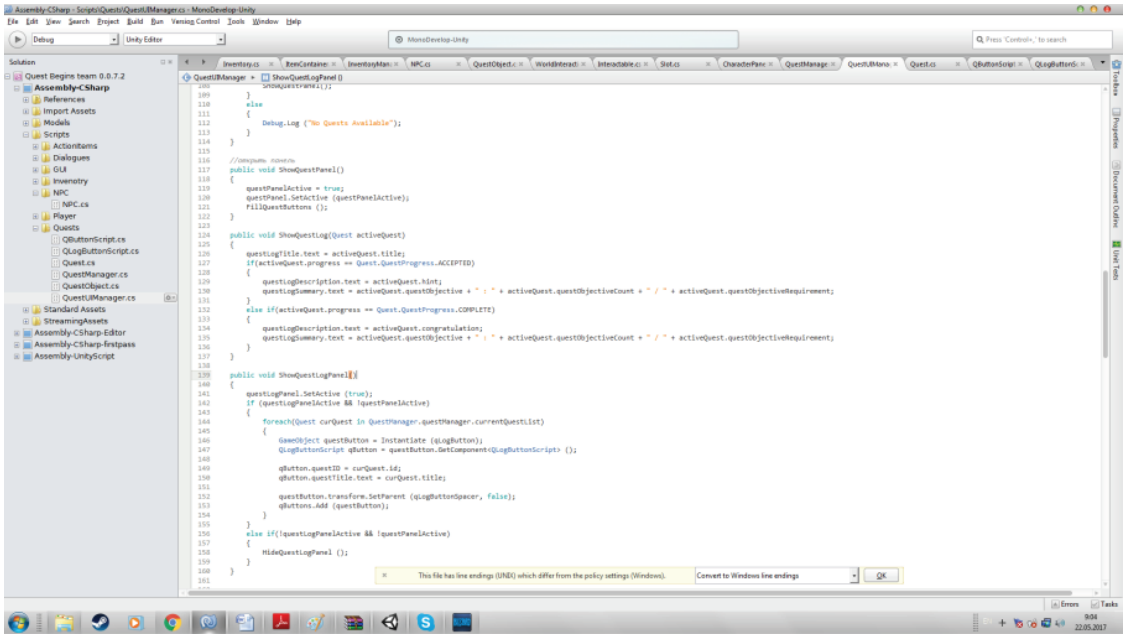
Несмотря на сильный функционал Lua[8], последние версии Unity не ведут поддержку его обновлений, в связи с чем могут возникать ошибки, которые не опытному разработчику решить будет сложно, поэтому данный язык не является первым в приоритете.

Язык программирования C#[3] – это высокоуровневый язык программирования, разработанный и до сих пор активно поддерживающий компанией Microsoft. C# на сегодняшний день является активно используемым в многочисленных сферах программной индустрии. На нем можно реализовать любые программные продукты для разных платформ, таких как Android, IOS и Линукс.

Данный язык программирования поддерживается не только компанией Microsoft, а также и различными сообществами, которые свидетельствуют о том, что C# на данный момент времени является одним из самых популярных языком программирования.

JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам.

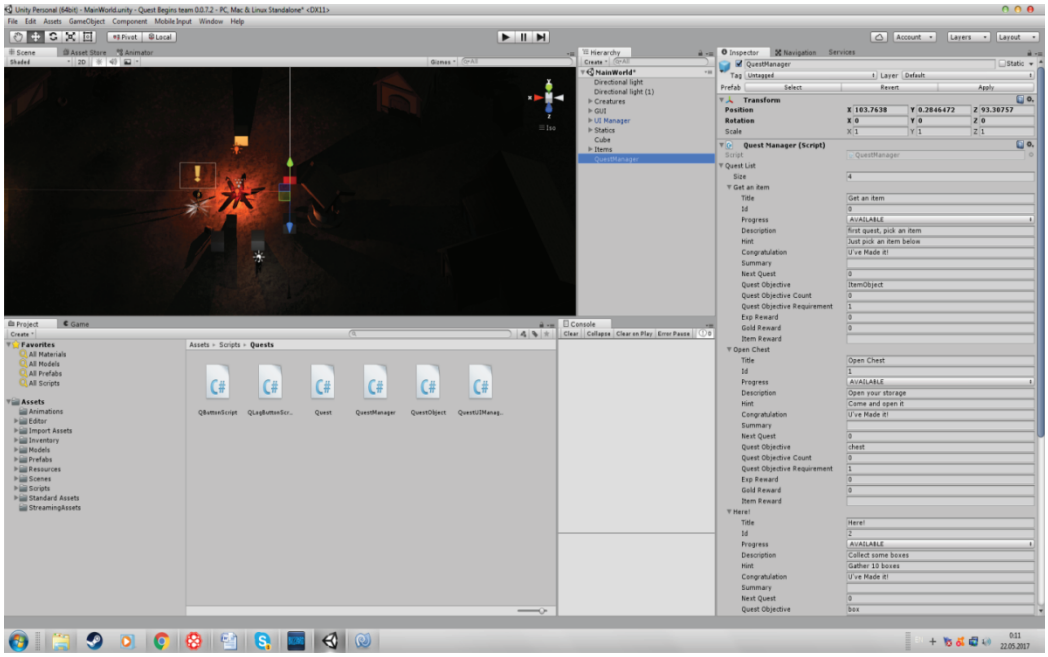
JavaScript является объектно-ориентированным языком[2], но используемое в языке прототипирование обуславливает отличия в работе с объектами по сравнению с традиционными класс-ориентированными языками. Кроме того, JavaScript имеет ряд свойств, присущих функциональным языкам — функции как объекты первого класса, объекты как списки, карринг, анонимные функции, замыкания — что придаёт языку дополнительную гибкость.


Assembly-CSharp - Scripts\Quest\QuestUIManager.cs - MonoDevelop-Unity
File Edit View Search Project Build Run Version Control Tools Window Help
Debug
Unity Editor
MonoDevelop-Unity
Solution
Quest Begins team 0.0.7.2
Assembly-CSharp
References
Import Assets
Models
Scripts
Actionitems
Dialogues
GUI
Invenotry
NPC
NPC.cs
Player
Quests
QButtonScript.cs
QLogButtonScript.cs
Quest.cs
QuestManager.cs
QuestObject.cs
QuestUIManager.cs
Standard Assets
StreamingAssets
Assembly-CSharp-Editor
Assembly-CSharp-firstpass
Assembly-UnityScript
QuestUIManager ▸ ShowQuestLogPanel ()
public void ShowQuestPanel()
public void ShowQuestLog(Quest activeQuest)
public void ShowQuestLogPanel()
This file has line endings (UNIX) which differ from the policy settings (Windows).
Convert to Windows line endings
OK
Errors
Tasks

2.5 Вывод по второй главе

Во второй главе было сформировано точное представление о том, что должна включать в себя система ветвлений игры. Был подробно описан интерфейс для разработки заданий, а так же пользовательский игровой интерфейс.

Были рассмотрены варианты языков программирования для написания программы, и среди них был выбран вариант, наиболее удовлетворяющий требованиям разработки, а так же выбрана среда программирования, максимально подходящая для разработки в среде Unity3d на языке C#.



В поле Size указывается общее количество заданий в игре, после указания значения Объект организует нужное количество элементов списка для заполнения всей необходимой информации о квесте.

Поле Title- название квеста, которое в дальнейшем будет отображаться на кнопках, которые необходимо нажать для получения информации о задании у неигрового персонажа или же в собственном журнале заданий, так же будет являться названием элемента списка.

Поле Id - уникальный идентификатор квеста, необходимый для осуществления поиска заданий в списках, очень важно следить за тем, чтобы у каждого задания был свой уникальный идентификатор, обратное может привести к непредвиденным ошибкам.

Поле Progress - состояние задания (задание можно получить, задание нельзя получить, задание получено, условия задания выполнены, задание завершено), меняется по мере выполнения задания.

Поле Description - описание задания, в которых предъявляются общие требования для его выполнения.

Поле Hint - более краткое и структурированное описание задания, в котором должен быть ясно изложен порядок действий, на случай если игрок не смог разобраться по Description.

Поле Congratulation - сообщение, выдаваемое по завершению задания.

