# АННОТАЦИЯ

Данная дипломная работа посвящена разработке алгоритмов взаимодействия объектов в виртуальном пространстве на примере компьютерной игры с применением языков программирования C#, JavaScript и игрового движка Unity3D. В качестве таких объектов рассмотрены модели предметов Гражданской Обороны, выступающие в роли объектов взаимодействия в тестовой среде компьютерной игры.

Назначением разработки алгоритмов взаимодействия является обеспечение гибкости редактирования и возможности масштабирования в рамках различных задач, а так же облегчение понимания при дальнейшей разработке и модификации игрового программного продукта.

Использование алгоритмов предполагается осуществлять на конкретных объектах виртуального пространства в тестовой среде разрабатываемой компьютерной игры. Алгоритмы разрабатываются для объектов, за которыми закреплено определенное событие и с которыми пользователь может осуществлять взаимодействие посредством использования клавиатуры и мыши.

Процесс разработки состоит как из самой разработки алгоритмов, так и настройки тестовой среды, для которой она осуществляется. Настройка тестовой среды предполагает моделирование объектов посредством заявленного программного обеспечения, расстановку смоделированных объектов в виртуальном пространстве и программирования объектов, с помощью сценариев взаимодействия, разработанных на основе алгоритмов, выполненных на языках программирования C# и JavaScript.

В качестве средства для моделирования объектов взаимодействия использована программа графического редактирования 3ds Max 2013. Для разработки тестовой среды использовался игровой движок Unity3D.

Оглавление

[ВВЕДЕНИЕ 7](#_Toc358030501)

[ГЛАВА 1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 10](#_Toc358030502)

[1.1. История становления компьютерных игр, как отдельного направление в области информационных технологий 10](#_Toc358030503)

[1.1.1. 1940-е годы 10](#_Toc358030504)

[1.1.2. 1950-е годы 11](#_Toc358030505)

[1.1.3. 1960-е годы. 11](#_Toc358030506)

[1.1.4. 1970-е годы 12](#_Toc358030507)

[1.1.5. 1980-е годы 13](#_Toc358030508)

[1.1.6. 1990-е годы 14](#_Toc358030509)

[1.1.7. 2000-е годы 15](#_Toc358030510)

[1.1.8. Наши дни 16](#_Toc358030511)

[1.2 Классификация компьютерных игр 17](#_Toc358030512)

[1.2.1. Классификация по жанру 17](#_Toc358030513)

[1.2.2. Классификация по количеству 18](#_Toc358030514)

[1.2.3. Классификация по визуальному представлению 19](#_Toc358030515)

[1.2.4. Классификация по платформам 19](#_Toc358030516)

[1.3. Разработка тренажеров для вооруженных сил 19](#_Toc358030517)

[2. ПОСТАНОВКА ЗАДАЧИ 22](#_Toc358030518)

[ГЛАВА 3. РАЗРАБОТКА АЛГОРИТМОВ ВЗАИМОДЕЙСТВИЯ ОБЪЕКТОВ 24](#_Toc358030519)

[3.1. Метод решения 24](#_Toc358030520)

[3.1.1. Описание принципа взаимодействия Main Object и Interactive Object 24](#_Toc358030521)

[3.1.2. Описание принципа взаимодействия Main Object с триггерами 25](#_Toc358030522)

[3.1.3. Описание классов 25](#_Toc358030523)

[3.2 Описание алгоритмов на РАЯ 26](#_Toc358030524)

[3.2.1. Алгоритмы взаимодействия Main Object и Interactive Object. 26](#_Toc358030525)

[3.2.2. Алгоритмы взаимодействия Main Object и триггеров 29](#_Toc358030526)

[3.3. Использованное программное обеспечение 31](#_Toc358030527)

[3.3.1. Autodesk 3D’s Max 31](#_Toc358030528)

[3.3.2. Unity 3D 32](#_Toc358030529)

[3.3.3. MonoDevelop 33](#_Toc358030530)

[Возможности среды: 34](#_Toc358030531)

[3.4 Выбранный язык программирования 35](#_Toc358030532)

[**3.4.1. Язык программирования C#** 35](#_Toc358030533)

[3.4.2. Язык программирования JavaScript 36](#_Toc358030534)

[3.5. Подготовка тестовой среды. 36](#_Toc358030535)

[ГЛАВА 4. КОНТРОЛЬНЫЙ ПРИМЕР 44](#_Toc358030536)

[4.1. Сценарий взаимодействия Main Object и Interactive Object. Включение или выключение освещения 44](#_Toc358030537)

[4.2. Сценарий взаимодействия Main Object и Interactive Object. Включение или выключение анимации 46](#_Toc358030538)

[4.3. Сценарий взаимодействия Main Object с триггерами. Включение или выключение аудиозаписи 48](#_Toc358030539)

[4.4. Сценарий взаимодействия Main Object с триггерами. Включение или выключение анимация объекта 50](#_Toc358030540)

[4.5. Сценарий взаимодействия Main Object с триггерами. Появляется или пропадает текстового сообщения 52](#_Toc358030541)

[4.6. Сценарий взаимодействия Main Object с триггерами. Комбинация из вышеперечисленных пунктов 53](#_Toc358030542)

[ЗАКЛЮЧЕНИЕ 55](#_Toc358030543)

[СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 57](#_Toc358030544)

[ПРИЛОЖЕНИЕ 58](#_Toc358030545)

[Код основных алгоритмов 58](#_Toc358030546)

# ВВЕДЕНИЕ

Тема дипломной работы посвящена разработке алгоритмов, которые организовывают взаимодействия между объектами, пространством и пользователем в трёхмерном виртуальном пространстве на примере компьютерной игры.

В последнее время все чаще можно услышать об увеличениях масштабов разработки программного обеспечения развлекательного характера. К числу таких продуктов относятся видеоигры. Спустя десятки лет индустрия видеоигр заняла фиксированное место на рынке наряду с другими развлечениями сферы мультимедиа, таких как кино, мультипликация, музыка. Это многомиллиардные компании, занимающиеся разработкой развлекательных продуктов и программных обеспечений, состоящих из миллионов строк программного кода и файлов мультимедиа.

Игры стали охватывать огромную аудиторию по всему миру, появляясь на различных игровых устройствах. Они стали сложнее и масштабнее: улучшилась обработка визуального пространства, обработка физики объектов и искусственный интеллект. Но направляющая методология разработки не изменилась. В виртуальном пространстве может находиться огромное количество разнообразных объектов, но их количество по-прежнему зависит от технических возможностей устройств, на которые рассчитана видеоигра. Интерактивность в такой программе сводится к взаимодействию между объектами. Они могут пересылать информацию друг другу, окружающему пространству и пользователю в том числе.

Основной проблемой является загруженность кода объектов взаимодействия, выполняющих, закрепленный за ними, определенный сценарий.

Цель: Разработать алгоритмы взаимодействия объектов таким образом, чтобы облегчить понимание при дальнейшей разработке и модификации игрового программного продукта.

Тема взаимодействия объектов привлекла мое внимание уже давно и вызвала большой интерес. На первом семестре пятого курса института я ознакомился с дисциплиной «Управление робототехническими системами», в которой рассматривались как сами роботы, так и модель их поведения и взаимодействие с окружающей средой. Так же как и в робототехнических системах в игровой продукции тоже встречается модель поведения и взаимодействие различных объектов друг с другом и с окружающим пространством. Объекты видеоигр обладать оригинальным поведением и набором действий, которыми они порождают события во время игрового процесса. Данная тема была подогрета изучением дисциплины «Языки программирования задач искусственного интеллекта» и моим увлечением видеоиграми в частности.

В работе используются слова, которые могут усложнить понимание материала. Поэтому в данном разделе будут описаны основные термины и определения к ним:

Видеоигра (Компьютерная игра) - компьютерная программа, предназначенная для развлечения или обучения, так же может быть использована для организации связи с партнёрами, или выступающая в качестве партнёра.

Игровой движок (Game Engine) - центральный программный компонент видеоигр, с помощью которого происходит разработка контента.

**Объект (Object)** — объектом является любой элемент в сцене.

Главный объект (Main Object) –объект в сцене, которым манипулирует пользователь на протяжении всего игрового процесса.

Объект взаимодействия (Interactive Object) – объект, у которого есть свой сценарий взаимодействия с пользователем или со сценой.

**Сцена (Scene)** — Сценой в является виртуальное пространство, так же совокупность объектов, материалов, анимации и некоторых настроек самой программы расположенных в этом пространстве.

Скрипт (Script) – Программа, которая содержит сценарий, описывающий определенную последовательность действий при различных обстоятельствах.

Триггер (Trigger) – Невидимый объект взаимодействия в определенном месте виртуального пространства, который используется для осуществления событий.

Взгляд (Raycast) – Невидимый луч, указывающий объект, который в данный момент времени отображается в центре камеры.

Камера (Camera) – Объект показывающий вид отображаемого пространства в сцене.

# ГЛАВА 1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

## 1.1. История становления компьютерных игр, как отдельного направление в области информационных технологий

Видеоигра (так же называется компьютерной игрой, если пользователь использует в качестве основного устройства компьютер) **-** программа, предназначенная для развлечения или обучения, так же может быть использована для организации связи с партнёрами, или имитирующая партнёра.

История видеоигр началась в 50-ых годах прошлого столетия. В те годы были созданы первые устройства, которые потом стали игровыми приставками, портативные игровые консоли и персональные ЭВМ, которые сегодня являются одной из самых старых и проверенных игровых платформ.

Термин «видеоигры» в современных реалиях часто применяется для обобщения всех видов игр, для взаимодействия с которыми используются различные устройства для улучшения эргономичности управления игровым процессом и отображения видеоинформации. К категории можно отнести компьютерные игры, игры для портативных устройств, игры для консолей и игры для мобильных устройств.

Чтобы лучше понять историю видеоигр необходимо рассмотреть данное направление на различных этапах формирования.

### 1.1.1. 1940-е годы

Компьютерные игры начинают свою историю в 1947 году. Тогда была разработана первая игра – ракетный симулятор. Эта разработка вряд ли смогла бы стать массовой – а массовость – это один из важнейших показателей современных игр. Она была создана с помощью катодно-лучевой трубки. Игрок обладал возможностью управлять световым пятном, которое изображало ракету, необходимую для устранения цели. Причем, в те времена катодно-лучевые трубки были очень редки и ограничены в возможностях, поэтому цель была просто нарисована и закреплена на поверхности экрана.

Разработка игровых устройств и программ занимала умы многих людей, работавших в высокотехнологичной сфере, людьми, работавшими с компьютерами, поэтому в 50-х годах можно отметить начало развития этой сферы.

### 1.1.2. 1950-е годы

Развитие компьютеров и средств отображения информации, позволило создать несколько заметных игровых проектов. Как и раньше, они рассматривались как достижения искусства программирования и современной в то время техники, а не как игры, в которые несут развлекательный характер и в которые могут играть все желающие.

В 1952 году Александр Дуглас разработал программу по названием OXO – компьютерный прототип игры в крестики-нолики. Эта программа была создана в процессе его научной деятельности, которая была посвящена человеко-машинному взаимодействию. Игра содержала модуль искусственного интеллекта, который позволял компьютеру взаимодействовать с человеком.

В 1958 году Уильям Хигинботэн создал игру Tennis for Two (Теннис для двоих) – примитивный симулятор игры в теннис. Часто именно эту игру называют первой настоящей видеоигрой. В качестве способа взаимодйствия в игре использовались контроллеры, имевшие джойстик и кнопку. В качестве устройства отображения информации игра использовала осциллограф. Устройство служило для развлечения посетителей Брукхэвенской Национальной Лаборатории в Нью-Йорке.

### 1.1.3. 1960-е годы.

В 60-х годах стали появляться проекты, которые вполне можно было сравнить с современными компьютерными играми. Однако, разрабатывались они, в большинстве случаев, на ЭВМ, которыми обладали учебные заведения.

В 1961 году Стив Рассел и его коллеги из Массачусетского университета в США написали игру Spacewar! (с восклицательным знаком). Она работала на компьютере PDP-1. Целью игры являлась атака космического корабля противника и уклонение от его огня, таким образои, чтобы не столкнуться со звездой.

В 1966 Ральф Баер разработал видеоигру под названием Chase, которая была примечательна тем, что впервые использовала в качестве устройства вывода информации обычный телевизор.

В 1969 Кен Томпсон из компании AT&T разработал игру Space Travel для Операционной Системы MULTICS. В последствии, игра была перенесена и на другие ОС, в результате чего она стала первым программным продуктом для ОС Unix.

### 1.1.4. 1970-е годы

70-е годы можно сравнить с настоящим игровым взрывом. Видеоигры стали развиваться уже в нескольких направлениях. Например, игровые автоматы, игры для компьютеров, установленных в учебных заведениях, игры для домашних компьютеров, а так же – игры для приставок. Особой популярность в то время стали пользоваться консоли от Atari и Magnavox.

Необычно развивалась ситуация среди пользователей домашних компьютеров. Например, в те времена некоторые игры распространялись в виде текстов, напечатанных в журналах или книгах. Таким образом, домашний пользователь ПК должен был самостоятельно набирать текст игры, производить отладку, компилировать программу, чтобы потом в неё поиграть.

В 1971 появился первый игровой автомат, который был запущен в Стэнфордском университете. На этот автомат была установлена игра Galaxy. В том же году Нолэн Басшнелл и Тед Дабни разработали игровой автомат, основанный на той же Spacewar!. Игра называлась Computer Space, и всего было выпущено 1500 экземпляров этого автомата. В том же году Дон Дэглоу разработал первую компьютерную симуляцию бейсбола на компьютере DEC PDP-10. В том же году Майк Мэйфил из Массачусетского технологического университета США создал игру Star Trek.

В 1972 была основана компания Atari, и тогда же вышла игра Atari PONG и соответствующее устройство к нему. Было выпущено 19000 копий этой игры.

В 1974 появились игры Maze War и Spasim – первые попытки разработки многопользовательских трехмерных шутеров. В том же году Брэд Фортнер создал авиасимулятор Airflight. Эта игра стала примером для создания Microsoft Flight Simulator.

В 1975 году была написана текстовая компьютерная игра Adventure. Компьютеры тех времен обладали скромными вычислительными способностями, поэтому текстовые игры обретали популярность. К тому же, существовали игры, которые "общались" с игроками благодаря печати игровой информации на принтере.

В 1979 году, выпущена всемирноизвестная игра Pac Man. Эта игра не потеряла актуальность и в наши дни. Она была занесена в Книгу Рекордов Гиннеса как самая распространенная игра в мире.

### 1.1.5. 1980-е годы

В 80-х годах игровые автоматы, которые обрели популярность в прошлом десятилетии, стали терять свои позиции. Однако начало 80-х можно считать золотой эрой игровых автоматов.

Теперь на первый план выходят игры для ПК, а так же – для игровых консолей. Примерами компьютеров тех времен являются ZX Spectrum, Apple II, Apple Macintosh, Commodore 64, IBM PC.

В это же время начинают появляться многие компании-издатели и компании-разработчики игр, известные и по сей день. Такие, как например - Electronic Arts, которая занимает одно из главных мест на современном рынке видеоигр.

Золотая эра игровых автоматов - 80-е стали причиной зарождения основ многих популярных на сегодня игровых жанров и вариантов построения игрового мира и игровой механики. Например – это трехмерные игры, сетевые игры, квесты, скроллинговые игры. Благодаря созданию новых аппаратных средств, в играх стало появляться неплохое звуковое оформление. В 1985 году была создана игра, которая стала прообразом одноименному портативному устройству. Речь конечно идет о всемирно известном Тетрисе, который написал наш соотечественник Алексей Пажитнов. В том же году вышла в свет Nintendo Entertainment System – игровая консоль, набравшая огромную популярность к началу 1990-х годов. Кстати, сегодня тоже можно поиграть в NES-игры. Nintendo и другие компании выпускают игровые консоли до сих пор.

В 1987 была создана ролевая игра Final Fantasy, которая стала символом японского игростроя. Final Fantasy является одной из популярнейших ролевых игр нашего поколеня.

В 1989 компания Nintendo выпустила карманную игровую консоль Game Boy. С этого момента у компании, которая начала свое существование с магазина игрушек, появились проблемы.

### 1.1.6. 1990-е годы

В 90-х годах индустрия разработки игрого программного обеспечения окрепла. Игры уже разрабатывались не одиночками, как это было ранее, а огромными командами профессиональных разработчиков.

