МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования

Пензенский государственный университет архитектуры и строительства

Факультет_____инженерно-строительный_____
Кафедра_____информационно-вычислительных систем___
Направление_09.03.02 «Информационные системы и технологии»_
Программа _____

Оценка «К защите»

_____ Заведующий кафедрой
_____ (_____)
«____»_____20__ г. «____»_____20__ г.

_____
(подпись секретаря ГЭК)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

на тему____Разработка компьютерной игры для изучения программирования_____
_____
_____

Научный руководитель
___к.т.н., доцент, Пышкина И. С.___
(должность, степень, фамилия, инициалы)

_____
(подпись)

Консультант по разделу_____
__к.т.н., доцент, Пышкина И. С___
(должность, степень, фамилия, инициалы, подпись)

Студент гр.____16ИСТ1_____

Клейменов Артем Андреевич
(фамилия, имя, отчество)

_____
(подпись)

_____
(дата)

Министерство науки и высшего образования российской федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
Пензенский государственный университет архитектуры и строительства
Факультет_____инженерно-строительный_____
Кафедра_____информационно-вычислительных систем___
Направление _09.03.02 «Информационные системы и технологии»_
Группа _____16ИСТ1_____

ЗАДАНИЕ
на выпускную квалификационную работу

Студент _____Клейменов Артем Андреевич_____
(фамилия, имя, отчество)

Тема выпускной работы:
____Разработка компьютерной игр для изучения программирования_____
_____
_____
_____

Время выполнения работы с _03.06.2020_ по _19.06_ 20_20_ г.
Руководитель выпускной квалификационной работы
_____к.т.н., доцент, Пышкина. И. С., ПГУАС_____
(фамилия, инициалы, должность, степень, место работы)

_____
Тема выпускной квалификационной работы и руководитель утверждены приказом № _06-09-920_ *от «*_26_*»* _____ноября_____ 2019 г.

Консультант по разделу ___Пышкина. И. С, доцент, к.т.н., ПГУАС_____
(фамилия, инициалы, должность, степень, место работы)

Консультант по разделу ____Пышкина. И. С, доцент, к.т.н., ПГУАС_____
(фамилия, инициалы, должность, степень, место работы)

Место выполнения работы _____ПГУАС_____

_____

Заведующий кафедрой ____Васин Л. А_____ «___» _____ 20__ г.

Задание принял к исполнению «___» _____ 20__ г.

_____
(подпись студента)

1. Содержание задания

1) Провести исследования данного вопроса
2) Провести анализ существующих решений
3) Изучить библиотеку Unity и ее инструмент разработки
4) Составить требования к программному средству
5) Реализовать игру по составленным требованиям

2. Задание и исходные данные по разделу

Разработка компьютерной игры для изучения программирования

Подпись консультанта _____

3. Рекомендуемая исходная литература

https://docs.unity3d.com – документация Unity

Подпись руководителя выпускной работы _____

КАЛЕНДАРНЫЙ ГРАФИК РАБОТЫ ПО РАЗДЕЛАМ ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ БАКАЛАВРА

№ п/п	**Перечень разделов работы**	**Срок выполнения**	**Отметки о выполнении**
1	Теоретическая часть	04.06.2020 – 10.06. 2020	
2	Проектирование программного средства	11.06.2020 – 17.06.2020	
3	Разработка программного средства	18.06.2020 – 24.06.2020	

Составлен «___» _____ 20___ г.

_____ _____

(Подпись руководителя) (Подпись студента)

РЕФЕРАТ

Выпускная квалификационная работа на тему «Разработка компьютерной игры для обучения программирования» содержит 104 листа, 98 рисунков

РАЗРАБОТКА КОМПЬЮТЕРНОЙ ИГРЫ ДЛЯ ОБУЧЕНИЯ ПРОГРАММИРОВАНИЯ, СРЕДА РАЗРАБОТКИ UNITY,

Целью работы является создание игры, в процессе прохождения которой пользователь должен ознакомиться с программированием или повысить свои навыки. Игра должна приносить удовольствие игроку чтобы избежать эмоционального напряжения.

Объектом разработки является система преподавания программирования.

Предметом разработки выпускной квалификационной работы игра обучающая программированию

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

На сегодняшний день, как подрастающему поколению, так и взрослым сложно представить жизнь без компьютеров и компьютерных игр. С появлением персональных компьютеров, с каждым годом, их роль в жизни людей постоянно растет. Компьютерные развлечения делают жизнь человека богаче и насыщеннее. Для кого-то игры это развлечение или отдых, для кого-то - работа. Это деятельность проникла во все сферы нашей жизни и стоит на одном уровне с просмотром фильмов, чтением книг или газет. Их функционал достаточно широк: развитие, образование, психология, досуг, развлечение.

Игра – это вид деятельности, направленный на обучение и усвоение в рамках игры, опыта, путем моделирования разных задач.

Из определения стало понятно, что основная цель игры – обучение.
А качество обучения в свою очередь влияет на подготовку специалистов. Именно образованность работников определяет, в каком мире мы будем жить. Ведь никому не хотелось пользоваться операционной системой в телефоне, где программист не смог предусмотреть некоторые ошибки. Что в дальнейшем может привести к утечке информации. Поэтому именно обучение, как положительную сторону компьютерной игры, будет максимально реализована в данной работе.

На мой взгляд существующие методы обучения компьютерным технологиям не являются в полной мере эффективными. Поскольку интересы современных людей стали отличатся от интересов людей 80-90-ых годов, под которых разрабатывалась система образования. Сейчас среднестатистическому ученику, обучение может показаться скучным и однообразным, что может убить его тягу к знаниям. Поэтому нужно идти в ногу со временем и добавлять в обучение что-то свежее, интересное.

Важнейшей деталью в процессе обучения является не только прослушивание лекций, но и решение различных занимательных задач. Будущему программисту важно научиться думать, находить правильные решения и быстро соображать. Что как раз компьютерная игра и может дать учащемуся.

Выбор пал на программирование не просто так. Программированию отведена очень важная роль в современном мире. Оно приносит огромную пользу в наши дни и используется в бизнесе, экономике, медицине, физике, биологии и т.д. Раньше однообразную работу человек делал своими руками, но с развитием технологий, многие профессии стали автоматизироваться и управляется посредством программного обеспечения и это стало обеспечивать скорость, точность операций и эффективность производства. С каждым годом число профессий будет все больше и больше автоматизироваться. И поэтому программисты очень востребовательные люди. На мой взгляд программирование - это ключ к светлому и технологичному будущему.

Объектом исследования дипломной работы, является система преподавания программирования.

Предметом изучения стала игра, обучающая программированию

Цель исследования. Целью работы является создание игры, в процессе прохождения которой игрок должен ознакомиться с программированием или повысить свои навыки. Игра должна приносить удовольствие игроку что бы избежать эмоционального напряжения.

Задачи. Для достижения поставленной цели необходимо решить следующие задачи:

— произвести исследование данного вопроса и анализ существующих решений;

— построить модели и алгоритмы разрабатываемой системы;

— выполнить программную реализацию средства.

Научная новизна работы состоит в разработке внедрения в учебный процесс, новых методов обучения в сфере программирования. Эта методика позволяет привлечь и заинтересовать больше людей разных возрастов к данной отрасли, за счет популярности в наше время компьютерных игр. И обучить их основам программирования, без помощи преподавательского состава. В какой-то степени даже автоматизировать практические занятия целой группы учащихся. Тем самым экономя такой ресурс как труд учителей.

В соответствии с намеченной целью и задачами мною были определены следующие методы исследования:

1. Изучение библиотеки Unity3D
2. Исследование существующих методов обучения программированию
3. Анализ компьютерных игр
4. Продумывание идеи обучающего аспекта игры
5. Создания прототипа
6. Реализация игры и основная работа над ней
7. Определение визуального стиля
8. Тестирование
9. Релиз игры
10. Опрос пользователей после прохождения игры
11. Анализ опроса

ГЛАВА 1. ПРЕДПРОЕКТНОЕ ИССЛЕДОВАНИЕ

1.1. Характеристика объекта исследования

Тема, связанная с изучением компьютерных игр, с каждым годом приобретает все большую актуальность в условиях информатизации современного общества. В настоящее время трудно представить ребенка, не осведомлённого о существовании компьютерных игр. Согласно ежегодно проводимым исследованиям почти каждый ребенок хотя бы раз в своей жизни играл в компьютерную игру.

Некоторые специалисты полагают, что компьютерные игры влияют на детей отрицательно, другие же отталкиваются от противоположной точки зрения. Я остановлюсь на рассмотрении основных достоинств и недостатков компьютерных игр, но при этом главной задачей для меня является вопрос, как именно воспринимаются игры, и насколько это возможно использовать в обучающей практике.

Прежде чем приступить к рассмотрению понятия «компьютерных игр», хотелось бы обратить внимание на главное, что именно подразумевается в слове «игра». Понятия игры, игрового обучения до сих пор вызывают споры в научных сообществах. Обозначу лишь некоторые определения, которые помогут мне в исследовании. В своих поисках обращусь к словарям. В. И. Даля, определяющего игру, как потеху, веселье или забаву. Ожегов пишет, что игра — это вид деятельности, мотив которой заключен не в ее результатах, а в самом непосредственном процессе.

Определение игры в образовании имеет схожее описание, но при этом обладает более узкой направленностью. По мнению А. Л. Катковой, «игра — это один из видов образовательной деятельности учащихся, мотивом которой является сам процесс, направленный на познание, освоение и преобразование

в качестве средства воспитания и обучения». Оставлю это понятие основополагающим для моего исследования.

Долгое время игру рассматривали как инертный по отношению к обучению процесс. Игра всегда являлась инструментом в обучении, способствующим повышению мотивации детей, их заинтересованности. Для нас интересны игры, которые носят дидактический характер, то есть обучающий. Но, поскольку наш окружающий мир склонен к изменениям, впрочем, как и общество с его модифицированными приоритетами, дидактические игры в условиях информатизации образования начинают приобретать новый вид, а именно — компьютерных обучающих игр.

Использование компьютерных игр в образовательном пространстве позволяет:

— повысить у детей положительную мотивацию учения;

— расширить объем используемой информации;

— использовать новые формы представления информации, в частности, визуально-наглядные;

— расширить набор применяемых учебных задач;

— активно включать учащихся в учебный процесс посредством применения игровой компьютерной деятельности;

— обеспечить условия для развития интеллектуальной активности, творческого мышления учащихся.

Известный психолог А. Г. Шмелев отмечал, что игры могут выполнять функцию психологической разгрузки, исполняя роль своеобразного психологического тренинга. Компьютерная игра при помощи виртуальной реальности создает для ученика новый мир возможностей, где он является не только его гостем, но и участником.

Виртуальность компьютерных игр позволяет заменить традиционную форму представления знаний (подразумеваются учебники, рассказ и другие классические методы) на реальное непосредственное динамичное воздействие. Ученик в ходе работы с игрой превращается из стороннего наблюдателя в создателя, влияющего на ход собственного обучения и испытывающего при этом эффект обратной связи. Также компьютерная игра способна удовлетворить потребности в познании. Она способна в необычной форме спроектировать ситуации и события (пожар, наводнение, шторм и т.д.), которые маловероятно могу произойти в реальном мире с человеком. Однако ситуации могут ему пригодиться в реальности в случае возникновения похожего момента из игры.

1.2. Типы компьютерных игр

Приведу в пример некоторые из видов компьютерных игр, включая их роль в формировании мышления ребенка.

Первый тип — приключенческие компьютерные игры. Их сценарий похож на мультфильм, но с интерактивным свойством, где участвующий может управлять ходом событий, изменяя его на свой лад. Сценарии подразумевают решение разных задач, где требуется неплохая сообразительность и развитое логическое мышление. К сожалению, подобный вид игр предполагает длительное сидение за компьютером, что может навредить здоровью ребенка. Дети семи лет могут работать за ПК не более десяти минут в день. Приключенческие компьютерные игры могут послужить серьезным раздражителем для гиперактивных детей.

Второй тип — игры стратегии. В основную цель входит завоевание вражеских поселений, набор очков, заключение союзов, управление ресурсами, войсками, энергией и так далее. Такая игра развивает в ребенке способность к формированию определенного ряда действий, тренируя в первую очередь многофакторное мышление. К сожалению, стратегии не подходят детям дошкольного и младшего школьного возраста в силу длительной продолжительности.

Третий тип — игры аркады. Это поуровневые сценарии, дробленые игры, где в качестве награды происходит переход к следующему эпизоду для свершения более сложной миссии. Здесь также предусмотрен набор очков и бонусов, в особенности, — за быстроту прохождения уровня, победу над сильным врагом, ценную находку, например, секретную дверь и т.д. Аркады детям с гиперактивностью не рекомендованы.

Четвертый тип — ролевые игры. Сценарий подразумевает поиск артефакта, человека или заклинания. Путь к цели преграждает хитрый враг, с которым приходится вступать в бой, но можно пойти на хитрость и обмануть.

Основное правило ролевых игр — использовать нужного персонажа в нужном месте и в нужное время. Ролевые игры хорошо тренируют скорость реакции, внимание, усидчивость, глазомер.

Пятый тип — симуляторы. Они позволяют спроектировать различные виды деятельности и обладают особыми педагогическими возможностями. К ним относят авиасимуляторы, автосимуляторы, медицинские и спортивные симуляторы и другие. Симуляторы моделируют важные для реального мира законы, создавая приближенную к реальности модель.

Шестой тип — логические игры. Подразумевают всевозможные головоломки, задачки, например, требуется переставить фигуры или составить рисунок. Логические игры часто используют в качестве подспорья в обучении, помогая ребенку освоить математику, чтение, письмо и прочее.

1.3. Обоснование выбора объекта исследования

В последние годы, начиная с 2008-го, в игровом мире становится популярным жанр интерактивного кино. Такая игра предполагает выбор игроком одной из нескольких сюжетных линий, которые ведут к той или иной концовке. Каждое действие, порой даже не самое важное, может привести к различным финалам. Данный жанр игр пока еще не используется в рамках образовательного процесса, но это дело времени. Интерактивные игры положительно влияют на психологические особенности игрока. Так возможность выбора сценарной ситуации игрового компьютерного мира помогает игроку осознанно принимать решения и в дальнейшем осознавать их результаты. Данная особенность этого жанра, безусловно, была бы полезна в обучении. Компьютерная игра выступает как универсальное средство приобретения опыта, своеобразным тренажером человеческих навыков и умений, необходимых для решения задач человеческой жизнедеятельности.

Благодаря своему структурному многообразию компьютерные игры представляют собой богатую возможность приобретения навыков деятельности и мышления, вычленяя основные функционально-деятельностные позиции для решения конкретных задач человеческого бытия.

Компьютерные игры психологически готовят человека к напряженным эмоциональным ситуациям, позволяют проявить способность действовать в кризисных ситуациях, осуществлять психическую саморегуляцию в момент замешательства.

Педагоги, имеющие непосредственный опыт работы с детьми, увлекающимися компьютерными играми, говорят о положительных аспектах уровня познавательной активности, любознательности, удовлетворенности результатом своей деятельности, а также волевых качеств, позволяющих сохранить и удержать процесс игры.

Компьютерная игра при соответствующем грамотном подходе помогает детям активизировать процесс обучения в изучении таких специализаций, как история, география, геометрия, английский язык и прочих, позволяя ребенку получить жизненный опыт через игровой сценарий, подвигнув его к творческой сфере и, что важно, к изучению программирования.

Если спросить учащихся, дети ответят, что компьютерные игры способствуют развитию навыков скорости реакции, способности к принятию самостоятельного решения, умению взвешивать и оценивать риск, приобретению настойчивости в достижении целей, развитию внимания к мельчайшим деталям, умению работать в команде.

Касаемо выбора компьютерных игр, специалисты IT-технологий рекомендуют играть в английские версии эпичных игр, стратегий, квестов, аркад и прочих жанров, поскольку эти игры обладают большей атмосферностью, продуманностью и многогранностью. Возвращаясь в их игровой мир, находишь для себя что-то новое. Производители компьютерных игр делают к ним регулярные обновления, дополняя, модифицируя, усложняя структуру и сценарий, характеры и способности героев. И вроде бы история в игре заканчивается, но к ней неизменно следует продолжение.

Однако важен один момент, — чтобы увлекаемость миром компьютерных игр не оторвала ребенка от реальности происходящего. Как и везде, впрочем, важна расстановка приоритетов, не забывая о запланированных в реальности делах и соблюдении несложных правил, помогающих сохранить здоровье подрастающего геймера.

Современная система образования в России требует внедрения информационных технологий в учебно-воспитательный процесс общеобразовательных учреждений. Это дает возможность педагогам использовать в учебном процессе компьютерные игры и программы.

Компьютерные игры являются неотъемлемой частью жизни современных детей, в том числе и детей младшего школьного возраста.

В настоящее время ученые ставят вопрос о возможности применения компьютера в младшем школьном возрасте. Л. Ф. Гарипов, В. В. Утемов отмечают, что основная цель применения компьютера в начальной школе заключается в использовании его в процессе изучения таких школьных дисциплин, как математика, русский язык, естествознание, музыка, изобразительное искусство, а также формирование интереса положительного отношения к компьютеру.

Существует две точки зрения о влиянии компьютерных игр. Одни авторы утверждают, что компьютерные игры обеспечивают качество учебно-воспитательного процесса и образования в целом: способствуют развитию творческого мышления, овладению, ими новыми знаниями, логическими операциями, дают представления о способах манипулирования предметами и символами. Другие же авторы утверждают, что чрезмерное увлечение компьютерными играми негативно сказывается на поведенческой, когнитивной и эмоциональной сферах ребенка.

По характеру и педагогической целесообразности компьютерные игры можно разделить на две группы: развивающие и развлекательные. У каждой из этих групп свои задачи. Меня больше интересуют развивающие игры, которые можно использовать в учебном процессе. Главное отличие развивающих компьютерных игр состоит в том, что в их создании принимают участие не только программисты и разработчики, но и педагоги.

По мнению Н. А. Максимовой и Т. И. Гавриловой, развивающие компьютерные игры не только эмоционально привлекательны для школьников, но и содержательны, а значит, их можно подчинить педагогическим целям.

Развивающие игры подразделяются на несколько видов:

— игры, развивающие логическое мышление и память ребенка;

— игры, улучшающие координацию движений (мелкую моторику рук);

— игры, развивающие навыки счета и чтения;

— игры, развивающие фантазию и объемное восприятие;

— игры, развивающие художественный вкус и музыкальный слух у ребенка.

Опыт школ, в которых на уроках в начальных классах применяются электронные учебные материалы на основе компьютерных игр, демонстрирует ряд положительных тенденций. В первую очередь, это уменьшение количества дидактических затруднений у учащихся, повышение активности и инициативности школьников, положительная динамика мотивации учения, формирование навыков использования новых информационных технологий для самообразования.Особенность методики применения компьютеров состоит в том, что компьютерные средства обучения являются интерактивными, они обладают способностью «откликаться» на действия ученика и учителя, «вступать с ними в диалог». Компьютер может использоваться на всех этапах процесса обучения: при объяснении «введении» нового материала, закреплении, повторении, контроле знаний, умений и навыков. При этом для ребенка он выполняет различные функции: учителя, рабочего инструмента, объекта обучения, игровой обучающей среды.

Для младшего школьного возраста характерны яркость и непосредственность восприятия, легкость вхождения в образы. Дети легко вовлекаются в любую деятельность, особенно игровую. В игровой модели учебного процесса проблемная ситуация проживается участниками в ее игровом воплощении, основу деятельности составляет игровое моделирование, часть деятельности учащихся происходит в условно-игровом

плане. Ученики действуют по игровым правилам. Итоги игры выступают в двойном плане — как игровой и как учебно-познавательный процесс.

1.4. Применение объекта исследования

Рассмотрим некоторые учебные компьютерные игры, которые можно использовать на уроках в школе и вузах.

Морской бой. На квадратах игрового поля располагаются корабли двух игроков: ученика и компьютера. Составляются 25 задач или примеров, при этом задачи — это условные «снаряды». Если ответ задачи совпал с номером квадрата, в котором находится малый корабль «противника», значит, корабль потоплен. Чтобы потопить большой корабль, необходимо «попасть» в него дважды. Выигрывает тот, кто раньше «потопит» корабли противника.

Уроки Кота Леопольда. Использован сюжет известного мультфильма. На игровом поле мышата Толстяк и Гений устроили для Леопольда пять засад (их название «западни»). Нажав на номер засады, в текстовом поле увидим четыре задания. Ответы вводятся с помощью клавиатуры, предварительно поставив в окно курсор. Нажимают на кнопку «Проверить». Появляются мышата и текст их обращения к Леопольду. Программа включается автономно, из презентации гиперссылка переводит к общей папке игры.

Сокровища подземелья. Правила игры. «Нажать на кнопку «Начинаем». На экране появляются сундуки. В них деньги, сокровища или загадки. Наводите курсор на каждый из сундуков и отвечайте на вопросы. При верном ответе сундук открывается. Сундуки надо открывать по порядку: деревянный, оловянный, медный, серебряный, золотой». Игра выполнена по программам математики (1-й класс) и русскому языку. В первой в ответы надо вводить полученные числа, во второй — сначала подобрать к правилам слова, а затем записать подряд номера этих слов и полученное число ввести в ответ.

Электронный букварь. На экране рисунки предметов и буквы русского алфавита. Нажав на один из рисунков, надо выбрать соответствующую букву и нажать на неё. При правильном выборе рисунок исчезает с экрана. Технология поможет малышам освоить русский алфавит. Необходимо

помнить, что на игры должно отводиться не более 15–20 минут урока в зависимости от поставленных учителем учебных задач.

Обобщая все выше сказанное, можем сделать вывод, что обучающие компьютерные игры обогащают учебный процесс, делают его более интересным и увлекательным, помогают развиваться каждому ученику.

1.5. Анализ влияния компьютерных игр

Компьютерные игры, игры онлайн, которые уже стали развиваться как самостоятельная индустрия XXI века, становятся неотъемлемой частью досуга людей всех возрастов. На сегодняшний день существует достаточное количество исследований, раскрывающих значимость игрового компонента в формировании культуры личности ребенка. В современном мире, мире информационных технологий, — на первый план выходят компьютерные игры, которые подменяют детям реальный мир виртуальным. Возникает проблема, как защитить своих детей от игровой зависимости? Есть ли способы конструктивно разрешить данную проблему, не потеряв своих детей в пространстве Сети?

Игровая компьютерная зависимость — это особо новая форма психологической зависимости, качественно отличающаяся от других форм зависимостей выходом в виртуальный мир, механизмами формирования и особенностями протекания на разных этапах возрастного развития. Привлекательность компьютерных игр заключается в том, что для многих детей и подростков они заменяют традиционный досуг, игры со сверстниками и даже общение с родителями.

Дети и подростки — это наиболее восприимчивая группа, которая чаще подвергается угрозе игровой компьютерной зависимости. Доказательством послужило исследование (мониторинг) общероссийского масштаба, которое показало, что численность детей и подростков в возрасте от 7-14 лет с различной степенью выраженности игровой компьютерной зависимости колеблется от 2% до 10% человек. Следовательно, проблема игровой компьютерной зависимости у детей и подростков актуальна, следовательно, необходимо разрабатывать профилактические, диагностические и коррекционно-развивающие мероприятия для устранения проблемы. Следовательно, надо обучать детей программированию с азов, с раннего

детского возраста, помогая, показывая грани плохого и хорошего в компьютерном мире, мире игр онлайн.

Многие исследователи (Д. Б. Эльконин, С. Л. Рубинштейн, Л. С. Выготский и др.) традиционную игру трактуют, как свободно протекающую активность, в рамках которой отсутствует материальный интерес, но присутствует собственный смысл, правила. Игра по своему содержанию символична и воссоздает социальные отношения и реализует передачу социального опыта. Ребенок в процессе традиционной игры выражают в ней свою индивидуальность, способности, возможности и желания.

Компьютерная игра имеет специфические особенности. Главное отличие от традиционной игры в том, что она происходит в условно-наглядной реальности (виртуальной). Игровая ситуация, по мнению Е. О. Смирнова, позволяет пробовать любое действо, в отношении другого объекта, при этом всегда можно вернуться, «переиграть» действие. Таким образом, компьютерные игры различаются по содержанию и форме взаимодействия с другими участниками игры, объектами виртуальной реальности и соответственно по-разному могут влиять на психическое состояние ребенка.

Предпосылкой возникновения и развития игровой компьютерной зависимости служит игровая компьютерная активность, которая отражает степень увлеченности компьютерными играми, имеет различную интенсивность, положительно связана с частотой и продолжительностью игры. Обнаружить игровую компьютерную активность можно по следующим признакам: ежедневная игра за компьютером продолжительностью от 4 и более часов (М. Гриффитс). Анализ исследований проблемы влияния игровой компьютерной зависимости на личность ребенка показал, что в психолого-педагогической науке существует, как минимум, две точки зрения. Первая точка зрения, которой придерживается И. Г. Белавена, В. Д. Горский и др., рассматривает компьютерные игры, как способ развития и преобразования деятельности человека за счет возникновения новых навыков, операций и

способов выполнения действий, новых видов деятельности. Таким образом, компьютерные игры вызывают положительный развивающий эффект на формирование личности ребенка.

Вторая позиция рассматривает компьютерные игры через значительное количество негативных последствий (например, низкий уровень коммуникативных способностей и др.). Частые сеансы компьютерной игры способствуют вырабатыванию привычки к виртуальному миру, следствием чего становится неадекватное восприятие реального мира ребенком. Разрушительные последствия игровой компьютерной зависимости для формирования личности ребенка и его социальной адаптации стали причиной для признания ее одной из форм поведенческой аддикции.

Анализ исследования проблемы игровой компьютерной зависимости позволил, в след за А. В. Гришиной, выделить психологические факторы развития игровой компьютерной зависимости:

1. Индивидуально-личностный фактор, который характеризуется низким уровнем умственных способностей, завышенной самооценкой, низким уровнем коммуникативных способностей, высоким уровнем тревожности, слабой волевой саморегуляцией и высокой склонностью к риску.