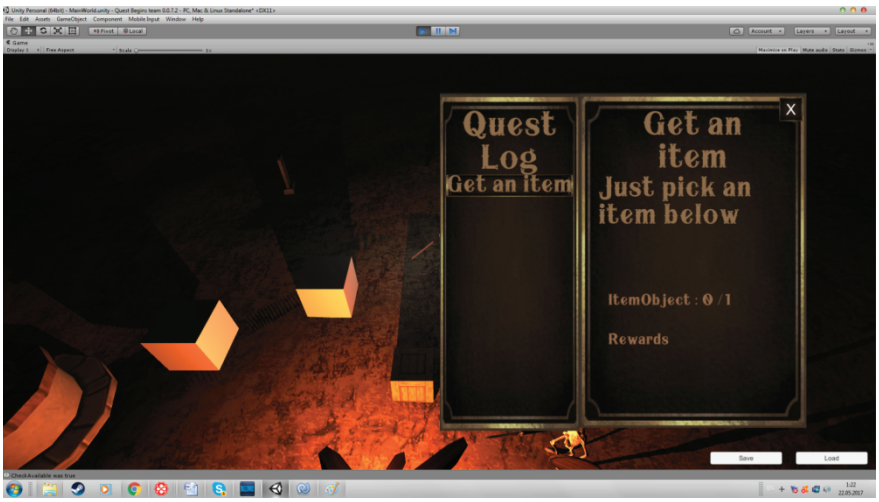
Поле Summary - фраза, произносимая тем, у кого задание будет выполнено.

Поле NextQuest - ссылка на уникальный идентификатор задания, которое можно будет получить по завершению квеста (так же может быть добавлено несколько последующих заданий через скрипт).

Поле QuestObjective – объект, необходимый для выполнения квеста. Представляет собой String переменную, в которую цель задания заносит необходимое значение с указанием количества.

Поле QuestObjectiveCount - количество выполненых пунктов конкретного задания, показывает количество объектов поля QuestObjective.

Available Quests
Open Chest
Talk to another NPC
Receivable Quests
Get an item
Open Chest
Open your storage
chest : 0 /1
Rewards
Accept
X
Save
Load


Quest Log
Get an item
Get an item
Just pick an item below
ItemObject : 0 / 1
Rewards
X
Save
Load

Поле QuestLog - содержит кнопки с уникальными идентификаторами и названиями заданий, которые игрок получил, но еще не выполнил, таким образом, он может отследить, какие задания у него имеются и что необходимо для их выполнения.

Поле с названием задания (правое верхнее) - содержит информацию о задании, аналогично той, что можно получить при разговоре с неигровым персонажем, за исключением отсутствия кнопок получения задания, принятия задания и отказа от задания.

3.1.4 Описание принципа реализации добавления предметов необходимых для задания

Добавления предметов для задания осуществляется по средствам записи в скрипты игровых объектов записи:

QuestManager.questManager.AddQuestItem (questObjective, itemAmount);

questObjective - название предмета необходимого для задания, которое можно узнать в игровом объекте QuestManager.

itemAmount - количество предметов которые метод добавит в поле QuestObjectiveCount из QuestManager.

3.2 Описание ключевых классов

Данный проект реализован на языке программирования C#. Он является объектно-ориентированным языком программирования и содержит инструментарий, позволяющий создавать и использовать различные классы для создания качественного кода, что и было использовано для написания прототипа игры.

3.2.1 ClassQuest.cs

Данный класс является базовым классом-контейнером отдельных неуникальных экземпляров квеста, который хранит всю информацию о каждом задании и не содержит никаких функций. Листинг программы представлен в приложении А.

3.2.2 QuestManager.cs

Данный класс является классом объекта QuestManager и необходим для хранения всего списка заданий, списка полученных заданий и совершения различного рода с этими списками. Содержит функции для работы со списками заданий, такие как «Awake», «QuestRequest», «AcceptQuest», «GiveUpQuest», «CompleteQuest», «CheckChainQuest», «AddQuestItem», «RequestAvailableQuest», «RequestAcceptedQuest», «RequestCompleteQuest», «CheckAvailableQuests», «CheckAcceptedQuests», «CheckCompletedQuests», «ShowQuestLog» . Листинг программы представлен в приложении Б.

Функция «Awake» вызывается один раз, как только игра будет запущена, вызов функции происходит перед вызовом функции Start и первым вызовом Update. Служит для того, чтобы при запуске игры объект QuestManager был уникальным экземпляром (паттерн одиночка), а также не уничтожался при запуске игры.

Функция «QuestRequest» служит для получения списка доступных и активных заданий у конкретного неигрового персонажа.

Функция «AcceptQuest» служит для того чтобы изменить состояние конкретного задания на принятое, в случае если оно находится в состоянии доступное.

Функция «GiveUpQuest» служит для того чтобы изменить состояние конкретного задания на доступное, в случае если оно находится в состоянии принятное.

Функция «CompleteQuest» служит для того чтобы изменить состояние конкретного задания на завершенное, если оно находится в состоянии принятое и все условия выполнения соблюдены.

Функция «CheckChainQuest» служит для того чтобы сделать следующий квест в цепочке доступным, по завершению задания.

Функция «AddQuestItem» служит для добавления предмета необходимого для задания в список предметов того задания, которому он требуется.

Функция «RequestAvailableQuest» проверяет, есть ли задание с конкретным id в списке доступных заданий, если да - возвращает true.

Функция «RequestAcceptedQuest» проверяет, есть ли задание с конкретным id в списке принятых заданий, если да - возвращает true.

Функция «RequestCompleteQuest» проверяет, есть ли задание с конкретным id в списке завершенных заданий, если да - возвращает true.

Функция «CheckAvailableQuests» проверяет, есть ли у конкретного NPC в списке доступные задания, если да - возвращает true.

Фукнция «CheckAcceptedQuests» проверяет, есть ли у конкретного NPC в списке сдаваемые задания, если да - возвращает true.

Функция «CheckCompletedQuests», проверяет, есть ли у конкретного NPC в списке завершенные задания, если да - возвращает true.

Функция «ShowQuestLog» передает в функцию класса QuestUIManager список текущих заданий.

3.2.3QuestUImanager.cs

Класс QuestUIManager осуществляет визуальную работу с заданиями в игре, такую как отображение панелей и добавление кнопок. Содержит функции «Start», «Awake», «Update», «CheckQuests», «ShowQuestPanel», «ShowQuestLog», «ShowQuestLogPanel», «HideQuestPanel», «HideQuestLogPanel», «FillQuestButtons», «ShowSelectedQuest». Листинг программы представлен в приложении В.

Функция «Awake» вызывается один раз, как только игра будет запущена, вызов функции происходит перед вызовом функции Start и первым вызовом Update. Так же служит для реализации паттерна одиночки.

Функция «Start» привязывает кнопки Accept, GiveUp, Complete к панели диалога и после этого отключает её.

Функция «Update» содержит ожидание прерывания с клавиатуры клавиши Q, после чего открывает/закрывает журнал заданий.

Функция «CheckQuests» проверяет, есть ли доступные или сдаваемые квесты у конкретного NPC, если есть - отображает панель заданий.

Функция «ShowQuestPanel» - непосредственное отображение панели квестов.

Функция «ShowQuestLog» - отображает информацию о задании в журнале заданий

Функция «ShowQuestLogPanel» - отображает панель журнала задач и формирует списки заданий.

Функция «HideQuestPanel» - закрывает панель заданий после разговора с NPC.

Функция «HideQuestLogPanel» - скрывает журнал задач.

Функция «FillQuestButtons» - заполняет кнопки заданий на панели квестов или в журнале заданий.

Функция «ShowSelectedQuest» - заполняет панель отображения задания при нажатии на соответствующую этому заданию кнопку в списке и заполняет всю информацию о задании.

3.2.4 ClassQLogButtonScript.cs

Данный класс является компонентом кнопки задания в списках журнала заданий и панели диалогов с NPC. Хранит id и название конкретного доступного для обработки по кнопке квеста. Содержит функции «ShowAllInfos», «ClosePanel». Листинг программы представлен в приложении Г.

Функция «ShowAllInfos» - вызывает функцию ShowQuestLog и передает в неё идентификатор квеста, привязанный к этой кнопке.

Функция «ClosePanel» - Скрывает панель журнала задач, необходима для создания события OnClickEvent.

3.2.5 ClassQButtonScript.cs

Данный класс является компонентом всех кнопок в панели у неигрового персонажа и отвечает за отображение кнопок Accept, GiveUp, Complete и корректного отображения маркеров заданий у всех NPC. Содержит функции «ShowAllInfos», «AcceptQuest», «GiveUpQuest», «CompleteQuest». Листинг программы представлен в приложении Д.

Функция «ShowAllInfos» - отображает необходимые кнопки в описании квеста.

Функция «AcceptQuest» - при клике вызывает функцию принятия квеста в QuestManager и обновляет маркеры у всех неигровых персонажей.

Функция «GiveUpQuest» - при клике вызывает функцию отказа от квеста в QuestManager и обновляет маркеры у всех неигровых персонажей.