Сегодняшним играм мы обязаны разработкам, которые были сделаны в 1990-е годы. Многие игры тех времен до сих пор популярны. Например – Star Craft 98-ого года выпуска, чемпионаты по которому проводятся и в наше время или Counter Strike 99-ого года выпуска, который является модификацией к игре Half Life, а так же первая часть сериия Alone in the Dark 92-ого года выпуска.

Одна из первых трехмерных игр была Quake, которая была разработана 1996 году и была выпущена благодаря появлению микросхем Voodoo, созданных компанией 3dfx. Этот набор микросхем значительно ускорял работоспособность ПК по часте трехмерной графики.

В 90-е переняли эстафету Интернет-игры. Ранее упомянутая Quake, разрешала организовывать многопользовательские баталии в Интернете, на равнее с такими играми, как StarCraft’а, EverQuest, Ultima Online, Age of Empires. Стали разрабатываться игры на Macromedia Flash. Подобные игры обычно являлись различными реализациями простых, но атмосфеных и увлекательных концепций, которые популярны и сегодня. Главная особенность Flash-игр заключалась в том, что они запускались в окне WEB-браузера, оснащенного соответствующими плагинами от компании Adobe.

### 1.1.7. 2000-е годы

В 2000-х годах игровая индустрия продолжала развиваться. В частности, на рынке игровых консолей присутствовали три сильных конкурента. Это Microsoft Xbox 360 (ее предшественница – Xbox – появилась в 2001 году), Sony Play Station III и Nintendo Wii.

Можно отметить, что игры этих времен (преимущественно – консольные) активно используют нестандартные игровые контроллеры – такие, как контроллер-гитару в игре Guitar Hero для Sony Play Station, контроллер в виде барабанов от Nintendo для некоторых ее игр. Кроме того, в эти времена вполне привычными стали беспроводные игровые манипуляторы. В 2010 году начались продажи Microsoft Kinect – устройства, отслеживающего перемещения, позволяющего превратить в игровой контроллер все тело игрока. В этом же году начали продавать PlayStation Move – игровой контроллер, положение которого отслеживается в трех измерениях. Существуют и другие подобные разработки, вдохновителем которых можно считать беспроводной Wii Remote от Nintendo с детектором движения, который является основным контроллером для Nintendo Wii.

В эти годы можно отметить бурное развитие онлайн-игр. Игровая графика и другие возможности игр шли в ногу со временем, а игры этих лет все еще популярны.

Нужно отметить, что в 2000-х годах разработчики уделяли большое внимание мобильным играм. Причем, среди мобильных платформ можно выделить как узкоспециализированные игровые, так и платформы более широкого профиля – такие, как ОС для мобильных телефонов, смартфонов и КПК.

В частности, если говорить о мобильных игровых платформах – то здесь мы имеем PlayStation Portable. Изначально она была выпущена в 2004-м году, за период 2004-2010 гг. были созданы несколько модификаций, ее развитие продолжается до сих пор.

В 2003-м была анонсирована, а в 2004-м выпущена Nintendo DS, развитие которой так же продолжается и в наши дни. Кроме того, Nintendo продолжает развивать серию Game Boy.

В 2003-м был выпущен мобильный телефон Nokia N-Gage, ориентированный на игровое применение, в 2004-м – Nokia N-Gage QD.

### 1.1.8. Наши дни

В настоящее время игры продолжают развиваться. Особое внимание сегодня уделяется разработке новых игровых интерфейсов (таких, как Microsoft Kinect, Nintendo Wii U) и мобильным играм.

Показательно то, что Kinect для Xbox 360 попал в Книгу рекордов Гиннеса как самое быстро продаваемое потребительское электронное устройство. В первые 60 дней после начала продаж Kinect было продано 8 миллионов экземпляров. Существует немало игр, совместимых с Kinect, но потенциал этого сенсора еще предстоит раскрыть.

В среде игровых консолей так же наблюдается оживление – в 2011 увидели свет Nintendo 3DS и Sony PlayStation Vita.

Традиционные игры – для стационарных игровых консолей и ПК, с привычным управлением, конечно же, не оставляют без внимания. Но на фоне новых технологических решений они смотрятся достаточно спокойно, не привлекают к себе повышенного внимания.

Сегодня разработка компьютерных игр – это гигантская индустрия. Бюджеты игровых проектов достигают десятков миллионов долларов, а объем рынков современных игр – компьютерных, консольных, мобильных, оценивается десятками миллиардов долларов.

## 1.2 Классификация компьютерных игр

Компьютерные игры классифицируются по нескольким критериям:

### 1.2.1. Классификация по жанру

Игра может относиться как к одному или к нескольким жанрам, а в

Редких исключениях — выявлять новое направление стилистики.

Жанр характеризуется целью игры. На данный момент самые актуальные жанры это:

Приключение (Adventure) — игра, обладающая целостным

Сюжетом, который пишут сценаристы из области кинематографа или художественных произведений. В этом случае пользователь в процессе выступает в роли зрителя игры, и сам раскрывает все нити заранее продуманного сюжета.

Боевик (Action) — игра, состоящая в основном из динамических, боевых сцен и перестрелок. Боевик так же может обладать кинематографичным сюжетом в угоду зрелищности.

Ролевая игра (Role Playing Game ) — игра, в которой пользователь управлят пресонажем или группой персонажей, обладающих определенным набором навыков и умений. В процессе игрового времени, персонажи могут получать новые навыки, совершенствовать имеющиеся, за счет выполнения различных заданий.

Стратегия (Strategy) — играя в игры этого жанра пользователь может почуствовать себя в роли главы финансового предприятия, командира отряда т.д. Он может заниматься строительством городов, командованием армией или ведением бизнеса. Игровой процесс может происходить как в режиме реального времени, так в пошаговом режиме.

Симулятор (Simulator) — игра, ,которая полностью или частично имитирует определенную сферу реальной жизни. Например, имитация управления подводной лодкой, гоночным автомобилем, космическим кораблем или самолётом.

Головоломка (Puzzle) — игра, в общей сложности, состоящая из решения головоломок и логических задач.

Образовательная игра — игра, которая обладает элементом обучающей

программы. В такой игре, обучение происходит через игровой процесс и способствует запоминаемости полученной информации.

Аркады — игры, в большинстве своем, предназначены для мобильных платформ с целью убиения времени в общественном транспорте.

### 1.2.2. Классификация по количеству

Игра может быть:

* однопользовательской — рассчитана на одного человека;
* многопользовательской — рассчитанной на игру нескольких человек.

### 1.2.3. Классификация по визуальному представлению

Игры могут использовать графическое отображение или не использовать его вообще (текстовые игры). Также могут быть двухмерными или трехмерными.

### 1.2.4. Классификация по платформам

Игры могут принадлежать одной платформе или быть

мультиплатформенными.

Классификация по платформам:

* ПК;
* Игровые консоли;
* Мобильные устройства.

Также по количеству платформ, на которые портирована игра:

* Мультиплатформенные игры (вышедшие на двух и более платформах).
* Одноплатформенные игры (эксклюзивны в рамках одной платформы).

## 1.3. Разработка тренажеров для вооруженных сил

****

Как это часто бывает, то, что бывает полезно для науки, полезно и для армии. Таким образом, следует заметить, что изначально игры, не были запланированы как средство развлечения, а разрабатывались учёными в университетах, как эксперименты с конечными автоматами и разработкой искусственного интеллекта, так и презентацией возможностей ЭВМ. Но армия заинтересовалась играми уже позднее, когда сама индустрия разработки игрового программного обеспечения обрела популярность и обратилась к вооруженным силам в качестве консультантов для получения информации о тактиках ведения боя и реконструкции военных действий в рамках виртуального пространства. После пары успешных проектов вооруженные силы сделали заказ на разработку тренажеров, основанных для игр, для обучения служащих в рамках виртуального представления.

По заказу Министерства обороны США две организации - Institute for Creative Technologies и Калифорнийский университет - разработали прототип компьютерного симулятора для корпуса Морской пехоты - Marine Doom. Затем американская военная академия "Вест-Поинт" включила в учебную программу тренировки с помощью серии тактических игр Close Combat, "Ближний бой", первая из которых вышла в 1996 г., а последняя, пятая, - в 2000-м. Разработчики из Atomic Games создали вполне убедительный симулятор действий роты сухопутных сил времен Второй мировой войны. Игрок в Close Combat, наблюдая за полем боя с высоты птичьего полета, руководил своим подразделением в рамках поля боя. Однако Close Combat все-таки оставался игрой, весьма далекой от реальности. Особенно ценен опыт игроков в стратегические игры, такие как Gеttysburg, созданный известным Сидом Мейером, автором прославленной игры Civilization. Сценарий Gеttysburg основан на реальных событиях, происходивших во время Гражданской войны в США, когда битва северян и южан при Геттисберге стала крупнейшим сражением. По неофициальным данным, Gеttysburg на некоторое время стал любимой игрой курсантов военных училищ США, проверявших свои теоретические знания и умения в виртуальном мире.

Игры, описывающие управление самолетами и вертолетами, также используются военными. К примеру, Microsoft Flight Simulator используется для обучения курсантов в 65 военных школах, где готовятся летчики ВМФ США. Компьютерные симуляторы широко применяются в Вооруженных Силах США для обучения стрелков, водителей, летчиков, танкистов, механиков, моряков и т.д. К примеру, наработки компании Novalogic, создавшей популярную игру Delta Force, используются для создания боевой системы управления и связи Land Warrior, которой в перспективе будет оснащен каждый американский солдат. Американские солдаты и офицеры играют в особые "стрелялки", "стратегии" и "квесты", чтобы получить новые знания и умения, которые будут использованы в условиях реального боя. Подавляющее большинство сценариев проведения военных операций - от диверсий и управления боевыми машинами до ведения крупномасштабных военных операций в масштабах отдельной страны или целой планеты - уже использованы авторами компьютерных игр. Созданные ими сценарии очень сложны, многовариантны и часто максимально приближены к действительности.

# 2. ПОСТАНОВКА ЗАДАЧИ

Разработка алгоритмов взаимодействия объектов в виртуальном пространстве.

Задача: Реализовать алгоритмы, удовлетворяющие архитектуре главного объекта иерархии Main Object с объектами взаимодействия и триггерами.

**Требования к алгоритмам:**

Алгоритмы взаимодействия должны включать в себя базовую концепцию и обладать:

- Возможностью применения в рамках объектов одного класса;

- Набором состояний с условиями взаимодействия и запуска, а так же самим событием;

- Возможностью редактирования на уровне интерфейса движка Unity3D;

- Возможностью отправлять и обрабатывать сообщения;

- Удобством сопровождения сценария;

- Ограничением программного кода каждого алгоритма в пределах одного допустимого сценария;

**Требование к тестовой среде:**

На основе алгоритмов реализовать различные варианты сценария для объектов взаимодействия и триггеров, с помощью которых, можно строить более сложные модели, на примере компьютерной игры.

Объект Main Object должен обладать следующими условиями взаимодействия и запуска:

- Для скриптов: взгляд (Raycast), нахождение в пределах допустимой дистанции до объекта взаимодействия;

- Для для триггеров: присутствие в самом триггере;

- Условие запуска: клавиши ввода, дополнительные условия, такие как предметы, сообщения отличаются типом;

**Требования к объектам взаимодействия и Main Object:**

Объекты взаимодействия и Main Object должны обладать следующим функционалом:

Функции объектов взаимодействия:

- Предоставление данных, которые можно изменять (посредством передачи сообщений и в режиме отладки);

- Воспроизведение событий, описанных сценарием;

- Каждый объект взаимодействия, должен обладать как минимум одним сценарием.

- Отображение и расположение в пределах видимости Main Object;

- Каждый объект взаимодействия должен быть смоделирован исходя из минимального соответствия с оригиналом.

Функции Main Object:

- Предоставление данных, которые можно изменять (посредством передачи сообщений и в режиме отладки);

- Управление Main Object осуществляется с помощью клавиатуры и мыши;

# ГЛАВА 3. РАЗРАБОТКА АЛГОРИТМОВ ВЗАИМОДЕЙСТВИЯ ОБЪЕКТОВ

## 3.1. Метод решения

Для реализации алгоритмов взаимодействия необходимо обозначить локальные требования для конкретных примеров сценария, удовлетворяющие постановке задачи.

### 3.1.1. Описание принципа взаимодействия Main Object и Interactive Object

После того как тестовая среда прогрузилась, пользователь может увидеть сцену и у него появляется возможность управления Main Object с помощью мыши и клавиатуры. Пользователь может навести перекрестие мыши на объект взаимодействия (**Interactive Object)**. Если пользователь выполнил данное действие и объект являлся активным, то срабатывает система Raycast. Согласно описываемому скрипту Main Object отправляет сообщение объекту взаимодействия «I am Main Object». Объект взаимодействия принимает сообщение от Main Object и проверяет у себя наличие возможных взаимодействий с данным собеседником. При положительном результате, объект взаимодействия включает подсвечивание и возвращает объекту Main Object сообщение «Good and I am <имя объекта>». Main Object принимает сообщение от объекта взаимодействия. Далее пользователь вводит необходимую клавишу или использует кнопки мыши. Тем самым отправляется сообщение объекту взаимодействия о нажатии соответствующей кнопки. Объект взаимодействия проверяет наличие реагирования на нажатую кнопку и на собственное свечение. Если результат положительный, то объект взаимодействие выполняет предписанное ему событие:

* Включается или выключается освещение;
* Включается или выключается анимация объекта;

После выполнения определенного события объект взаимодействия может стать неактивным или продолжить оставаться активным.

### 3.1.2. Описание принципа взаимодействия Main Object с триггерами

После того как сцена прогрузилась у пользователя появляется возможность управления Main Object. Пользователь может переместить Main Object в зону с триггером. В этом случае скрипт триггера при очередной проверке нахождения Main Object в своей зоне возвращает положительный результат. Далее проверяется условие доступности проигрывания события, которое дает результат true. Таким образом, за этим следует возможное событие:

Происходит событие:

* Включается или выключается аудиозапись;
* Включается или выключается анимация объекта;
* Появляется или пропадает текстового сообщения;
* Комбинация из вышеперечисленных пунктов;

После выполнения определенного события триггер может стать неактивным или продолжить оставаться активным. Также триггер может стать неактивным в случае, если first-person-controller покинет зону срабатывания триггера.

### 3.1.3. Описание классов

В тестовой среде должны быть представлены следующие классы объектов для Interactive Object:

table – объект данного типа представляет собой письменный стол, для которого допустимым событием является включение (выключение) анимации открытия или закрытия ящика стола.

light – объект данного типа представляет собой переключатель света, для которого допустимым событием является включение (выключение) освещения в помещении.

В тестовой среде должны быть представлены следующие классы объектов для триггеров:

generator – объект данного типа представляет собой дизельный генератор, для которого допустимым событием является включение (выключение) аудиозаписи работы генератора

door – объект данного типа представляет дверь, для которого допустимым событием является включение (выключение) анимации открытия или закрытия двери.