2. Социально-психологические факторы: проблемы во взаимоотношениях с родителями, проблемы, связанные с общением со сверстниками (статус или место в группе).

Игровая компьютерная зависимость в психологическом аспекте приводит к негативным изменениям в эмоционально-волевой сфере ребенка, а также в усилении чувства социального отчуждения и стремлении играть вновь и вновь.

Позитивное влияние компьютерных игр в том, что у игроков более развито логическое мышление, эмоциональная устойчивость. В отличие от своих сверстников, которые не играют в компьютерные игры, играющие дети

обладают умением конкурировать, у таких детей высокий уровень развития творческих способностей (И. В. Бурлаков, О. К. Тихомиров). Игровая компьютерная деятельность является средством компенсации недостатка любви, внимания, положительных эмоций и общения. Это и является одной из причин, почему у детей появляется игровая компьютерная активность. Как считают многие исследователи, компьютерные игры способны выполнять функцию психологической разгрузки, таким образом, они дают возможность выразить вытесняемые агрессию, чувство гнева и злости (И. А. Васильева, Е. М. Осипова).

По результатам своего исследования, я пришел к выводу о том, что компьютерные игры — это возможность моментально переместиться в другую реальность, либо оказаться одновременно в двух параллельных мирах. Привлекательно в них то, что их содержание наполнено яркими образами, завораживающими сюжетами, сверхвозможностями. Существование в виртуальном мире приводит к тому, что начинается раздвоение целостности личности. Ведь в компьютерной игре мы сами выбираем себе героя, к примеру, наделяем его всевозможными способностями, т.е. всем тем, чего нам не хватает в реальной жизни.

Для большинства детей компьютерная игра — это возможность по-другому, более позитивно принять себя. По мнению И. В. Бурмистрова, мотивом игровой деятельности является желание примерить иную идентичность, близкую к идеальному «Я». Итак, положительное влияние игр на личность — возможность принять себя через игрового персонажа. Происходит идентификация с героем, следовательно, появляется вера и в свои возможности в реальной жизни. Если вовремя начать контролировать игровую компьютерную активность, знать интересы своего ребенка, то можно сформировать представление о компьютерной игре, как о развивающей игре. Именно такой вид компьютерной игры позволяет ребенку развивать свои интеллектуальные и творческие способности.

1.6. Обзор аналогичных информационно-вычислительных систем

1.6.1. CodeMonkey

CodeMonkey это браузерная игра, которая помогает обучить новичков писать различные программы при помощи CoffeeScript. Это легкий язык, который основывается на Ruby и Python и преобразуется в JavaScript. Чаще всего CoffeeScript использует верстальщики для веб-приложений. Суть игры в том, что пользователь программой управляет поведением главного героя – обезьяной, для того что бы она смогла собирать все бананы. Эта игра предназначена больше для детей, но даже взрослым она пойдет по душе.

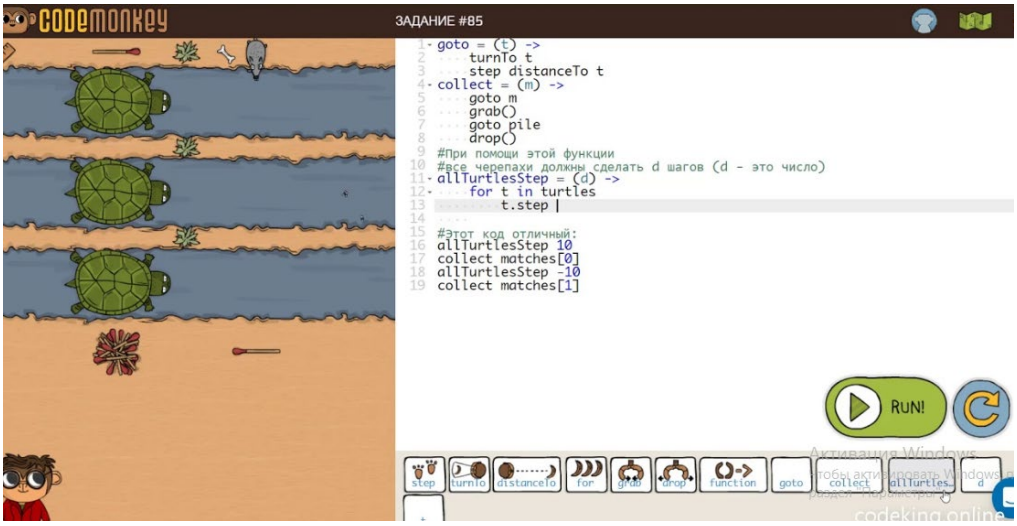

Рисунок 1.1 – Игра «CodeMonkey»

1.6.2. CodeCombat

CodeCombat это онлайн игра позволяющая изучить в увлекательной форме такие языки программирования как JavaScript, Python, HTML, CSS. В начале игры пользователь будет перемещать своего героя по разным локациям игры, используя лишь основные команды. По мере прохождения уровней команды будут усложнятся.

Игра поддерживает небсколько режимов, можно играть в одиночную игру или участвовать в многопользовательских играх – все это позволяет в интересной форме отработать полученные ранее навыки программирования.

Рисунок 1.2 – Игра «CodeCombat»

1.6.3. Code.org

Это программное средство создано в большинстве случаев для детей, но, как уверяют его разработчики, игра подойдет и взрослым, подросткам и детям. Обучение начинается с самых базовых аспектов — с обучения пользования компьютерной мыши. В игре используется scratch- визуальная событийно-ориентированная среда программирования, состоящий из блоков действий. Играя пользователь изучит азы программирования и узнать, как алгоритм превращается в код JavaScript и как создастся компьютернпя игру из блоков действий.

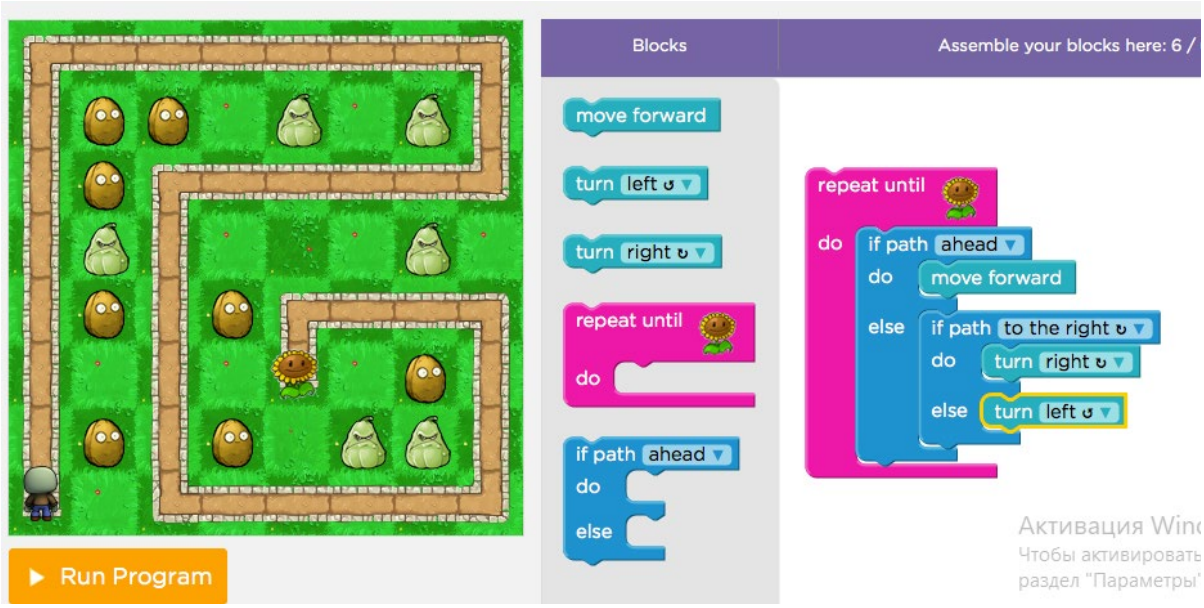

Рисунок 1.3 – Игра «Code.org»

1.6.4. Ruby Warrior

Эта игра позволяет изучить язык программирования Ruby. Ее цель управлять главным героем – рыцарем с помощью кода и попутно выполнять легкие игровые задачи:

– убивать с монстров
– проходить подземелья

В этом продукте присутствуют как простые, так и трудные задания, соответствующие игрока, уровню владения языком. В процессе игры в Ruby Warrior, можно улучшить знания циклов и знание условий конструкций, создания и вызова методов.

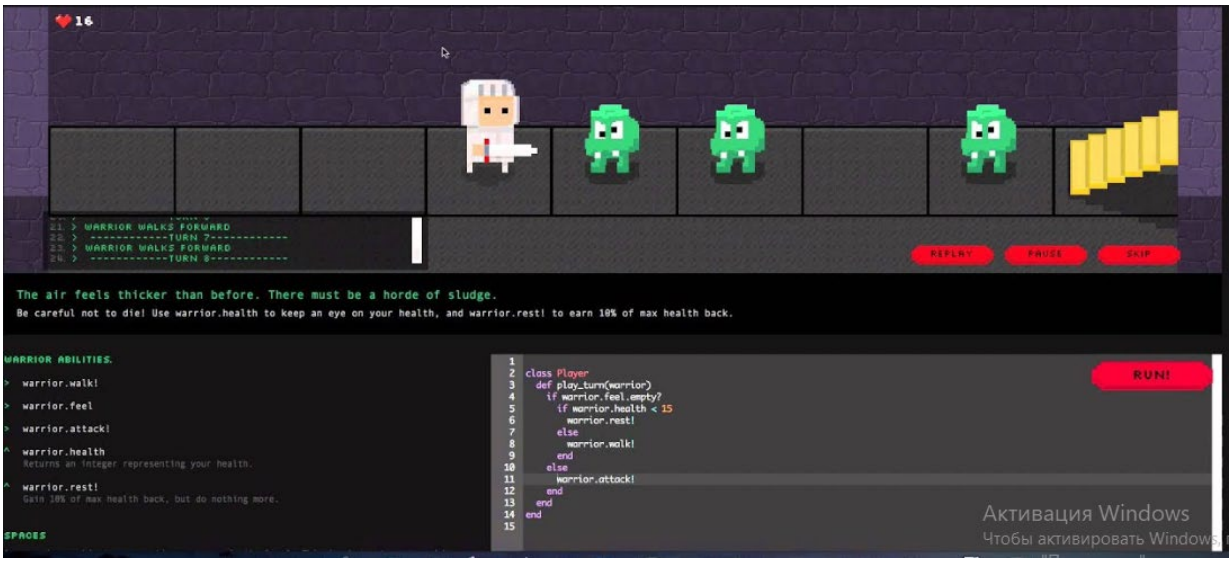

Рисунок 1.4 – Игра «Ruby Warrior»

1.6.5. Elevator Saga

Эта игра помогает пользователям оценить их знания в JavaScript во процессе решения задач, которые связанные с транспортировкой людей и перемещением лифта наиболее оптимальным способом. Здесь представлены здания из нескольких этажей и лифтов, перевозящие людей.

Необходимо запрограммировать перемещение лифтов так, чтобы перевезти нужное количество пассажиров за определенный срок. Начинается задача с транспортировки 15 человек, потом задачи становятся тяжелее. В процессе игры и по мере прохождения уровней будет увеличиваться этажность и поголовье лифтов, а условия становятся все строже.

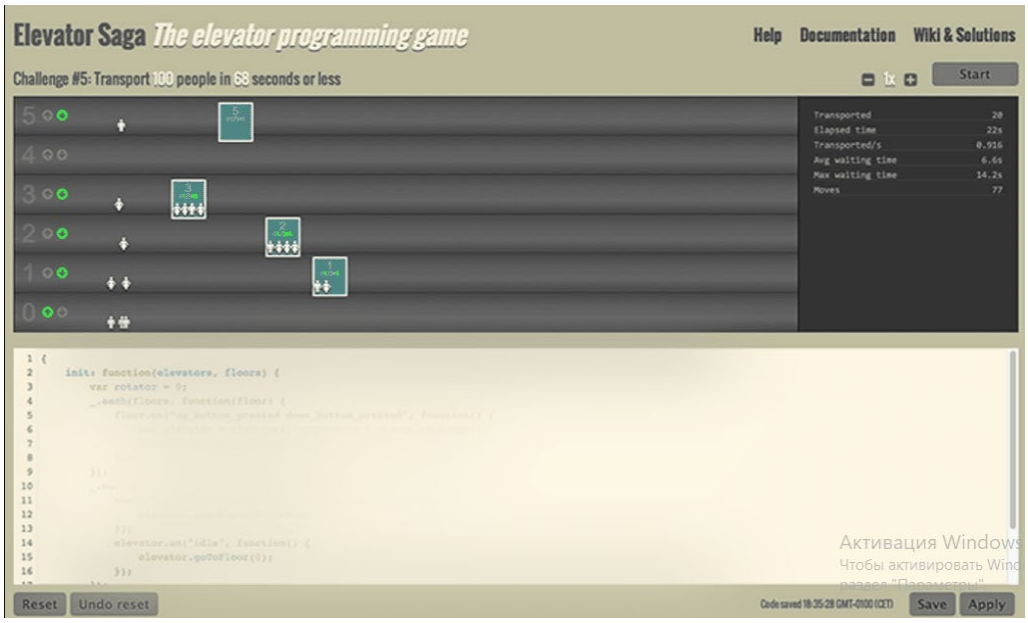

Рисунок 1.5 – Игра «Elevator Saga»

Таблица 1. Достоинства и недостатки аналогов

Аналог	Достоинства	Недостатки	+	-
Code Monkey	Браузерная игра	Поддерживает один язык	3	3
	Возможность самому сделать игру	Примитивная графика		
	Малый порог вхождения	Простые задачи		
Code Combat	Браузерная игра	Примитивная графика	5	1
	Поддерживает несколько яз. программирования			
	Сложные уровни			
	Создавалась компанией			
	Малый порог вхождения			
Code.org	Браузерная игра	Не поддерживает языки программирования	2	3
	Малый порог вхождения	Простые задачи		
		Примитивная графика		
Ruby Warrior	Браузерная игра	Поддерживает один язык	3	3
	Сложные уровни	Примитивная графика		
	Создавалась компанией	Высокий порог вхождения		
Elevator Saga	Браузерная игра	Поддерживает один язык	3	3
	Сложные уровни	Примитивная графика		
	Создавалась компанией	Высокий порог вхождения		

1.7. Выводы

Исходя из данных, проведенных исследовательским «Левада-центром», россияне все больше предпочитают отдавать своих детей в сферу обучения IT технологий. В наше время любой успешный бизнес, любая организация, включая муниципальные образования — пусть и незначительная, но IT-компания. Возьмем штаты крупных банковских структур, где трудятся тысячи программистов. Нефтегазовые компании, заводы, консалтинговые фирмы и другие бизнес-сферы требуют привлечения опытных специалистов IT технологий. Знание основ программирования пригодится, даже если ребенок предпочтет выбор другой профессии.

На мой взгляд, обучение программированию учит основам системности, развивает логику, что свойственно как алгоритмам, так и людям их пишущим, поскольку дети лучше воспринимают информацию. Обучение программированию детей способствует активному формированию универсальных навыков.

Наиболее подходящим возрастом, гарантирующим овладение навыком программированию, считают детский возраст. Обучать ребенка программированию «с пеленок» не следует, вначале дети должны получить комплекс определенного опыта, постигнув традиционные образовательные азы. Однако, ребенку, познакомившемуся с основами программирования еще в дошкольном возрасте, уже будет легче принять новые знания в будущем. Вполне вероятно, подрастающего специалиста вопрос с выбором профессии в дальнейшем беспокоить уже не будет.

Плюсом к раннему изучению программирования можно назвать стимуляцию развития мозга. Вычислительные алгоритмы помогаю развить логическое мышление ребенка, которое считается основой всей образовательной сферы. На западе уже научно доказали, — дети, с раннего возраста постигающие азы программирования, в дальнейшем показывают

выдающийся успех в любых точных науках, будь то высшая математика или физика. Обучаясь программирования, ребенок получит уникальные знания, он поймет и осознает сложность математических вычислительных законов.

С автоматизационным процессом, происходящим во всем мире, к обществу пришли новые возможности. Специалистов, которых, так или иначе, коснулась область программирования, только в одной Америке за последние пятнадцать лет стало больше на двадцать процентов. За последние пять лет в США были подготовлены около восьмидесяти тысяч учителей программирования. В шесть раз увеличилось количество учеников, которые серьезным образом изучают программирование и даже сдают по нему экзамен. Восемьдесят процентов учеников, изучающих программирование в начальной и средней школах, продолжают его изучение, поступая в высшие учебные заведения.

Когда дети выходят играть во двор, родители учат их поведению с незнакомцами. Но в онлайн-игры ребенок может играть свободно, даже не подозревая о возрасте игрока. В офлайн детей учат, как правильно переходить дорогу. В мире онлайн ребенок может перейти по незнакомой ссылке, хакеры могут удаленно подключиться к компьютеру. Поведению в мире онлайн чаще всего ребенок учится сам, к сожалению, такой подход не дает всего пакета знаний поведения в мире онлайн.

К сожалению, государственные образовательные программы далеко не всегда поспевают за трендовыми новинками мира IT-технологий. Я же считаю программирование — фундаментальный, а не факультативный предмет и оно уже кардинальным образом изменило нашу повседневность, поменяв привычки поездки в авто, заботу о здоровье, сферу общения с друзьями, обучение в школах и т.д. Системность познания азов программирования поможет сформировать у ребенка комплекс привычек, которые в дальнейшем пригодятся и в повседневной обычной жизни. В наше время — это такой же

равнозначный предмет, как и русский язык, литература, математика, алгебра, геометрия, английский язык, физика.

ГЛАВА 2. ОПИСАНИЕ РАЗРАБОТАННОГО ПРИЛОЖЕНИЯ

2.1. Общая информация

Разработанное приложение – это компьютерная игра «Cyber Cod» от 3-ого лица, в стиле Cyberpunk. Цель которой является автоматизировать процесс обучения программированию в школах и вузах, и так же снизить нагрузку с, и без того тяжелой деятельности учителей.

«Cyber Cod» - это игра, направленная на практическое применение программирования, по нарастающей сложности. В ней задача игрока, писать код для движения главного героя и движений посторонних предметов, собирая бонусы для прохождения уровней. Пользователь должен будет написать такой код, что бы запустив программу, были собраны все бонусы, не задев препятствий и не допустив никаких остановок. В противном случае уровень начнется с самого начала.

Визуальный и музыкальный стиль игры Cyber Cod (рисунок 2.1) был выбран не с проста, так как моя игра связана с программированием, ей нужно было задать соответственный технологический стиль. И я пришел к выбору совместить жанр «Cyberpunk» и атмосферу фильма «Tron». Cyberpunk – очень популярный жанр научной фантастики, отражающий технологический прогресс человечества в эпоху компьютеров (рисунок 2.2). Фильм «Tron» так же был создан в жанре Cyberpunk, но его графика и визуальная часть была настолько интересной своими неоновыми узорами, что породила отдельный подстиль. Рисунок 2.1.

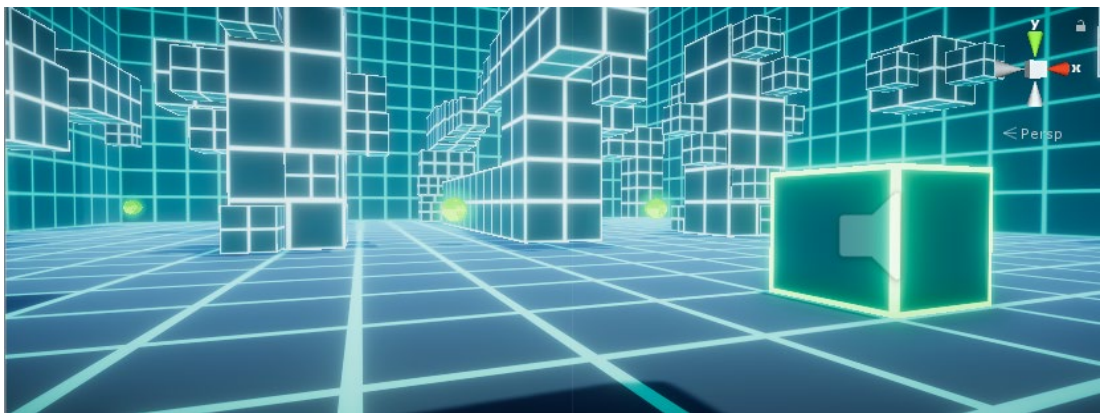

Рисунок 2.1 – Стиль игры «Cyber Code»

Рисунок 2.2 – Стиль Cyberpunk

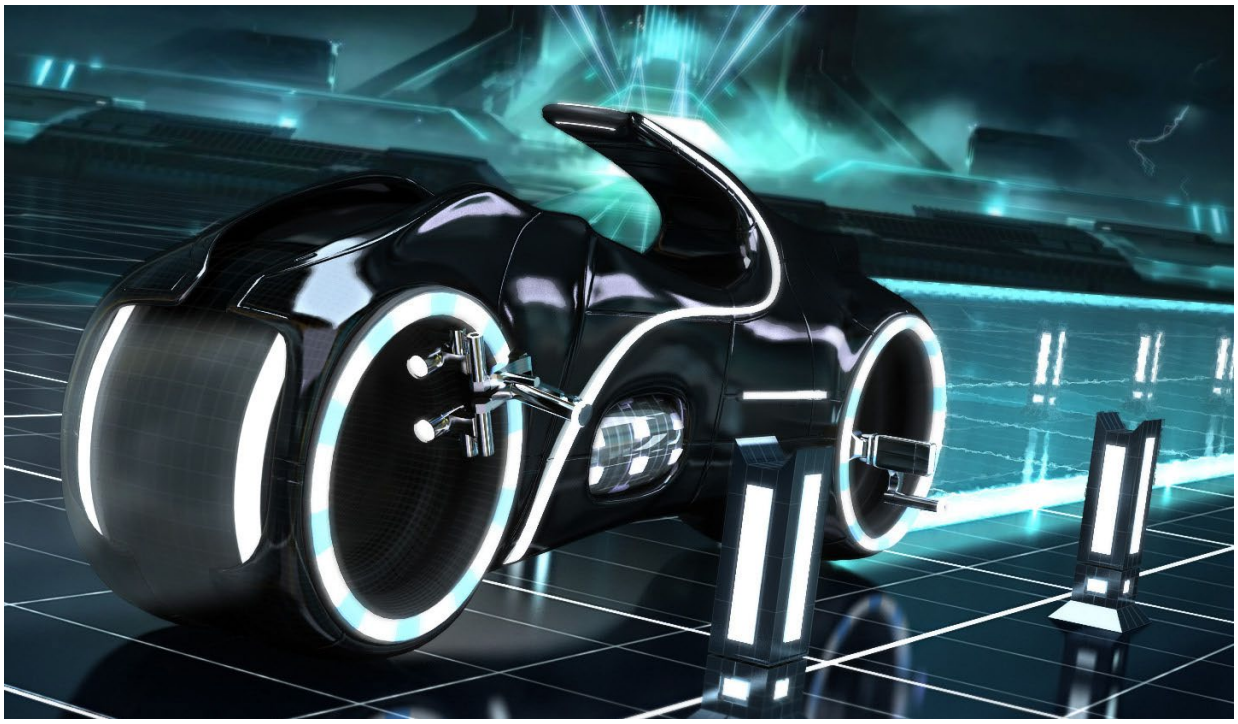

Рисунок 2.3 – Фильм «Tron»

2.2. Задачи

Для реализации своих задумок я составил для себя перечень главных задач. И в процессе проектирования ими руководствовался:

- Реализовать способность передвигаться в игре с помощью ввода команд в текстовом редакторе.
- Создать объекты, которые нужно собрать, для прохождения уровня путем написания программы.
- Сделать задания по нарастающей сложности.
- Сделать красивую графику, соответствующую потребностям молодежи

1. Реализовать способность передвигаться в игре с помощью ввода команд в текстовом редакторе.

Такой способ управления позволяет освоить программирование, понять, как работают его базовые механизмы. Так же этот способ максимально приближен к работе программиста и заставляет игрока практиковаться в написании кода.

2. Создать объекты, которые нужно собрать, для прохождения уровня путем написания программы.

Для того что бы заинтересовать пользователя игрой и мотивировать к написанию кода, следует создать в сцене объекты, коснувшись которых пользователь перейдет на уровень сложности выше. Повышение сложности задач, встающих перед игроком, является одним из главных правил в педагогике для достижения нужных результатов в учебе. Подробнее об этом в следующей задаче.

3. Сделать задания по нарастающей сложности.

Одним из приемов успешного обучения является подбор для игрока не одного, а небольшого ряда заданий нарастающей сложности. Первое задание

выбирается несложным для того, чтобы пользователь, смог решить его и почувствовать себя знающими и опытными. Далее следуют задания по сложнее. Например, можно использовать специальные сдвоенные задания: первое задание готовит базу для решения последующей, более сложной задачи. Проходя уровни все сложнее и сложнее, игрок будет совершенствоваться.

4. Сделать красивую графику, соответствующую потребностям молодежи

Сейчас графику в играх тяжело отличить от реального мира, поэтому они пользуются популярностью у подрастающего поколения. Но игры с такой детализацией очень трудозатратны и требует слаженной работы сразу нескольких специалистов, таких как: художник, 3d-модельер, аниматор, левел-дизайнер, геймдизайнер. Но так как над проектом работаю я один (студент), выполняя одновременно работы целой команды профессионалов. Думаю, было бы правильным найти золотую середину между трудозатратами и качеством детализации. Т.е. сделать графику не сильно простой, но и сильно сложной. В этом есть и свои плюсы. С увеличением графики растет и технические требования к игре, но в моей игре требования будут не высокие, что увеличит число потенциальных пользователей.

2.3. Обзор информационной системы

Запустив приложение, пользователь в первую очередь увидит главное меню игры. Где пока что есть две рабочих кнопки («PLAY» и «QUIT»). Рисунок 2.4. Шрифт выбран был соответствующий технологическому стилю игры – «Electronic Highway sing»

Рисунок 2.4 – Главное меню игры

После нажатия кнопки «PLAY» загрузится первый уровень и с этого момента начинается игра со всеми ее функциями. Рисунок 2.5.