Функция «CompleteQuest» - при клике вызывает функцию завершенияквеста в QuestManager и обновляет маркеры у всех неигровых персонажей.

3.2.6 ClassNPC.cs

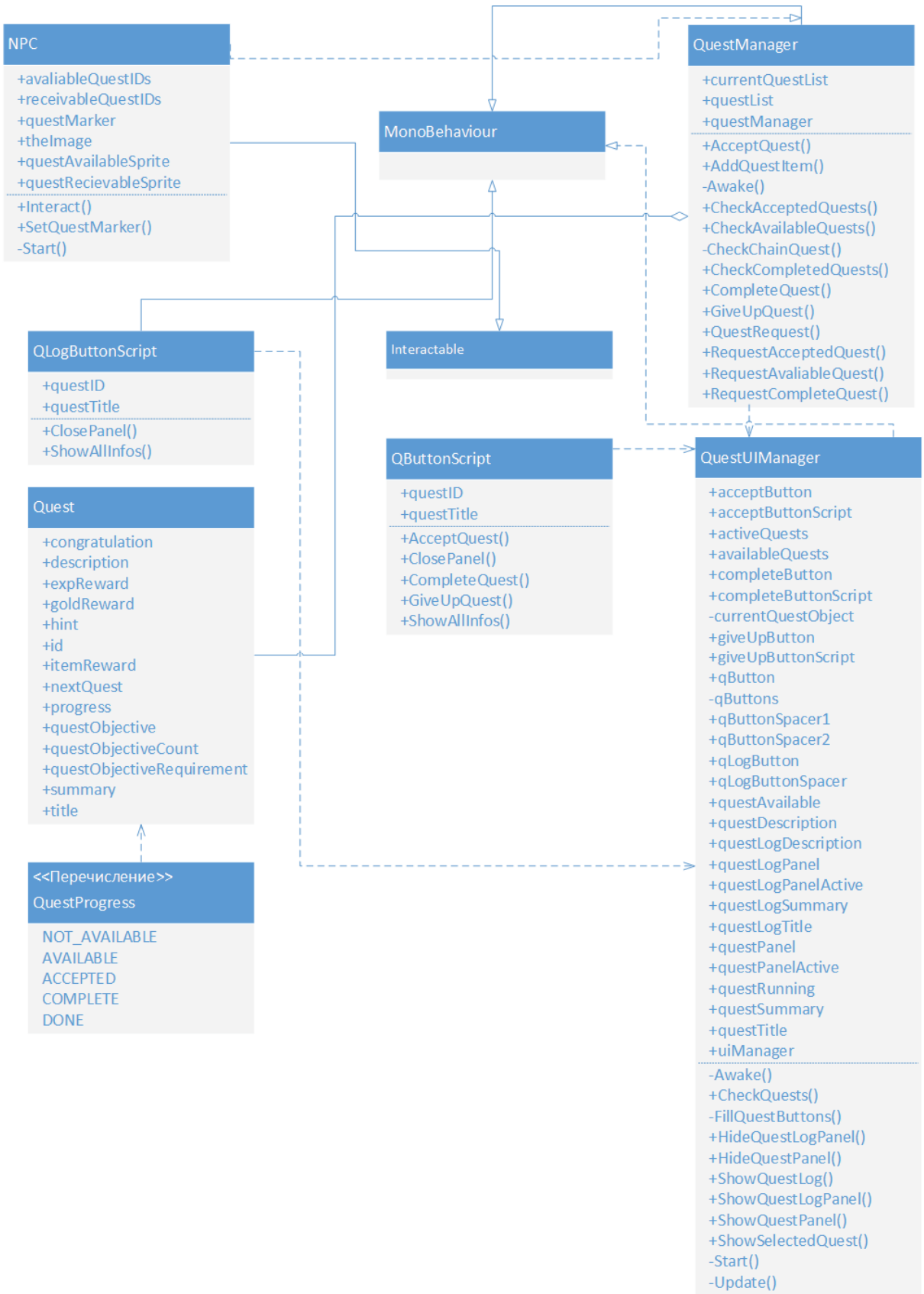
Класс является компонентом всех неигровых персонажей, у которых есть возможность взять или сдать задание. Содержит списки доступных и принимаемых заданий конкретного NPC и обновляет маркеры. Содержит функции «Start», «SetQuestMarker», «Interact». Листинг программы представлен в приложении Е.

Функция «Start» - вызывает SetQuestMarker после начала игры.

Функция «SetQuestMarker» - Проверяет состояние активных заданий, и в зависимости от результата выставляет необходимый маркер.

Функция «Interact» - вызывает метод CheckQuests и передает в него ссылку на NPC, к которому привязан этот компонент.

Для наглядной демонстрации взаимодействия между классами была создана диаграмма классов (Рисунок 8).


NPC
+avaliableQuestIDs
+receivableQuestIDs
+questMarker
+theImage
+questAvailableSprite
+questRecievableSprite
+Interact()
+SetQuestMarker()
-Start()
MonoBehaviour
QuestManager
+currentQuestList
+questList
+questManager
+AcceptQuest()
+AddQuestItem()
-Awake()
+CheckAcceptedQuests()
+CheckAvailableQuests()
-CheckChainQuest()
+CheckCompletedQuests()
+CompleteQuest()
+GiveUpQuest()
+QuestRequest()
+RequestAcceptedQuest()
+RequestAvaliableQuest()
+RequestCompleteQuest()
QLogButtonScript
+questID
+questTitle
+ClosePanel()
+ShowAllInfos()
Interactable
QButtonScript
+questID
+questTitle
+AcceptQuest()
+ClosePanel()
+CompleteQuest()
+GiveUpQuest()
+ShowAllInfos()
QuestUIManager
+acceptButton
+acceptButtonScript
+activeQuests
+availableQuests
+completeButton
+completeButtonScript
-currentQuestObject
+giveUpButton
+giveUpButtonScript
+qButton
-qButtons
+qButtonSpacer1
+qButtonSpacer2
+qLogButton
+qLogButtonSpacer
+questAvailable
+questDescription
+questLogDescription
+questLogPanel
+questLogPanelActive
+questLogSummary
+questLogTitle
+questPanel
+questPanelActive
+questRunning
+questSummary
+questTitle
+uiManager
-Awake()
+CheckQuests()
-FillQuestButtons()
+HideQuestLogPanel()
+HideQuestPanel()
+ShowQuestLog()
+ShowQuestLogPanel()
+ShowQuestPanel()
+ShowSelectedQuest()
-Start()
-Update()
Quest
+congratulation
+description
+expReward
+goldReward
+hint
+id
+itemReward
+nextQuest
+progress
+questObjective
+questObjectiveCount
+questObjectiveRequirement
+summary
+title
<<Перечисление>>
QuestProgress
NOT_AVAILABLE
AVAILABLE
ACCEPTED
COMPLETE
DONE

3.3 Описание алгоритмов

У пользователя имеется 6 способов взаимодействия с программой:

1. Диалог с неигровым персонажем
2. Нажатие кнопки в списке доступных и завершенных заданий
3. Нажатие кнопки Accept
4. Нажатие кнопки GiveUp
5. Нажатие кнопки Complete
6. Нажатие кнопки в списке журнала заданий

3.3.1Диалог с неигровым персонажем

1. При взаимодействии игрока с неигровым персонажем происходит вызов метода Interact. В случае если панель заданий еще не активна - происходит вызов метода CheckQuests, с параметром, содержащим ссылку на игровой объект со скриптом NPC.

2. В методе CheckQuests происходит вызов метода QuestRequestc параметром questObject, который он получил от Interact.

3. Метод QuestRequestпросматривает весь список квестов и сравнивает, есть ли в нем идентификаторы тех же заданий, что имеются в списке доступных идентификаторов у NPC, если такие имеются - добавляет в список доступных заданий.

4. Метод QuestRequest просматривает весь список квестов и сравнивает, есть ли в нем идентификаторы тех же заданий, что имеются в списке активных идентификаторов у NPC, если такие имеются - добавляет в список активных заданий.

5. В случае если список доступных или активных заданий не пуст - происходит вызов метода ShowQuestPanel.

6. В ShowQuestPanel происходит активация панели заданий и вызов метода FillQuestButtons.

7. Метод FillQuestButtons просматривает весь список доступных и активных заданий NPC и создает каждому заданию в списке свою кнопку с именем и id в соответствующем поле.

3.3.2 Нажатие кнопки в списке доступных и завершенных заданий

1. При нажатии кнопки с названием задания - вызывается событие которое активирует метод ShowAllInfos.