## 3.2 Описание алгоритмов на РАЯ

### 3.2.1. Алгоритмы взаимодействия Main Object и Interactive Object.

**3.2.1.1. Включение или выключение освещения. Сценарий для Main Object**

Если взгляд Main Object обращен на Interactive Object то

Если Interactive Object относится к классу light то

Interactive Object активный;

Отправить Interactive Object сообшение: «I am Main Object»;

Если Main Object получает ответ «Good and I am light» то

Ожидается нажатие кнопки «F»;

Если кнопка «F» нажата то

Отправить Interactive Object сообшение о нажатии кнопки «F»;

КЕсли

КЕсли

КЕсли

КЕсли

**3.2.1.2. Включение или выключение освещения. Сценарий для Interactive Object**

Если получено сообщение от объекта то

Если объектом является Main Object то

Включить самосвечение;

Отправить Main Object сообщение «Good and I am light»;

Если получено сообщение о нажатии кнопки «F» то

Если значение activity объекта light\_01 false то

Значение activity объекта light\_01 становится true;

Включается освещение;

Иначе

Значение activity объекта light\_01 становится false;

Выключается освещение;

КЕсли

КЕсли

КЕсли

Кесли

**3.2.1.3. Включение или выключение анимации. Сценарий для Main Object**

Если взгляд Main Object обращен на Interactive Object то

Если Interactive Object относится к классу table то

Interactive Object активный;

Отправить Interactive Object сообшение: «I am Main Object»;

Если Main Object получает ответ «Good and I am table» то

Ожидается нажатие кнопки «F»;

Если кнопка «F» нажата то

Отправить Interactive Object сообшение о нажатии кнопки «F»;

КЕсли

КЕсли

КЕсли

КЕсли

**3.2.1.4. Включение или выключение анимации. Сценарий для Interactive Object**

Если получено сообщение от объекта то

Если объектом является Main Object то

Включить самосвечение;

Отправить Main Object сообщение «Good and I am table»;

Если получено сообщение о нажатии кнопки «F» то

Если значение activity объекта table\_01 false то

Значение activity объекта table\_01 становится true;

Включить первую половину анимации объекта;

Иначе

Включить ворую половину анимации объекта;

Значение activity объекта table\_01 становится false;

КЕсли

КЕсли

КЕсли

Кесли

### 3.2.2. Алгоритмы взаимодействия Main Object и триггеров

**3.2.2.1. Включение или выключение аудиозаписи**

Если check = 1 то

Если объект имеет класс MainObject и нажата кнопка «F» то

audio.clip = audiom;

Запуск аудиозапись;

КЕсли

Иначе

Выключение аудиозапись;

КЕсли

**3.2.2.2. Включается или выключается анимация объекта**

Если объект имеет класс MainObject то

Ground.animation.clip = animom;

Запуск анимации;

КЕсли

**3.2.2.3. Появляется или пропадает текстового сообщения**

Если check = 1 то

Если объект имеет класс MainObject то

Если значение activity объекта hello false то

Значение activity объекта hello становится true;

message.text = hello;

Отображается надпись «Welcome to testing environment Downground»;

Иначе

Значение activity объекта table\_01 становится false;

Надпись пропадает;

КЕсли

КЕсли

КЕсли

## 3.3. Использованное программное обеспечение

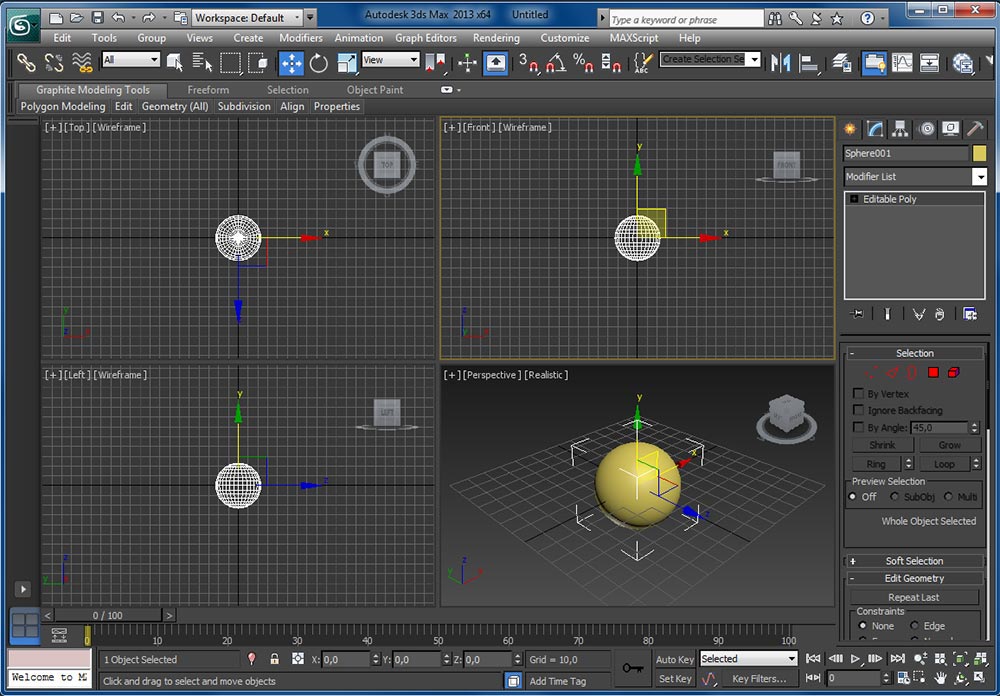
### 3.3.1. Autodesk 3D’s Max



Используемая в качестве графического редактора в дипломной работе программа 3D’s Max является полнофункциональной профессиональной программной системой для моделирования, редактирования и обработки [анимации](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%B0%D0%BD%D0%B8%D0%BC%D0%B0%D1%86%D0%B8%D1%8F). Данный программный пордукт создан компанией [Autodesk](http://ru.wikipedia.org/wiki/Autodesk). Он обладает самыми современными средствами для художников и специалистов в области [мультимедиа](http://ru.wikipedia.org/wiki/%D0%9C%D1%83%D0%BB%D1%8C%D1%82%D0%B8%D0%BC%D0%B5%D0%B4%D0%B8%D0%B0).

3ds Max содержит обширный инструментарий со средствами для моделирования разнообразных по форме и сложности трёхмерных моделей, , основанных на реальных прототипах или фантастических объектов окружающей действительности, с применением множества подходов и механизмов.

*Пример интерфейса Autodesk 3D’s Max*

****

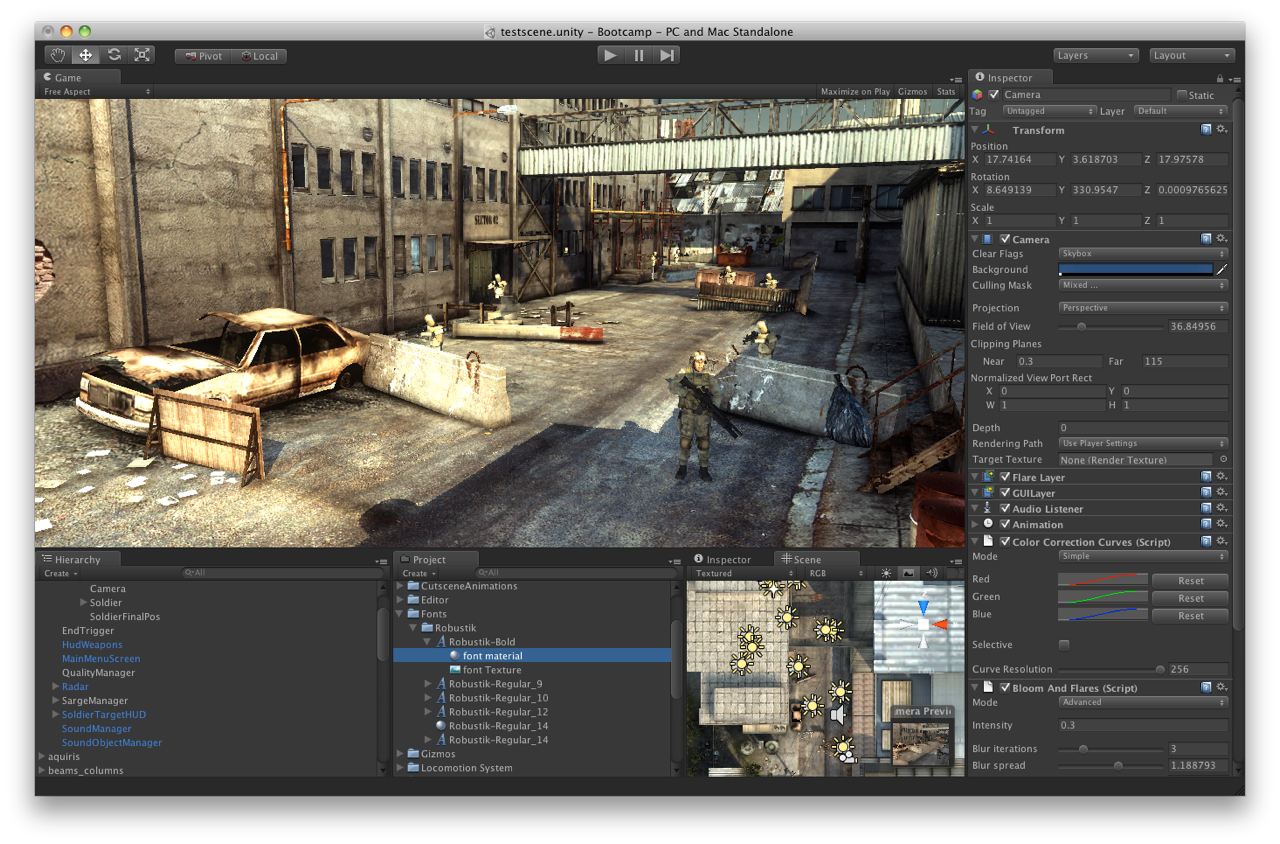
### 3.3.2. Unity 3D



Unity — это игровой движок для разработки [двух-](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D0%BA%D0%B0#.D0.94.D0.B2.D1.83.D1.85.D0.BC.D0.B5.D1.80.D0.BD.D0.B0.D1.8F_.D0.B3.D1.80.D0.B0.D1.84.D0.B8.D0.BA.D0.B0) или [трёхмерных](http://ru.wikipedia.org/wiki/%D0%A2%D1%80%D1%91%D1%85%D0%BC%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D0%BA%D0%B0) приложений и игр, предназначенный для операционных системам [Windows](http://ru.wikipedia.org/wiki/Microsoft_Windows) и [OS X](http://ru.wikipedia.org/wiki/Mac_OS_X). Созданные с помощью Unity игровое программное обеспечение работает на таких операционных системах, как Windows, OS X, [Android](http://ru.wikipedia.org/wiki/Android), [Apple iOS](http://ru.wikipedia.org/wiki/Apple_iOS), [Linux](http://ru.wikipedia.org/wiki/Linux), а также на игровых приставках [Wii](http://ru.wikipedia.org/wiki/Wii), [PlayStation 3](http://ru.wikipedia.org/wiki/PlayStation_3) и [XBox 360](http://ru.wikipedia.org/wiki/XBox_360). Компьютерные игры и другая продукция, созданные с помощью Unity, поддерживают набор библиотек [DirectX](http://ru.wikipedia.org/wiki/DirectX) и [OpenGL](http://ru.wikipedia.org/wiki/OpenGL).

Этот движок позволяет создать сцену, на которой будут размещены импортированные из 3D’s Max объекты взаимодействия. Unity 3D использует такие языки программирования как [C#](http://ru.wikipedia.org/wiki/C_Sharp), [JavaScript](http://ru.wikipedia.org/wiki/JavaScript) и [Boo](http://ru.wikipedia.org/wiki/Boo).

*Пример интерфейса Unity 3D.*



### 3.3.3. MonoDevelop



Изначально, объекты не имеют никаких взаимодействий, поэтому встроенное в используемый движок Unity 3D, приложение MonoDevelop, позволяет их написать.

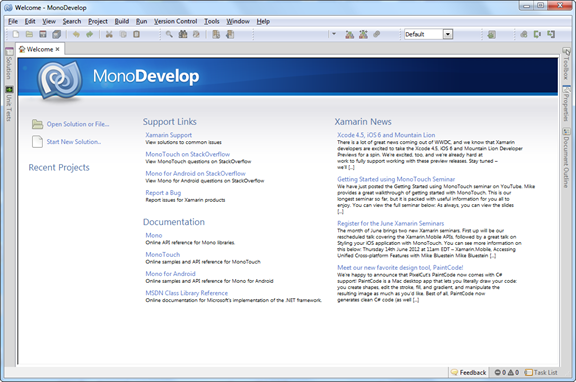
MonoDevelop — является средой разработки, предназначенной для создания приложений на следующих языках: CIL, C#, Nemerle, Java, Boo, Vala, Visual Basic .NET, C и C++.

Возможности среды:

* Встроенный отладчик
* Сворачивание кода
* Автодополнение кода
* Браузер классов
* Подсветка синтаксиса
* Модульное тестирование
* Поддержка плагинов

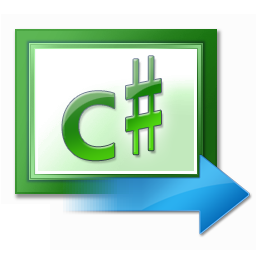
Таким образом предоставлен полный функционал из программного обеспечения для подготовки тестовой среды. На начальном этапе разработки в графической среде 3ds Max создаются модели объектов игрового пространства. Затем, они помещаются в движок Unity3D, где с помощью среды MonoDevelop обеспечивается написание взаимодействие для этих объектов.

*Пример MonoDevelop.*

**

## 3.4 Выбранный язык программирования

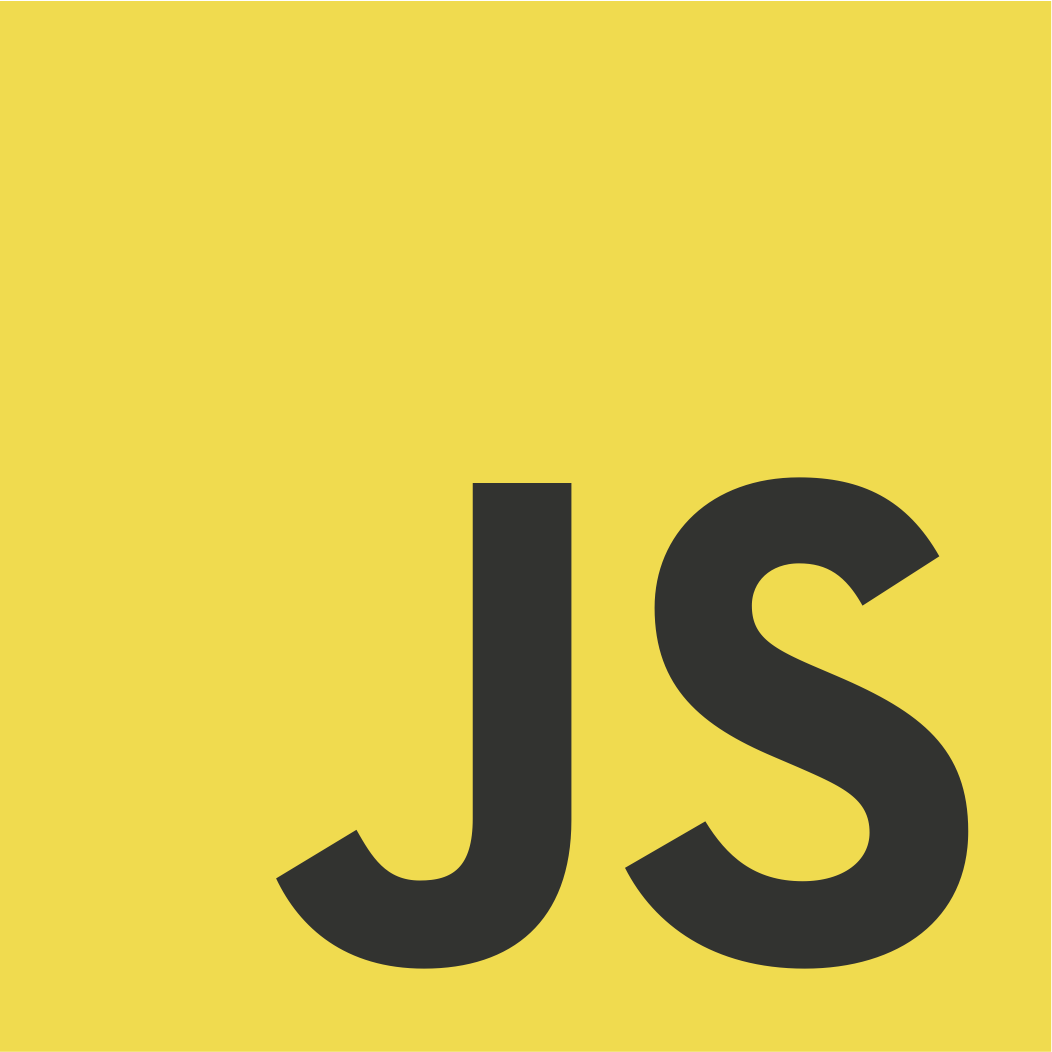
**3.4.1. Язык программирования C#**



В качестве первого ключевого языка программирования в данной дипломной работе выбран С#.