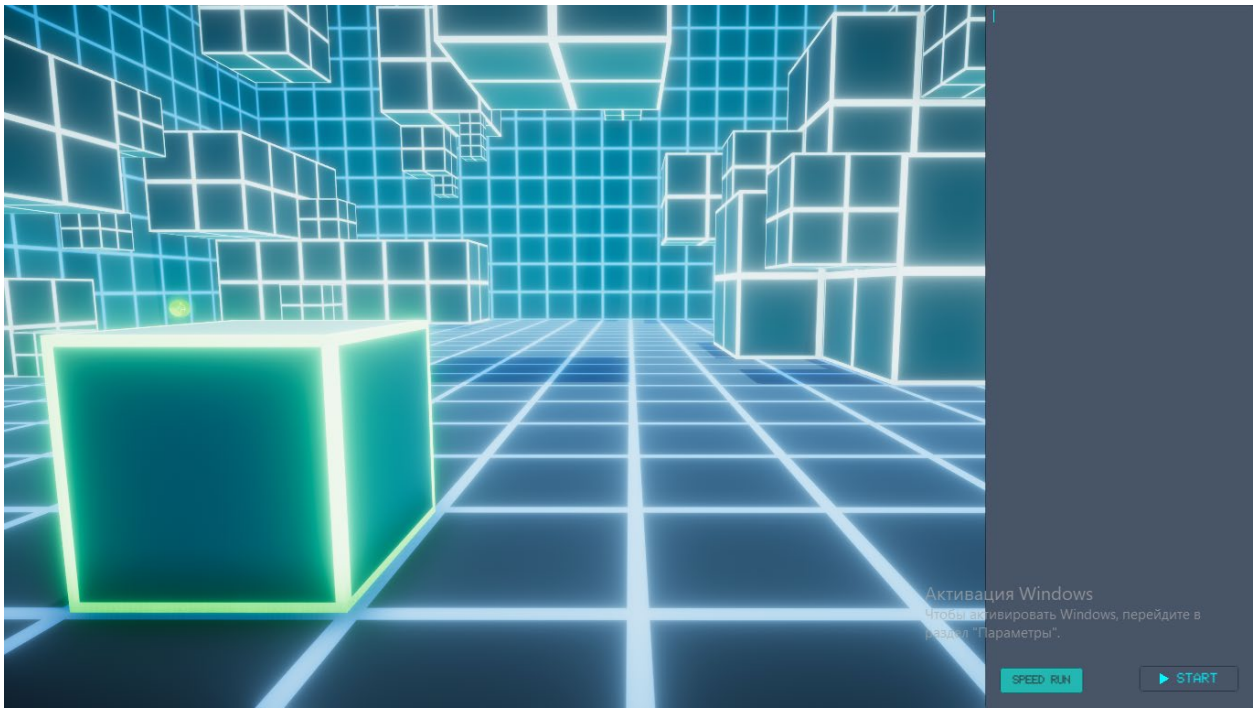

Рисунок 2.5 – Первый уровень игры

Справа находится текстовое поле (Рисунок 2.6) где пользователь пишет любую из существующих команд и кнопкой «START» запускает ее. Пока выполняется код движения главного героя, кнопка исчезает, до того момента пока не выполнятся все команды. Я это сделал для того что бы в процессе выполнения нельзя было запускать какие-либо команды. Если же игрок напишет команду, которую не существует. То код не сработает.

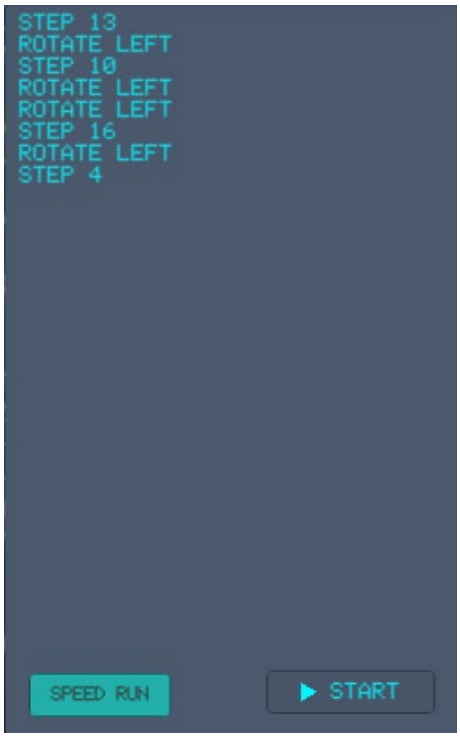

Рисунок 2.6 – Текстовое поле

Квадрат с зеленым неоном это аватар игрока, которым он будет управлять написанием кода, вводя его в текстовое поле справа. Этот объект может:

- Ходить вперед на любое количество клеток («STEP N», где N – количество клеток)
- Повернуться вправо и влево («ROTATE RIGHT» и «ROTATE LEFT»)
- Прыгнуть на две клетки вперед («JUMP»)
- Управлять платформами («DOOR.OPEN/CLOSE» и «DOOR.UP/DOWN»)

Для того что бы пройти уровень и перейти к следующему, игроку необходимо двигаясь, собрать все объекты-бонусы на уровне, не разу не остановившись. Выглядят эти объекты так – рисунок 2.7.

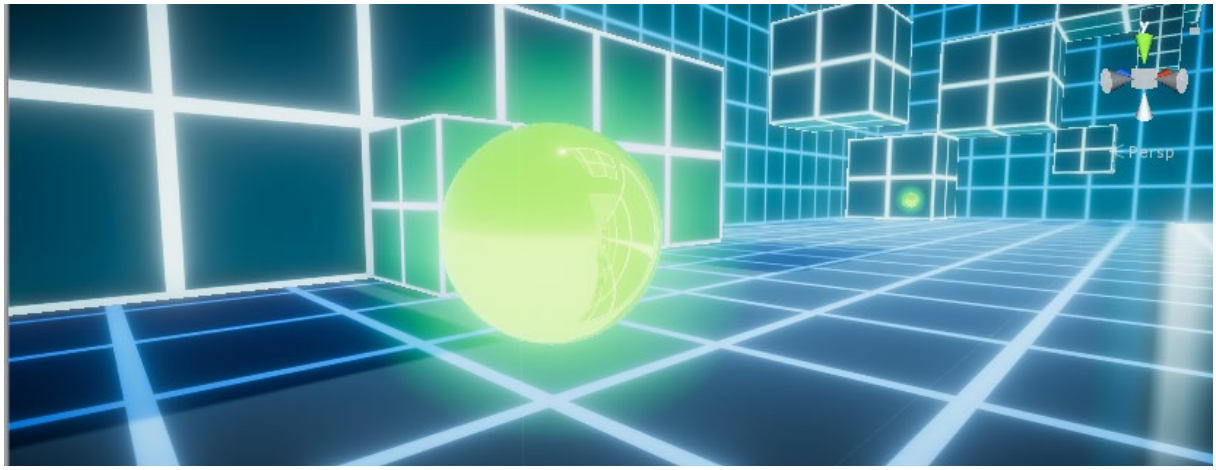

Рисунок 2.7 – Подбираемые объекты для прохождения уровня

Что бы собрать первый бонус игроку необходимо проложить следующий путь, рисунок 2.8. И реализовать его в текстовом поле игры, написав программу для движения, рисунок 2.9. Это программа должна, задействовать часть существующих команд и будет выглядеть следующим образом:

— STEP 13
— ROTATE LEFT
— STEP 10

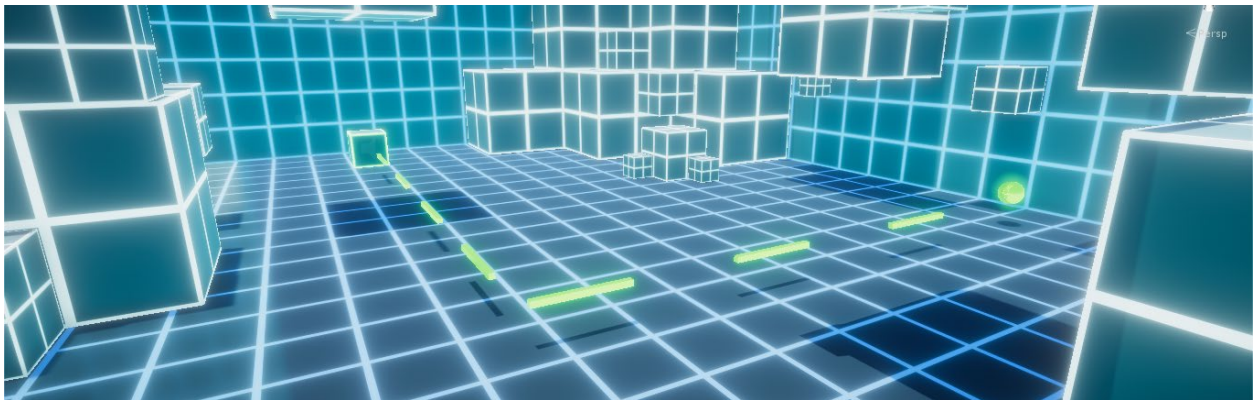

Рисунок 2.8 – Путь к бонусу

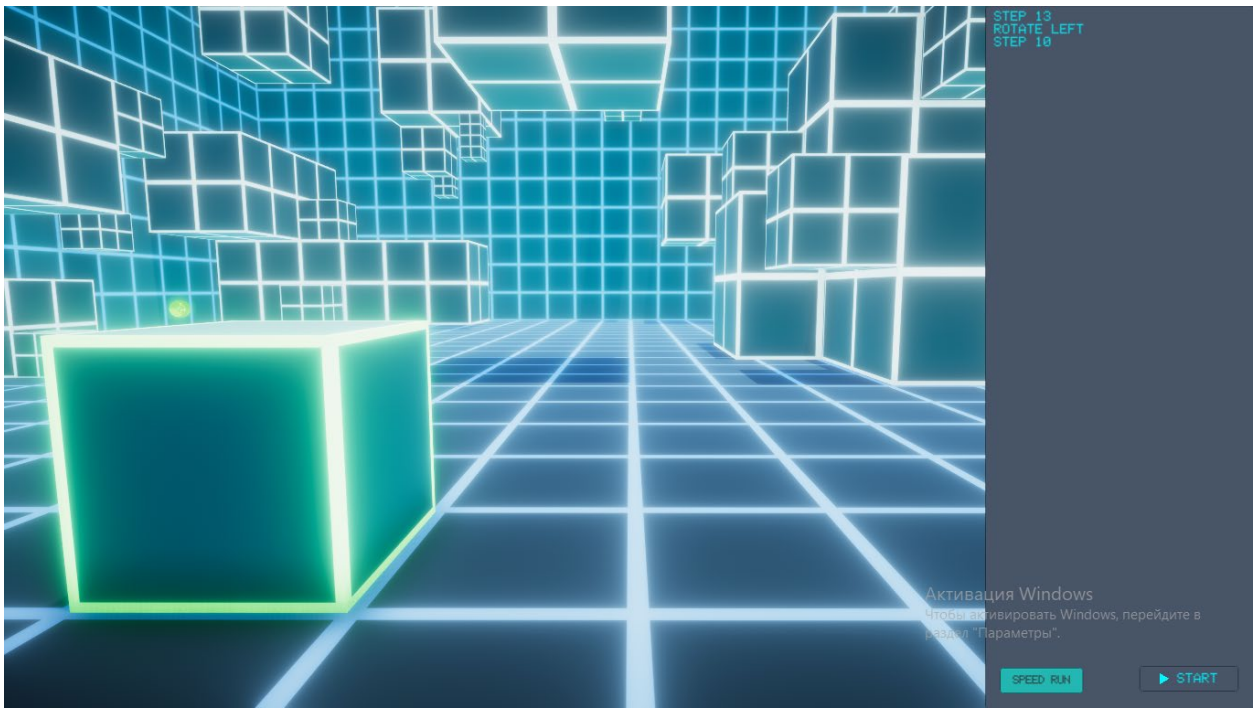

Рисунок 2.9 – Программа для движения к бонусу

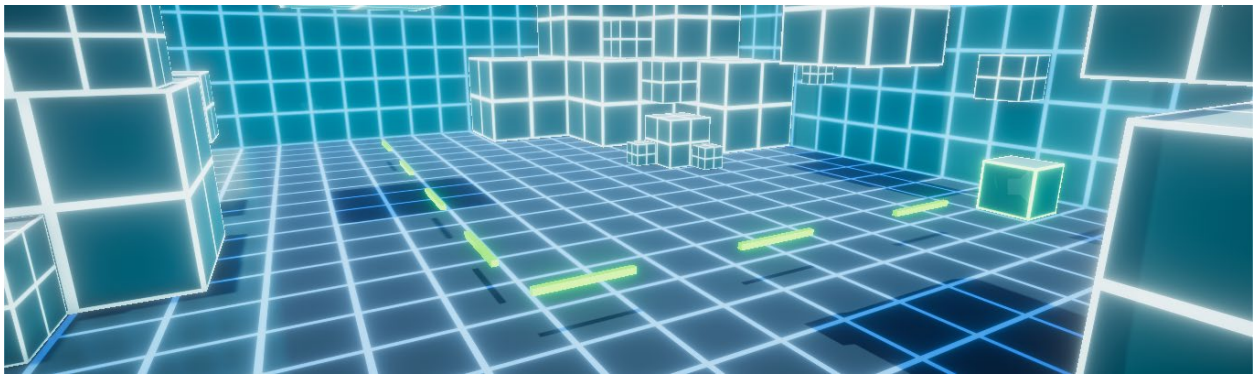

Рисунок 2.10 – Завершение пути

Как только пользователь соберет все бонусы в сцене, скрипт воспроизведет анимацию «Level completed», рисунок 2.11 и запустит следующий уровень.

Рисунок 2.11 – Анимация «Level completed»

Теперь рассмотрим уровень (Рисунок 2.12), в котором используются команда «JUMP». Это команда предназначена для перепрыгивания препятствий, перепрыгивания через обрыв и сбора бонусов которые находиться высоко. Прыжок осуществляется по направлению камеры главного героя.

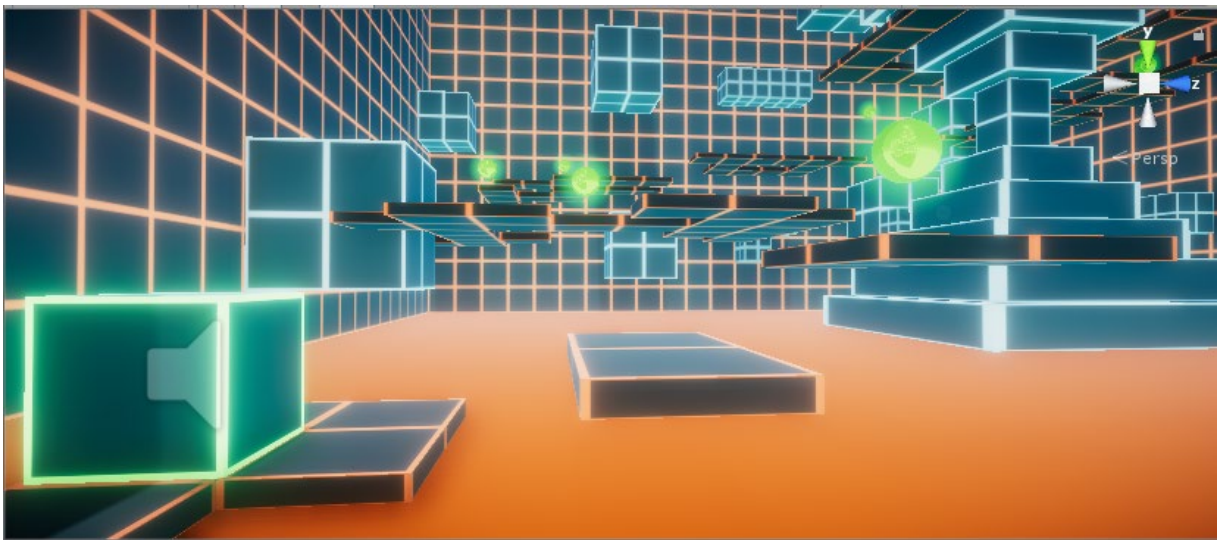

Рисунок 2.12 – Уровень с новой командой «JUMP»

Для того что бы собрать первый бонус и не упасть в лаву, игроку необходимо сделать один шаг, и два раза прыгнуть. Код в текстовом поле

будет выглядеть как на рисунке 2.13. После запуска, которого пользователь благополучно соберет бонус, рисунок 2.14.

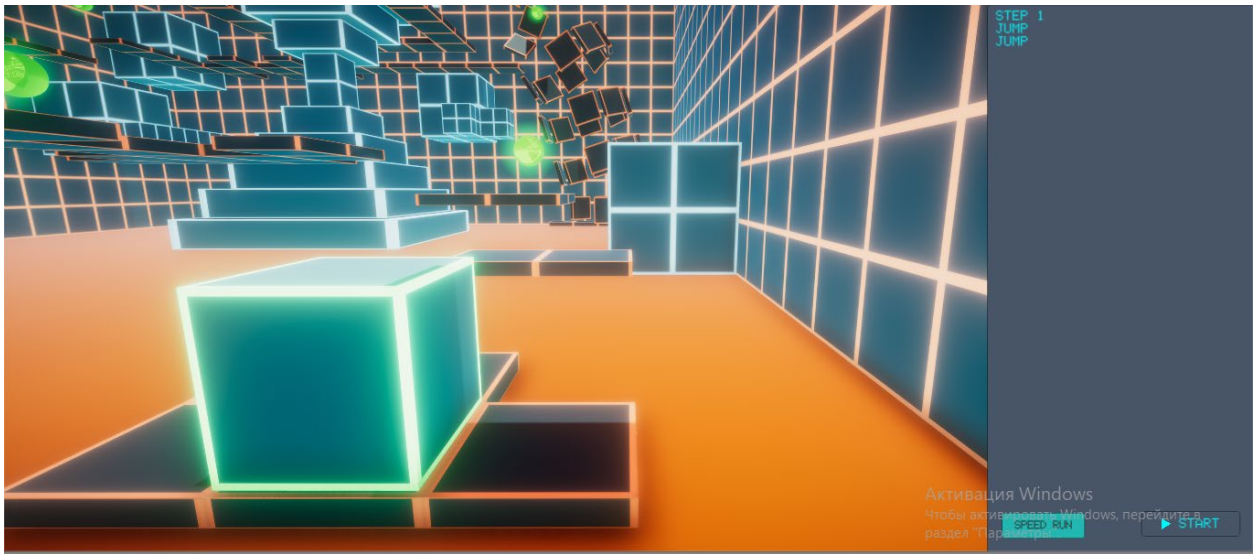

Рисунок 2.13 – Код для сбора первого бонуса

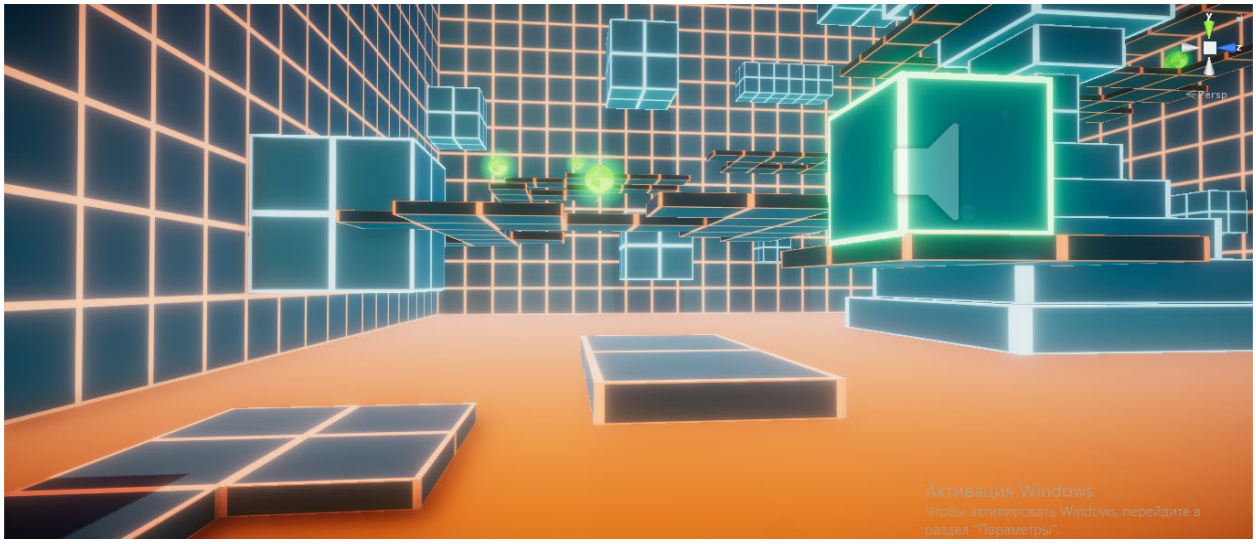

Рисунок 2.14 – Объект после выполнения кода

Перейду к последнему уровню где доступны команды «DOOR.OPEN/CLOSE» и «DOOR.UP/DOWN». Эти команды двигают платформы. Возьмем, например, команду «DOOR.UP/DOWN». Встав на соответствующую платформу. И запустив этот код. Платформа поднимет игрока до нужной высоты как на рисунке 2.14 и 2.15

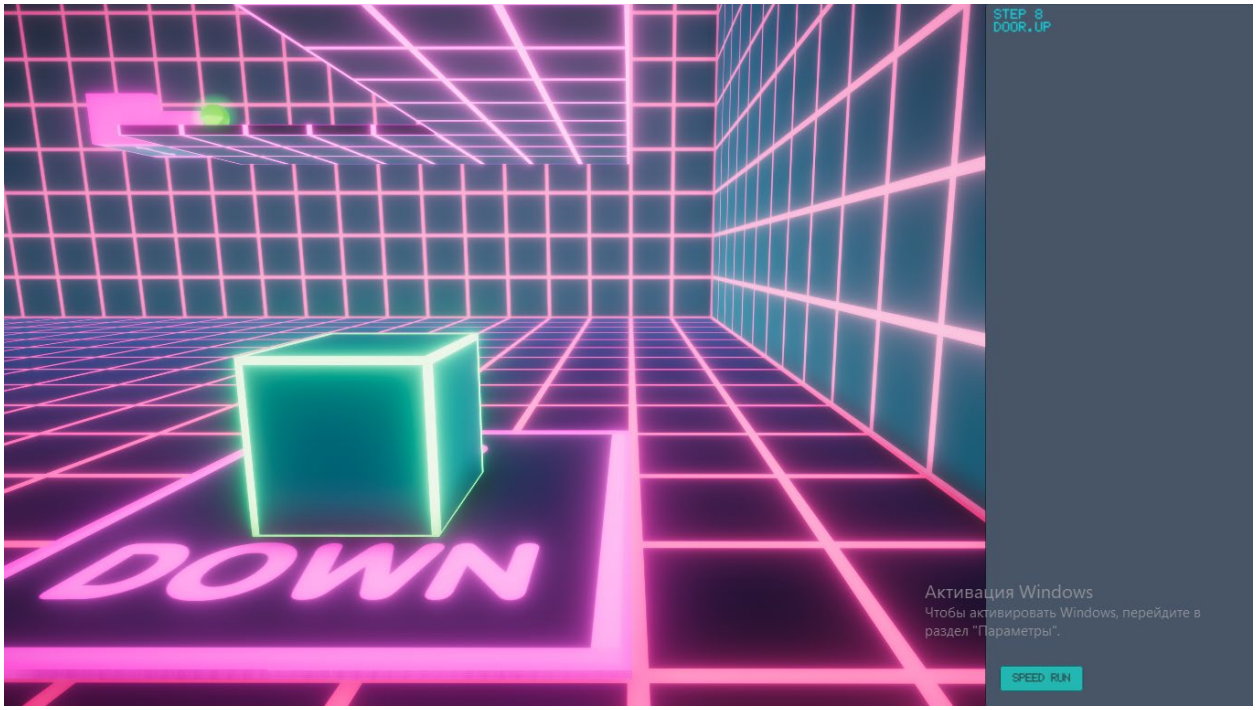

Рисунок 2.14 – До применения команды «DOOR.UP»

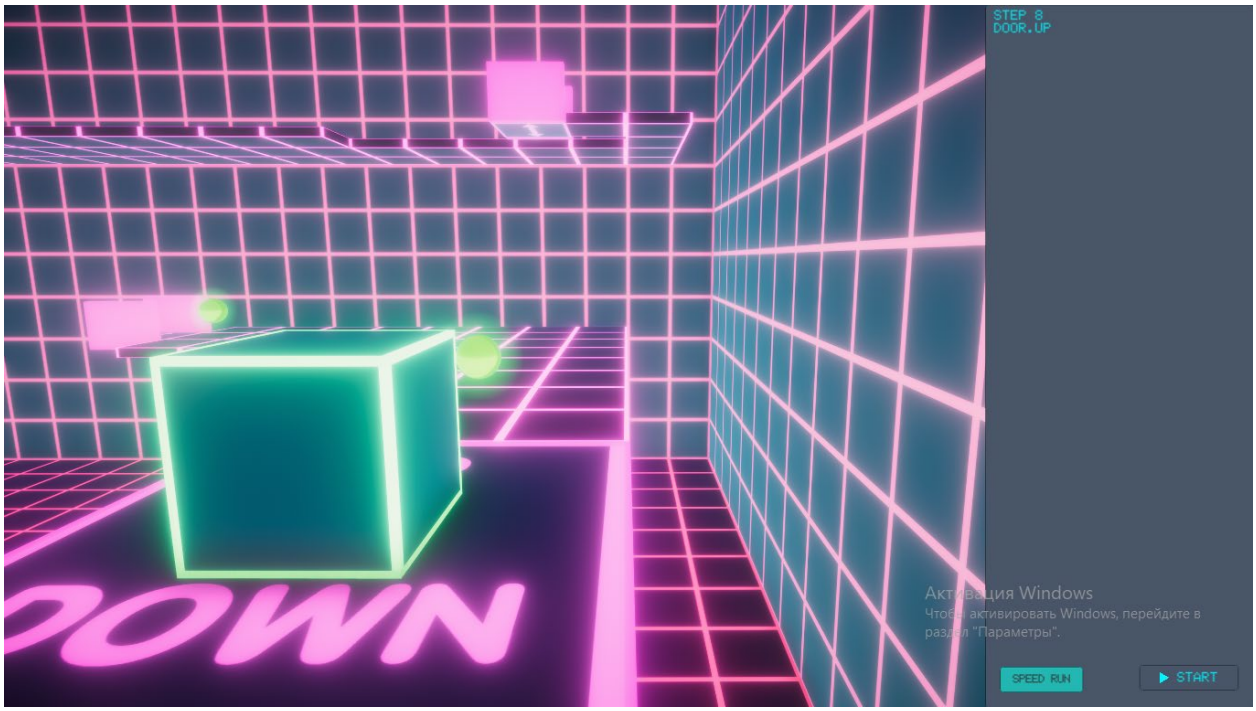

Рисунок 2.15 – После применения команды «DOOR.UP»