2. ShowAllInfos вызывает метод ShowSelectedQuest и активирует одну из кнопок Accept, GiveUp, Complete в зависимости от состояния задания на момент нажатия кнопки.

3. Метод ShowSelectedQuest заполняет поле описания задания соответствующими значениями элемента из списка заданий.

3.3.3 Нажатие кнопки accept

1. При нажатии на кнопку происходит вызов метода AcceptQuest класса QButtonScript.

2. AcceptQuest вызывает метод AcceptQuest класса QuestManager, который проверяет, есть ли данный квест в списке доступных, если да - переводит его в состояние полученного задания.

3. Метод AcceptQuest класса QButtonScript закрывает панель заданий и обновляет маркеры всех NPC.

3.3.4 Нажатие кнопки giveup

1. При нажатии на кнопку происходит вызов метода GiveUpQuest класса QButtonScript.

2. GiveUpQuest вызывает метод GiveUpQuest класса QuestManager, который проверяет, есть ли данный квест в списке полученных, если да - переводит его в состояние доступного задания.

3. Метод GiveUpQuest класса QButtonScript закрывает панель заданий и обновляет маркеры всех NPC.

3.3.5 Нажатие кнопки complete

1. При нажатии на кнопку происходит вызов метода CompleteQuest класса QButtonScript.

2. CompleteQuest вызывает метод CompleteQuest класса QuestManager, который проверяет, есть ли данный квест в списке выполненных, если да - переводит его в состояние выполненного задания.

3. Метод CompleteQuest класса QButtonScript закрывает панель заданий и обновляет маркеры всех NPC.

3.3.6 Нажатие кнопки в списке журнала заданий

1. После нажатия кнопки происходит вызов метода ShowAllInfos класса QLogButtonScript, который в свою очередь вызывает ShowQuestLog класса QuestManager, передавая ему id задания, хранящиеся на этой кнопке.

2. ShowQuestLog проверяет текущий список заданий, и если в нем встречается соответсвующий переданному id, - вызывается метод ShowQuestLog класса QuestUIManager.

3. ShowQuestLog заполняет поле описания задания соответствующими значениями элемента из списка заданий.

3.4 Контрольный пример

В соответствии с ТЗ, была реализована система проектирования алгоритмов ветвлений игры. Для нее были реализованы следующие функции:

1. Взаимодействие с неигровым персонажем и открытие меню диалога.

2. Просмотр списка доступных заданий.

3. Просмотр списка принимаемых заданий.

4. Просмотр подробной информации о задании.

5. Возможность получения задания.

6. Возможность отмены задания.

7. Возможность завершения задания, при выполнении необходимых условий.

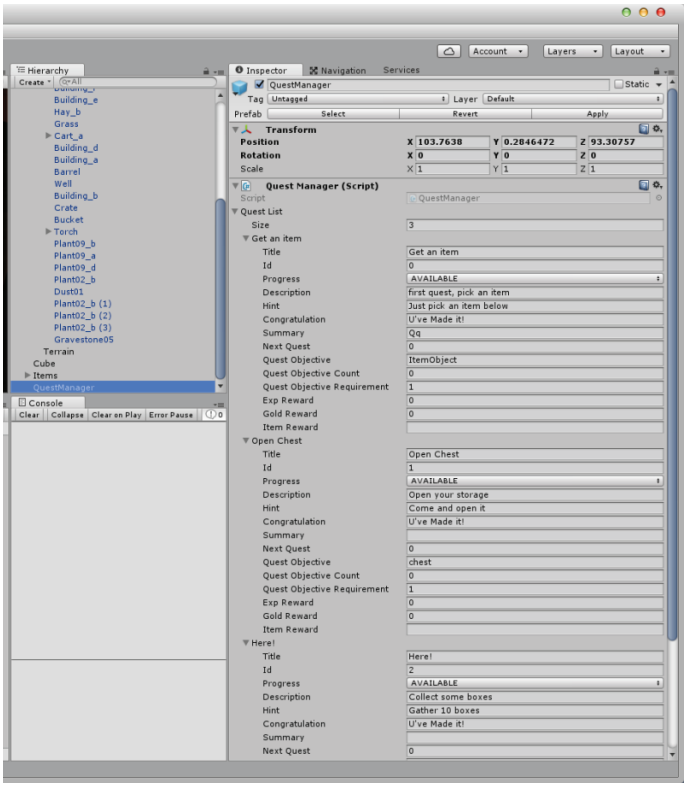
8. Просмотр полученных заданий в собственном журнале заданий.

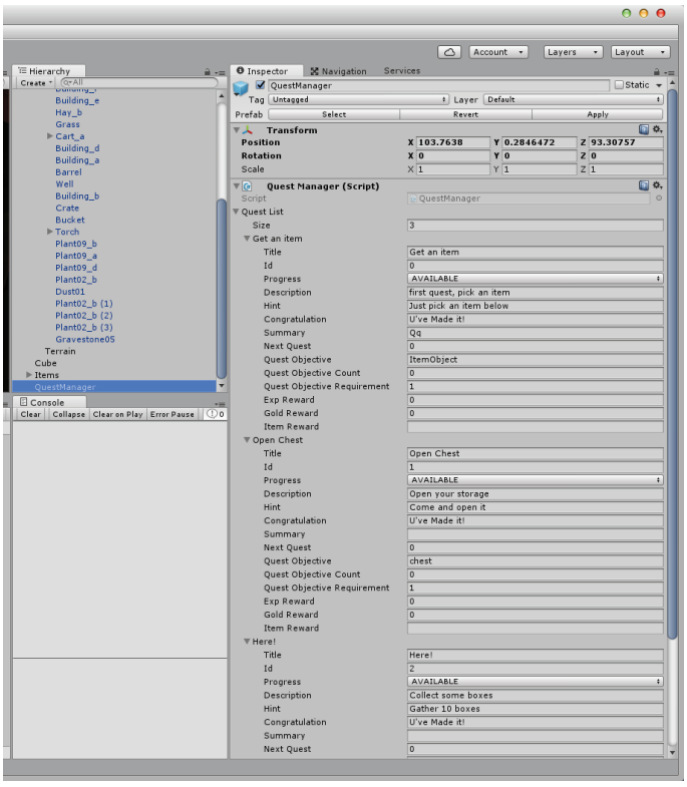
9. Добавление заданий по средствам сцены редактора заданий.

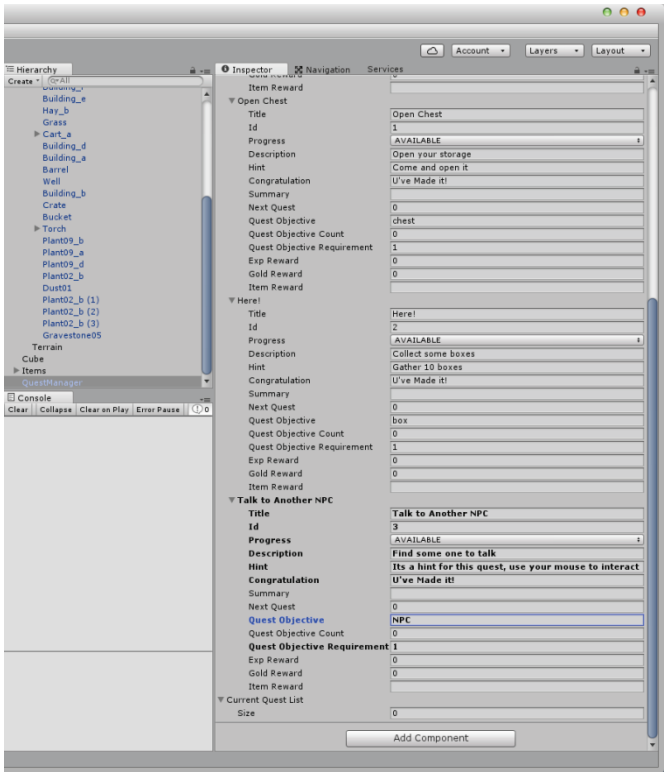
10. Изменение заданий по средствам сцены редактора заданий.