C# относится к языкам с [C-подобным синтаксисом](http://ru.wikipedia.org/w/index.php?title=C-%D0%BF%D0%BE%D0%B4%D0%BE%D0%B1%D0%BD%D1%8B%D0%B9_%D1%81%D0%B8%D0%BD%D1%82%D0%B0%D0%BA%D1%81%D0%B8%D1%81&action=edit&redlink=1), из которых его синтаксис скорее близок к [C++](http://ru.wikipedia.org/wiki/C%2B%2B) и [Java](http://ru.wikipedia.org/wiki/Java). Унаследовав многое от своих предшественников — языков [C++](http://ru.wikipedia.org/wiki/C%2B%2B), [Java](http://ru.wikipedia.org/wiki/Java_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)) и др. — С#, опирается на практическое их использования и исключает некоторые модели, определившие себя как проблематичные при разработке программного продукта. Например, C# в отличие от C++ не поддерживает [множественное наследование](http://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BD%D0%B0%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) классов.

### 3.4.2. Язык программирования JavaScript

****

**JavaScript**— прототипно-ориентированный сценарный язык программирования. JavaScript часто применяется как встраиваемый язык для доступа к объектам приложений. Наиболее широкое применение находит как язык сценариев для отображения взаимодействий в веб-страницах. JavaScript так же был подвержен влиянию многих других языкоа. Во время его разработки была поставлена цель ссоздать язык программирования похожий на Java, но отличающимся в легкости обращения пользователями, далекими от программирования. Языком JavaScript не является собственностью какой-либо организации или компании. Это отличие является достоинством языка, используемого в веб-разработке.

## 3.5. Подготовка тестовой среды.

Для подготовки тестовой среды, необходимо виртуально пространство с объектами взаимодействия. В данной ситуации, таким пространством является тестовый уровень из разрабатываемой компьютерной игры. В этом пространстве присутствуют помещения, которые являются прототипом убежища Гражданской Обороны. Убежище оснащено основными атрибутами Гражданской Обороны. Каждое помещение выполняет определенную функцию и содержит объекты взаимодействия. Для тестирования контрольного примера в пространство были помещены триггеры, скрипты, элементы освещения, Main Object, объекты взаимодействия, аудиозаписи.

План помещений:

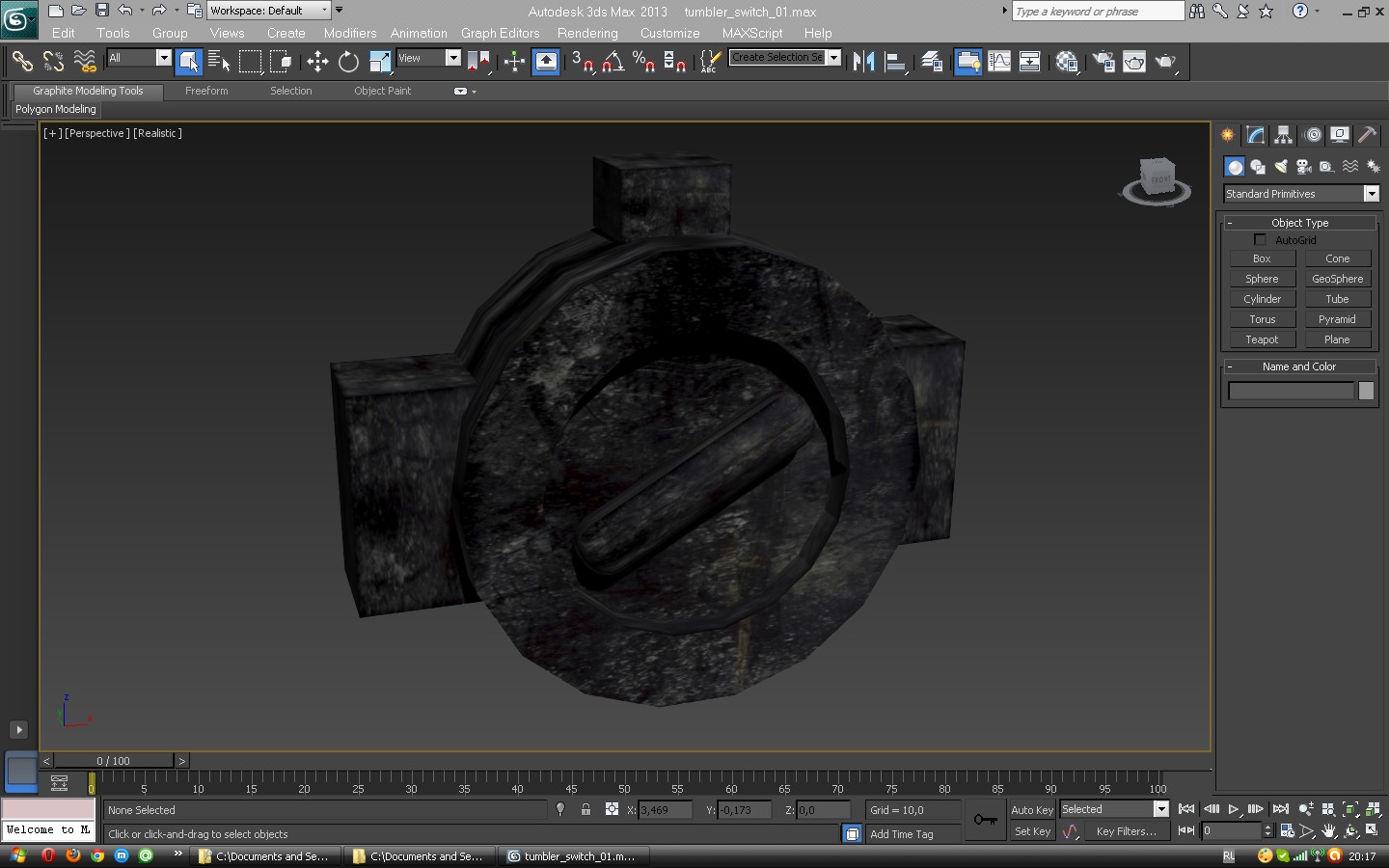


Список помещений:

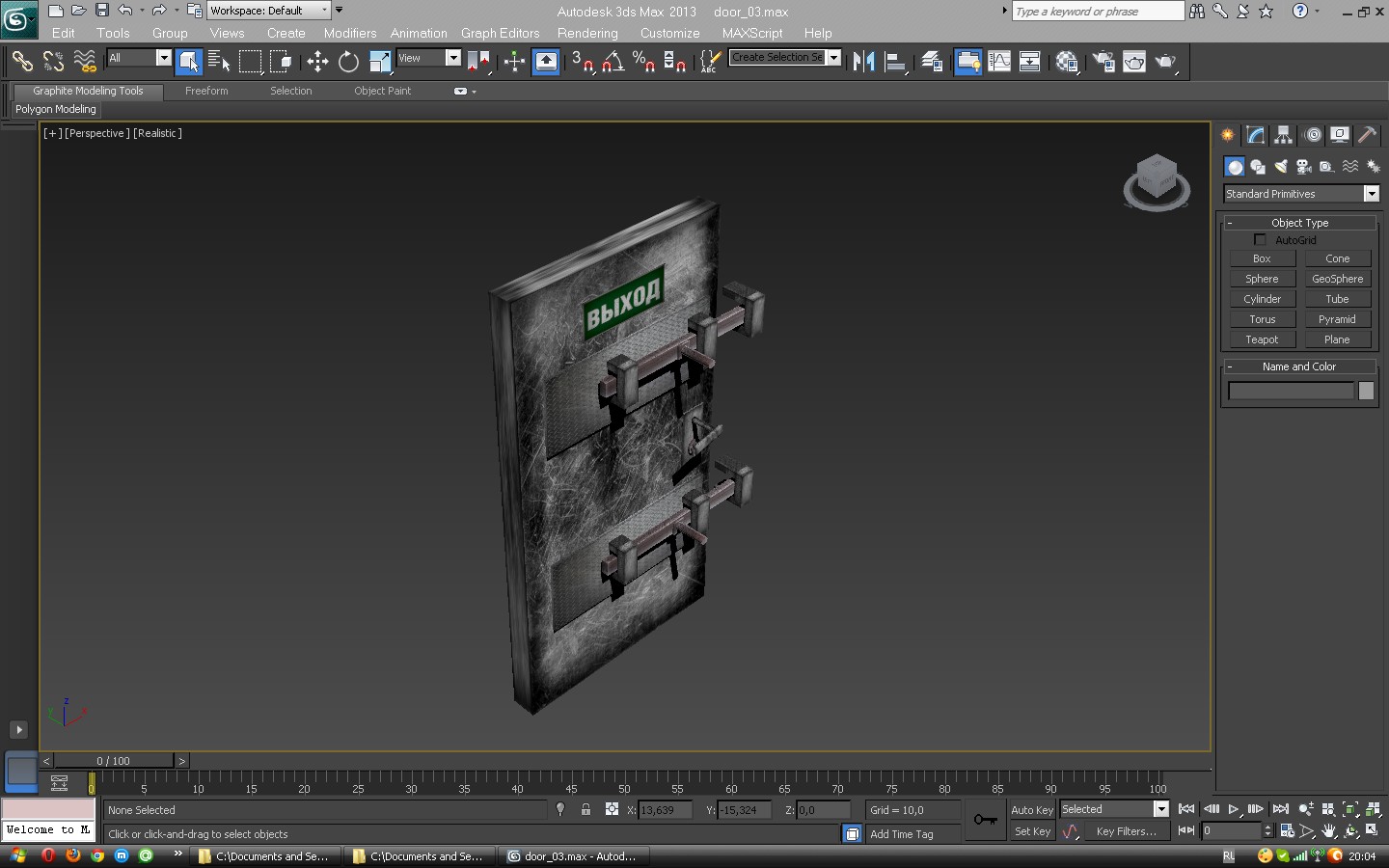
* 01\_Exit\_01;
* 02\_Дизельный генератор;
* 03\_Склад топлива;
* 04\_Комната с трубами;
* 05\_Стенды;
* 06\_Фильтры;
* 07\_Баки с водой;
* 08\_Exit\_02;
* 09\_Комната\_01;
* 10\_Комната\_02;
* 11\_Коридор\_01;

Список объектов взаимодействия:

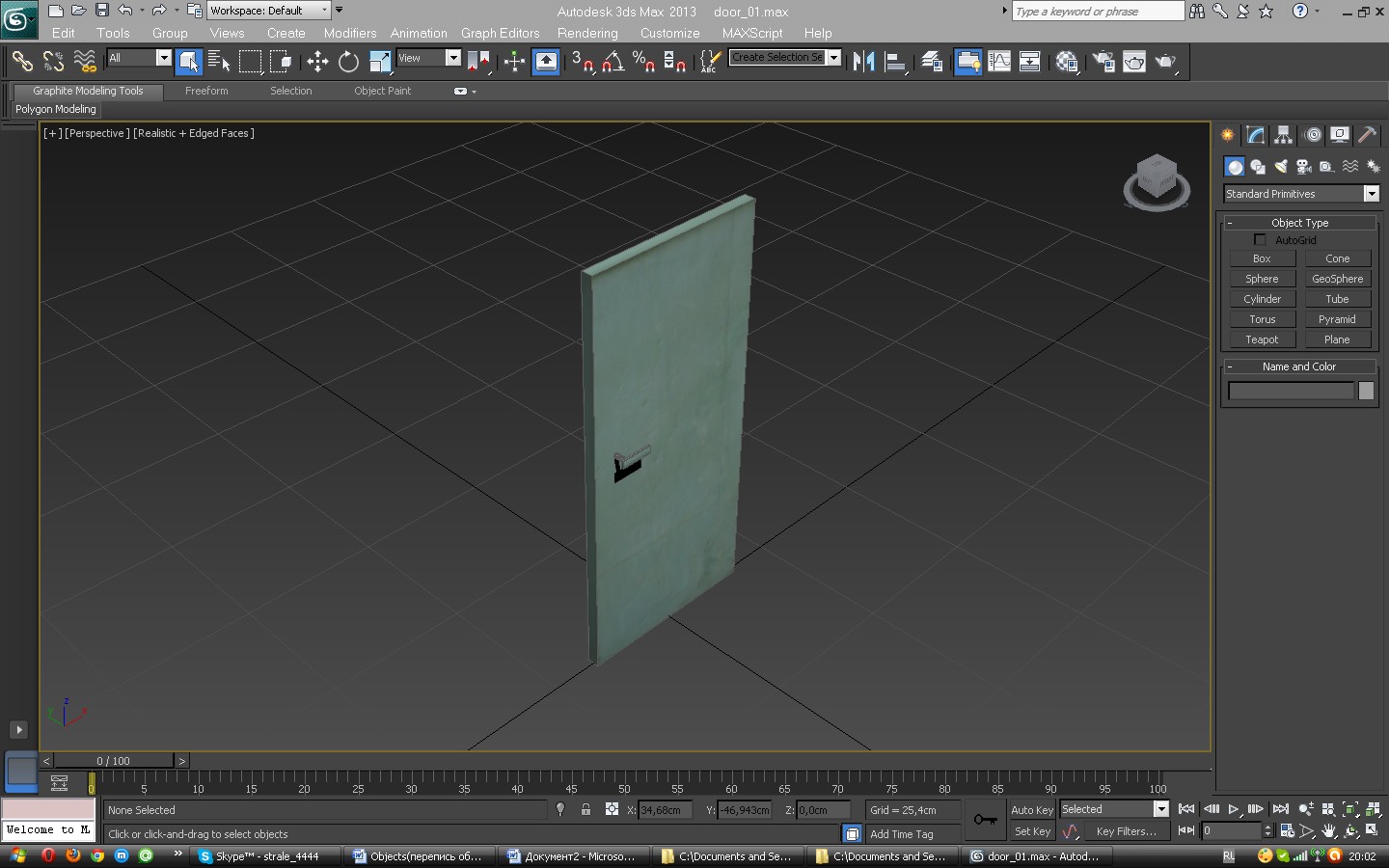
переключатель света\_1;



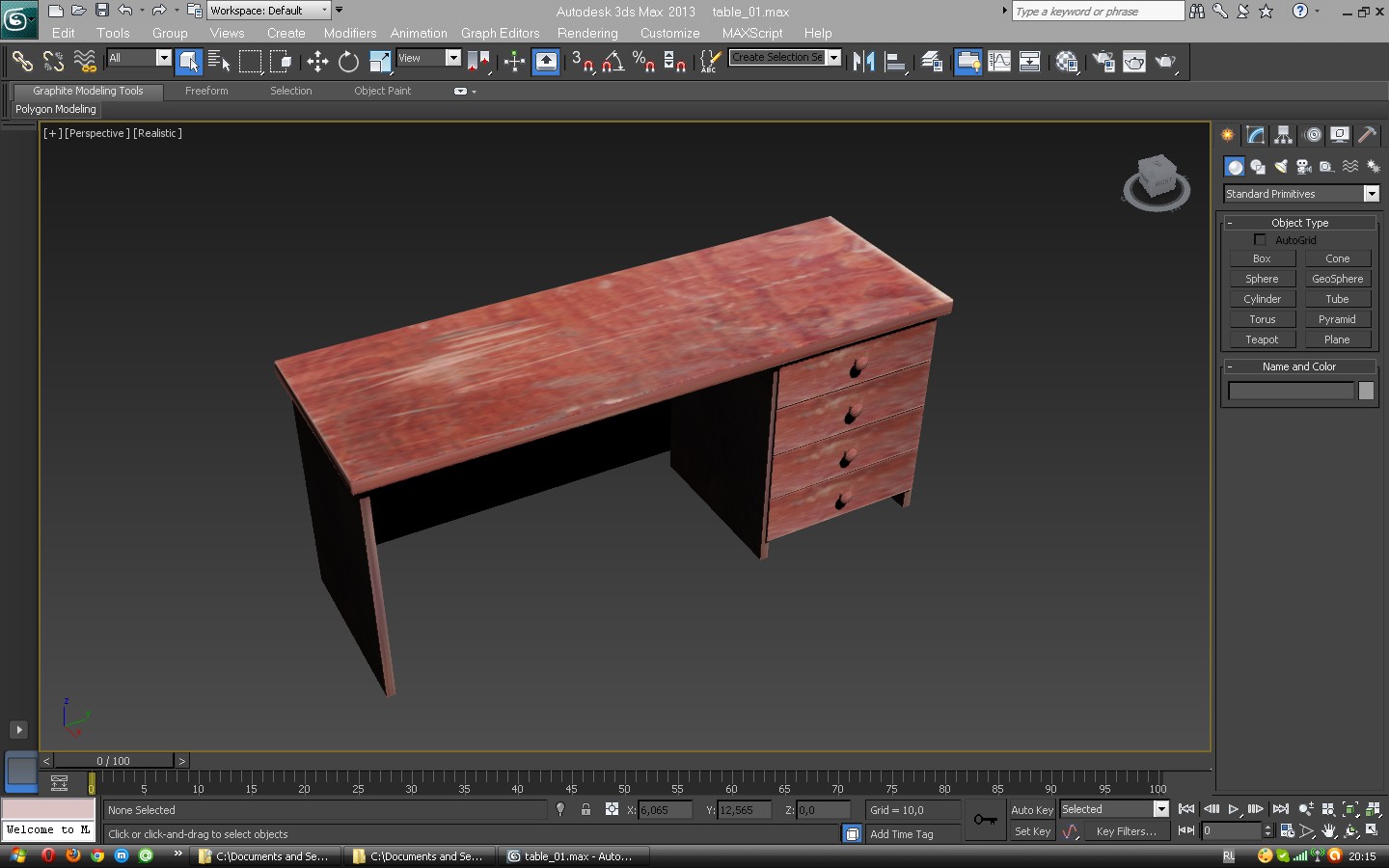
дверь\_1;