Последняя команда, которую рассмотрим «DOOR.OPEN/CLOSE». Она двигает платформы вправо и влево как на рисунке 2.16 и 2.17

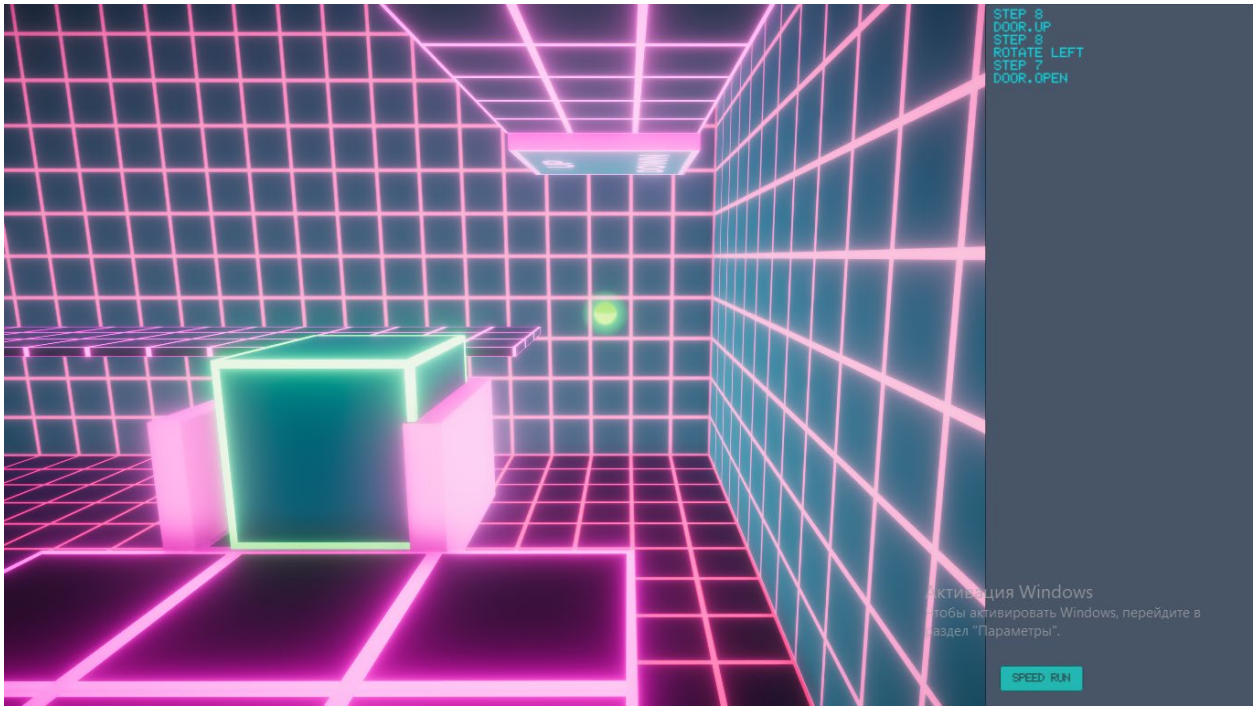

Рисунок 2.16 – До применения команды «DOOR.OPEN»

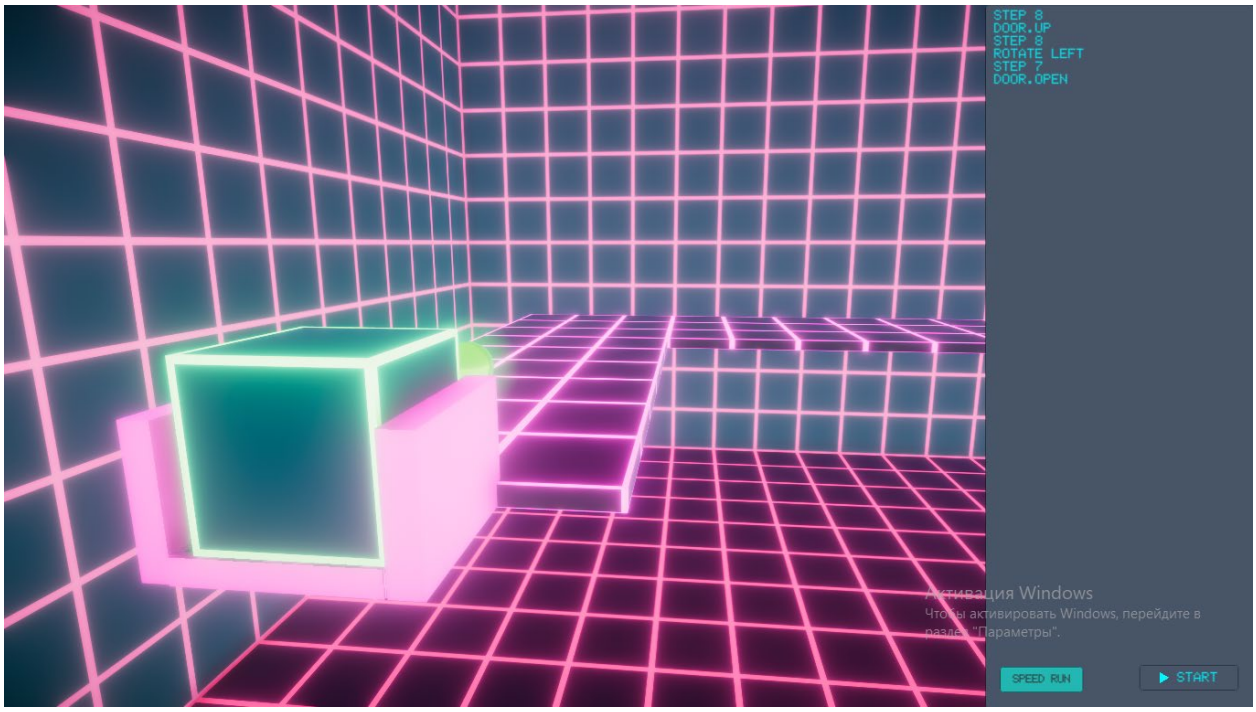

Рисунок 2.17 – После применения команды «DOOR.OPEN»

На этом заканчивается команды в игре. Теперь перейдем к описанию алгоритмов информационной системы.

2.4. Описание схем и UML-диаграмм информационной системы

2.4.1 Блок-схема

В этой схеме на рисунке 2.18 описаны все алгоритмы и процессы игры. Например, здесь можно увидеть, что «Game Over» происходит, когда выполняется условие касания с препятствием, после чего начинается загрузка уровня под индексом X. А так как этот индекс не менялся, это действие можно расценивать как перезагрузка уровня. «Level Completed» в свою очередь происходит если в сцене уже не осталось бонусов которых собрал игрок и после чего индекс уровня увеличивается на один.

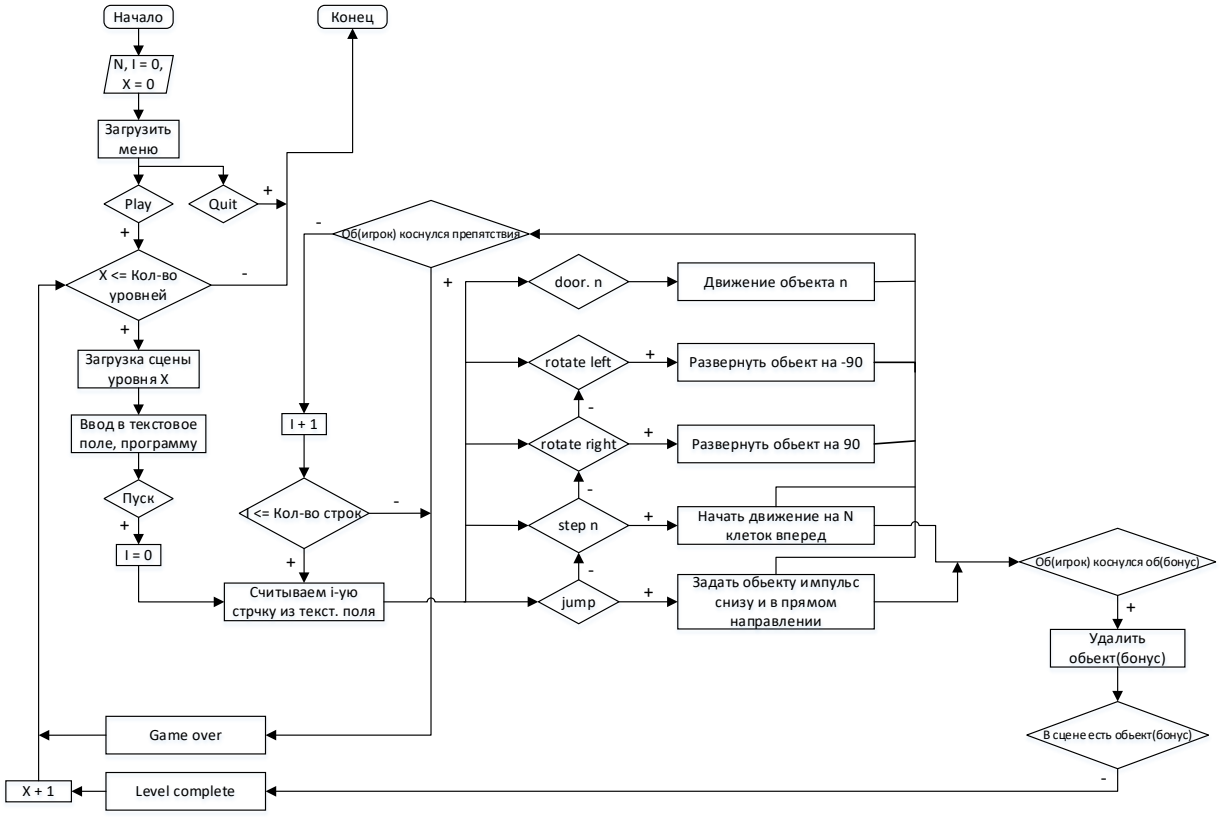

Рисунок 2.18 – Блок схема алгоритма игры

Все эти действия схема наглядно показывает, как и в какой последовательности работает игра. Часто бывает так что надо быстро вникнуть в работу информационной системы. Блок схема как раз ускоряет этот

процесс. И даже человек без технического образования сможет понять сложные алгоритмы системы.

2.4.2. Диаграмма состояний

Диаграмма состояний рисунок 2.19, показывает, как объект переходит из одного состояния в другое. И служит для моделирования динамических концепций системы. Округленные прямоугольники представляют собой состояния, через которые проходит игрок в течение пользования ИС. Стрелками изображаются переходы между состояниями, которые вызваны выполнением методов описываемого диаграммой объекта.

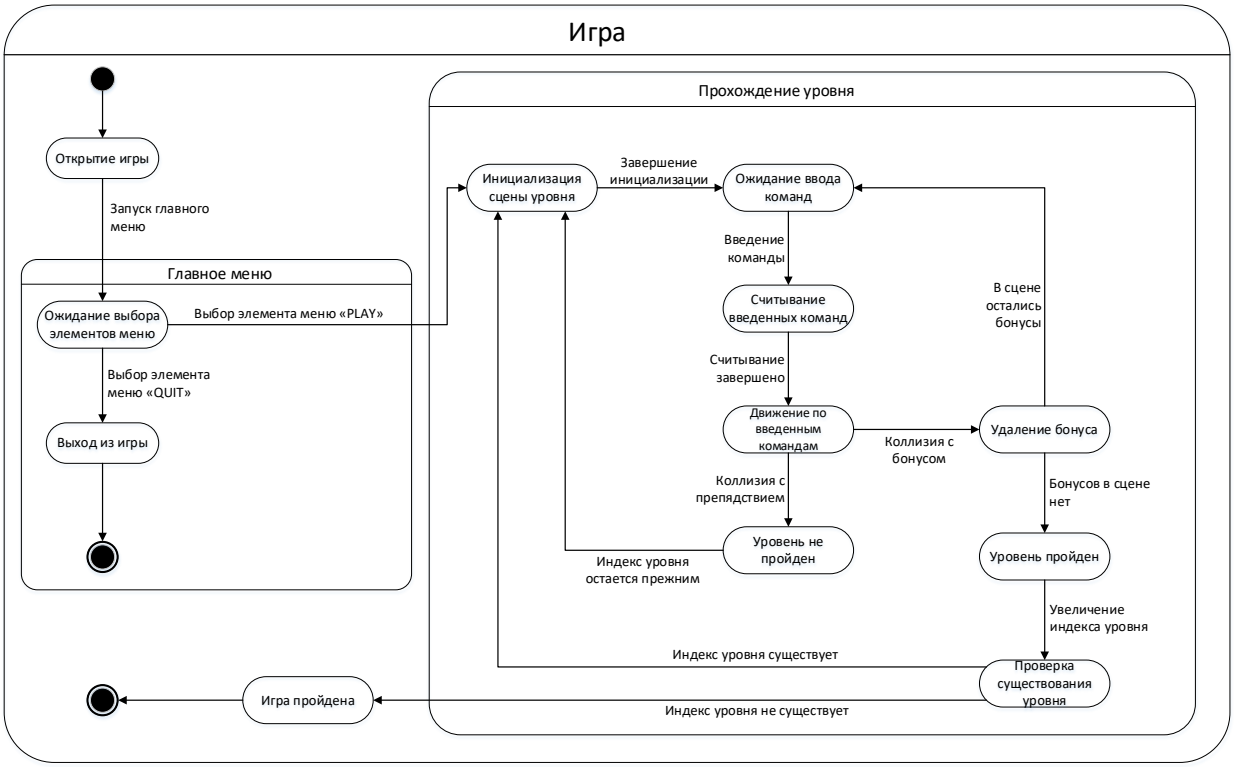

Рисунок 2.19 – UML диаграмма состояний

В этой диаграмме можно увидеть, что в данной ИС существует два главных состояния, это «Главное меню» и «Прохождение уровня». Можно заметить, что состояние «Прохождение уровня» вызывается только из подсостояния «Ожидания выбора элемента меню». Это означает, что игра не начнется если пользователь не выберет элемент «PLAY» в главном меню.

Перейдя в состояние «Прохождение уровня». Первым что срабатывает это подсостояние «Инициализация уровня». В этот момент загружаются и

сцена, объекты в сцене, пользовательский интерфейс и т.д. После, наступает подсостояние «Ожидание ввода команд». Игра ждет пока пользователь введет команды в текстовое поле.

Введя команды, и после чего побывав в подсостоянии «Считывание введённых команд». Игра начнет двигать объекты, а это уже ключевое подсостяние, так как от этого зависит коснемся бонуса или препятствия? Если же коллизия произошла с препятствием, то тут все просто. Переходим в подсосотояние «Уровень не пройден». Что бы оказаться в подсостоянии «Уровень пройден» необходимо собрать все объекты-бонусы. Это значит придется несколько раз побывать в подсостоянии «Удаление бонуса». Так как в сцене много несколько таких объектов.

2.4.3. Диаграмма последовательностей

Диаграмма последовательностей относится к UML диаграммам, описывающим поведение системы и рассматривает взаимодействие объектов во времени. Другими словами, диаграмма последовательностей показывает временные особенности передачи и приема сообщений объектами.

В данной диаграмме на рисунке 2.20 я выделил 7 главных объектов:

- Игрок (Пользователь который играет или собирается играть)
- Приложение (Файл который запускает игрок)
- Движок игры (Программное обеспечение игры)
- Главное меню (Пользовательский интерфейс для управления режимами игры)
- Скрипт (язык сценариев, описывающий действия выполняемых системой)
- Сцена уровня (Игровое пространство с объектами внутри)
- Текстовое поле (Поле куда пользователь пишет программу)

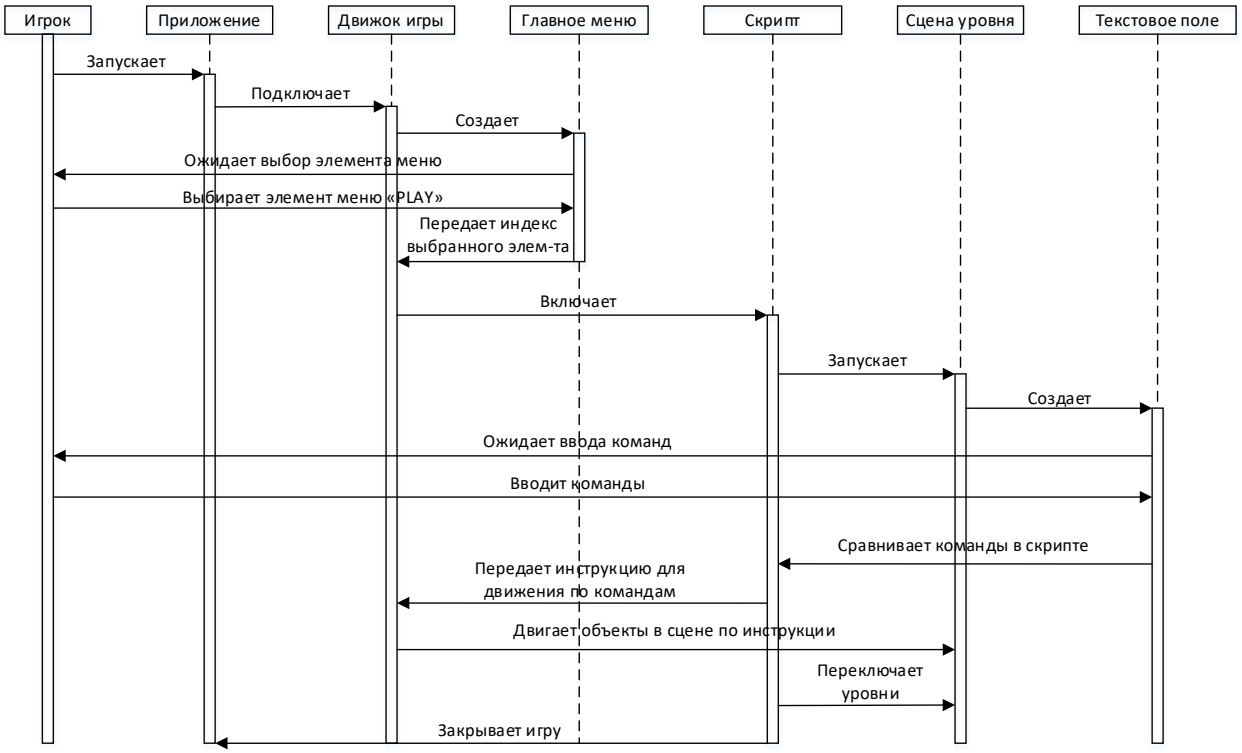

Рисунок 2.20 – Диаграмма последовательностей

2.4.4. Диаграмма вариантов использования

Диаграмма вариантов использования на рисунке 2.21, является представлением системы в процессе ее работы. Данная диаграмма состоит из актеров, вариантов использования и отношений между ними.

Суть данной диаграммы состоит в проектируемой системе представления в виде множества актеров, взаимодействующих с системой с помощью так называемых вариантов использования.

Рисунок 2.21 – Диаграмма последовательностей

В этой диаграмме, разработанной под мою информационную систему, можно заметить всего двух участников: Пользователь и Игровой движок. Пользователь выполняет чаще всего процессы выбора действий, таких как:

- Выбор элементов меню
- Выбор команд для ввода в текстовое поле
- Выбор момент времени для нажатия на кнопки

– Выбор момент времени для запуска приложения

Игровой движок выполняет процессы автоматизации:

– Автоматизирует переход между уровнями
– Автоматизирует движение главного героя
– Автоматизирует удаление объектов
– Автоматизирует отрисовку уровня и элементов пользовательского интерфейса

Как только пользователь запустил игру, этот процесс автоматически подключит процессы «Загрузку главного меню» и «Запуск фоновой музыки» через компонент <<include>>. Если позже игрок выбрал элемент меню «Play», этот процесс так же через <<include>> включит процесс «Отрисовка уровня».

Включение (include) в языке UML — это разновидность отношения зависимости между процессами, выполнение которого, приводит к включению другого.

Отношение расширения (extend) определяет взаимосвязь базового варианта использования с другим вариантом использования, функциональное поведение которого задействуется базовым не всегда, а только при выполнении дополнительных условий.

2.5. Выводы

В данной главе я описал алгоритмы, стилистику, геймплей проекта. Рассмотрел поставленные задачи и выполнил их в полном объеме:

- Реализовал способность передвигаться в игре с помощью ввода команд в текстовом редакторе.
- Создал объекты, которые нужно собрать, для прохождения уровня путем написания программы.
- Сделал задания по нарастающей сложности.
- Сделал красивую графику, соответствующую потребностям молодежи

В этом проекте для разработки и создания игры был выбран язык программирования C#.

Разработка велась на популярном и обладающим широким функционалом, движке Unity в силу своей простоты.

Игра не лишена минусов и не является идеальной. Но я считаю это дело времени. Следует поработать детальнее с подбираемыми объектами и сделать их модель интереснее. Добавить больше уровней для прохождения и больше команд что должно благополучно повлиять на обучение. И возможно поработать над оптимизацией, что бы внедрение прошло успешно как можно в больших учебных заведениях.

На мой взгляд игра получилась современной, благодаря чему привлечет молодое поколение, простой и понятной в управлении и подойдет не только детям, но и взрослым, желающим в простой игровой манере закрепить знания языка.

ГЛАВА 3. РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА

3.1. Создание игровой сцены

Для создания игровой сцены необходимо, открыть игровой движок Unity. И во вкладке Project (Проект) в папке Assets (Асеты), создать папки Scenes, Material, Script как на рис 3.1. В них будут располагаться основные элементы игры.

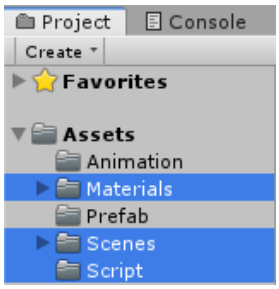

Рисунок 3.1 – Создание папок

В папке Scenes разместятся все созданные сцены уровней (один уровень равен одной сцене) (Рис. 3.2).

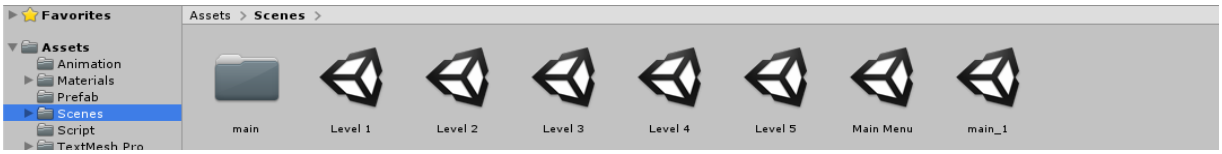

Рисунок 3.2 – Папка «Scenes»

В папке Material располагаются все звуки, шрифты, текстуры, карты для текстур, картинки, skybox'ы (Рис. 3.3).

Рисунок 3.3 – Папка «Materials»

Папка Script. В ней будут храниться файлы со скриптами (Рис. 3.4)

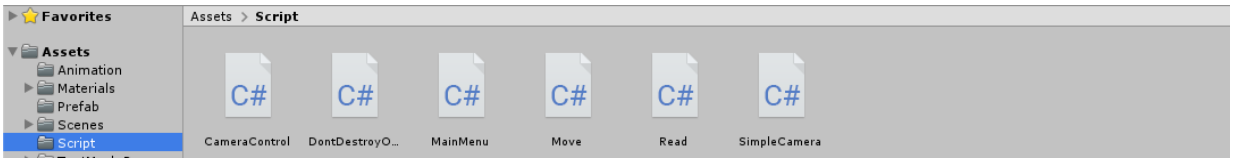

Рисунок 3.4 – Папка «Script»

Теперь можно приступать к созданию сцены. В папке Scenes необходимо нажать правую кнопку мыши Create > Scene. В сцене нужно создать игровое пространство, где будут происходить все действия. Create > 3D Object > Quad. Как на рис. 3.5.

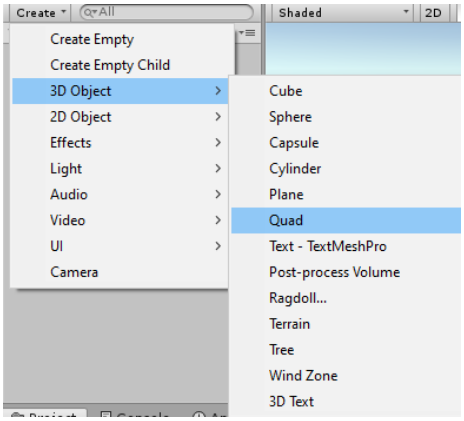

Рисунок 3.5 – Создание плоскости

Quad – это плоскость, которая с одной стороны имеет текстуру (Рис. 3.6), а с другой стороны, прозрачна (Рис. 3.7). Это позволяет не отрисовывать невидимую сторону плоскости. Что повышает производительность.