Пример работы со спроектированной и разработанной подсистемой представлен ниже:

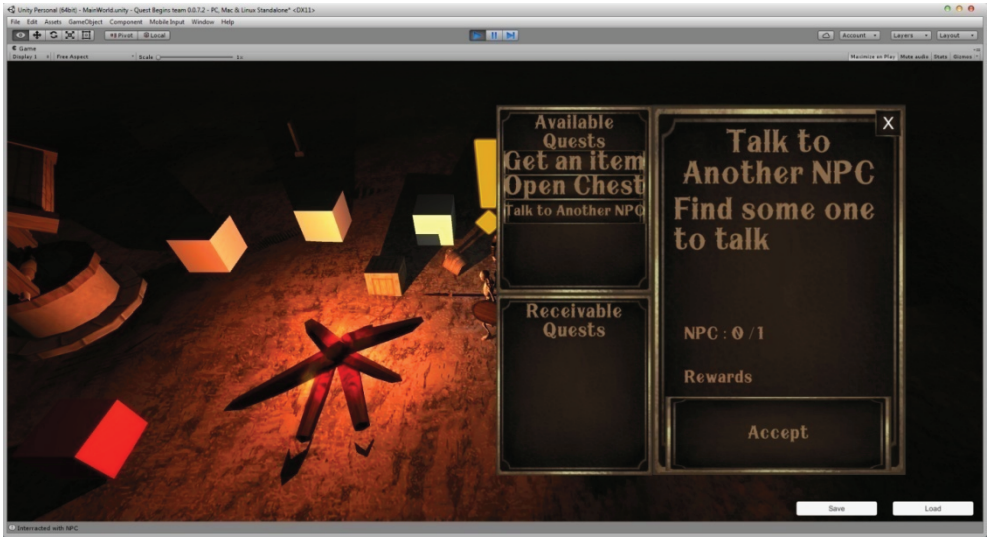
1) Для добавления задания переходим на игровой объект QuestManager (рисунок 9).


Account
Layers
Layout
Hierarchy
Create
Building_e
Hay_b
Grass
Cart_a
Building_d
Building_a
Barrel
Well
Building_b
Crate
Bucket
Torch
Plant09_b
Plant09_a
Plant09_d
Plant02_b
Dust01
Plant02_b (1)
Plant02_b (2)
Plant02_b (3)
Gravestone05
Terrain
Cube
Items
QuestManager
Console
Clear
Collapse
Clear on Play
Error Pause
Inspector
Navigation
Services
QuestManager
Static
Tag
Untagged
Layer
Default
Prefab
Select
Revert
Apply
Transform
Position
X 103.7638
Y 0.2846472
Z 93.30757
Rotation
X 0
Y 0
Z 0
Scale
X 1
Y 1
Z 1
Quest Manager (Script)
Script
QuestManager
Quest List
Size
3
Get an item
Title
Get an item
Id
0
Progress
AVAILABLE
Description
first quest, pick an item
Hint
Just pick an item below
Congratulation
U've Made it!
Summary
Qq
Next Quest
0
Quest Objective
ItemObject
Quest Objective Count
0
Quest Objective Requirement
1
Exp Reward
0
Gold Reward
0
Item Reward
Open Chest
Title
Open Chest
Id
1
Progress
AVAILABLE
Description
Open your storage
Hint
Come and open it
Congratulation
U've Made it!
Summary
Next Quest
0
Quest Objective
chest
Quest Objective Count
0
Quest Objective Requirement
1
Exp Reward
0
Gold Reward
0
Item Reward
Here!
Title
Here!
Id
2
Progress
AVAILABLE
Description
Collect some boxes
Hint
Gather 10 boxes
Congratulation
U've Made it!
Summary
Next Quest
0


Account
Layers
Layout
Hierarchy
Create
Building_e
Hay_b
Grass
Cart_a
Building_d
Building_a
Barrel
Well
Building_b
Crate
Bucket
Torch
Plant09_b
Plant09_a
Plant09_d
Plant02_b
Dust01
Plant02_b (1)
Plant02_b (2)
Plant02_b (3)
Gravestone05
Terrain
Cube
Items
QuestManager
Console
Clear
Collapse
Clear on Play
Error Pause
Inspector
Navigation
Services
QuestManager
Static
Tag
Untagged
Layer
Default
Prefab
Select
Revert
Apply
Transform
Position
X 103.7638
Y 0.2846472
Z 93.30757
Rotation
X 0
Y 0
Z 0
Scale
X 1
Y 1
Z 1
Quest Manager (Script)
Script
QuestManager
Quest List
Size
3
Get an item
Title
Get an item
Id
0
Progress
AVAILABLE
Description
first quest, pick an item
Hint
Just pick an item below
Congratulation
U've Made it!
Summary
Qq
Next Quest
0
Quest Objective
ItemObject
Quest Objective Count
0
Quest Objective Requirement
1
Exp Reward
0
Gold Reward
0
Item Reward
Open Chest
Title
Open Chest
Id
1
Progress
AVAILABLE
Description
Open your storage
Hint
Come and open it
Congratulation
U've Made it!
Summary
Next Quest
0
Quest Objective
chest
Quest Objective Count
0
Quest Objective Requirement
1
Exp Reward
0
Gold Reward
0
Item Reward
Here!
Title
Here!
Id
2
Progress
AVAILABLE
Description
Collect some boxes
Hint
Gather 10 boxes
Congratulation
U've Made it!
Summary
Next Quest
0


Account
Layers
Layout
Hierarchy
Inspector
Navigation
Services
Open Chest
Title
Open Chest
Id
1
Progress
AVAILABLE
Description
Open your storage
Hint
Come and open it
Congratulation
U've Made it!
Summary
Next Quest
0
Quest Objective
chest
Quest Objective Count
0
Quest Objective Requirement
1
Exp Reward
0
Gold Reward
0
Item Reward
Here!
Title
Here!
Id
2
Progress
AVAILABLE
Description
Collect some boxes
Hint
Gather 10 boxes
Congratulation
U've Made it!
Summary
Next Quest
0
Quest Objective
box
Quest Objective Count
0
Quest Objective Requirement
1
Exp Reward
0
Gold Reward
0
Item Reward
Talk to Another NPC
Title
Talk to Another NPC
Id
3
Progress
AVAILABLE
Description
Find some one to talk
Hint
Its a hint for this quest, use your mouse to interact
Congratulation
U've Made it!
Summary
Next Quest
0
Quest Objective
NPC
Quest Objective Count
0
Quest Objective Requirement
1
Exp Reward
0
Gold Reward
0
Item Reward
Current Quest List
Size
0
Add Component
Create
Building_e
Hay_b
Grass
Cart_a
Building_d
Building_a
Barrel
Well
Building_b
Crate
Bucket
Torch
Plant09_b
Plant09_a
Plant09_d
Plant02_b
Dust01
Plant02_b (1)
Plant02_b (2)
Plant02_b (3)
Gravestone05
Terrain
Cube
Items
QuestManager
Console
Clear
Collapse
Clear on Play
Error Pause

Available Quests
Get an item
Open Chest
Talk to another NPC
Receivable Quests
Get an item
first quest, pick an item
ItemObject : 0 / 1
Rewards
Accept
X
Save
Load


Available Quests
Get an item
Open Chest
Talk to Another NPC
Receivable Quests
Talk to Another NPC
Find some one to talk
NPC : 0 /1
Rewards
Accept
X
Save
Load

Quest Log
Get an item
Get an item
Just pick an item below
ItemObject : 0 /1
Rewards
X
Save
Load

Available
Quests
Open Chest
Talk to another NPC
Receivable
Quests
Get an item
Get an
item
U've Made it!
ItemObject : 1 / 1
Rewards
Complete
X
Save
Load

Available
Quests
Open Chest
Talk to another NPC
Receivable
Quests
Rewards
Complete
X
Save
Load

3.5 Вывод по третьей главе

В третьей главе были описаны принципы взаимодействия игрока с интерфейсом игровых заданий, принципы создания заданий во время разработки игры, а так же их удаление, редактирование полей списка заданий.

Так же был разработан программный код, осуществляющий игровую логику по созданию алгоритмов ветвлений игры, и описаны алгоритмы работы программы.

После разработки функционала был проведен тест игровых заданий, который полностью соответствует ожидаемым результатам.

ЗАКЛЮЧЕНИЕ

При выполнении выпускной квалификационной работы был разработан система, позволяющая проектировать алгоритмы ветвлений игры, которая предлагала пользователю интерфейс для разработки заданий:

1. Добавление новых заданий.
2. Редактирование заданий.
3. Удаление заданий.

Был так же добавлен интерфейс работы с заданиями во время игры:

1. Получение списка доступных и сдаваемых заданий у неигрового персонажа.
2. Получение списка заданий в собственном журнале заданий.
3. Возможность просмотра подробной информации о заданиях.
4. Возможность получать, отменять и завершать задания, в зависимости от их стадии выполнения.

При рассмотрении аналогов было выявлено, что не все из них удовлетворяют определенным требованиям игроков. Была составлена таблица достоинств и недостатков ветвлений игры каждого рассмотренного аналога (Таблица 1). Разработанная система была создана с учетом всех предъявленных качеств.