дверь\_2;



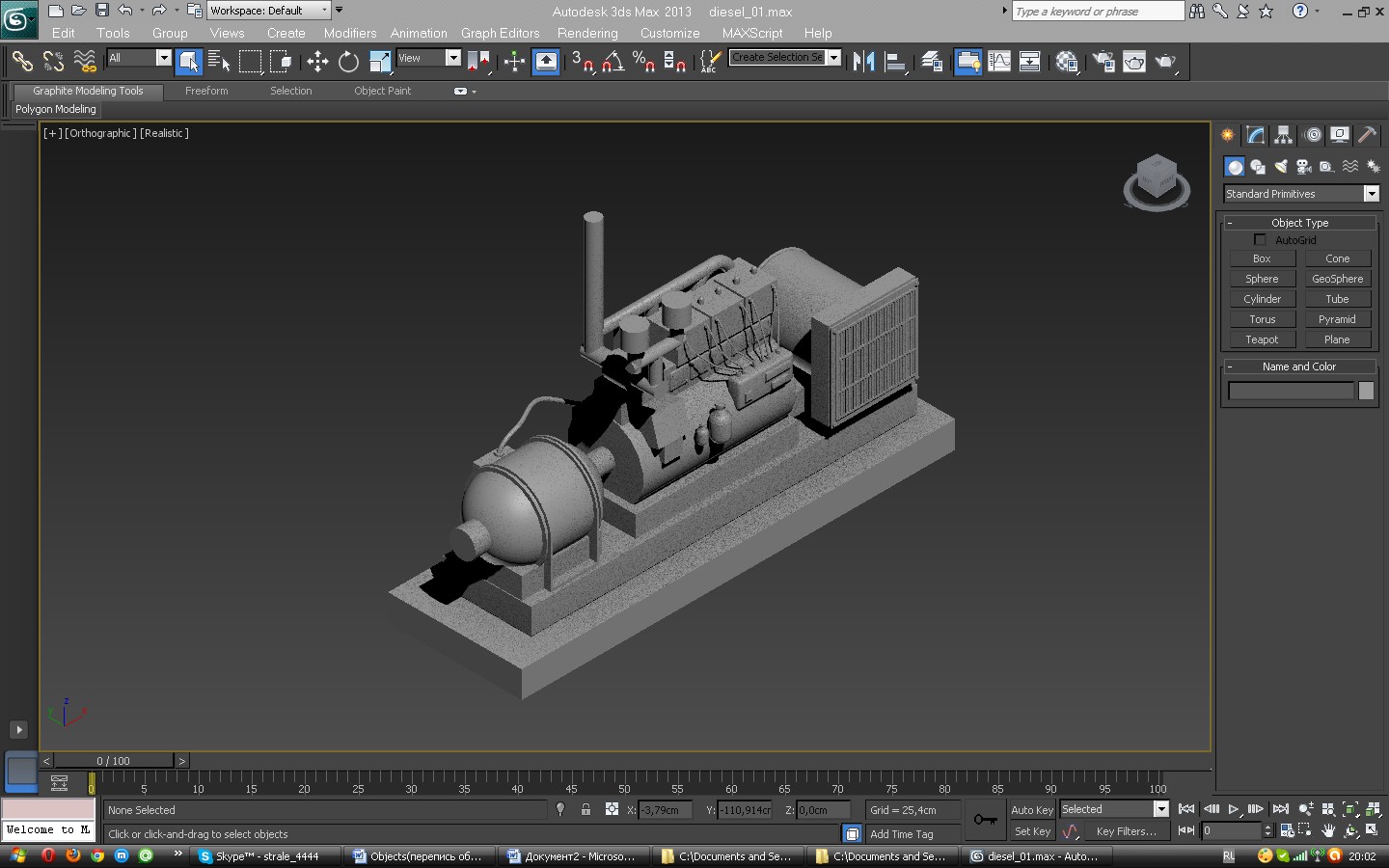
стол\_1;



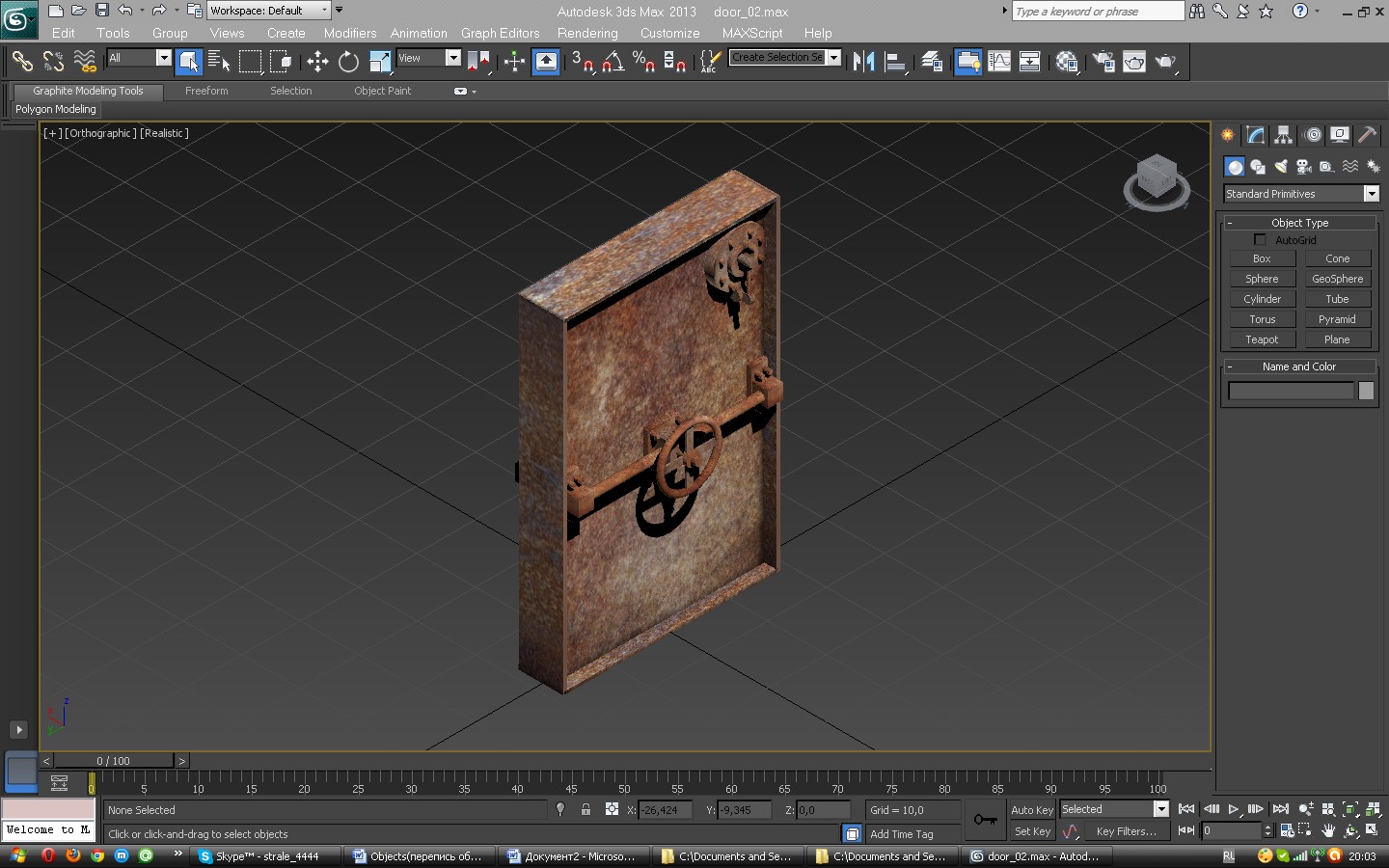
стол\_2;



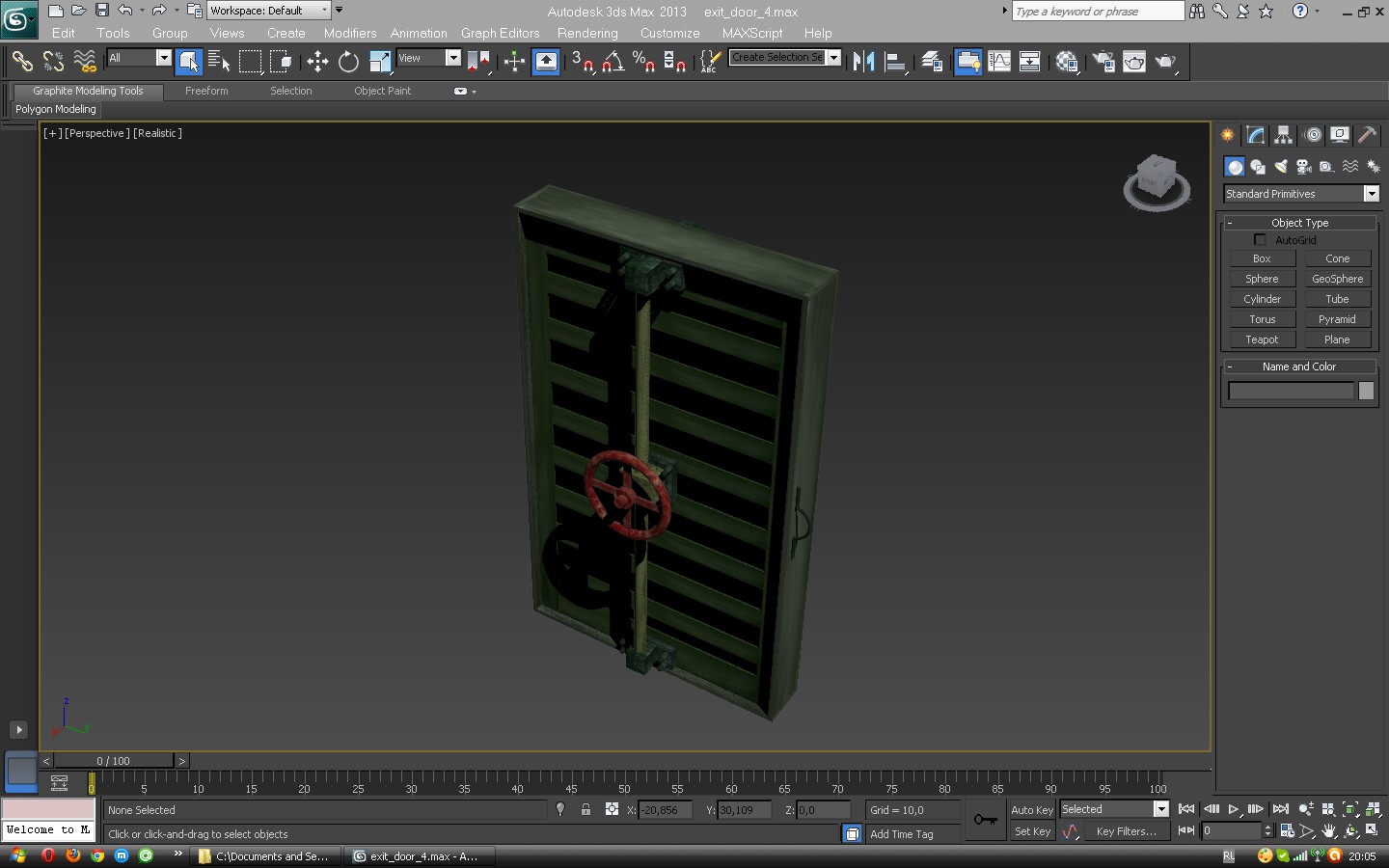
дизельный генератор;



гермодверь\_1;



гермодверь\_2;



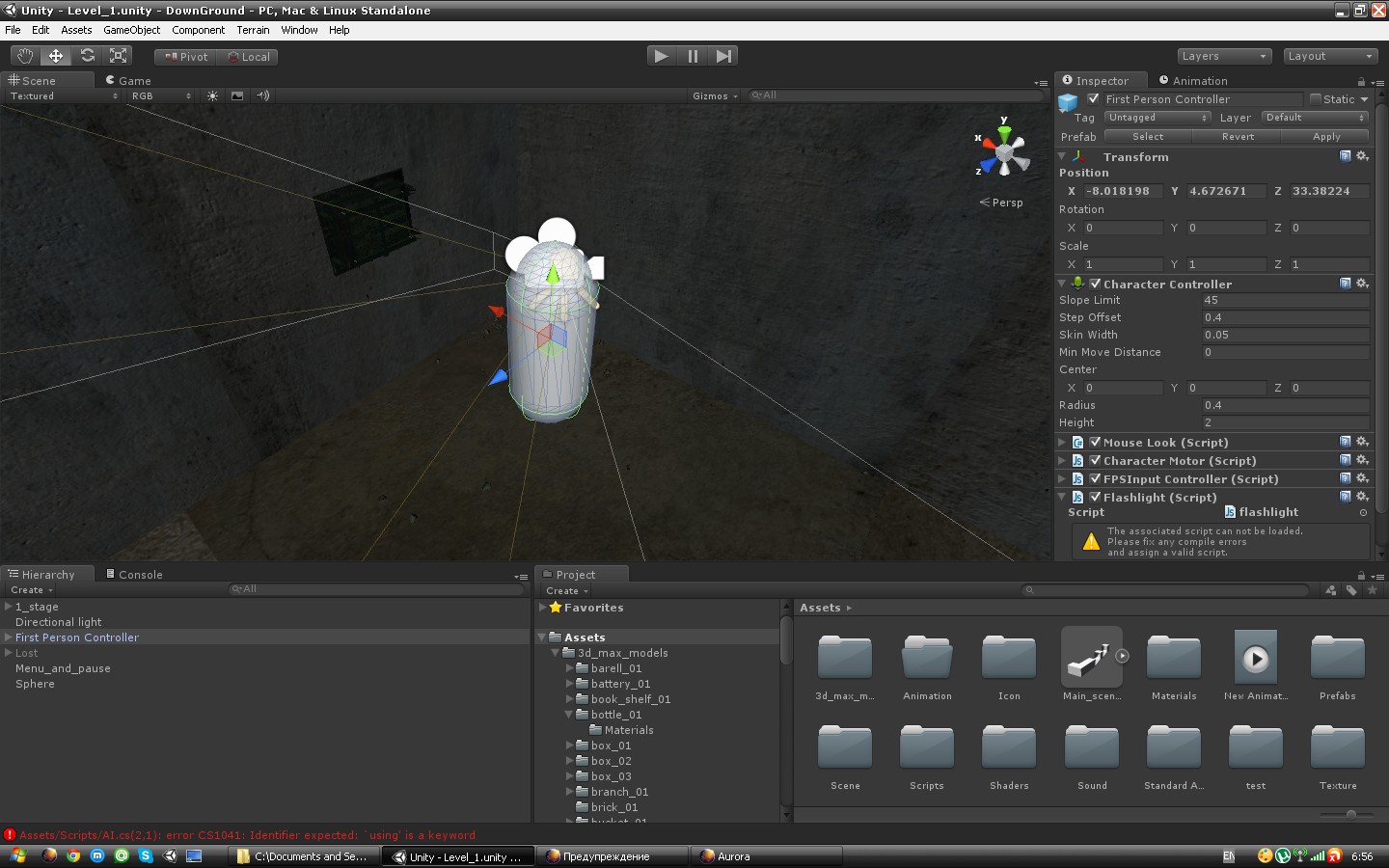
Список триггеров:

* trigger\_audio;
* trigger\_animation;
* trigger\_text;

Список скриптов:

* light\_mainobj
* light\_interobj
* anim\_mainobj
* anim\_interobj

Main Object:



Main Object – это объект, имитирующий передвижение персонажа по помещениям. Объект состоит из:

* графической оболочки, которая выполнена в виде капсулы;
* камеры, которая имитирует вид виртуального пространства из глаз персонажа;
* скриптов, которые описывают взаимодействие персонажа с окружающей средой;
* элемента освещения, прикрепленного к персонажу в качестве фонарика;

# ГЛАВА 4. КОНТРОЛЬНЫЙ ПРИМЕР

Для выполнения ручного тестирования разработанных алгоритмов в подготовленной тестовой среде были написаны и продемонстрированы следующие сценарии.