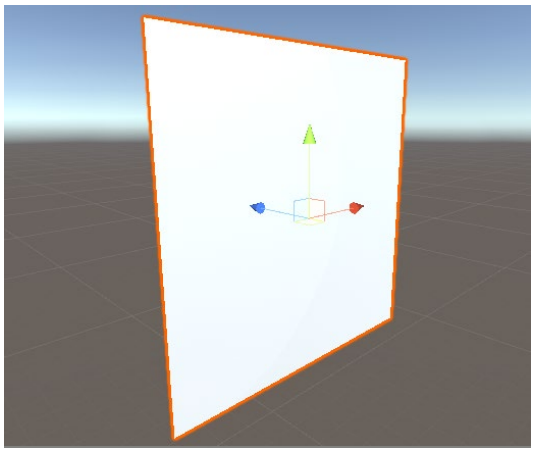

Рисунок 3.6 – Плоскость, вид спереди

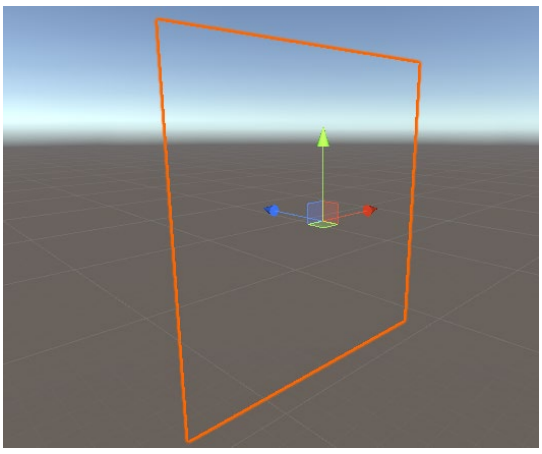

Рисунок 3.7 – Плоскость, вид сзади

Для большей атмосферности я создал SkyBox. В папке с материалами нужно нажимать ПКМ > Create > Material. Далее в самом материале изменить его шейдер (Shader) на Skybox > Procedural. И настроить его по-своему усмотрению как на рис. 3.8.

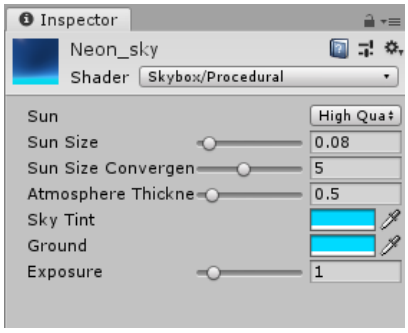

Рисунок 3.8 – Настройка скайбокса

По итогу, после применения нового SkyBox'а сцена немного изменилась и стала интереснее. Рисунок 3.9 и 3.10.

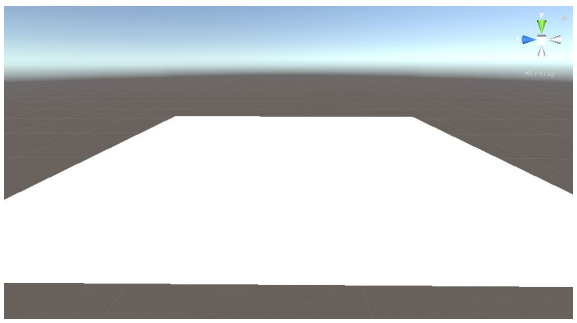

Рисунок 3.9 – Скайбокс до настройки

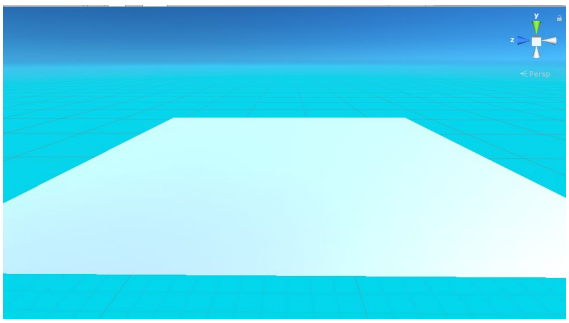

Рисунок 3.10 – Скайбокс после настройки

Осталось наложить текстуру на игровое пространство. Для этого в Photoshop'е необходимо создать две картинки текстуры как на рисунке 3.11 и 3.12.

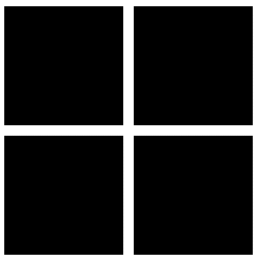

Рисунок 3.11

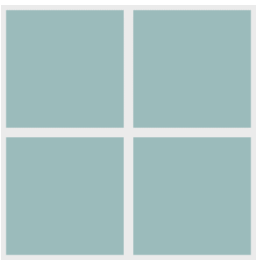

Рисунок 3.12

Что бы создать текстуру, в папке Material, в инспекторе созданной текстуры нужно включить галочку на Emission (Излучение). Перекинуть два рисунка в Albedo и Color и изменить показатель «Tiling», так как он увеличивает повтор рисунка на текстуре. Рисунок 3.13.

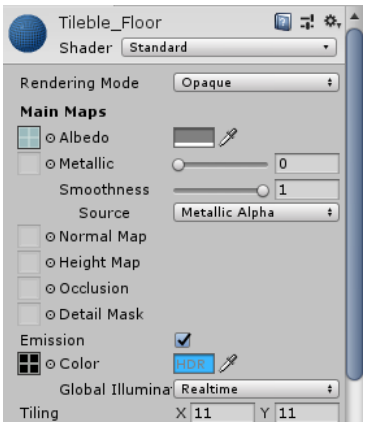

Рисунок 3.13 – Настройка текстуры

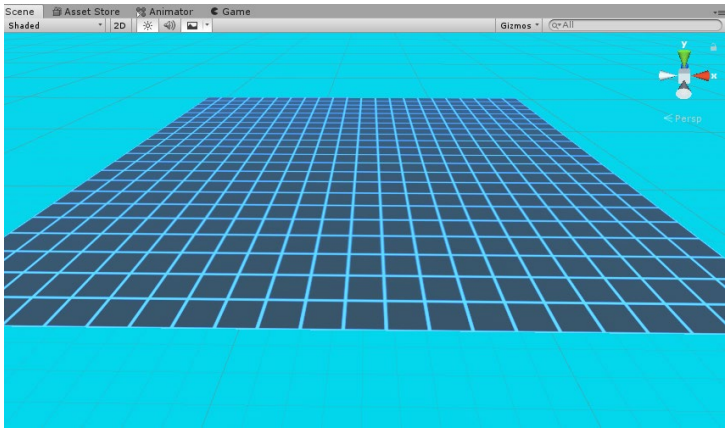

Рисунок 3.14 – Плоскость с текстурой

Первый показатель (Albedo) отвечает за оттенок всего рисунка. А второй показатель (Emission) изменяет цвет белого, в рисунке текстуры. Тем самым имитируя излучение конкретных узоров.

3.2. Постобработка

Следующий шагом это постобработка. Постобработка – это полноэкранные фильтры и эффекты применяемые к изображению камеры, до того, как картинка появиться на экране. Это может радикально улучшить визуальное представление моего приложения с небольшим временем настройки.

Для этого в иерархии (Hierarchy) нужно выбирать камеру. В инспекторе (Inspector) нажимать «Add Component» и добавить «Post-process Layer» и «Post-process Volume» как на рисунке 3.15.

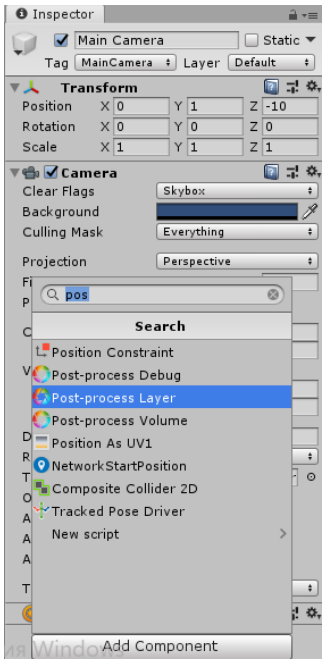

Рисунок 3.15 –Добавление компонентов

Создав файл постобработки в папке «Material». Нажимать ПКМ > Create > Post-process profile. В этом файле будут храниться все настройки эффектов постобработки. В инспекторе (Inspector) этого файла нажимаю «Add effect». Добавляю следующие эффекты и настраиваю их по своим предпочтениям:

- Ambient Occlusion
- Auto Exposure
- Bloom
- Color Gradient
- Vignette

В инспектор камеры, в компонент «Post-process Volume», в элемент «Profile», перетащить файл постобработки и включаю элемент «Global». В компоненте «Post-process Layer», в элементе «Layer» из выпадающего списка выбираю «Everything». Рисунок 3.16 и 3.17.

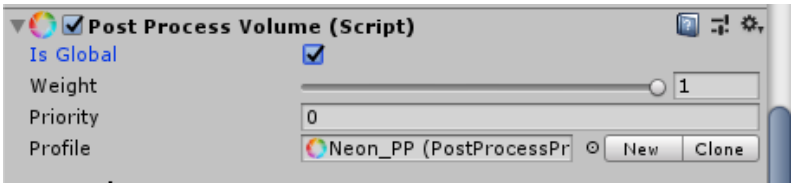

Рисунок 3.16 – Настройка Post-process Volume

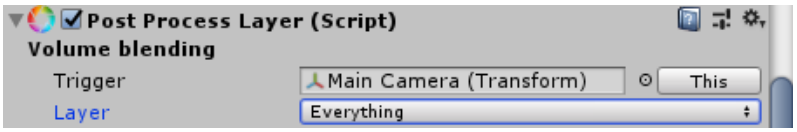

Рисунок 3.17 – Настройка Post-process Layer

Следующим действием добавить эффект отражения в игровом пространстве. Для этого, Нажимать в иерархии (Hierarchy) ПКМ > Light > Reflection Probe. Выбирать мод - «Edit bounding volume», Type – «Realtime». И растянуть его по всему пространству игрового поля. Рисунок 3.18 и 3.19.

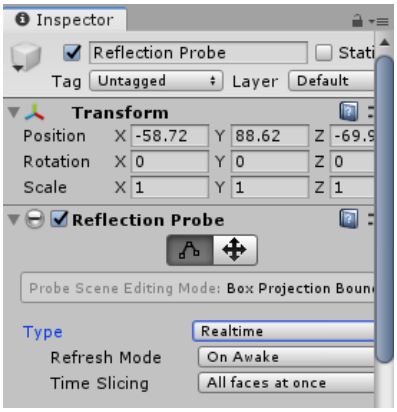

Рисунок 3.18 – Reflection Probe

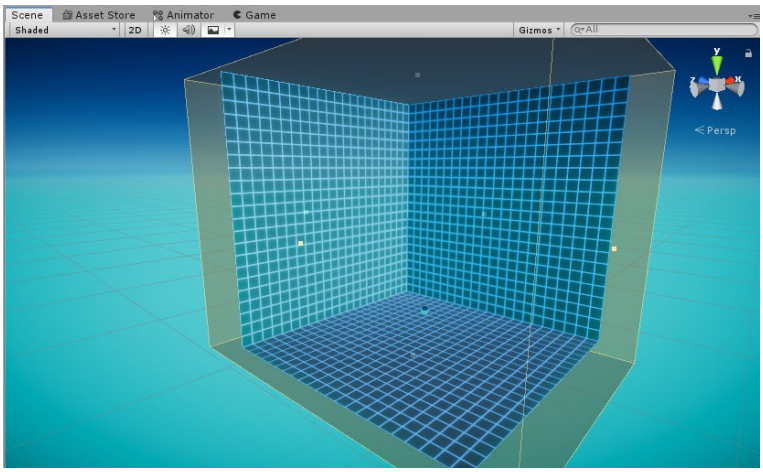

Рисунок 3.19 – Создания области Reflection Probe

Для сравнения продемонстрировав сцену до постобработки и после. Картинка на рисунке 3.20 стала гораздо мягче, появился эффект неонового излучения. Рисунок 3.20 и 3.21

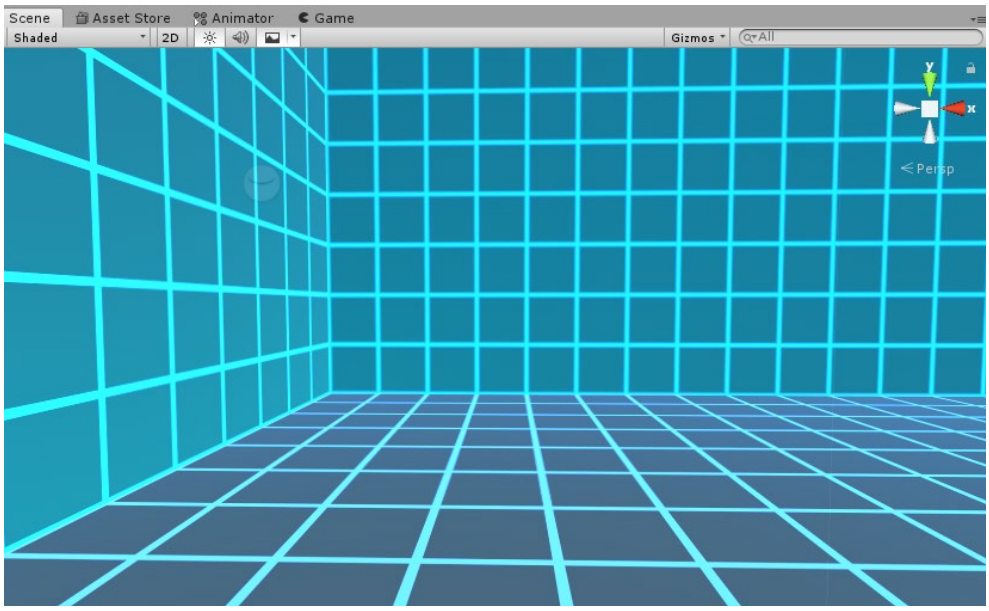

Рисунок 3.20 – Сцена без постобработки

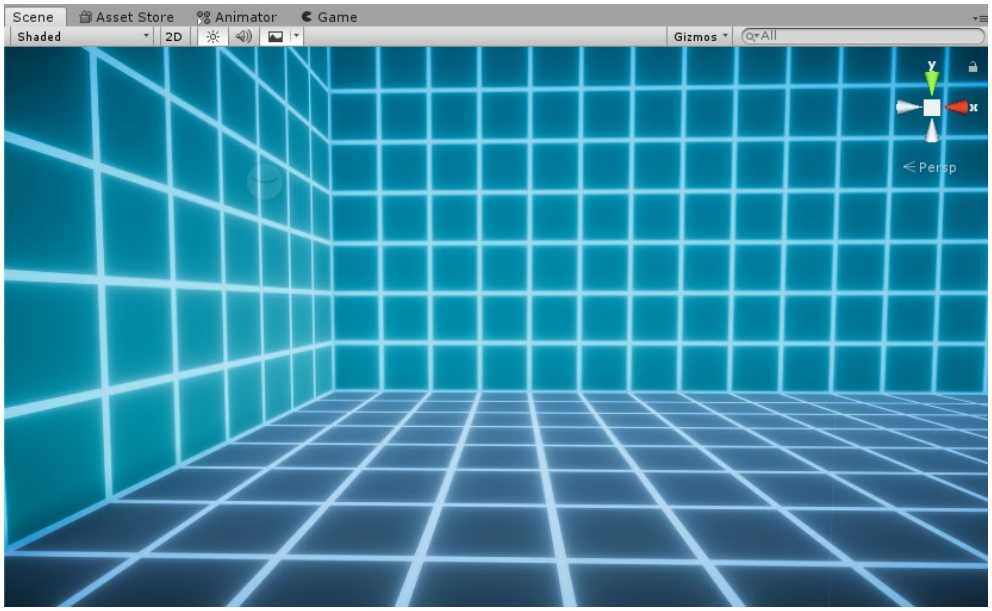

Рисунок 3.21 – Сцена с постобработкой

3.3. Создание препятствий

В иерархии (Hierarchy) необходимо нажимать ПКМ > 3D Object > Cube. Расположить его в пределах игрового пространства. Рисунок 3.22.

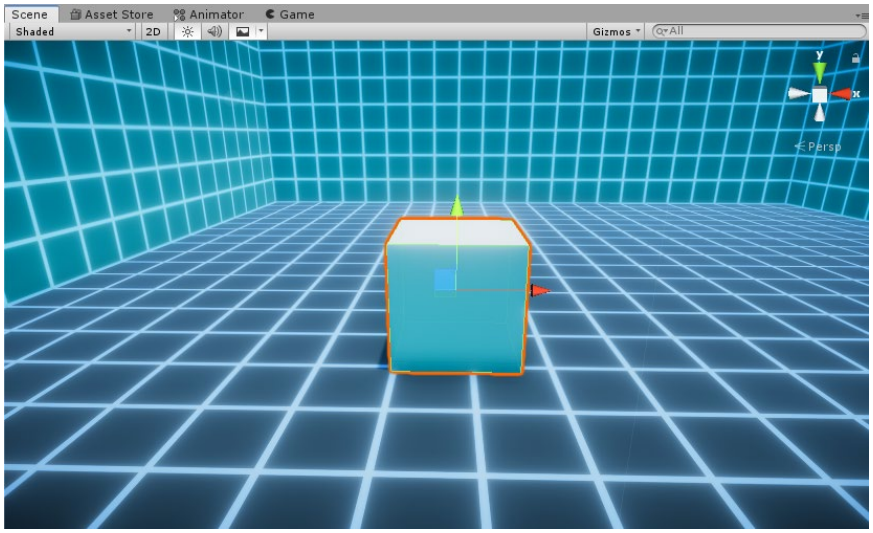

Рисунок 3.22 – Созданное препятствие

Создав новую текстуру, в папке с материалами нужно нажимать ПКМ > Create > Material. В инспекторе (Inspector) материала, элемент «Albedo» заполнить рисунком 3.11, а элемент «Color» - рисунком 3.12. И отредактировать цвет в обоих элементах как на рисунке 3.25. В итоге получим текстуру, которую применим к препятствию. Рисунок 3.26.

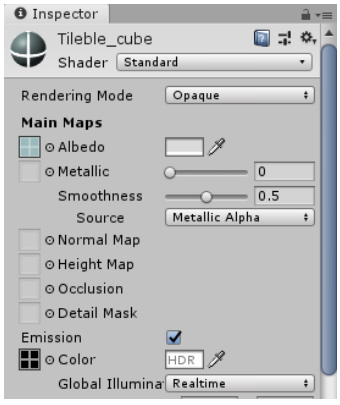

Рисунок 3.25 – Нас-ка текстуры препятствия

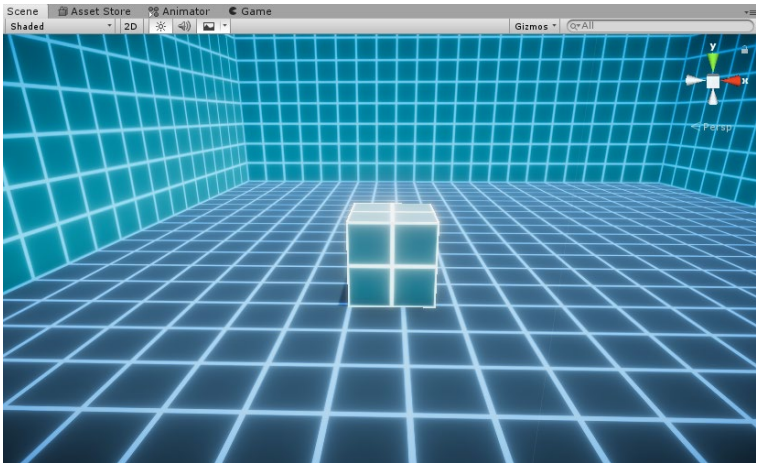

Рисунок 3.26 – Препятствие с текстурой

Для уровня, одно препятствие, это очень мало и не интересно. Следует создать и расставить их так, чтобы усложнить уровень и заполнить игровое пространство дабы не было больших пустых мест. Рисунок 3.27

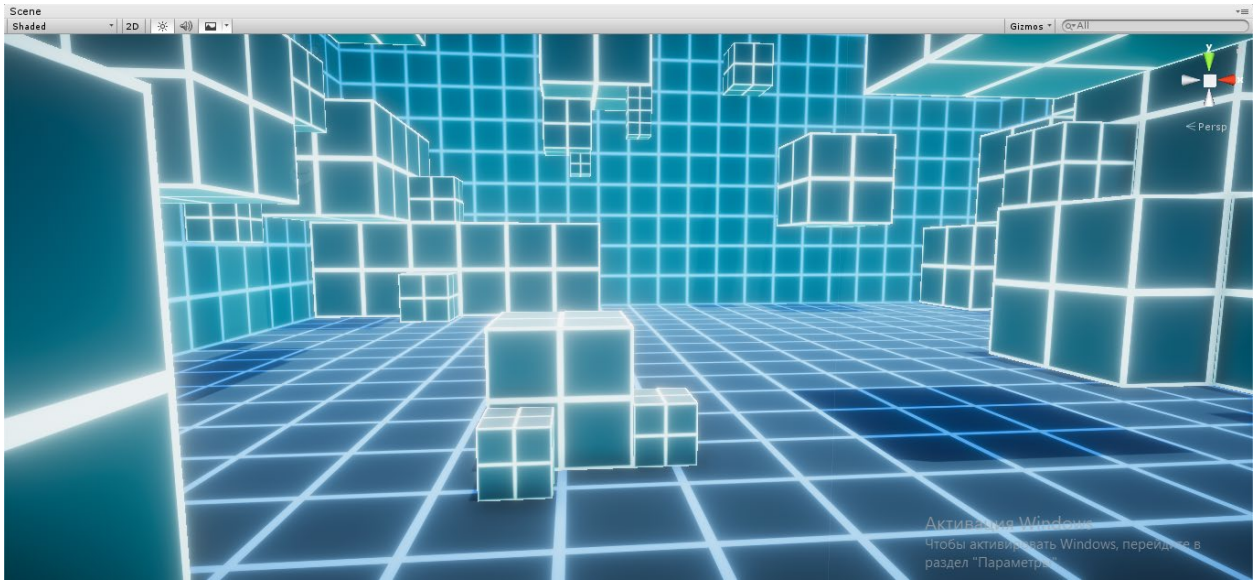

Рисунок 3.27 – Сцена с добавлением всех препятствий

3.4. Создание главного героя

Что бы создать новый куб, в иерархии (ПКМ > 3D Object > Cube). Следует сделать его размер по размеру клетки текстуры пола. Рисунок 3.28

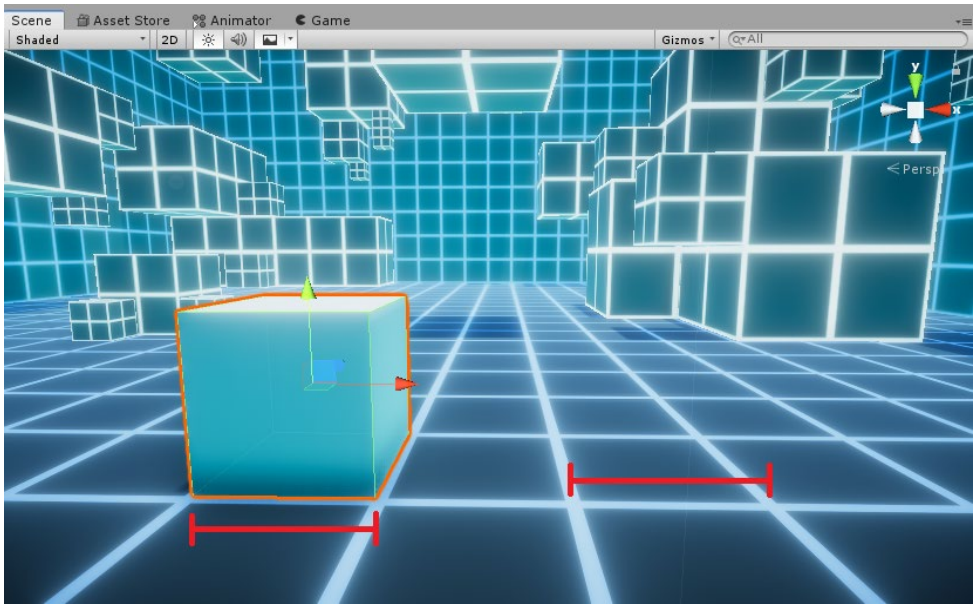

Рисунок 3.28 – Создание главного героя

В папке с материалами нужно создать новую текстуру. Рисунок для нее создам в Photoshop’е (рисунок 3.29 и 3.30)

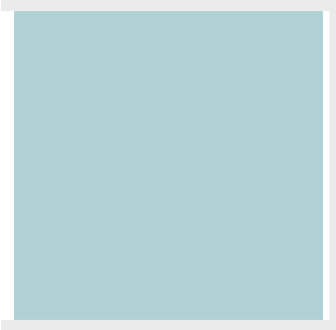

Рисунок 3.29

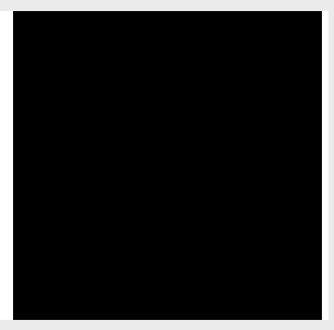

Рисунок 3.30

В инспекторе (Inspector) материала, элемент «Albedo» требуется заполнить рисунком 3.29, а элемент «Color» - рисунком 3.30. И отредактировать цвет в обоих элементах. Что-бы выделить объект главного героя, от остальных объектов. Цвет неона необходимо выбыть зеленый и увеличить интенсивность свечения этого цвета с 0 до 1.4169. Рисунок 3.31.