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

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Буч Г., Рамбо Д., Джекобсон А. UML. Руководство пользователя. Изд. 2-е. М.: ДМК, 2006 – 101 с.
2. Макурина Т.А., Гудков П.А. Объектно-ориентированное программирование. Методические указания к курсовому проектированию. Пенза: 2005.
3. Дж. Рихтер. Программирование на платформе Microsoft .NetСПб.: Международная школа программирования, 2016 – 29 с.
4. Blizzard Entertainment: Diablo II [Электронный ресурс] // http://eu.blizzard.com/ru-ru/: Blizzard Entertainment, 2017. — Режим доступа: http://eu.blizzard.com/ru-ru/games/d2/, свободный — Загл. с экрана. (Дата обращения: 20.05.2017).
5. The Elder Scrolls: Arena [Электронный ресурс] // https://elderscrolls.bethesda.net/: The Elder Scrolls: Official Site, 2016. — Режим доступа: www.neverwinter.net.ru, свободный — Загл. с экрана. (Дата обращения: 20.05.2017).
6. NeverwinterNights (NWN) - Скрижали судьбы [Электронный ресурс] // http://www.neverwinter.net.ru/: NeverwinterNights (NWN) - Скрижали судьбы, 2017. 20февраля. — Режим доступа: http://www.neverwinter.net.ru/, свободный — Загл. с экрана. (Дата обращения: 20.05.2017).
7. Dark Souls вики | Fandom powered by Wikia [Электронный ресурс] // http://www.wikia.com/: Explore | Fandom powered by Wikia, 2017. 27 марта. — Режим доступа: http://ru.darksouls.wikia.com/wiki/Dark_Souls_Wiki, свободный — Загл. с экрана. (Дата обращения: 20.05.2017).
8. Создание игры на ваших глазах — часть 3: Прикручиваем скриптовый язык к Unity (UniLua) [Электронный ресурс] // https://habrahabr.ru/: Хабрахабр, 2014. 6февраля. — Режим доступа: https://habrahabr.ru/post/211576/, свободный — Загл. с экрана. (Дата обращения: 20.05.2017).

9. Как устроены задания популярной игры «TheelderScrolls:Arena» [Электронный ресурс] // https://vc.ru/: vc.ru — новости бизнеса, стартапы, кейсы, 2015. 19августа. — Режим доступа: https://vc.ru/p/arena-quests/, свободный — Загл. с экрана. (Дата обращения: 20.05.2017).

10. Рецензия на игру DarkSouls [Электронный ресурс] // http://www.lki.ru/: ЛКИ| сайт журнала «Лучшие Компьютерные Игры» - главная страница, 2004. 12 декабря. — Режим доступа: www.lki.ru/text.php?id=240, свободный — Загл. с экрана. (Дата обращения: 20.05.2017).

11. Особенности бесплатного кроссплатформенного движка Unity3d [Электронный ресурс] // https://gamesmaker.ru/: Создание игр и разработка игровых движков, 2013. 27 декабря. — Режим доступа: https://gamesmaker.ru/3d-game-engines/unity3d/osobennosti-besplatnogo-krossplatformennogo-dvizhka-unity3d, свободный — Загл. с экрана. (Дата обращения: 20.05.2017).

12. Лучший игровой движок по версии пользователей хабра [Электронный ресурс] // https://habrahabr.ru/: habrahabr.ru— новости бизнеса, стартапы, кейсы, 2015. 6августа. — Режим доступа: https://habrahabr.ru/post/307952/, свободный — Загл. с экрана. (Дата обращения: 20.05.2017).

13. 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), свободный — Загл. с экрана. (Дата обращения: 20.05.2017).

ПРИЛОЖЕНИЕ А

Quest.cs

```
[ System.Serializable]
public class Quest
{

    public enum QuestProgress { NOT_AVAILABLE, AVAILABLE, ACCEPTE
D, COMPLETE, DONE}

    public string title;
    public int id;
    public QuestProgress progress;
    public string description;
    public string hint;
    public string congratulation;
    public string summary;
    public int nextQuest;

    //имена вещей необходимых для квеста
    public string questObjective;
    public int questObjectiveCount;
    public int questObjectiveRequirement;

    public int expReward;
    public int goldReward;
    public string itemReward;
}
```

ПРИЛОЖЕНИЕ Б

QuestManager.cs

```
public class QuestManager : MonoBehaviour {

    public static QuestManager questManager;

    //Хранение всех квестов в игре
    public List<Quest> questList = new List<Quest>();
    //взятые квесты
    public List<Quest> currentQuestList = new List<Quest> ();

    void Awake()
    {
        // проверяем, нет ли других квест менеджеров (аналог син
глтона)
        if (questManager == null)
        {
            questManager = this;
        }
        else if (questManager != this)
        {
            Destroy (gameObject);
        }
        DontDestroyOnLoad (gameObject);
    }


    public void QuestRequest(NPC NPCQuestObject)
    {
        //доступные квесты
        if (NPCQuestObject.avaliableQuestIDs.Count > 0)
        {
            for(int i = 0; i < questList.Count; i++)
            {
                for(int j = 0; j < NPCQuestObject.avaliableQuest
IDs.Count; j++)
                {

                    if(questList[ i].id == NPCQuestObject.avaliab
leQuestIDs[ j]  && questList[ i].progress == Quest.QuestProgress.AV
AILABLE)
                    {

                        //AcceptQuest (NPCQuestObject.avaliableQ
uestIDs [j]);
                        QuestUIManager.uiManager.questAvailable
= true;
                        QuestUIManager.uiManager.availableQuests
```

```
.Add (questList[ i] );
                    }
                }
            }
        }
        // активные квесты
        for(int i = 0; i < currentQuestList.Count; i++)
        {
            for(int j = 0; j < NPCQuestObject.receivableQuestIDs
.Count; j++)
            {
                if(currentQuestList[ i] .id == NPCQuestObject.rece
ivableQuestIDs[ j]  && currentQuestList[ i] .progress == Quest.Quest
Progress.ACCEPTED || currentQuestList[ i] .progress == Quest.Quest
Progress.COMPLETE)
                {
                    Debug.Log ("Quest ID receivable: " + NPCQues
tObject.receivableQuestIDs [ j]  + " is " + currentQuestList [ i] .p
rogress);
                    //CompleteQuest (NPCQuestObject.receivableQu
estIDs [j]);
                    QuestUIManager.uiManager.questRunning = true
;
                    QuestUIManager.uiManager.activeQuests.Add (q
uestList[ i] );
                }
            }
        }
    }


    // принять квест

    public void AcceptQuest(int questID)
    {
        for(int i = 0; i < questList.Count; i++)
        {
            if(questList[ i] .id == questID && questList[ i] .progre
ss == Quest.QuestProgress.AVAILABLE)
            {
                currentQuestList.Add (questList [ i] );
                questList [ i] .progress = Quest.QuestProgress.ACC
EPTED;
            }
        }
    }



    // отказаться от квеста

    public void GiveUpQuest(int questID)
    {
```

```
        for (int i = 0; i < currentQuestList.Count; i++)
        {
            if(currentQuestList[ i] .id == questID && currentQuest
List[ i] .progress == Quest.QuestProgress.ACCEPTED)
            {
                currentQuestList [ i] .progress = Quest.QuestProgr
ess.AVAILABLE;
                currentQuestList [ i] .questObjectiveCount = 0;
                currentQuestList.Remove (currentQuestList [ i] );
            }
        }
    }

    // завершить квест

    public void CompleteQuest(int questID)
    {
        for(int i = 0; i < currentQuestList.Count; i++)
        {
            if (currentQuestList [ i] .id == questID && currentQue
stList [ i] .progress == Quest.QuestProgress.COMPLETE)
            {
                currentQuestList [ i] .progress = Quest.QuestProgr
ess.DONE;
                currentQuestList.Remove (currentQuestList [ i] );
                //добавить награду
            }
        }
        // проверить квесты в цепочке
    }

    // проверяем квест в цепочке

    void CheckChainQuest(int questID)
    {
        int tempID = 0;
        for(int i = 0; i < questList.Count; i++)
        {
            if(questList[ i] .id == questID && questList[ i] .nextQu
est > 0)
            {
                tempID = questList [ i] .nextQuest;
            }
        }
        if (tempID > 0)
        {
            for(int i = 0; i < questList.Count; i++)
            {
                if(questList[ i] .id == tempID && questList[ i] .pro
gress == Quest.QuestProgress.NOT_AVAILABLE)
                {
                    questList [ i] .progress = Quest.QuestProgress
.AVAILABLE;
```

```
                }
            }
        }
    }

    //ADD ITEMS

    public void AddQuestItem(string questObjective, int itemAmou
nt)
    {
        // прочесываем весь список чтобы найти квест, которому н
еобходим данный предмет
        for (int i = 0; i < currentQuestList.Count; i++)
        {
            if (currentQuestList [ i] .questObjective == questObje
ctive && currentQuestList [ i] .progress == Quest.QuestProgress.AC
CEPTED)
            {
                currentQuestList [ i] .questObjectiveCount += item
Amount;
            }
            //проверка на то что квест будет выполнен при достиж
ении нужного количества предметов

            if (currentQuestList [ i] .questObjectiveCount >= curr
entQuestList [ i] .questObjectiveRequirement && currentQuestList [
i] .progress == Quest.QuestProgress.ACCEPTED)
            {
                currentQuestList [ i] .progress = Quest.QuestProgr
ess.COMPLETE;
            }
        }
    }

    // добавить удаление предметов

    // BOOLS

    public bool RequestAvaliableQuest(int questID)
    {

        // проверяем весь список квестов
        for (int i = 0; i < questList.Count; i++)
        {
            // если ID есть в списке всех квестов и его статус д
оступен - возвращаем true
            if(questList[ i] .id == questID && questList[ i] .progre
ss == Quest.QuestProgress.AVAILABLE)
            {
                return true;
            }
        }
        return false;
```

```
    }

    public bool RequestAcceptedQuest(int questID)
    {
        for (int i = 0; i < questList.Count; i++)
        {
            if(questList[ i] .id == questID && questList[ i] .progre
ss == Quest.QuestProgress.ACCEPTED)
            {
                return true;
            }
        }
        return false;
    }

    public bool RequestCompleteQuest(int questID)
    {

        for (int i = 0; i < questList.Count; i++)
        {
            if(questList[ i] .id == questID && questList[ i] .progre
ss == Quest.QuestProgress.COMPLETE)
            {
                return true;
            }
        }
        return false;
    }

    //BOOLS 2

    public bool CheckAvailableQuests(NPC NPCQuestObject)
    {
        for (int i = 0; i < questList.Count; i++)
        {
            for (int j = 0; j < NPCQuestObject.avaliableQuestIDs
.Count; j++)
            {
                if (questList[ i] .id == NPCQuestObject.avaliableQ
uestIDs[ j]  && questList[ i] .progress == Quest.QuestProgress.AVAIL
ABLE)
                {
                    return true;
                }
            }
        }

        return false;

    }

    public bool CheckAcceptedQuests(NPC NPCQuestObject)
    {
```

```
        for (int i = 0; i < questList.Count; i++)
        {
            for (int j = 0; j < NPCQuestObject.receivableQuestID
s.Count; j++)
            {
                if (questList[ i] .id == NPCQuestObject.receivable
QuestIDs[ j]  && questList[ i] .progress == Quest.QuestProgress.ACCE
PTED)
                {
                    return true;
                }
            }
        }

        return false;

    }

    public bool CheckCompletedQuests(NPC NPCQuestObject)
    {
        for (int i = 0; i < questList.Count; i++)
        {
            for (int j = 0; j < NPCQuestObject.receivableQuestID
s.Count; j++)
            {
                if (questList[ i] .id == NPCQuestObject.receivable
QuestIDs[ j]  && questList[ i] .progress == Quest.QuestProgress.COMP
LETE)
                {
                    return true;
                }
            }
        }

        return false;

    }

    public void ShowQuestLog(int questID)
    {
        for(int i = 0; i < currentQuestList.Count; i++)
        {
            if(currentQuestList[ i] .id == questID)
            {
                QuestUIManager.uiManager.ShowQuestLog (currentQu
estList[ i] );
            }
        }
    }
}
```