## 4.1. Сценарий взаимодействия Main Object и Interactive Object. Включение или выключение освещения

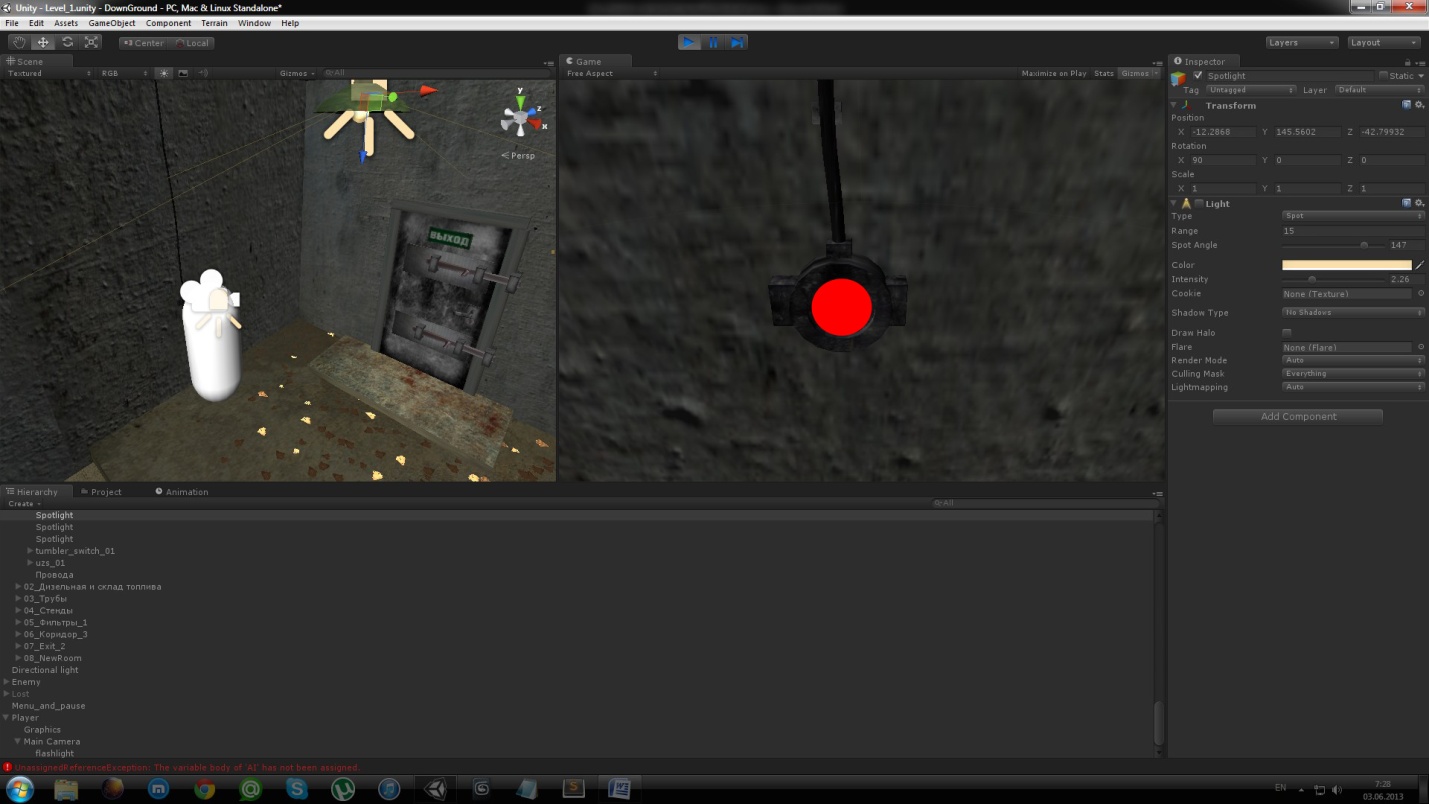
Описание:

Пользователь включает и выключает освещение.

Предусловие:

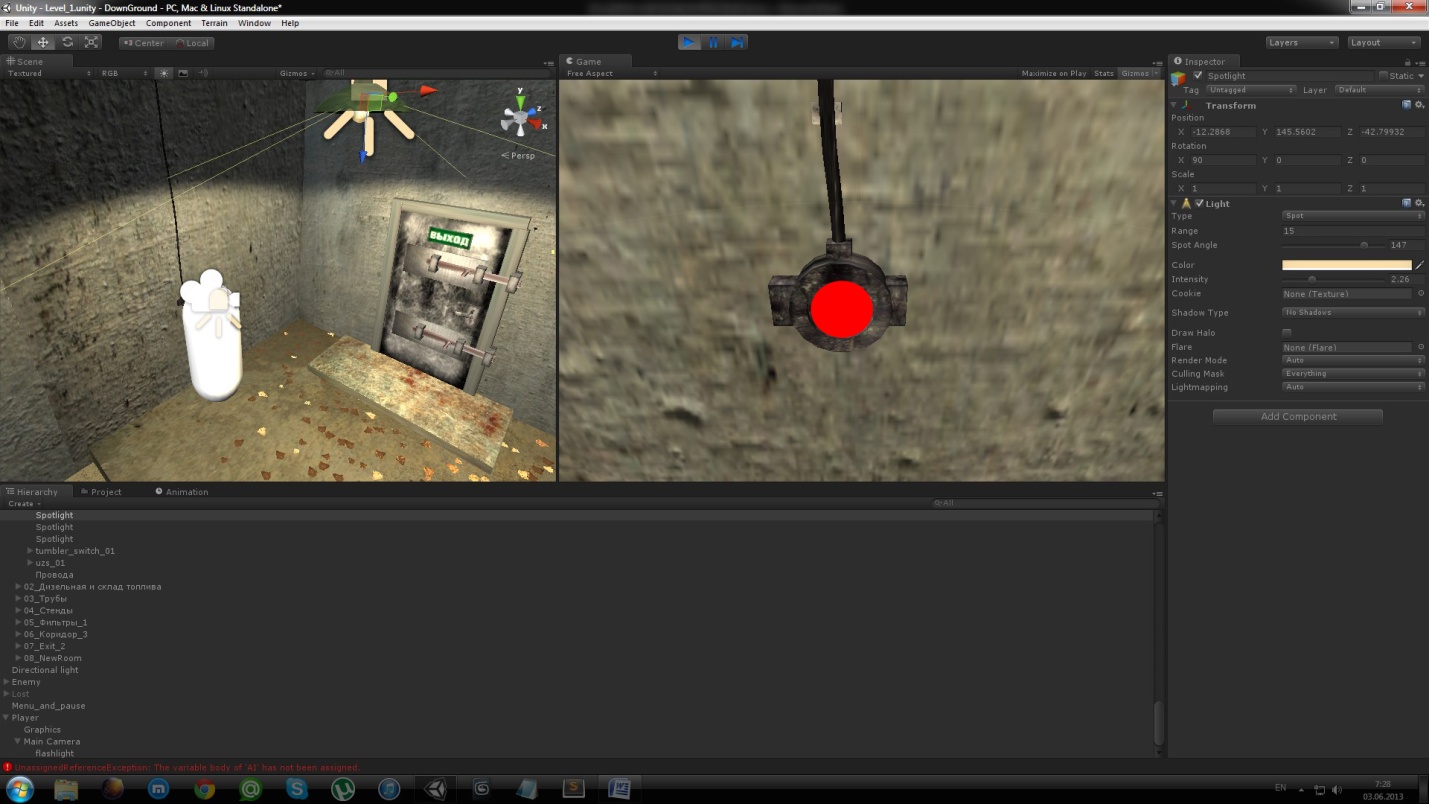
В сцене присутствует переключатель освещения и Main Object. У Main Object в окне Inspector отображается скрипт взаимодействия с переключателем освещения. У переключателя освещения в окне Inspector отображается скрипт взаимодействия с Main Object, а так же снят чек-бокс с поля activity.

Шаг 1: Пользователь, управляя Main Object, наводит взгляд на переключатель освещения.



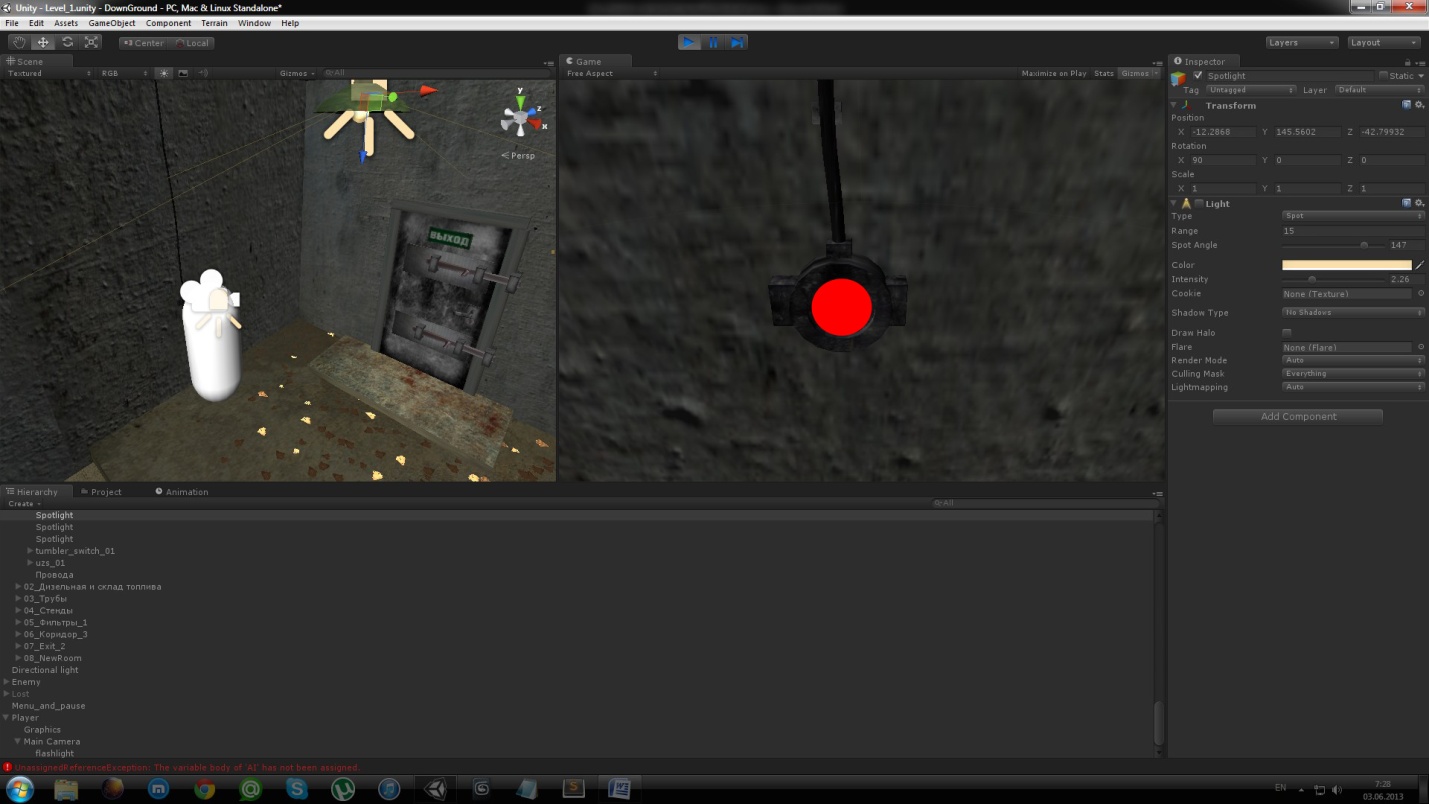
Ожидаемый результат: Переключатель освещение подсвечивается.

Шаг 2: Пользователь нажимает кнопку «F».



Ожидаемый результат: Включается освещение. Чек-бокс поля activity становится отмеченным.

Шаг 3: Пользователь нажимает кнопку «F».



Ожидаемый результат: Выключается освещение. Чек-бокс поля activity не отечен.

## 4.2. Сценарий взаимодействия Main Object и Interactive Object. Включение или выключение анимации

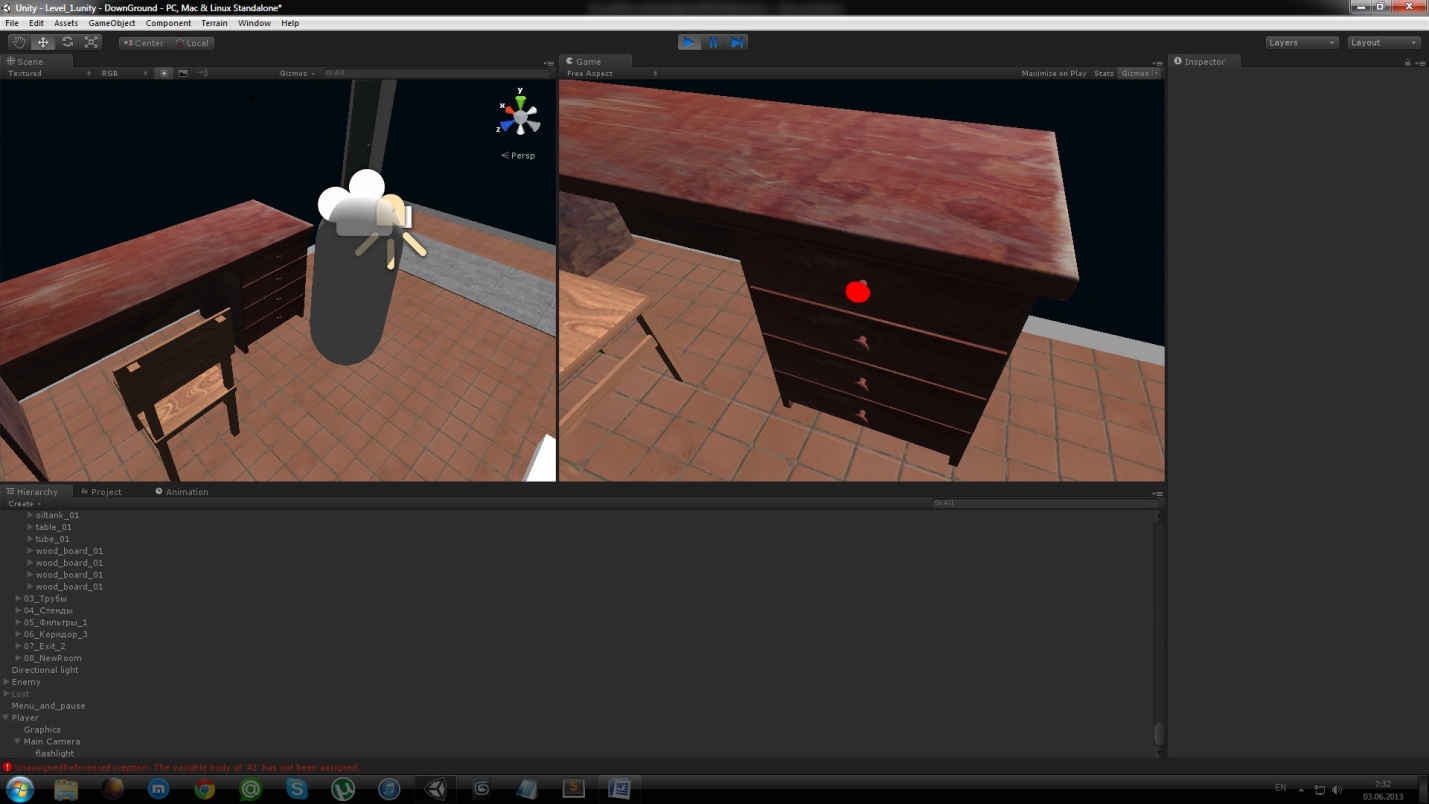
Описание:

Пользователь открывает и закрывает ящик стола.