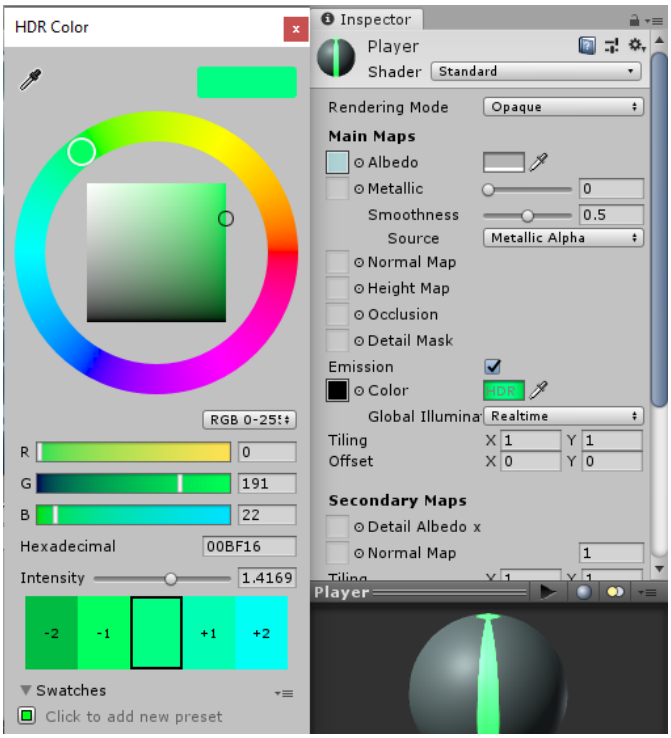

Рисунок 3.31 – Настройка текстуры главного героя

Применив текстуру главный, герой будет выглядеть следующим образом. Рисунок 3.32. Не смотря на то что главный герой впроекте изображен просто, он хорошо вписывается в окружение игрового пространства. Сохраняя стилистику кибер игры. И в тот же момент его можно отличить от других объектов и понять, что он в игре такой один.

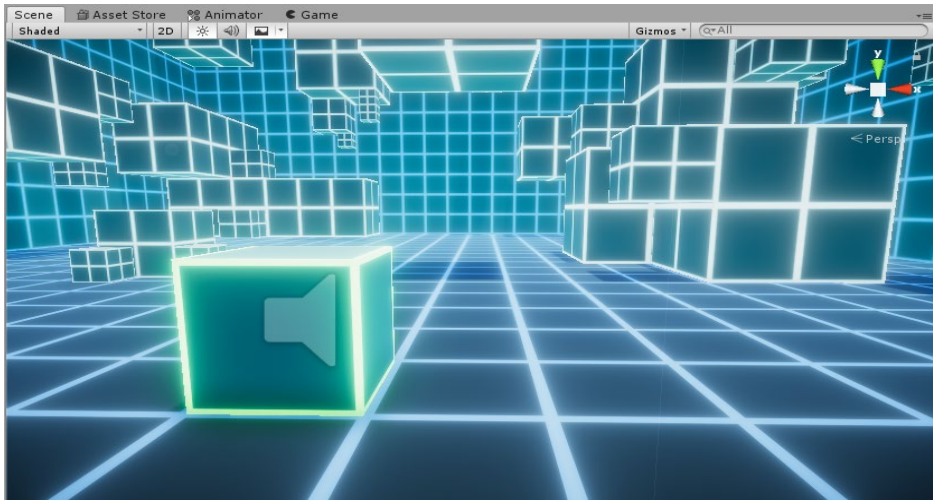

Рисунок 3.32 – Главный герой после применения текстур

3.5. Создание подбираемых предметов

Требуется создать новые объекты (Sphere). Расставить их на игровом поле, в тех местах в которых игрок их должен собрать. После, создать текстуру для этих объектов в папке с материалами. Выставить настройки текстуры так как на рисунке 3.33. И применить ее к нужным объектам.

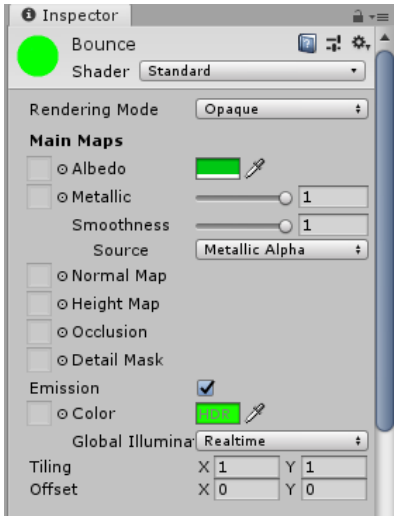

Рисунок 3.33 – Настойка текстуры подбираемых предметов

Выделив эти объекты в сцене, следует применить к ним компонент «Halo». И выставлю размер сияния на 0.48 единиц. Рисунок 3.34.

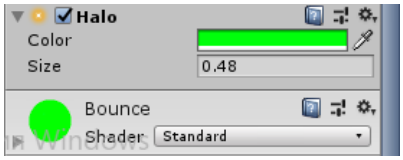

Рисунок 3.34 – Настройка компонента «Halo»

В итоге сферы будут выглядеть как что-то сияющее и самое главное заметное.

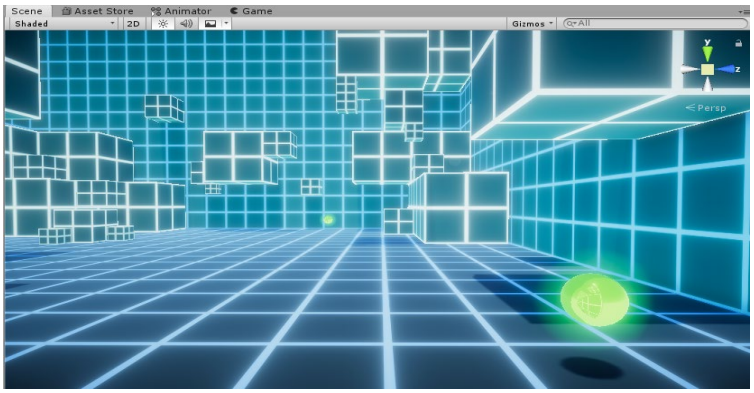

Рисунок 3.35– сферы после применения текстур

3.6. Создание текстового поля для ввода команд

Текстовое поле в моей работе будет выполнять роль редактора кода. Её функции – это ввод текста, удаление текста, запуск команд нажатием кнопки. Для создания текстового поля следует использовать стандартные элементы пользовательского интерфейса (UI), а именно «Input Field». В иерархии (Hierarchy) нажимаю ПКМ > UI > Input Field. Создастся небольшое текстовое поле размером с один столбец. Рисунок 3.36.

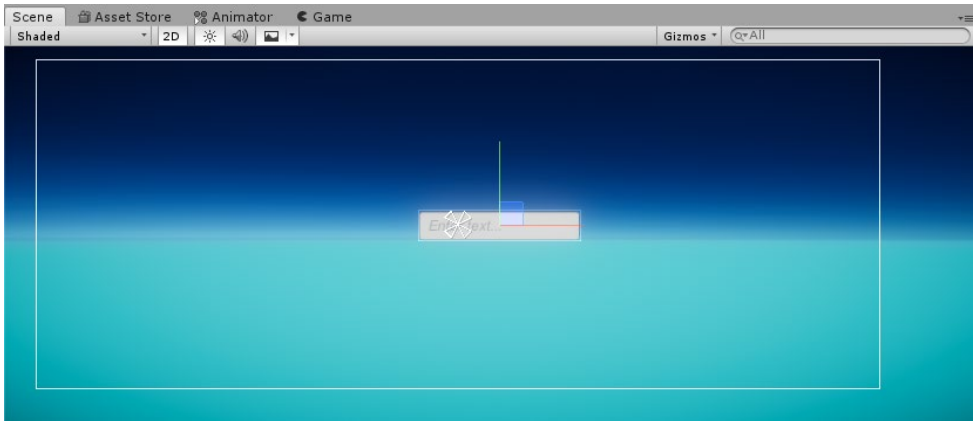

Рисунок 3.36 – Созданный «Input Field»

Размер текстового поля для реализации моей работы недостаточен. Тем более, когда запись текста в несколько строчек пока не возможен. А как я знаю в большинстве популярных редакторов кода, текст считывается построчно как на рисунке 3.37.

```
535         target.transform.position = transform.position;
536         step = Convert.ToInt32(words_2[j + 1]);
537         step_float = step * 11.37f;
538         progress = 0;
539         progress += Time.deltaTime * speed_run;
540         progress_dbl = Convert.ToDouble(progress);
541         progress_dbl = Math.Round(progress, 1);
542         progress_round = (float)progress_dbl;
543         start = target.transform.position;
544         target.transform.position = Vector3.Lerp(start,
545         ch_run = true;
```

Рисунок 3.37 – Пример кода в Visual Studio

Что бы мой редактор кода был приближен к настоящим редакторам. Следует его настроить. Первое – это сделать размер поля по всей высоте экрана. Для этого в инспекторе InputField, выполняю привязку к правому краю

по всей высоте экрана. Это нужно для того что бы играя на мониторах разного разрешения, текстовое поле было адаптивное. Рисунок 3.38.

Рисунок 3.38 – Настройка привязки поля

Второе – это задать размеры и позицию поля как на рисунке 3.39. Изменяя такие параметры как: Pivot, Pos X, Top, Width, Bottom. В итоге должно получиться, большое поле. Рисунок 3.40.

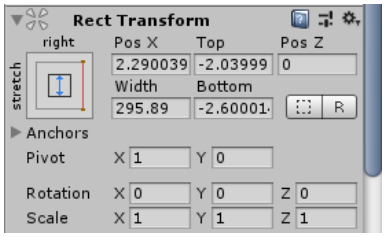

Рисунок 3.39 – Настройка размера и позиционирования

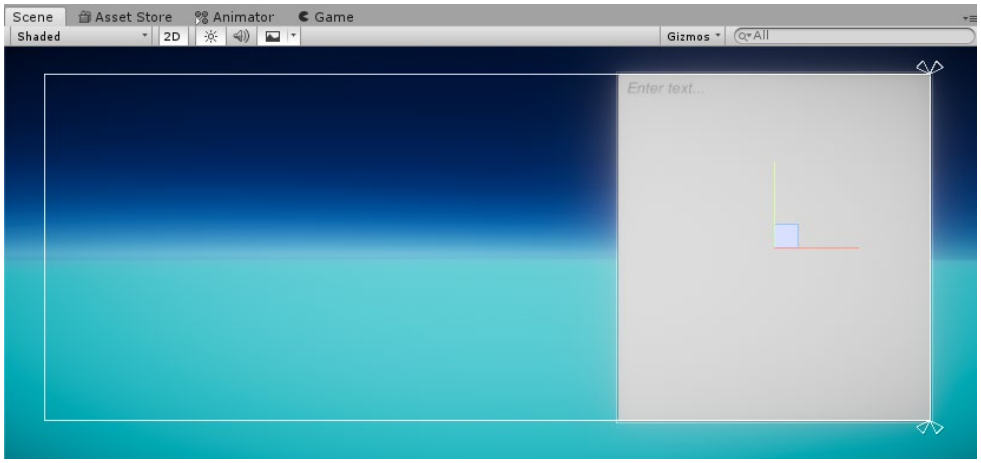

Рисунок 3.40 – Текстовое поле с изменёнными параметрами

Третье это сделать возможность переходить на новую строку в любой момент нажатием «Enter». Для этого в инспекторе (Inspector) InputField’a изменяем параметр «Line Type» на «Multi Line Newline». Рисунок 3.41.

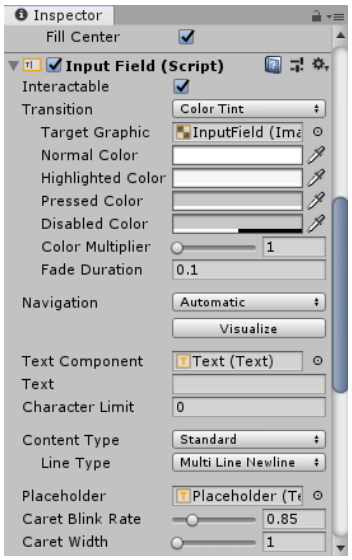

Рисунок 3.41 – Параметр «Line Type»

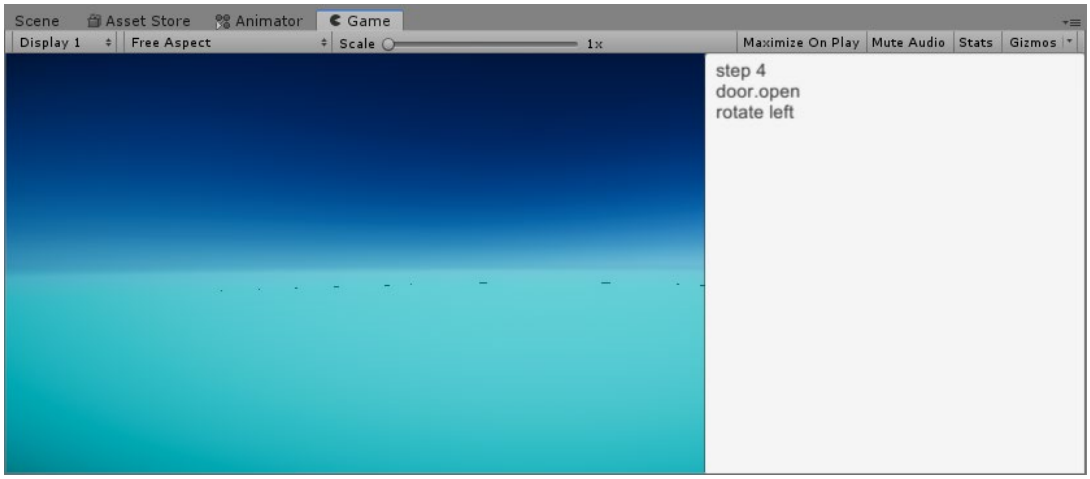

Рисунок 3.42 – Демонстрация Мульти строки

Теперь можно поработать над дизайном текстового поля, необходимо изменить цвет фона, шрифт, цвет букв. И сделать это так что бы дизайн вписывался в стилистику игры. Шрифт я поменять на «Electronic high», Цвет букв сделать ярко-голубым и фон темным. Рисунок 3.43

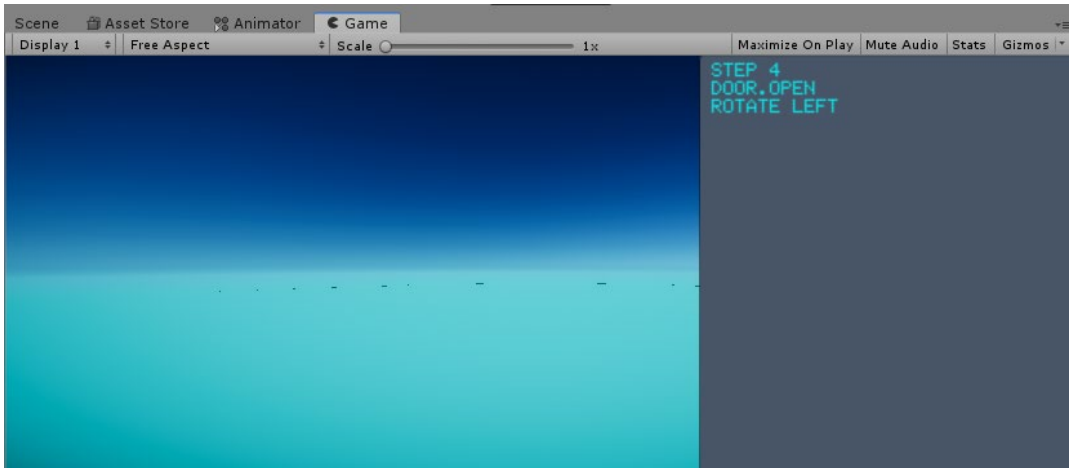

Рисунок 3.43 – Текстовое поле изменения внешнего вида

Так же во всех редакторах кода присутствует кнопка компиляции программы. В инспекторе нужно создать кнопку (ПКМ > UI > Button). Поменять у нее шрифт, цвет фона и цвет букв. И добавить картинку «Пуск», предварительно нарисовав ее в Photoshop. Что получиться в итоге посмотрим на рисунке 3.44.

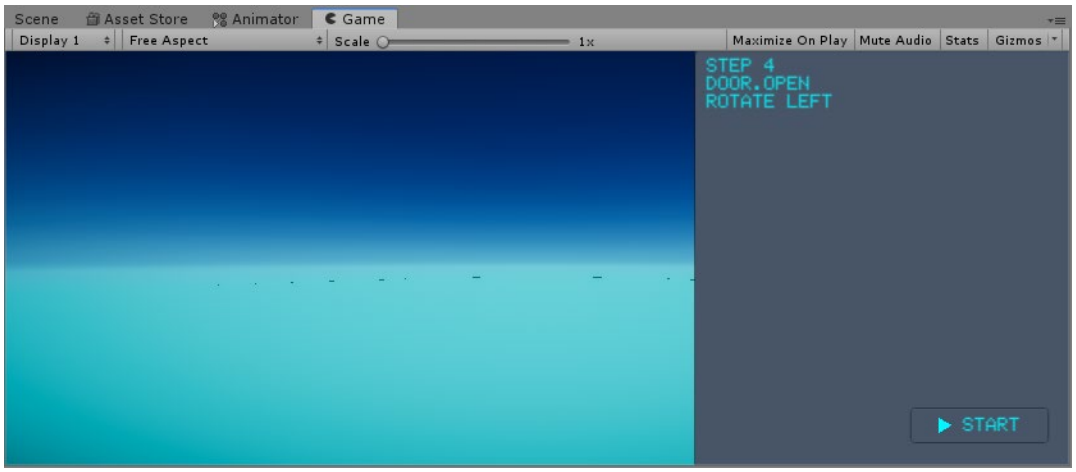

Рисунок 3.44 – Добавление кнопки запуска

3.7. Движение главного героя

Что бы герой двигался по заданным командам, необходим считать данные с текстового поля нажатием кнопки «START», идентифицировать команды, и выполнить скрипт по веденным командам.

Для считывания данных следует создать метод Check_void(). При срабатывании этого метода, скрипт создаст переменную тип string «a» и запишет в нее весь текст из текстового поля. Следующая строчка кода разделит все команды и заполнит ими массив с помощью функции Split, так как пока в переменной «a» сплошной текст. Разделителем служит символ переноса строки «\n», так как на одной строке одна команда. Рисунок 3.45.

```
public void Check_void()
{
    check = true;
    string a = Text.text;
    words = a.Split(new char[] { '\n' }, StringSplitOptions.RemoveEmptyEntries);
    ch_run = false;
    ch_rot = false;
    rot_ch = 1;
    x = 0;
    int h = 0;
```

Рисунок 3.45 – Метод считывания текста

Написав в текстовое поле команды как на рисунке 3.43 и нажав кнопку «START». Работа скрипта будет работать следующим образом.

```
string a = Text.text;
(a = «step 4
door.open
rotate left»)

words = a.Split(new char[] { '\n' }, StringSplitOptions.RemoveEmptyEntries);
(words[0] = step 4
 words[1] = door.open
 words[2] = rotate left)
```

3.7.1. Команда «STEP»

Перед тем как идентифицировать команду «STEP», ее нужно разделить на две части. Первая - это само слово step, вторая часть – количество шагов. Выполнив код:

```
words_2 = words[x].Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);
(words_2[0] = step
 words_2[1] = 4)
```

Программа разделит текст “step 4”, где разделителем является пробел.

Теперь если текст в переменной «words_2» равен слову «step», программа запустит метод Run(), убедившись в том что главный герой не двигается (check == true) и не находится в воздухе (cubeIsGround == true). Рисунок 3.46

```
words_2 = words[x].Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);

if (words_2[j] == "step" )
{
    check = true;
    if (check == true && cubeIsGround == true) Run();
}
```

Рисунок 3.46 – Идентификация команды «STEP»

Движение игрока выполняет метод Run() рисунок 3.47. В переменную «step» записывается конвертированное значение из переменной «words_2[1]». И «step» будет умножаться на длину пути одного шага.

```
public void Run()
{
    if(ch_run == false)
    {
        target.transform.position = transform.position;
        step = Convert.ToInt32(words_2[j + 1]);
        step_float = step * 11.37f;
        progress = 0;
        progress += Time.deltaTime * speed_run;
        progress_dbl = Convert.ToDouble(progress);
        progress_dbl = Math.Round(progress, 1);
        progress_round = (float)progress_dbl;
        start = target.transform.position;
        target.transform.position = Vector3.Lerp(start, target.transform.position + (target.transform.forward * step_
        ch_run = true;
    }
    progress = 0;
    progress += Time.deltaTime * speed_run;
    transform.position = Vector3.MoveTowards(transform.position, target.transform.position, progress);
    if (transform.position == target.transform.position)
    {
        check = false;
        ch_run = false;
        x++;
        j = 0;
    }
}
```

Рисунок 3.47 – Метод Run()

Что бы заставить игрока двигаться плавно на n-ое количество шагов, нужно создать пустой объект (Target_Move) внутри объекта главного героя. И с помощью скрипта задаю ту позицию объекту Target_Move, которая соответствует пройденным n-ым шагам. Далее благодаря функции MoveTowards(x, y, z), где x – координата начала движения, y – координата конца движения, z – скорость движения, движок переместит объект главного героя к объекту Target_Move как на рисунке 3.48.

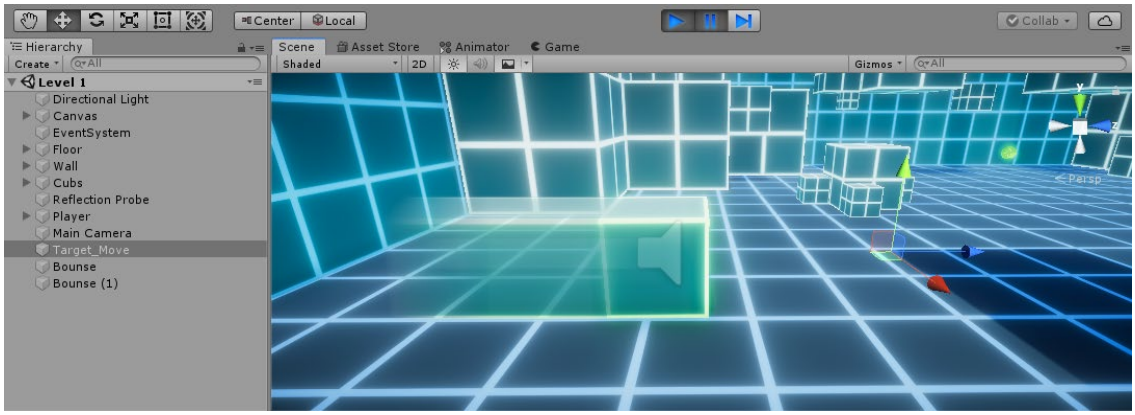

Рисунок 3.48 – Движение к объекту Target_Move

3.7.2. Команда «ROTATE LEFT» и «ROTATE RIGHT»

Идентифицировав команду поворота «Rotate right/left». Скрипт запустит метод Rotate_L() или Rotate_R(), убедившись в том что игрок находится на земле(cubeIsGround == true) и не выполняет поворот(rot_ch = 1). Рисунок 3.49.

```
if (words[x] == "rotate right")
{
    rot_ch = 1;
    if (rot_ch == 1 && cubeIsGround == true) Rotate_L();
}

if (words[x] == "rotate left")
{
    rot_ch = 1;
    if (rot_ch == 1 && cubeIsGround == true) Rotate_R();
}
```

Рисунок 3.49 – Идентификация команды поворота

Метод Rotate_R() и Rotate_L() выполняет поворот объекта на 90 градусов. При помощи функции Quaternion.RotateTowards(r, s, progress_rot). Где r – начальный угол поворота, s – конечный угол поворота, progress_rot – скорость поворота (градусов в секунду). Рисунок 3.50.

```
public void Rotate_L()
{
    if(ch_rot == false)
    {
        progress_rot = 0;
        y = transform.localEulerAngles.y;
        to = y + 90f;
        target.transform.Rotate(Vector3.up, 90);
        r = transform.rotation;
        ch_rot = true;
        s = transform.rotation;
        s = Quaternion.Euler(0, to, 0);
    }
    progress_rot += Mathf.Round(Time.deltaTime * speed_rot);
    transform.rotation = Quaternion.RotateTowards(r, s, progress_rot);
    if (progress_rot == 90f)
    {
        rot_ch = 0;
        x++;
        ch_rot = false;
    }
}
```

Рисунок 3.50 – Метод поворота главного героя

В итоге выполнив данный скрипт, Unity плавно развернет объект в ту сторону, какую указал пользователь. Рисунок 3.51 и 3.52.

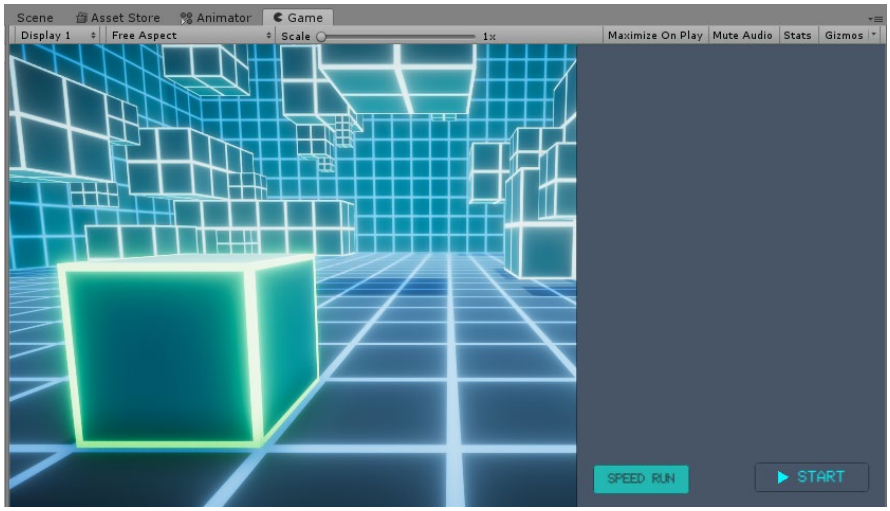

Рисунок 3.51 – Объект до применения команды

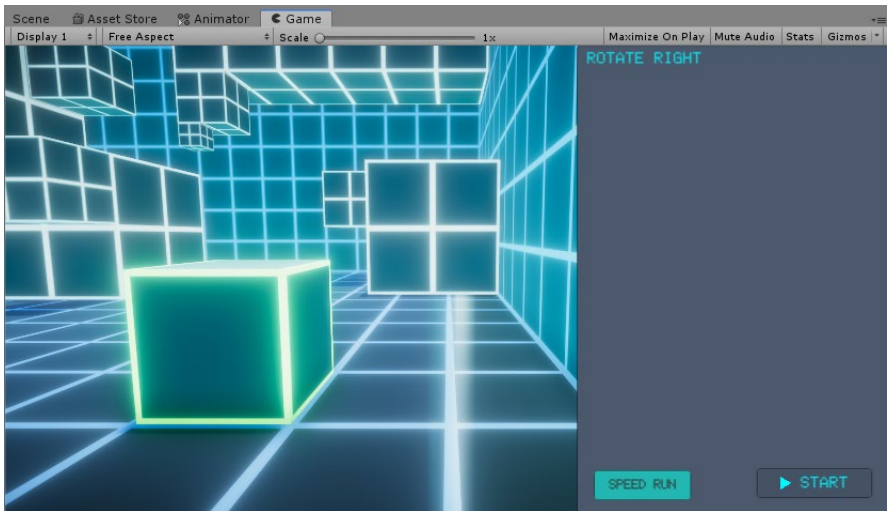

Рисунок 3.52 – Объект после применения команды

3.7.3. Команда «JUMP»

Для срабатывания команды «JUMP». Должны быть соблюдены уже три параметра. Первый параметр – игрок должен стоять на земле. Второй – игрок не должен двигаться вперед. Третий параметр, игрок не должен разворачиваться. Рисунок 3.53.

```
if (words[x] == "jump")
{
    if (cubeIsGround == true && ch_run == false && ch_rot == false) Jump();
}
```

Рисунок 3.53 – Идентификация команды «JUMP»

Что бы иметь функцию прыжка для игрока, нужно применить силу к твердому телу. Но пока объект игрока не имеет твердости и не имеет физики. Исправить это очень просто. Нужно выделить объект главного героя, в инспекторе (Inspector) нажать кнопку «Add Component» и искать компонент «RigidBody» как на рисунке 3.54.