ПРИЛОЖЕНИЕ В

QuestUIManager.cs

```
public class QuestUIManager : MonoBehaviour {

    public static QuestUIManager uiManager;
    //bools
    public bool questAvailable = false;
    public bool questRunning = false;
    public bool questPanelActive = false;
    public bool questLogPanelActive = false;
    // панели
    public GameObject questPanel;
    public GameObject questLogPanel;
    // ссылка на нашего НПС
    private NPC currentQuestObject;
    // списки квестов
    public List <Quest> availableQuests = new List<Quest> ();
    public List <Quest> activeQuests = new List<Quest> ();

    //кнопочки
    public GameObject qButton;
    public GameObject qLogButton;
    //для манипуляций со всеми кнопками во время открытия лога и
 диалога
    private List<GameObject> qButtons = new List<GameObject> ();

    public GameObject acceptButton;
    public GameObject giveUpButton;
    public GameObject completeButton;

    // спейсеры для кнопок
    //qButton available
    public Transform qButtonSpacer1;
    // qButton running
    public Transform qButtonSpacer2;
    //весь quest Log
    public Transform qLogButtonSpacer;

    //иоформация о квестах
    public Text questTitle;
    public Text questDescription;
    public Text questSummary;

    //иоформация в квестлогах
    public Text questLogTitle;
    public Text questLogDescription;
    public Text questLogSummary;
```

```
    public QButtonScript acceptButtonScript;
    public QButtonScript giveUpButtonScript;
    public QButtonScript completeButtonScript;

    void Start()
    {
        acceptButton = GameObject.Find ("QuestCanvas").transform
.FindChild("QuestPanel").transform.FindChild("QuestDescription")
.transform.FindChild("GameObject").transform.FindChild("AcceptBu
tton").gameObject;
        acceptButtonScript = acceptButton.GetComponent<QButtonSc
ript> ();

        giveUpButton = GameObject.Find ("QuestCanvas").transform
.FindChild("QuestPanel").transform.FindChild("QuestDescription")
.transform.FindChild("GameObject").transform.FindChild("GiveUpBu
tton").gameObject;
        giveUpButtonScript = giveUpButton.GetComponent<QButtonSc
ript> ();

        completeButton = GameObject.Find ("QuestCanvas").transfo
rm.FindChild("QuestPanel").transform.FindChild("QuestDescription
").transform.FindChild("GameObject").transform.FindChild("Comple
teButton").gameObject;
        completeButtonScript = completeButton.GetComponent<QButt
onScript> ();

        acceptButton.SetActive (false);
        giveUpButton.SetActive (false);
        completeButton.SetActive (false);

    }

    void Awake()
    {
        if(uiManager == null)
        {
            uiManager = this;
        }
        else if(uiManager != this)
        {
            Destroy (gameObject);
        }
        DontDestroyOnLoad(gameObject);

        HideQuestPanel ();
    }



    // Update is called once per frame
    void Update ()
    {
```

```
        if(Input.GetKeyDown(KeyCode.Q))
        {
            questLogPanelActive = !questLogPanelActive;
            //открыть квестлог
            ShowQuestLogPanel();
        }
    }

    public void CheckQuests(NPC questObject)
    {
        currentQuestObject = questObject;
        QuestManager.questManager.QuestRequest (questObject);
        if((questRunning || questAvailable) && !questPanelActive
)
        {
            //отобразить панель квестов
            ShowQuestPanel();
        }
        else
        {
            Debug.Log ("No Quests Available");
        }
    }

    //открыть панель
    public void ShowQuestPanel()
    {
        questPanelActive = true;
        questPanel.SetActive (questPanelActive);
        FillQuestButtons ();
    }

    public void ShowQuestLog(Quest activeQuest)
    {
        questLogTitle.text = activeQuest.title;
        if(activeQuest.progress == Quest.QuestProgress.ACCEPTED)
        {
            questLogDescription.text = activeQuest.hint;
            questLogSummary.text = activeQuest.questObjective +
" : " + activeQuest.questObjectiveCount + " / " + activeQuest.qu
estObjectiveRequirement;
        }
        else if(activeQuest.progress == Quest.QuestProgress.COMP
LETE)
        {
            questLogDescription.text = activeQuest.congratulatio
n;
            questLogSummary.text = activeQuest.questObjective +
" : " + activeQuest.questObjectiveCount + " / " + activeQuest.qu
estObjectiveRequirement;
        }
    }
```

```
    public void ShowQuestLogPanel()
    {
        questLogPanel.SetActive (true);
        if (questLogPanelActive && !questPanelActive)
        {
            foreach(Quest curQuest in QuestManager.questManager.
currentQuestList)
            {
                GameObject questButton = Instantiate (qLogButton
);
                QLogButtonScript qButton = questButton.GetCompon
ent<QLogButtonScript> ();

                qButton.questID = curQuest.id;
                qButton.questTitle.text = curQuest.title;

                questButton.transform.SetParent (qLogButtonSpace
r, false);
                qButtons.Add (questButton);
            }
        }
        else if(!questLogPanelActive && !questPanelActive)
        {
            HideQuestLogPanel ();
        }
    }


    public void HideQuestPanel()
    {
        questPanelActive = false;
        questAvailable = false;
        questRunning = false;

        questTitle.text = "";
        questDescription.text = "";
        questSummary.text = "";

        availableQuests.Clear ();
        activeQuests.Clear ();

        for(int i = 0; i < qButtons.Count; i++)
        {
            Destroy (qButtons[i]);
        }

        qButtons.Clear ();
        questPanel.SetActive (questPanelActive);
    }

    //закрыть квест лог
    public void HideQuestLogPanel()
    {
```

```
        questLogPanelActive = false;

        questLogTitle.text = "";
        questLogDescription.text = "";
        questLogSummary.text = "";

        // очищаем список кнопок

        for(int i = 0; i < qButtons.Count; i++)
        {
            Destroy (qButtons[i]);
        }

        qButtons.Clear ();
        questLogPanel.SetActive (false);
    }


    // заполнение кнопок на панели квестов
    void FillQuestButtons()
    {
        foreach(Quest availableQuest in availableQuests)
        {
            GameObject questButton = Instantiate (qButton);
            QButtonScript qBScript = questButton.GetComponent<QB
uttonScript> ();

            qBScript.questID = availableQuest.id;
            qBScript.questTitle.text = availableQuest.title;

            questButton.transform.SetParent (qButtonSpacer1, fal
se);
            qButtons.Add (questButton);
        }
        foreach(Quest activeQuest in activeQuests)
        {
            GameObject questButton = Instantiate (qButton);
            QButtonScript qBScript = questButton.GetComponent<QB
uttonScript> ();

            qBScript.questID = activeQuest.id;
            qBScript.questTitle.text = activeQuest.title;

            questButton.transform.SetParent (qButtonSpacer2, fal
se);
            qButtons.Add (questButton);
        }
    }

    //показать квест при нажатии на соответствующую кнопку
    public void ShowSelectedQuest(int questID)
    {
        for(int i = 0; i < availableQuests.Count; i++)
```

```
        {
            if (availableQuests [ i] .id == questID)
            {
                questTitle.text = availableQuests [ i] .title;
                if(availableQuests[ i] .progress == Quest.QuestPro
gress.AVAILABLE)
                {
                    questDescription.text = availableQuests [ i] .
description;
                    questSummary.text = availableQuests [ i] .ques
tObjective + " : " + availableQuests[ i] .questObjectiveCount + "
/ " + availableQuests[ i] .questObjectiveRequirement;
                }
            }
        }

        for(int i = 0; i < activeQuests.Count; i++)
        {
            if(activeQuests[ i] .id == questID)
            {
                questTitle.text = activeQuests [ i] .title;
                if(activeQuests[ i] .progress == Quest.QuestProgre
ss.ACCEPTED)
                {
                    questDescription.text = activeQuests [ i] .hin
t;
                    questSummary.text = activeQuests [ i] .questOb
jective + " : " + activeQuests [ i] .questObjectiveCount + " / " +
 activeQuests [ i] .questObjectiveRequirement;
                }
                else if (activeQuests[ i] .progress == Quest.Quest
Progress.COMPLETE)
                {
                    questDescription.text = activeQuests [ i] .con
gratulation;
                    questSummary.text = activeQuests [ i] .questOb
jective + " : " + activeQuests [ i] .questObjectiveCount + " / " +
 activeQuests [ i] .questObjectiveRequirement;
                }
            }
        }

    }
}
```