Предусловие:

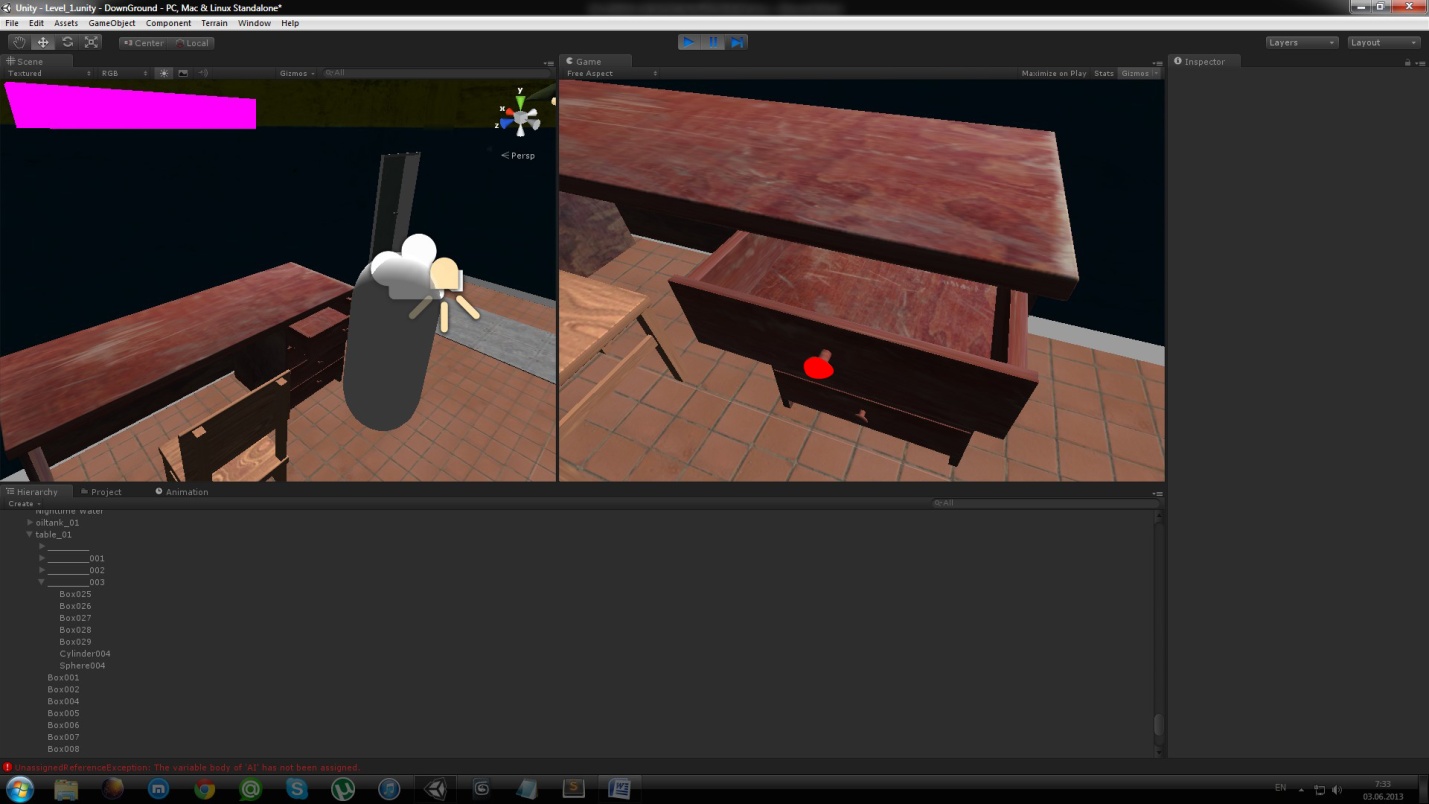
В сцене присутствует стол и Main Object. У Main Object в окне Inspector отображается скрипт взаимодействия со столом. У стола в окне Inspector отображается скрипт взаимодействия с Main Object, а так же снят чек-бокс с поля activity.

Шаг 1: Пользователь, управляя Main Object, наводит взгляд на ящик стола.



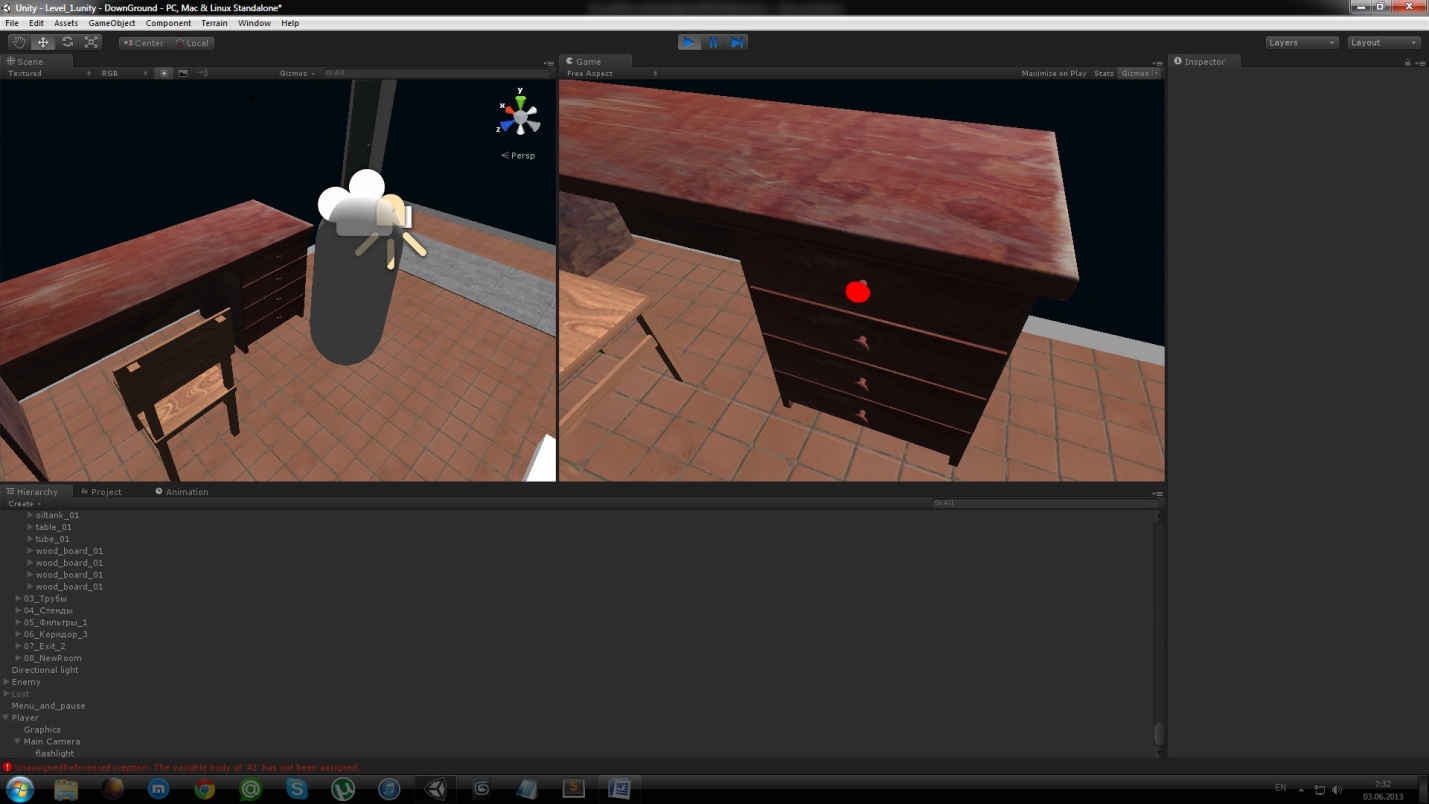
Ожидаемый результат: Ящик стола подсвечивается.

Шаг 2: Пользователь нажимает кнопку «F».



Ожидаемый результат: Ящик выдвигается. Чек-бокс поля activity становится отмеченным.

Шаг 3: Пользователь нажимает кнопку «F».



Ожидаемый результат: Ящик задвигается. Чек-бокс поля activity не отечен.

## 4.3. Сценарий взаимодействия Main Object с триггерами. Включение или выключение аудиозаписи

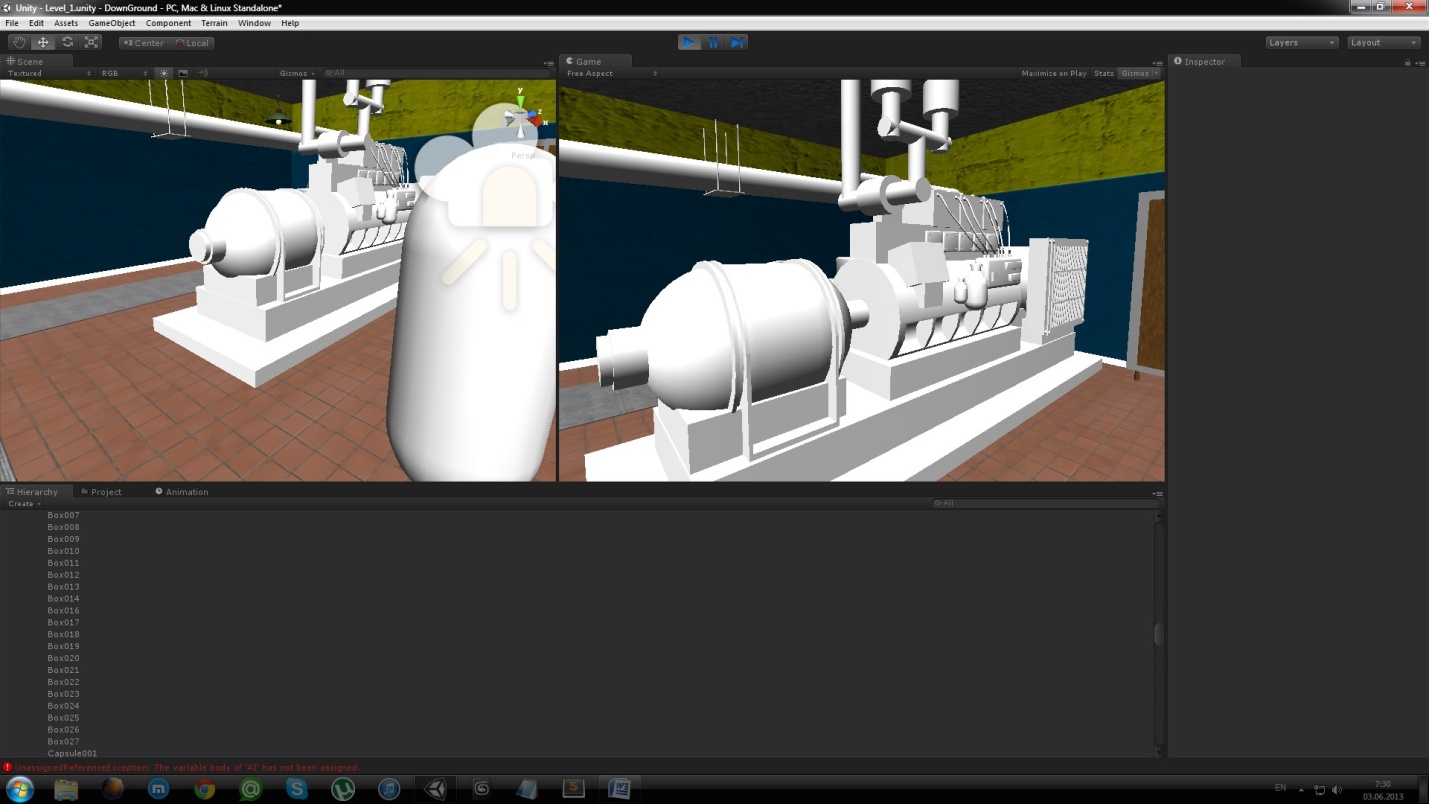
Описание:

Пользователь включает и выключает дизельный генератор.

Предусловие:

В сцене присутствует дизельный генератор и Main Object. В сцене создается объект Cube (куб) и переименовывается trigger\_audio. В Inspector (меню настроек объекта) в разделе Box Collider указывается значение «Is Trigger». Удаляется компонент Mesh Renderer, который отвечает за визуальное отображение куба. Проверяется наличия тэга «Main Object» на главном объекте, для уведомления триггера о существовании главного объекта. Скрипт накладывается на trigger\_audio. В параметр check ставиться значение 1.

Шаг 1: Пользователь, управляя Main Object подходит к дизельному генератору и нажимает, кнопку «F».



Ожидаемый результат: Включается звук работающего мотора. Параметр check принимает значение 0.

Шаг 2: Пользователь нажимает кнопку «F».



Ожидаемый результат: Выключается звук работающего мотора. Параметр check принимает значение 1.

## 4.4. Сценарий взаимодействия Main Object с триггерами. Включение или выключение анимация объекта

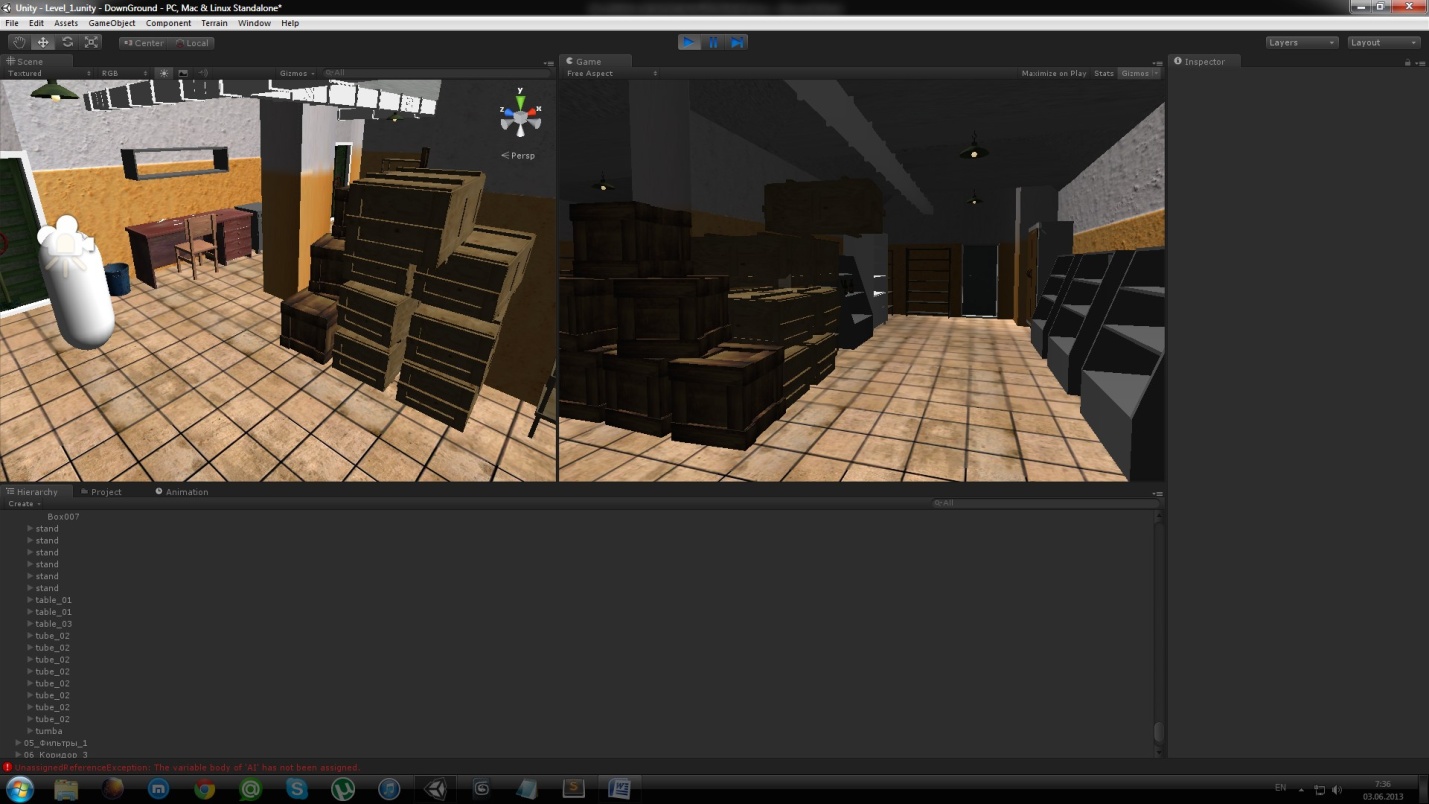
Описание:

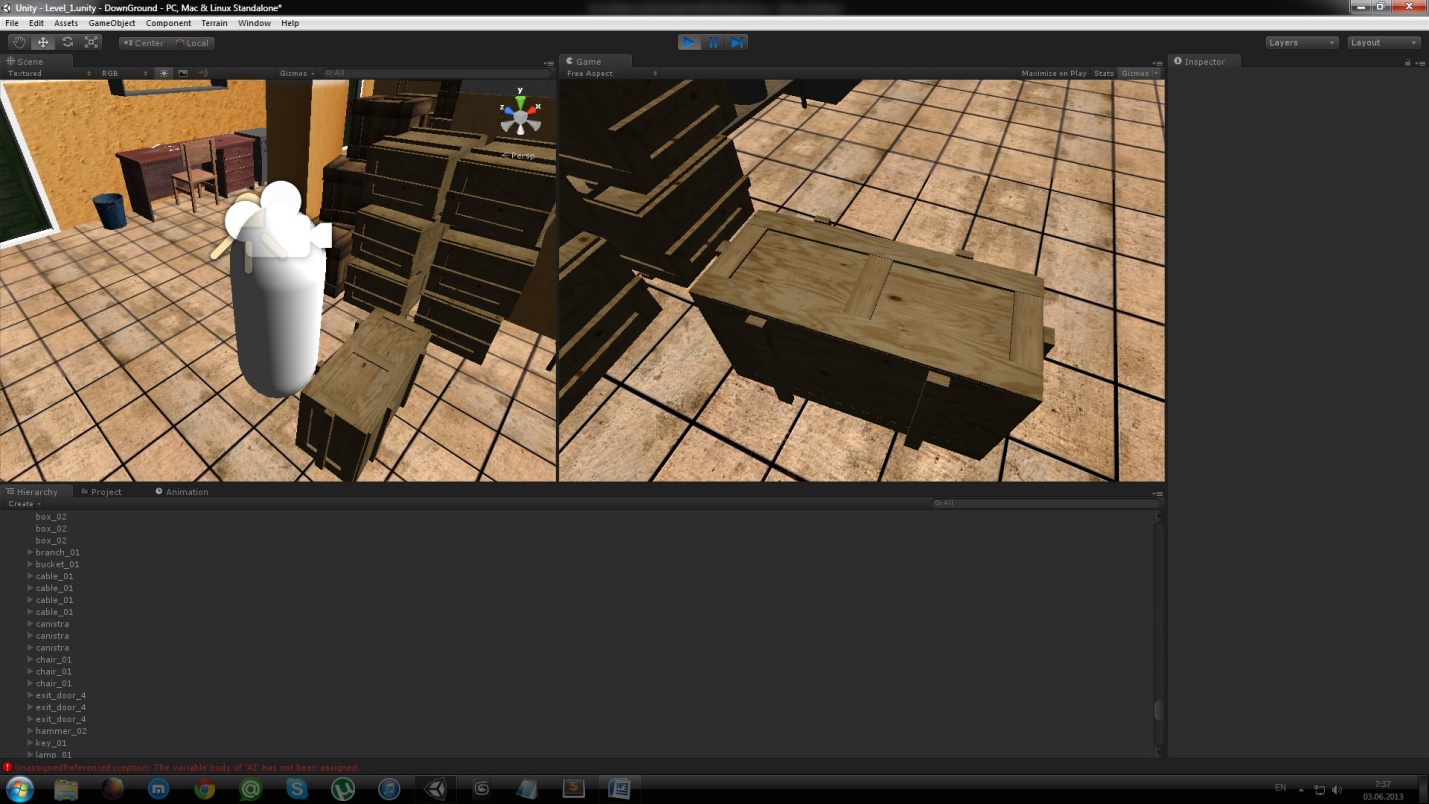
Пользователь заходит в помещение, с полки падает ящик.

Предусловие:

В сцене присутствует ящик и Main Object. В сцене создается объект Cube (куб) и переименовывается trigger\_animation. В Inspector (меню настроек объекта) в разделе Box Collider указывается значение «Is Trigger». Удаляется компонент Mesh Renderer, который отвечает за визуальное отображение куба. Проверяется наличия тэга «Main Object» на главном объекте, для уведомления триггера о существовании главного объекта. Скрипт накладывается на trigger\_animation. В параметр check ставиться значение 1. Параметр Ground принимает значение Cube. Параметр animom принимает значение выбранной анимации.

Шаг 1: Пользователь, управляя Main Object подходит к ящику.





Ожидаемый результат: Ящик падает.

## 4.5. Сценарий взаимодействия Main Object с триггерами. Появляется или пропадает текстового сообщения

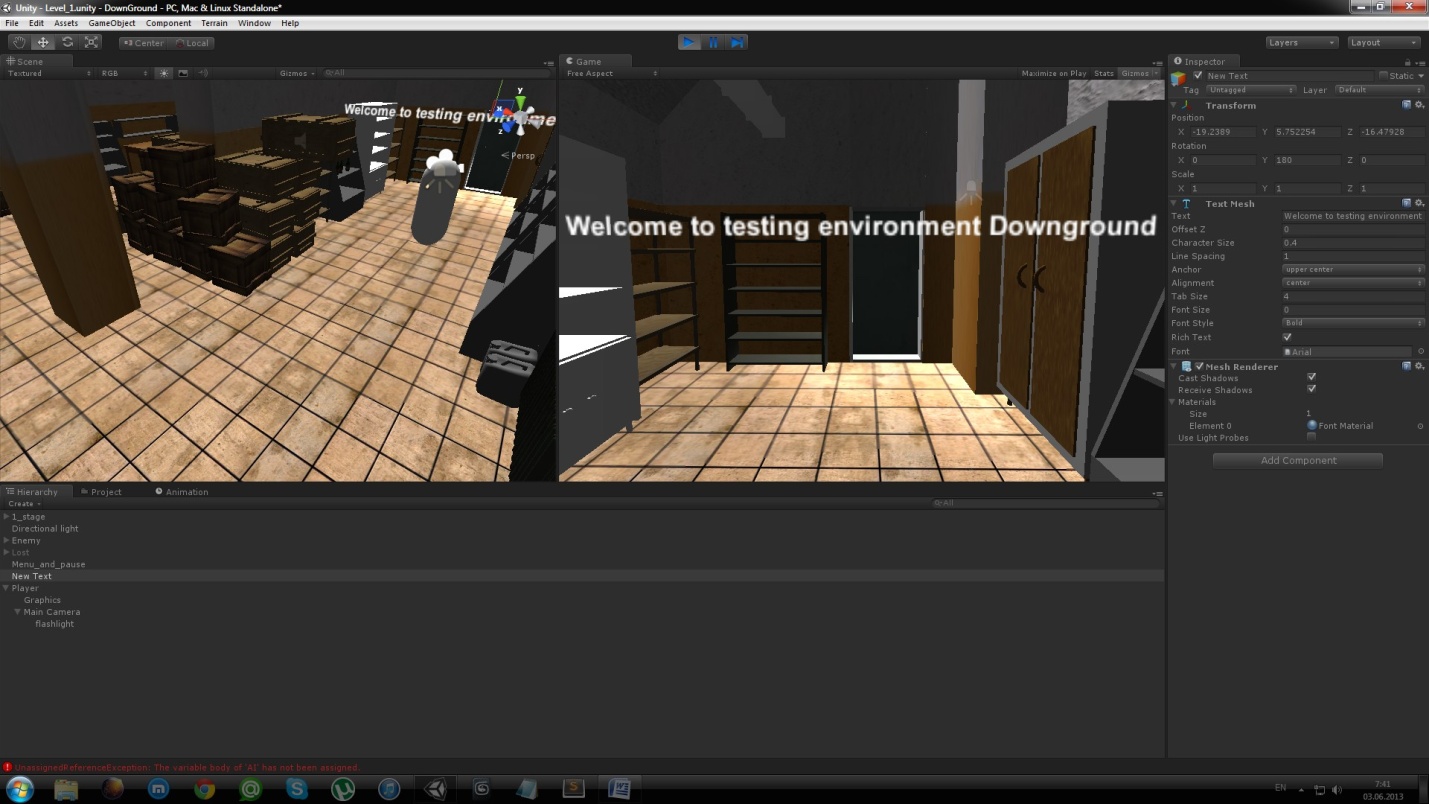
Описание:

Пользователь заходит в помещение, появляется текстовое сообщение.

Предусловие:

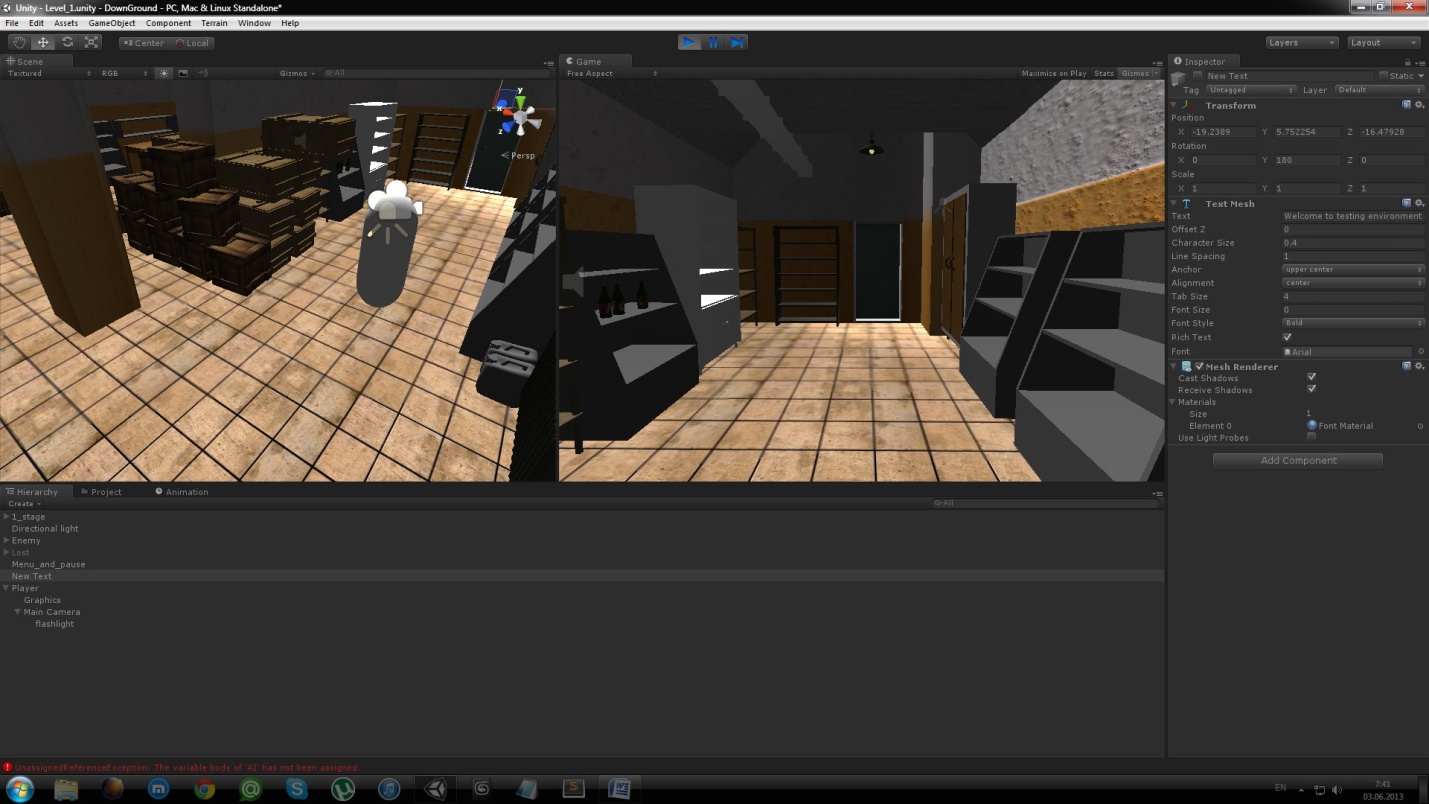
В сцене присутствует текст и Main Object. В сцене создается объект Cube (куб) и переименовывается trigger\_text. В Inspector (меню настроек объекта) в разделе Box Collider указывается значение «Is Trigger». Удаляется компонент Mesh Renderer, который отвечает за визуальное отображение куба. Проверяется наличия тэга «Main Object» на главном объекте, для уведомления триггера о существовании главного объекта. Скрипт накладывается на trigger\_text. В параметр check ставиться значение 1. Параметр Ground принимает значение Cube. Параметр message.text принимает значение выбранногой текста.

Шаг 1: Пользователь, управляя Main Object заходит в помещение.



Ожидаемый результат: Появляется текстовое сообщение: «Welcome to testing environment Downground». Параметр check принимает значение 0;

Шаг 2: Пользователь, управляя Main Object проходит чуть глубже помещения.



Ожидаемый результат: Пропадает текстовое сообщение: «Welcome to testing environment Downground»; Параметр check принимает значение 1;

## 4.6. Сценарий взаимодействия Main Object с триггерами. Комбинация из вышеперечисленных пунктов

Описание:

Пользователь заходит в помещение, появляется текстовое сообщение, и сверху падает ящик. Пользователь подходит к дизельному генератору, затем включает и выключает шум мотора.

Предусловие:

Для выполнения данной необходимы предусловия последних трех тест-кейсов.

Шаг 1: Пользователь, управляя Main Object заходит в помещение.

Ожидаемый результат: Появляется текстовое сообщение: «Welcome to testing environment Downground». Сверху падает ящик. Параметры check принимает значение обоих триггеров 0;

Шаг 2: Пользователь, управляя Main Object проходит чуть глубже помещения.

Ожидаемый результат: Пропадает текстовое сообщение: «Welcome to testing environment Downground»; Параметр check принимает значение 1;

Шаг 3: Пользователь, управляя Main Object подходит к дизельному генератору и нажимает, кнопку «F».

Ожидаемый результат: Включается звук работающего мотора. Параметр check принимает значение 0.

Шаг 4: Пользователь нажимает кнопку «F».

Ожидаемый результат: Выключается звук работающего мотора. Параметр check принимает значение 1.

# ЗАКЛЮЧЕНИЕ

В данной дипломной работе был произведен основной разбор принципов взаимодействия. Результатом этого разбора, стал тестовой уровень для компьютерной игры, показывающий возможности игрового движка, а так же знаний накопленных в процессе изучения данной темы. Стало ясно, что для взаимодействия объектов необходимо установить внутренний диалог между ними с помощью обмена сообщениями. Такой подход позволяет строить сложные цепочки событий, позволяющих описать огромной количество ситуаций как реальной жизни так и игровой механики в целом. Данный метод связи используется при создании:

* трехмерных мультипликационных фильмов, ,чтобы сократить процесс прорисовки кадра;
* тренажеров, направленных на обучение персонала различных структур;
* компьютерных игр, направленных на обучение;
* компьютерных игр направленных на развлечение;
* системы взаимодействия в робототехнике, позволяющей обмениваться информацей сразу нескольким роботам ;

В соответствии с дизайн документом, который был создан для описания разрабатываемой компьютерной игры, планируется развить следующие направляющие:

* Увеличение количества помещений;
* Увеличение количества объектов окружающей среды;
* Увеличение количества объектов взаимодействия;
* Повышение качества визуализации;
* Рефакторинг программного кода для оптимизации работы продукта;
* Моделирование тела персонажа;
* Повышение качества сюжетной составляющей;
* Разработка сложных ситуаций;
* Портирование продукта на портативные устройства;
* Продвижение программного продукта в плане реализации для целевой аудитории.

# СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Sue Blackman “Beginning 3DGame Development with Unity” 2011
2. Will Goldstone “Unity Game Development Essentials” 2009
3. Стиллмен Э., Грин Дж. - Изучаем C#. Включая C# .NET 4.0 и Visual Studio 2010. 2-е издание (Бестселлеры O'Reilly) – 2012
4. Фримен Эр., Фримен Эл., Сьерра К., Бейтс Б. - Паттерны проектирования - 2011