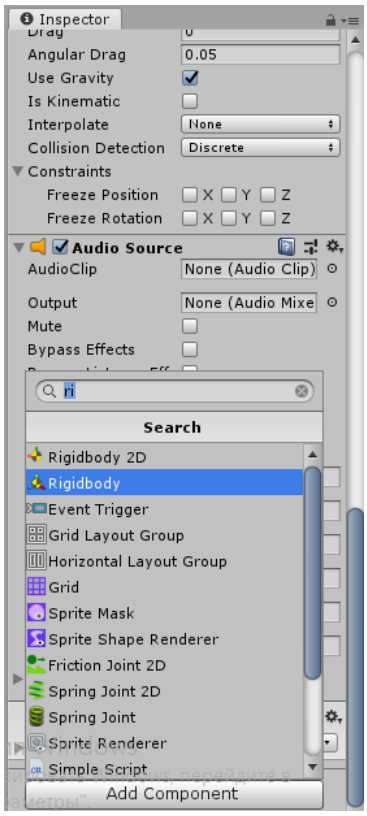

Рисунок 3.54 – Добавление компонента «RigidBody»

Дальше в скрипте применить две силы объекту. Рисунок 3.55. Первая сила будет толкать игрока вверх что бы он смог оттолкнуться от земли (rigid.AddForce(Vector3.up * 3.5f, ForceMode.Impulse)), вторая сила толкает вперед, что бы игрок передвигался немного вперед, а не прыгал на одном месте (rigid.AddRelativeForce(Vector3.forward * 0.79999f, ForceMode.Impulse)). Наглядно на рисунке 3.56 и 3.57.

```
public void Jump()
{
    audio.PlayOneShot(Jump_Sound);
    rigid.AddForce(Vector3.up * 3.5f, ForceMode.Impulse);
    rigid.AddRelativeForce(Vector3.forward * 0.79999f, ForceMode.Impulse);
    x++;
    cubeIsGround = false;
    rigid.freezeRotation = true;
}
```

Рисунок 3.55 – Метод «Jump»

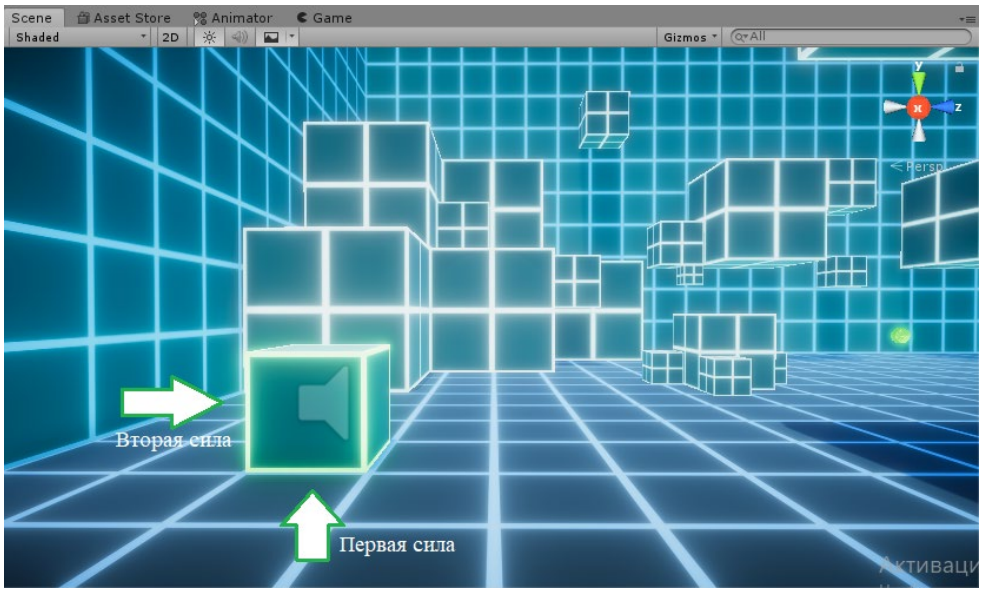

Рисунок 3.56 – Объект до применения сил

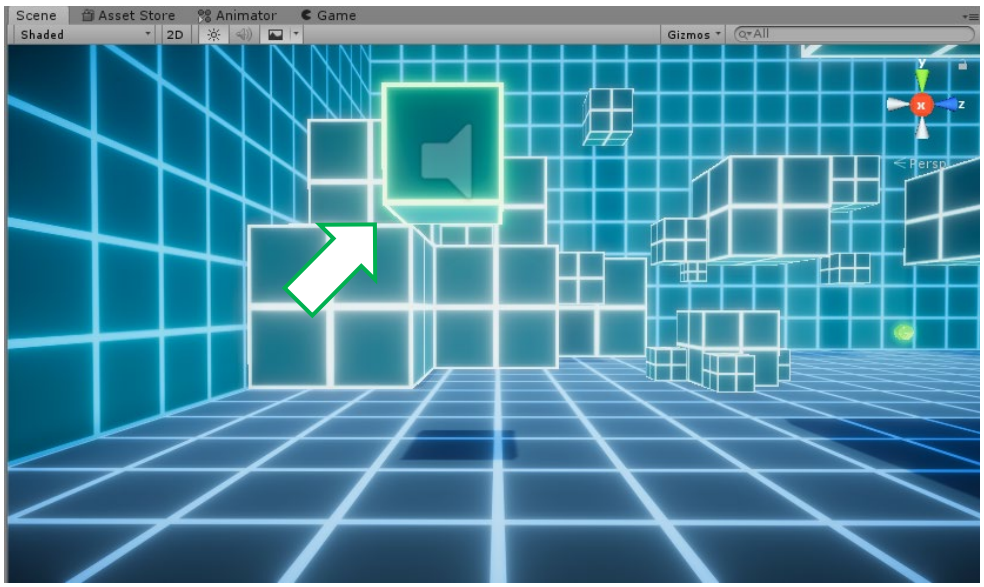

Рисунок 3.57 – Объект после применения сил

3.7.4. Команда «DOOR.OPEN» и «DOOR.CLOSE»

Эта команда будет двигать платформы вправо и влево, на которые должен будет встать игрок, чтобы переместиться с одного места на другое как на рисунке 3.58 и 3.59.

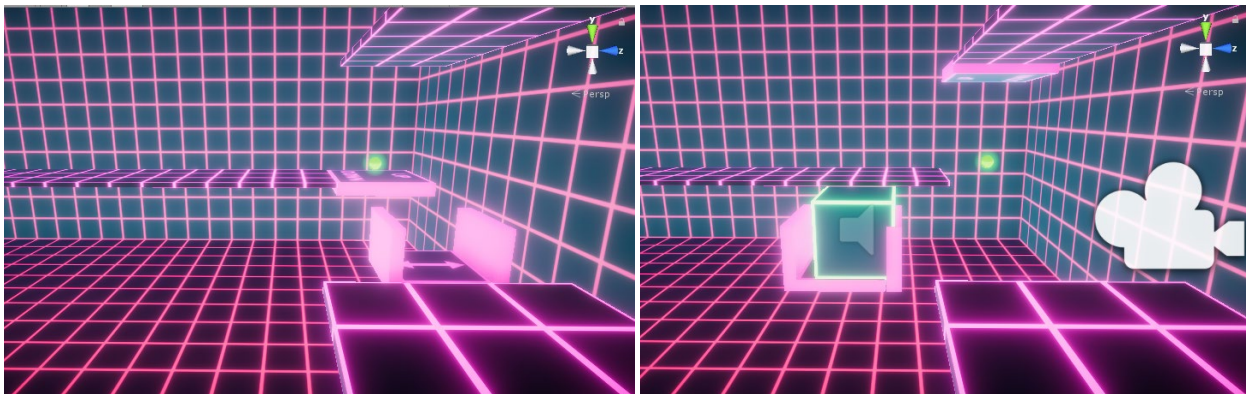

Рисунок 3.58 и 3.59 – Применение команды «DOOR.OPEN»

Что бы можно было двигать объекты, на которых нет скрипта, необходимо создать публичный игровой объект в самом коде как на рисунке 3.60. И в самой программе Unity, заполнить переменные игровыми объектами. Рисунок 3.61 и 3.62.

```
public GameObject Door;
public GameObject Door2;
public GameObject Door3;
public GameObject Door4;
public GameObject Door5;
```

Рисунок 3.60 – Создание публичных переменных

Рисунок 3.61.и 3.62 – Переменные до и после заполнения

Теперь я могу получить координаты объектов, которые лежат в переменных Door - Door5. У меня есть доступ к их компоненту трансформ (Transform). И это значит, что я могу ими управлять.

Прежде чем сработает метод движения объектов Door_control(). Слкдует записать в переменные координаты всех объектов по оси Z. Потому что именно по этой оси будут двигаться объекты. Рисунок 3.63. И далее проверить находится ли объект на земле или нет. Рисунок 3.64.

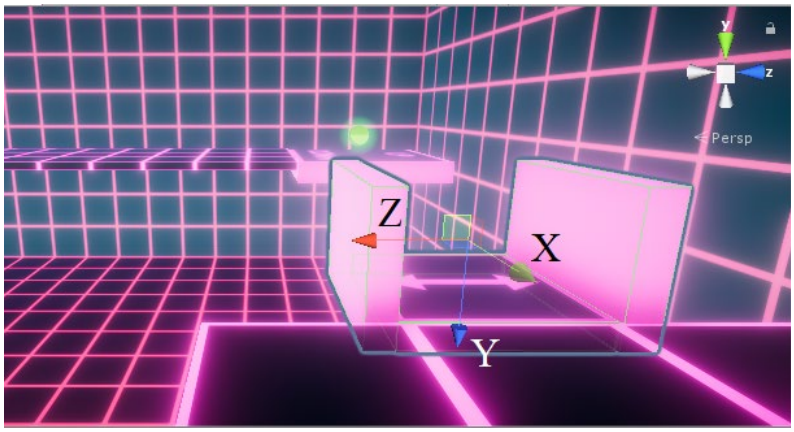

Рисунок 3.63 – Оси объекта

```
if (words[x] == "door.open")
{
    if(ch_lr == 1)
    {
        start_lr = Door2.transform.localPosition.z;
        start_lr2 = Door4.transform.localPosition.z;
        ch_lr = 0;
    }
    door_close = true;
    if(cubeIsGround == true) Door_control();
```

Рисунок 3.64 – Код срабатывающий перед запуском метода Door_control()

Метод Door_control() смотрит в каком положении в данный момент находятся передвигаемые объекты. И изменяет их координаты по оси Z на 17.012 единиц. При помощи строчки кода Door.transform.position = Vector3.MoveTowards(start_door, new Vector3(e2, e3, door_x), 3f * Time.deltaTime). Рисунок 3.65.

```
void Door_control()
{
    door_close = true;
    if (door_close == true)
    {

        if (words[x] == "door.close")
        {
            door_x = start_lr + 17.012f;
            door_x2 = start_lr2 + 17.012f;
        }
        if (words[x] == "door.open")
        {
            door_x = start_lr - 17.012f;
            door_x2 = start_lr2 - 17.012f;
        }
        start_door = Door2.transform.position;
        start_door3 = Door4.transform.position;
        door_close = false;

    }
    float e2 = Door2.transform.localPosition.x;
    float e3 = Door2.transform.localPosition.y;
    float r2 = Door4.transform.localPosition.x;
    float r3 = Door4.transform.localPosition.y;
    Door2.transform.position = Vector3.MoveTowards(start_door, new Vector3(e2, e3, door_x), 3f * Time.deltaTime);
    Door4.transform.position = Vector3.MoveTowards(start_door3, new Vector3(r2, r3, door_x2), 3f * Time.deltaTime);
    if (Door2.transform.localPosition.z == door_x)
    {
        ch_lr = 1;
        x++;
    }
```

Рисунок 3.65 – Метод Door_control()

3.7.5. Команда «DOOR.UP» и «DOOR.DOWN»

Команды «DOOR.UP» и «DOOR.DOWN», так же работают с движением платформ. Только они двигают такой тип платформ, который смещается по оси Y, т.е. вверх и вниз как на рисунке 3.66 и 3.67.

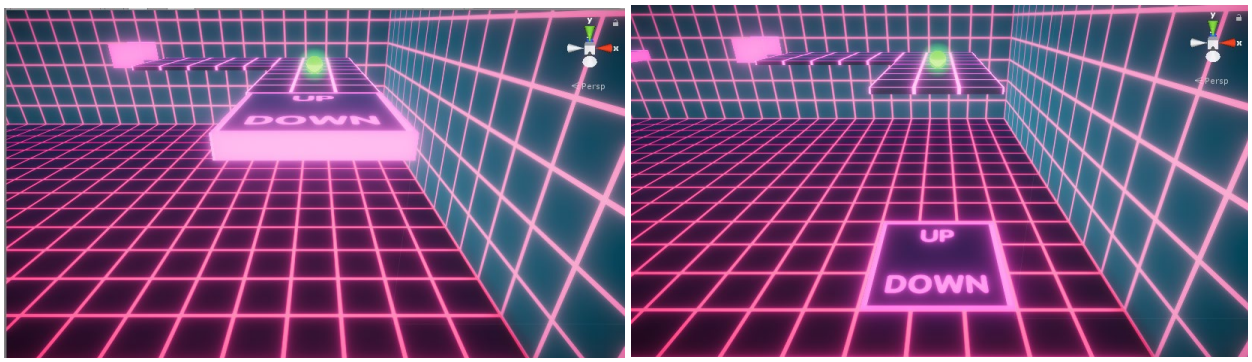

Рисунок 3.66 и 3.67 – Работа команд «DOOR.UP» и «DOOR.DOWN»

Перед выполнением метода Door_Control_Up() скрипт должен записать в переменные, текущие позиции всех платформ (UP DOWN). И убедиться, что главный герой находиться на земле и не выполняет в данный момент команду. Рисунок 3.68.

```
if (words[x] == "door.up")
{
    if (ch_ud == 1)
    {
        start_ud = Door.transform.localPosition.y;
        start_ud2 = Door3.transform.localPosition.y;
        start_ud3 = Door5.transform.localPosition.y;
        ch_ud = 0;
    }
    door_close = true;
    if (cubeIsGround == true) Door_control_UP();

}

if (words[x] == "door.down")
{

    if (ch_ud == 1)
    {
        start_ud = Door.transform.localPosition.y;
        start_ud2 = Door3.transform.localPosition.y;
        start_ud3 = Door5.transform.localPosition.y;
        ch_ud = 0;
    }
    door_close = true;
    if (cubeIsGround == true) Door_control_UP();
}
```

Рисунок 3.68 - Код срабатывающий перед запуском метода Door_control_UP()

Метод Door_control_UP(), двигает платформу вверх и вниз на 4 единицы по оси Y в зависимости от введенной команды, при помощи функции MoveTowards. Рисунок 3.69.

```
void Door_control_UP()
{
    door_close = true;
    if (door_close == true)
    {

        if (words[x] == "door.up")
        {
            door_y = start_ud + 4f;
            door_y2 = start_ud2 + 4f;
            door_y3 = start_ud3 + 4f;
        }

        if (words[x] == "door.down")
        {
            door_y = start_ud - 4f;
            door_y2 = start_ud2 - 4f;
            door_y3 = start_ud3 - 4f;
        }
        start_door = Door.transform.position;
        start_door2 = Door3.transform.position;
        start_door4 = Door5.transform.position;
        door_close = false;
    }
    float e2 = Door.transform.localPosition.x;
    float e3 = Door.transform.localPosition.z;
    float r2 = Door3.transform.localPosition.x;
    float r3 = Door3.transform.localPosition.z;
    float v2 = Door5.transform.localPosition.x;
    float v3 = Door5.transform.localPosition.z;
    Door.transform.position = Vector3.MoveTowards(start_door, new Vector3(e2, door_y, e3), 0.8f * Time.deltaTime);
    Door3.transform.position = Vector3.MoveTowards(start_door2, new Vector3(r2, door_y2, r3), 0.8f * Time.deltaTime);
    Door5.transform.position = Vector3.MoveTowards(start_door4, new Vector3(v2, door_y3, v3), 0.8f * Time.deltaTime);
    if (Door.transform.localPosition.y == door_y)
    {
        ch_ud = 1;
        x++;
    }
```

Рисунок 3.69 – Метод Door_control_UP()

3.8. Коллизия с посторонними предметами

В моем проекте два типа соприкосновения с объектами. Первый это соприкосновение со стенами и препятствиями, которое приводит к перезагрузке уровня. Второе соприкосновение с подбираемыми предметами, оно завязано на прохождении уровня.

Для реализации соприкосновения с препятствием понадобился готовый метод в библиотеке Unity, называется «OnCollisionEnter». OnCollisionEnter вызывается, когда collider/rigidbody начал соприкосновение с другим rigidbody/collider. Что бы метод не вызывался при соприкосновении со всеми объектами, а только с определенными. следует присвоить препятствиям тэг – «Obstacle», как на рисунке 3.70. Предварительно создав его «Add Tag…».

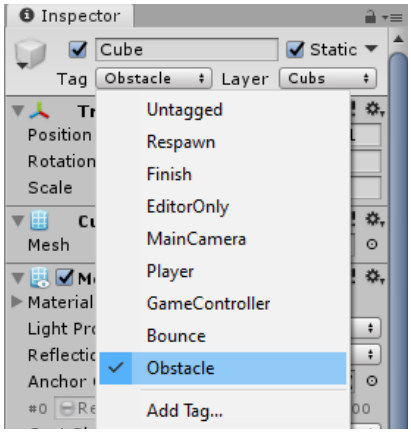

Рисунок 3.70 – Присваивание тэга «Obstacle»

Далее в скрипте нужно создать метод «OnCollisionEnter» и писать программу, которая будет работать при соприкосновении с объектом с тэгом «Obstacle». Рисунок 3.71. В моем случае это воспроизведение звуков проигрыша, перезапуск текущего уровня и воспроизведение анимации «Game over». Рисунок 3.72.

```
void OnCollisionEnter(Collision collision)
{
    if (collision.gameObject.tag == "Obstacle")
    {
        audio.PlayOneShot(GameOver_sound);
        game_over.SetActive(true);
        Invoke("Game_Over", 2.5f);
    }
    cubeIsGround = true;
    rigid.freezeRotation = false;
    rigid.constraints = RigidbodyConstraints.None;
}
```

Рисунок 3.71 – Метод «OnCollisionEnter»

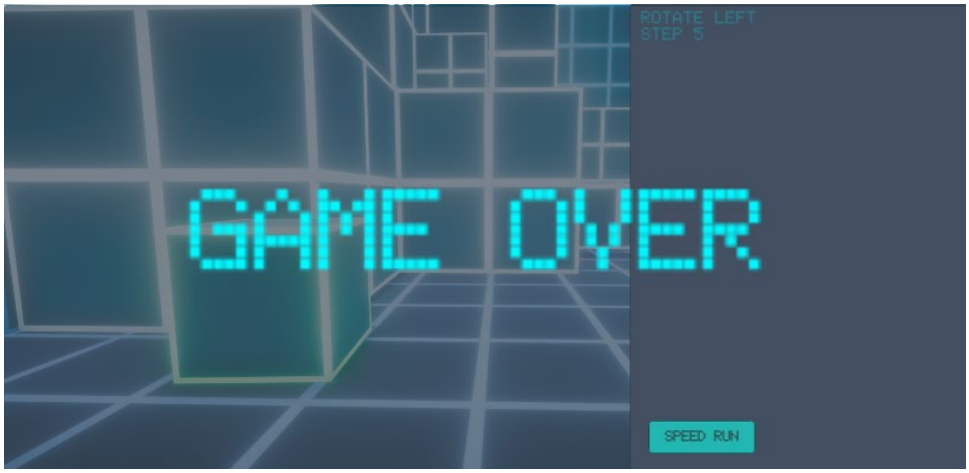

Рисунок 3.72 – Воспроизведение анимации «Game over»

Для реализации соприкосновения с подбираемыми объектами. Необходимо использовать метод «OnTriggerEnter». Так как на этих объектах не должно быть компонента «RigidBody». Что бы при соприкосновении эти объекты не замедляли главного героя и можно было проходить сквозь них.

Реализуется это следующим образом. Выбирав объект и в инспекторе (Inspector) в компоненте «Sphere Collider» ставить галочку напротив «Is Trigger» как на рисунке 3.73

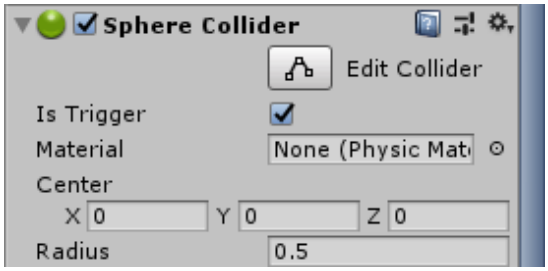

Рисунок 3.73 – Превращение объекта в триггер

При вызове метода «OnTriggerEnter» воспроизведется звук, и удалится объект которого коснулся главный герой. Рисунок 3.74

```
void OnTriggerEnter(Collider collider)
{
    if (collider.gameObject.tag == "Bounce")
    {
        audio.PlayOneShot(Bonus);
        Destroy(collider.gameObject);
        sum++;
        if(sum == Bounce.Length)
        {
            Invoke("Next_lvl", 2.5f);
            lvl_comp.SetActive(true);
        }
    }
}
```

Рисунок 3.74 – Метод «OnTriggerEnter»

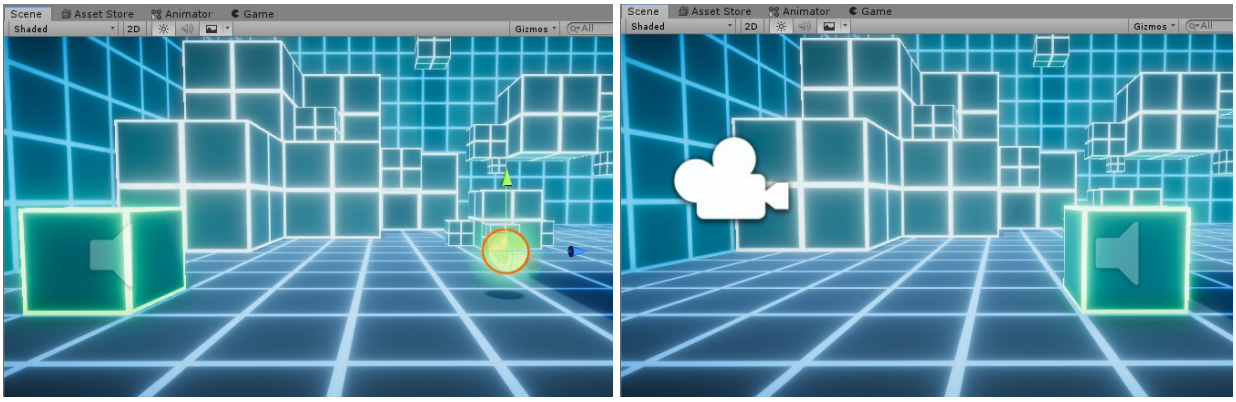

Рисунок 3.75 и 3.76 – Удаление объекта после соприкосновения

3.9 Выводы

Глава «Разработка программного средства» похожа на методические указания по созданию обучающей игры. В ней я разобрал и детально описал проектирование основных объектов и скриптов игры. Показал:

- как создавалась сцена уровня и ее окружение
- как создавался пользовательский интерфейс
- как осуществляется движение главного героя
- как взаимодействуют объекты друг с другом
- как работают основные скрипты игры

Этот проект не дался мне легко и для достижения конечного результата мне пришлось пройти через большое количество ошибок, недоработок, багов, которые в итоге стали мне уроком. Заставили глубже изучить в игровой движок Unity и его компоненты.

ГЛАВА 4. ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ

4.1. Определение информационной безопасности

Информационная безопасность, представляет собой набор методов, предназначенных для защиты данных от несанкционированного доступа или изменений, как при хранении, так и при передаче с одного устройства на другое. Чаше это называется безопасностью данных. Поскольку информация стала одним из самых важных активов 21-го века, усилия по обеспечению безопасности информации становятся все более жесткими.

Вопрос информационной безопасности является важным аспектом для каждой игровой студии в индустрии видеоигр. На первый взгляд может показаться, что проблем с безопасностью в играх нет и быть не может. Но, как мы можем видеть в последних новостях, ни одна компания не застрахована от рисков безопасности, независимо от того, насколько она велика или мала.