ПРИЛОЖЕНИЕ Г

QLogButtonScript.cs

```
public class QLogButtonScript : MonoBehaviour {

    public int questID;
    public Text questTitle;

    public void ShowAllInfos()
    {
        //QuestUIManager.uiManager.ShowQuestLog (questID);
        QuestManager.questManager.ShowQuestLog (questID);
    }

    public void ClosePanel()
    {
        QuestUIManager.uiManager.HideQuestLogPanel ();
    }
}
```

ПРИЛОЖЕНИЕ Д

QButtonScript.cs

```
public class QButtonScript : MonoBehaviour {

    public int questID;
    public Text questTitle;


    public void ShowAllInfos()
    {
        QuestUIManager.uiManager.ShowSelectedQuest (questID);

        if (QuestManager.questManager.RequestAvaliableQuest (que
stID))
        {
            QuestUIManager.uiManager.acceptButton.SetActive (tru
e);
            QuestUIManager.uiManager.acceptButtonScript.questID
= questID;
        }
        else
        {
            QuestUIManager.uiManager.acceptButton.SetActive (fal
se);
        }
        if (QuestManager.questManager.RequestAcceptedQuest (ques
tID))
        {
            QuestUIManager.uiManager.giveUpButton.SetActive (tru
e);
            QuestUIManager.uiManager.giveUpButtonScript.questID
= questID;
        }
        else
        {
            QuestUIManager.uiManager.giveUpButton.SetActive (fal
se);
        }
        if (QuestManager.questManager.RequestCompleteQuest (ques
tID))
        {
            QuestUIManager.uiManager.completeButton.SetActive (t
rue);
            QuestUIManager.uiManager.completeButtonScript.questI
D = questID;
        }
        else
        {
```

```
            QuestUIManager.uiManager.completeButton.SetActive (f
alse);
        }
    }

    public void AcceptQuest()
    {
        QuestManager.questManager.AcceptQuest (questID);
        QuestUIManager.uiManager.HideQuestPanel ();


        // обновляем маркеры у всех НПС
        NPC[] currentQuestNPCs = FindObjectsOfType (typeof(NPC))
 as NPC[];
        foreach(NPC obj in currentQuestNPCs)
        {
            obj.SetQuestMarker ();
        }

    }

    public void GiveUpQuest()
    {
        QuestManager.questManager.GiveUpQuest (questID);
        QuestUIManager.uiManager.HideQuestPanel ();


        // обновляем маркеры у всех НПС
        NPC[] currentQuestNPCs = FindObjectsOfType (typeof(NPC))
 as NPC[];
        foreach(NPC obj in currentQuestNPCs)
        {
            obj.SetQuestMarker ();
        }

    }

    public void CompleteQuest()
    {
        QuestManager.questManager.CompleteQuest (questID);
        QuestUIManager.uiManager.HideQuestPanel ();


        // обновляем маркеры у всех НПС
        NPC[] currentQuestNPCs = FindObjectsOfType (typeof(NPC))
 as NPC[];
        foreach(NPC obj in currentQuestNPCs)
        {
            obj.SetQuestMarker ();
        }

    }
```

```
    public void ClosePanel()
    {
        QuestUIManager.uiManager.HideQuestPanel ();
        QuestUIManager.uiManager.acceptButton.SetActive (false);
        QuestUIManager.uiManager.giveUpButton.SetActive (false);
        QuestUIManager.uiManager.completeButton.SetActive (false
);
    }
}
```

ПРИЛОЖЕНИЕ Е

NPC.cs

```
public class NPC : Interactable
{
    public List<int> avaliableQuestIDs = new List<int> ();
    public List<int> receivableQuestIDs = new List<int> ();

    public GameObject questMarker;
    public Image theImage;

    public Sprite questAvailableSprite;
    public Sprite questRecievableSprite;

    void Start()
    {
        SetQuestMarker ();
    }



    public void SetQuestMarker()
    {
        if (QuestManager.questManager.CheckCompletedQuests (this
))
        {
            questMarker.SetActive (true);
            theImage.sprite = questRecievableSprite;
            theImage.color = Color.yellow;
            Debug.Log ("CheckComplete was true");
        }
        else if (QuestManager.questManager.CheckAvailableQuests
(this))
        {
            questMarker.SetActive (true);
            theImage.sprite = questAvailableSprite;
            theImage.color = Color.yellow;
            Debug.Log ("CheckAvailable was true");
        }
        else if
            (QuestManager.questManager.CheckAcceptedQuests (this
))
        {
            questMarker.SetActive (true);
            theImage.sprite = questRecievableSprite;
            theImage.color = Color.gray;
            Debug.Log ("CheckAccepted was true");
        }
        else
```

```
        {
            questMarker.SetActive (false);
            Debug.Log ("No quests available or accepted");
        }
    }

    public override void Interact()
    {
        if(!QuestUIManager.uiManager.questPanelActive)
        {
            QuestUIManager.uiManager.CheckQuests (this);
            //QuestManager.questManager.QuestRequest (this);
            Debug.Log ("Interracted with NPC");
        }
    }

    public override void InteractUpdate()
    {
        if (Input.GetKeyDown (KeyCode.V))
        {
            Debug.Log ("Lololo");
            SetQuestMarker ();
        }
    }
}
```