Хотя одна студия не похожа на другую, и их методы и подход к работе отличаются, существуют общие проблемы безопасности, которые могут влиять и влияют на каждую компанию, выпускающую видеоигры. Чтобы должным образом описать эти проблемы безопасности, была выбрана одна из ведущих компаний, занимающихся видеоиграми. Информация этого раздела будет обобщена для того, чтобы ее можно было легко применить к любой компании, занимающейся видеоиграми.

4.2. Принципы информационной безопасности

Основные компоненты информационной безопасности чаще всего суммируются с помощью так называемой триады ЦРУ: конфиденциальность, целостность и доступность.

Конфиденциальность – это, пожалуй, первое, что приходит на ум, когда вы думаете об информационной безопасности. Чтобы обеспечить конфиденциальность, вы должны определить, кто пытается получить доступ к данным. И нужно будет блокировать попытки тех, кто не имеет на это разрешения. Пароли, шифрование, аутентификация и защита от атак проникновения – все эти методы, разработаны для обеспечения конфиденциальности.

Целостность данных означает поддержание данных в их правильном состоянии и предотвращение их изменению, случайно или злонамеренно. Многие из методов, обеспечивающих конфиденциальность, также защищают целостность данных - в конце концов, хакер не может изменить данные, к которым он не может получить доступ, - но есть и другие инструменты, которые помогают обеспечить глубокую защиту целостности: контрольные суммы могут помочь вам проверить данные целостность, например, программное обеспечение для контроля версий и частые резервные копии могут помочь вам восстановить данные в правильном состоянии, если это будет необходимо. Целостность также охватывает концепцию отказа от прав: вы должны доказать, что вы сохранили целостность своих данных, особенно в юридическом контексте.

Доступность является зеркальным отражением конфиденциальности: Вам необходимо убедиться, что к вашим данным не могут получить доступ неавторизованные пользователи, вам также необходимо убедиться, что к ним могут получить доступ те, у кого есть соответствующие разрешения. Обеспечение доступности данных означает согласование сетевых и

вычислительных ресурсов с ожидаемым объемом доступа к данным и реализацию хорошей политики резервного копирования для целей аварийного восстановления.

4.3. Информационная безопасность в играх

Средства, с помощью которых принципы применяются к программным продуктам, принимают форму политики безопасности. Это не часть аппаратного или программного обеспечения безопасности. Скорее всего это документ, который компания разрабатывает, основываясь на своих конкретных потребностях и причудах, чтобы определить, какие данные должны быть защищены и каким образом. Эта политика безопасности направляют решения организации относительно приобретения инструментов кибербезопасности, а также предписывают поведение и обязанности сотрудников.

Например, Play Market. Ее способы обезопасить пользователей их сервиса, состоят:

- Во-первых, после публикации, идёт автоматическая проверка. Проверяется много данных. Описание приложения, график, возрастной рейтинг контент и прочая. Нейронная сеть тут может среагировать как угодно и забанить приложение если найдет в ней что-то неладное.
- Во-вторых, иногда после публикации, игру проверяет ещё и человек. Что он проверяет - неизвестно и не будет известно, чтобы недобросовестные разработчики не могли найти способ обойти проверку.
- В-третьих, периодически нейросеть проверяет все приложения. Может проверить через час после одобрения приложения, а может через месяц или год.

4.4. Выводы

Современная индустрия видеоигр обширна по размеру и продолжает расти. Новые компании постоянно создаются, продаются, приобретаются и даже закрываются. Проблемы информационной безопасности являются критически важным аспектом для каждой организации в индустрии видеоигр. Благодаря успешному созданию и внедрению любого корпоративного плана безопасности каждая из этих компаний сможет поддерживать надлежащую безопасность и инфраструктуру конкурентоспособности на современном рынке видеоигр. Информация описанная в этом разделе должна актуальна для большинства компаний, занимающихся разработкой видеоигр.

Информация актуальна на момент написания ВКР, но через год и более появятся компоненты, которые должны быть заменены или обновлены. Возникают новые процедуры безопасности, создается и покупается новое оборудование, производится новое ПО. Мир постоянно развивается и меняется, и индустрия видеоигр ничем не отличается.

ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы была достигнута основная цель – реализована и создана игра, обучающая программированию «Cyber Code». Игра получилась привлекательной и в своем технологическом стиле Cyberpunk. Управление у нее простое и понятное, что позволяет сразу приступить к практическим занятиям по программированию, не тратя дополнительное время на изучение ее принципов. Эта игра помимо свей обучающей функции, так же принесет пользователю удовольствие, позволит повысить мотивацию в изучении программирования и прибавит заинтересованность в данной теме. Что позволит в простой игровой манере закрепить знания.

Но все равно игра «Cyber Code» не лишена минусов и не является идеальной. Следует поработать детальнее с подбираемыми объектами и сделать их модель интереснее. Добавить больше уровней для прохождения и больше команд. Что должно благополучно повлиять на обучение. Поработать над оптимизацией, что бы внедрение прошло успешно как можно в больших учебных заведениях.

Все поставленные задачи были выполнены в полном объеме:

— произвел исследование данного вопроса и анализ существующих решений;

— построил модели и алгоритмы разрабатываемой системы;

— выполнил программную реализацию средства.

Исследование такого вопроса как, роль игр, обучающих программированию, показало, что использование компьютерных игр в образовательном пространстве позволит:

— повысить положительную мотивацию учения;

— расширить объем используемой информации;

— использовать новые формы представления информации, в частности, визуально-наглядные;

— расширить набор применяемых учебных задач;

— активно включать учащихся в учебный процесс посредством применения игровой компьютерной деятельности;

— обеспечить условия для развития интеллектуальной активности, творческого мышления учащихся.

Но стоит не допускать игровой зависимости, что бы для детей и подростков игры не заменяли традиционный досуг, общение с друзьями и родителями. Для этого необходимо разрабатывать профилактические, диагностические и коррекционно-развивающие мероприятия для устранения проблемы. Я считаю, надо обучать программированию с азов, с раннего детского возраста, помогая, показывая грани плохого и хорошего в компьютерном мире, мире онлайн игр.

В своей работе я решил обучать программированию. Выбор пал на эту отрасль не спроста. Я считаю, что программирование - фундаментальный предмет, и он уже кардинальным образом изменил нашу повседневность жизнь, поменяв такие привычки как: поездки в авто, заботу о здоровье, сферу общения с друзьями, обучение в школах и т.д. Системность познания азов программирования поможет сформировать комплекс знаний, которые в дальнейшем пригодятся повседневной обычной жизни. В наше время — это такой же равнозначный предмет, как и русский язык, литература, математика, алгебра, геометрия, английский язык, физика.

СПИСОК ЛИТЕРАТУРЫ

1. Unity in Action: Multiplatform Game Development in C# with Unity 5 by Joe Hocking
2. Introduction to Game Programming: Using C# and Unity 3D by Vahe Karamian
3. Паласиос, Хорхе Unity 5.x. Программирование искусственного интеллекта в играх / Хорхе Паласиос. - М.: ДМК Пресс, 2016.
4. C# Game Programming Cookbook for Unity 3D by Jeff W. Murray
5. Unity в действии. Мультиплатформенная разработка на C#. - М.: Питер, 2018
6. Хорхе, Паласиос Unity 5.x. Программирование искусственного интеллекта в играх. Руководство / Паласиос Хорхе. - М.: ДМК Пресс, 2017.
7. Learning C# by Developing Games with Unity 3D Beginner's GuideSep by Terry Norton
8. Beginning 3D Game Development with Unity 4: All-in-one, multi-platform game development (Technology in Action) by Sue Blackman
9. Крис Диккенсон. Unity 5, оптимизация игр.: Диалектика. – 2015.
10. https://docs.unity3d.com/ru/2018.4/Manual/UnityManual.html – Документация Unity
11. Дошкольная педагогика с основами методик воспитания и обучения: Учебник / Под ред. Гогоберидзе А. Г., Солнцевой О.В.. - СПб.: Питер, 2017. - 480 с.
12. Педагогика. Учебник для ВУЗов. Стандарт третьего поколения / Под ред. П. Тряпицыной. - СПб.: Питер, 2018. - 16 с.
13. Абрамов, С.А. Математические построения и программирование / С.А. Абрамов. - М.: Наука, 2016. - 192 с.
14. Ахо Альфред В. Структуры данных и алгоритмы: Вильямс / пер. с английского и ред. Минько А. А., Ахо Альфред В., Хопкрофт Джон Э., Ульман Джеффри Д. — М. и др.: Вильямс, 2001. – 382 с.

ПРИЛОЖЕНИЕ А

(обязательное)

Спецификация

Обозначение	Наименование	Примечание
	Документация	
ВКР-09.03.02-150469 -19 81	Пояснительная записка	

ПРИЛОЖЕНИЕ Б. Исходный код игры

Move.cs

```
using System;
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.UI;

public class Move : MonoBehaviour
{
    public float speed_run = 5.0f;
    public float speed_rot = 50.0f;
    public float Rot = 60.0f;
    public float Step = 5f;

    public GameObject target;
    public GameObject Door;
    public GameObject Door2;
    public GameObject Door3;
    public GameObject Door4;
    public GameObject Door5;
    public GameObject button;
    public GameObject game_over;
    public GameObject lvl_comp;
    public GameObject[] Bounce;

    public InputField Text;
    public Button But;
    public Transform target_2;

    bool rot_bool = false;
    bool ch_run, ch_rot, cubeIsGround,ch_j,door_close = true;
    bool check, split,aud = false;

    float y,to, w, p, l, step_float, progress_round, up, e, door_x, door_x2,
door_y,door_y2, door_y3, start_lr, start_lr2, start_ud, start_ud2,start_ud3;
    private float progress;
    private float progress_rot;

    string[] words, words_2;

    [SerializeField] AudioClip Bonus;
    [SerializeField] AudioClip GameOver_sound;
    [SerializeField] AudioClip Jump_Sound;
    [NonSerialized] int rot_ch = 0;
    int x,j,sum = 0;
    int v = 1;
    int step,lvl;
    int ch_ud = 1;
    int ch_lr = 1;

    double progress_dbl;

    Vector3 test;
    Vector3 start;
    Vector3 jump;
    Vector3 start_door;
    Vector3 start_door2;
    Vector3 start_door3;
```

```
    Vector3 start_door4;

    Quaternion s;
    Quaternion r;

    Rigidbody rigid;
    AudioSource audio;

    void Start()
    {
        audio = GetComponent<AudioSource>();
        Bounce = GameObject.FindGameObjectsWithTag("Bounce");
        But.GetComponent<Button>();
        rigid = GetComponent<Rigidbody>();
    }

    public void Update()
    {
        if (x == words.Length && sum != Bounce.Length)
        {
            if (aud == false)
            {
                audio.PlayOneShot(GameOver_sound);
                aud = true;
            }
            game_over.SetActive(true);
            Invoke("Game_Over", 2.5f);
        }

        if (x < words.Length) But.gameObject.SetActive(false);
        else if(x >= words.Length) But.gameObject.SetActive(true);
        words_2 = words[x].Split(new char[] { ' ' },
StringSplitOptions.RemoveEmptyEntries);

        if (words_2[j] == "step" )
        {
            check = true;
            if (check == true && cubeIsGround == true) Run();
        }

        if (words[x] == "rotate right")
        {
            rot_ch = 1;
            if (rot_ch == 1 && cubeIsGround == true) Rotate_L();
        }

        if (words[x] == "rotate left")
        {
            rot_ch = 1;
            if (rot_ch == 1 && cubeIsGround == true) Rotate_R();
        }

        if (words[x] == "jump")
        {
            if (cubeIsGround == true && ch_run == false && ch_rot == false) Jump();
        }

        if (words[x] == "door.open")
        {
            if(ch_lr == 1)
            {
                start_lr = Door2.transform.localPosition.z;
                start_lr2 = Door4.transform.localPosition.z;
                ch_lr = 0;
```

```
        }
        door_close = true;
        if(cubeIsGround == true) Door_control();

    }

    if (words[x] == "door.close")
    {
        if (ch_lr == 1)
        {
            start_lr = Door2.transform.localPosition.z;
            start_lr2 = Door4.transform.localPosition.z;
            ch_lr = 0;

        }
        door_close = true;
        if (cubeIsGround == true) Door_control();
    }

    if (words[x] == "door.up")
    {
        if (ch_ud == 1)
        {
            start_ud = Door.transform.localPosition.y;
            start_ud2 = Door3.transform.localPosition.y;
            start_ud3 = Door5.transform.localPosition.y;
            ch_ud = 0;
        }
        door_close = true;
        if (cubeIsGround == true) Door_control_UP();

    }

    if (words[x] == "door.down")
    {

        if (ch_ud == 1)
        {
            start_ud = Door.transform.localPosition.y;
            start_ud2 = Door3.transform.localPosition.y;
            start_ud3 = Door5.transform.localPosition.y;
            ch_ud = 0;
        }
        door_close = true;
        if (cubeIsGround == true) Door_control_UP();
    }
}

void OnTriggerEnter(Collider collider)
{
    if (collider.gameObject.tag == "Bounce")
    {
        audio.PlayOneShot(Bonus);
        Destroy(collider.gameObject);
        sum++;
        if(sum == Bounce.Length)
        {
            Invoke("Next_lvl", 2.5f);
            lvl_comp.SetActive(true);
        }
    }
}

void Next_lvl()
```

```
    {
        int lvl = SceneManager.GetActiveScene().buildIndex;
        SceneManager.LoadScene(lvl = lvl + 1);
    }

    void Game_Over()
    {
        lvl = SceneManager.GetActiveScene().buildIndex;
        SceneManager.LoadScene(lvl);
    }

    void Door_control()
    {
        door_close = true;
        if (door_close == true)
        {

            if (words[x] == "door.close")
            {
                door_x = start_lr + 17.012f;
                door_x2 = start_lr2 + 17.012f;
            }
            if (words[x] == "door.open")
            {
                door_x = start_lr - 17.012f;
                door_x2 = start_lr2 - 17.012f;
            }
            start_door = Door2.transform.position;
            start_door3 = Door4.transform.position;
            door_close = false;

        }
        float e2 = Door2.transform.localPosition.x;
        float e3 = Door2.transform.localPosition.y;
        float r2 = Door4.transform.localPosition.x;
        float r3 = Door4.transform.localPosition.y;
        Door2.transform.position = Vector3.MoveTowards(start_door, new Vector3(e2, e3,
door_x), 3f * Time.deltaTime);
        Door4.transform.position = Vector3.MoveTowards(start_door3, new Vector3(r2, r3,
door_x2), 3f * Time.deltaTime);
        if (Door2.transform.localPosition.z == door_x)
        {
            ch_lr = 1;
            x++;
        }

    }

    void Door_control_UP()
    {
        door_close = true;
        if (door_close == true)
        {

            if (words[x] == "door.up")
            {
                door_y = start_ud + 4f;
                door_y2 = start_ud2 + 4f;
                door_y3 = start_ud3 + 4f;
            }

            if (words[x] == "door.down")
            {
                door_y = start_ud - 4f;
```

```
                door_y2 = start_ud2 - 4f;
                door_y3 = start_ud3 - 4f;
            }
            start_door = Door.transform.position;
            start_door2 = Door3.transform.position;
            start_door4 = Door5.transform.position;
            door_close = false;
        }
        float e2 = Door.transform.localPosition.x;
        float e3 = Door.transform.localPosition.z;
        float r2 = Door3.transform.localPosition.x;
        float r3 = Door3.transform.localPosition.z;
        float v2 = Door5.transform.localPosition.x;
        float v3 = Door5.transform.localPosition.z;
        Door.transform.position = Vector3.MoveTowards(start_door, new Vector3(e2, door_y,
e3), 0.8f * Time.deltaTime);
        Door3.transform.position = Vector3.MoveTowards(start_door2, new Vector3(r2,
door_y2, r3), 0.8f * Time.deltaTime);
        Door5.transform.position = Vector3.MoveTowards(start_door4, new Vector3(v2,
door_y3, v3), 0.8f * Time.deltaTime);
        if (Door.transform.localPosition.y == door_y)
        {
            ch_ud = 1;
            x++;
        }

    }

    void OnCollisionEnter(Collision collision)
    {
        if (collision.gameObject.tag == "Obstacle")
        {
            audio.PlayOneShot(GameOver_sound);
            game_over.SetActive(true);
            Invoke("Game_Over", 2.5f);
        }
        cubeIsGround = true;
        rigid.freezeRotation = false;
        rigid.constraints = RigidbodyConstraints.None;
    }

    public void Jump()
    {
        audio.PlayOneShot(Jump_Sound);
        rigid.AddForce(Vector3.up * 3.5f, ForceMode.Impulse);
        rigid.AddRelativeForce(Vector3.forward * 0.79999f, ForceMode.Impulse);
        x++;
        cubeIsGround = false;
        rigid.freezeRotation = true;
    }

    public void Run()
    {
        if(ch_run == false)
        {
            target.transform.position = transform.position;
            step = Convert.ToInt32(words_2[j + 1]);
            step_float = step * 11.37f;
            progress = 0;
            progress += Time.deltaTime * speed_run;
            progress_dbl = Convert.ToDouble(progress);
            progress_dbl = Math.Round(progress, 1);
            progress_round = (float)progress_dbl;
            start = target.transform.position;
```

```
            target.transform.position = Vector3.Lerp(start, target.transform.position +
(target.transform.forward * step_float), progress_round);
            ch_run = true;
        }
        progress = 0;
        progress += Time.deltaTime * speed_run;
        transform.position = Vector3.MoveTowards(transform.position,
target.transform.position, progress);
        if (transform.position == target.transform.position)
        {
            check = false;
            ch_run = false;
            x++;
            j = 0;
        }
    }

    public void Rotate_L()
    {
        if(ch_rot == false)
        {
            progress_rot = 0;
            y = transform.localEulerAngles.y;
            to = y + 90f;
            target.transform.Rotate(Vector3.up, 90);
            r = transform.rotation;
            ch_rot = true;
            s = transform.rotation;
            s = Quaternion.Euler(0, to, 0);
        }
        progress_rot += Mathf.Round(Time.deltaTime * speed_rot);
        transform.rotation = Quaternion.RotateTowards(r, s, progress_rot);
        if (progress_rot == 90f)
        {
            rot_ch = 0;
            x++;
            ch_rot = false;
        }
    }

    public void Rotate_R()
    {
        if (ch_rot == false)
        {
            progress_rot = 0;
            y = transform.localEulerAngles.y;
            to = y - 90f;
            target.transform.Rotate(Vector3.up, -90);
            r = transform.rotation;
            ch_rot = true;
            s = transform.rotation;
            s = Quaternion.Euler(0, to, 0);
        }
        progress_rot += Mathf.Round(Time.deltaTime * speed_rot);
        transform.rotation = Quaternion.RotateTowards(r, s, progress_rot);
        if (progress_rot == 90f)
        {
            rot_ch = 0;
            x++;
            ch_rot = false;
        }
    }

    public void Check_void()
```

```
    {
        check = true;
        string a = Text.text;
        words = a.Split(new char[] { '\n' }, StringSplitOptions.RemoveEmptyEntries);
        ch_run = false;
        ch_rot = false;
        rot_ch = 1;
        x = 0;
        int h = 0;
        print(a);


public void Lvl_compleate()
    {
        int lvl_com = SceneManager.GetActiveScene().buildIndex;
        if (lvl_com == 1)
        {
            Text.text = "step 13" + "\n" +
                "rotate left" + "\n" +
                "step 10" + "\n" +
                "rotate left" + "\n" +
                "rotate left" + "\n" +
                "step 16" + "\n" +
                "rotate left" + "\n" +
                "step 4" + "\n";
        }

        if (lvl_com == 2)
        {
            Text.text = "rotate right" + "\n" +
                "step 2" + "\n" +
                "rotate left" + "\n" +
                "step 16" + "\n" +
                "rotate right" + "\n" +
                "step 7" + "\n" +
                "rotate right" + "\n" +
                "rotate right" + "\n" +
                "step 19" + "\n" +
                "rotate left" + "\n" +
                "rotate left" + "\n" +
                "step 8" + "\n" +
                "rotate right" + "\n" +
                "step 10" + "\n";
        }

        if (lvl_com == 3)
        {
            Text.text = "step 4" + "\n" +
                "jump" + "\n" +
                "rotate left" + "\n" +
                "step 1" + "\n" +
                "rotate right" + "\n" +
                "step 3" + "\n" +
                "jump" + "\n" +
                "step 6" + "\n" +
                "rotate left" + "\n" +
                "step 3" + "\n" +
                "jump" + "\n" +
                "step 9" + "\n" +
                "rotate left" + "\n" +
                "step 3" + "\n" +
                "jump" + "\n" +
                "jump" + "\n" +
                "step 1" + "\n" +
```

```
        "rotate left" + "\n" +
        "step 7" + "\n" +
        "rotate right" + "\n" +
        "step 7" + "\n" +
        "rotate right" + "\n" +
        "step 5" + "\n";
}

if (lvl_com == 4)
{
    Text.text = "step 1" + "\n" +
        "jump" + "\n" +
        "jump" + "\n" +
        "rotate left" + "\n" +
        "step 1" + "\n" +
        "jump" + "\n" +
        "step 2" + "\n" +
        "rotate left" + "\n" +
        "step 4" + "\n" +
        "rotate right" + "\n" +
        "step 4" + "\n" +
        "jump" + "\n" +
        "jump" + "\n" +
        "step 2" + "\n" +
        "jump" + "\n" +
        "rotate right" + "\n" +
        "step 2" + "\n" +
        "rotate right" + "\n" +
        "step 1" + "\n" +
        "rotate left" + "\n" +
        "step 3" + "\n" +
        "jump" + "\n" +
        "step 5" + "\n" +
        "jump" + "\n" +
        "step 3" + "\n" +
        "rotate right" + "\n" +
        "step 12" + "\n" +
        "jump" + "\n" +
        "step 1" + "\n";
}

if (lvl_com == 5)
{
    Text.text = "step 8" + "\n" +
        "door.up" + "\n" +
        "step 8" + "\n" +
        "rotate left" + "\n" +
        "step 7" + "\n" +
        "door.open" + "\n" +
        "step 8" + "\n" +
        "rotate right" + "\n" +
        "step 14" + "\n" +
        "door.down" + "\n" +
        "step 2" + "\n" +
        "door.up" + "\n" +
        "rotate right" + "\n" +
        "step 17" + "\n" +
        "rotate right" + "\n" +
        "step 2" + "\n" +
        "rotate right" + "\n" +
        "door.close" + "\n" +
        "step 2" + "\n" +
        "door.open" + "\n" +
        "step 6" + "\n" +
```

```
                "door.down" + "\n" +
                "step 2" + "\n" +
                "door.up" + "\n" +
                "rotate right" + "\n" +
                "step 7" + "\n" +
                "jump" + "\n" +
                "step 2" + "\n";
        }
    }
}
```

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

public class CameraControl : MonoBehaviour
{

    public float panSpeed = 5f;
    Quaternion s;

public float movementSpeed = 1;
    public float speedH = 2;
    public float speedV = 2;

    public float yaw = 0;
    public float pitch = 0;
    float to,y;

    void Start()
    {
        Cursor.lockState = CursorLockMode.Locked;
        Cursor.visible = false;
    }

    void Update()
    {
        Quaternion pos = transform.rotation;
        if (Input.GetKey("e"))
        {
            y = transform.localEulerAngles.y;
            to = y + 90f;
            s = Quaternion.Euler(0, to, 0);
            transform.rotation = Quaternion.RotateTowards(transform.rotation, s,
Time.deltaTime * 30f);
        }

        Quaternion pos2 = transform.rotation;
        if (Input.GetKey("q"))
        {
            y = transform.localEulerAngles.y;
            to = y - 90f;
            s = Quaternion.Euler(0, to, 0);
            transform.rotation = Quaternion.RotateTowards(transform.rotation, s,
Time.deltaTime * 30f);
        }

        if (Input.GetKey("space"))
        {
            y = transform.localEulerAngles.x;
            to = y - 90f;
            s = Quaternion.Euler(to, 0, 0);
            transform.rotation = Quaternion.RotateTowards(transform.rotation, s,
Time.deltaTime * 10f);
        }

        if (Input.GetKey("c"))
        {
            y = transform.localEulerAngles.x;
            to = y + 90f;
```

```
            s = Quaternion.Euler(to, 0, 0);
            transform.rotation = Quaternion.RotateTowards(transform.rotation, s,
Time.deltaTime * 10f);
        }

float h = Input.GetAxis("Horizontal");
        float v = Input.GetAxis("Vertical");

transform.position += transform.TransformDirection(Vector3.forward/20) * v +
transform.TransformDirection(Vector3.up/20) * h;

        if (Input.GetKeyDown(KeyCode.Escape))
        {
            Cursor.lockState = CursorLockMode.None;
            Cursor.visible = true;
        }
```

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

public class SimpleCamera : MonoBehaviour
{
    public Transform target;
    public float speed_cam = 6f;
    private Vector3 _position;
    public LayerMask maskObstacles;

    void Start()
    {
        _position = target.InverseTransformPoint(transform.position);
    }

    // Update is called once per frame
    void Update()
    {
        var currentPosition = target.TransformPoint(_position);
        transform.position = Vector3.Lerp(transform.position,currentPosition, speed_cam *
Time.deltaTime);
        var carrentRotation = Quaternion.LookRotation(target.position -
transform.position);
        transform.rotation = Quaternion.Lerp(transform.rotation, carrentRotation,
speed_cam * Time.deltaTime);

        RaycastHit hit;
        if(Physics.Raycast(target.position,transform.position - target.position,out hit,
Vector3.Distance(transform.position, target.position), maskObstacles))
        {
            transform.position = hit.point;
            transform.LookAt(target);
        }
    }
}
```

MainMenu.cs

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

public class MainMenu : MonoBehaviour
{
    public void PlayGame()
    {
        int lvl = SceneManager.GetActiveScene().buildIndex;
        SceneManager.LoadScene(lvl = lvl + 1);
    }

    public void Quit()
    {
        Application.Quit();
    }
}
```