Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Космических и информационных технологий
институт

Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О.В. Непомнящий
подпись инициалы, фамилия
«___» _____ 2018 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 Информатика и вычислительная техника
код и наименование специализации

Разработка трехмерной игры в жанре стэлс-экшен на платформе «Unity»
тема

Руководитель	_____ подпись, дата	доцент, канд. техн. наук должность, ученая степень	А.В. Редькин инициалы, фамилия
Выпускник	_____ подпись, дата		Н.М. Кулеш инициалы, фамилия
Нормоконтролер	_____ подпись, дата	доцент, канд. техн. наук должность, ученая степень	В.И. Иванов инициалы, фамилия

Красноярск 2018

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

С начала 2000-х годов в игровой индустрии ощутимо возрастает популярность игр от независимых разработчиков. В последнее время подобные игры получили особо высокую популярность, в связи с более доступными средствами разработки и дистрибуции. Независимые игры создаются, как правило, отдельными, небольшими коллективами разработчиков или маленькими независимыми компаниями. Разработчики независимых игр, зачастую, не имеют финансовой поддержки от издателя и обладают небольшим бюджетом, либо не обладают им вовсе. Ввиду своей независимости, разработчики не имеют операционных ограничений или творческих ограничений и не нуждаются в одобрении издателя, что является обязательным для разработчиков крупных проектов [1], [2].

Распространение интернета и служб цифровой дистрибуции позволило распространять игры, не осуществляя розничных продаж. Это дало возможность разработчикам издавать игры через такие службы, как Xbox Live Arcade, Steam или Playstation Store [2].

Цифровая дистрибуция и дешёвые инструменты вроде Unity придали импульс к появлению нового поколения независимых разработчиков. Этот сектор в принципе хорошо развивался, но теперь небольшие по численности команды вместо работы над низкобюджетными проектами для небольшой аудитории сотрудничают с производителями консолей и Steam-сообществом, чтобы создавать настоящие хиты [5].

1 Анализ технического задания

1.1 Цель и задачи

Целью проекта: разработка игрового приложения для персональных компьютеров на базе операционных систем Windows.

Задачи проекта:

– создать и интегрировать трехмерные модели в среду Unity;

– реализовать передвижение персонажей в пространстве и взаимодействие с интерактивным кружением;

– реализовать искусственный интеллект.

1.2 Обзор выбранного жанра

Важным и самым значимым элементом игр жанра стелс – это скрытное и бесшумное прохождение уровней, путем выполнения поставленных перед игроком задач. Основными инструментами игрового процесса современных стелс-игр являются укрытия, затененные места уровней, обходные пути, рукопашный бой и бесшумное оружие. Несмотря на то, что в отдельных играх скрытность может быть единственным способом выиграть, в большинстве игр обычно предусматриваются другие способы и стили достижения поставленной задачи. Если игрок будет обнаружен врагом, обычно требуется спрятаться на некоторое время, пока враги не прекратят поиск. Таким образом, важность приобретают планирование и метод проб и ошибок. В некоторых играх игрок может выбирать между убийством врагов или простым его оглушением. Если необходимость в скрытности прохождения необязательна, либо в недостаточной степени реализована в игре, игроки все же могут пробовать

избегать сражений по моральным причинам или для демонстрации своего мастерства [3].

Игры жанра экшен акцентированы на координации и скорости реакции игрока. Главный персонаж должен перемещаться по уровню, собирать предметы, избегать препятствий и сражаться с врагами своими естественными навыками, а также оружием и другими инструментами, находящимися в их распоряжении. В конце уровня или группы уровней игрок должен часто побеждать босса, который является более сложным и часто основным противником на протяжении игры [4].

1.3 Техническое задание

В результате выполнения данного проекта должна быть разработана одиночная трехмерная игра от третьего лица в жанре стэлс-экшен. Игра должна быть реализована в виде приложения для персональных компьютеров на базе операционных систем Windows с помощью среды разработки Unity.

Описание игровой механики

Игра состоит их нескольких схожих по своей структуре уровней с одноэтажным зданием, включающим в себя несколько комнат, соединенных общим коридором. Игроку необходимо незаметно проникнуть в здание и похитить портфель с документами, а затем добраться до точки выхода незамеченным, что считается успешным прохождением уровня. Портфель находится под охраной, которая патрулирует внутренние помещения и периметр здания. Игроку предоставляются два способа проникновения в здание: окна и входные двери. Также игрок может оглушать неприятелей, подкравшись к ним со спины. Охранники способны обращать внимание на оставленные открытыми игроком двери и окна, вследствие чего они начинают

проверять прилегающую территорию. Обнаружение игрока охранником приводит к провалу задания.

1.4 Инструменты разработки

1.4.1 Сравнительный анализ трехмерных редакторов

3Ds Max

3Ds Max обладает большим количеством инструментов для моделирования разнообразных архитектурных проектов. 3Ds Max дает возможность очень гибко управлять частицами, создавая самые разнообразные эффекты (моделирование анимированных массивов объектов, имитации всевозможных природных явлений, таких как брызги накатывающихся волн, дым). В программу был интегрирован фотореалистичный визуализатор, позволяющий добиться высокой правдоподобности просчитываемого изображения. Также в редакторе 3Ds Max присутствуют средства для анализа и настройки освещенности трехмерного проекта. Инструмент скиннинга Geodesic Voxel позволяет обрабатывать сложную геометрию, которая не является непроницаемой и может содержать не множащиеся или перекрывающиеся компоненты. Инструменты для анимации персонажей позволяют эффективно выполнять процедурную анимацию и оснастку двуногих персонажей и толпы. Модификаторы Skin и CAT Muscle обеспечивают более точное и плавное управление скелетной деформацией. Сложные механизмы и персонажей можно оснащать нестандартными скелетами, используя кости 3ds Max, модули решения задач инверсной кинематики (IK) и настраиваемые средства оснастки. Объединение одно- и двусторонних связей между контроллерами помогает создавать упрощенные интерфейсы 3D-анимации. Существует возможность внедрять пользовательские элементы управления анимацией, которые

объединяют элементы интерфейса, например регуляторы и наборные счетчики, с анимируемыми треками объекта и управление несколькими атрибутами с помощью одного регулятора. Набор инструментов для работы с ключевыми кадрами и процедурной анимации позволяет управлять практически всеми параметрами в сцене. Просмотр и редактирование анимационных траекторий осуществляются непосредственно на видовом экране. Для редактирования времени и значений отдельных ключей скольжения, перемещения и масштабирования, а также наборов ключей применяется функция Dopesheet [6].

Maya

Программный продукт Maya позволяет решать сложные производственные задачи по созданию персонажей и цифровой анимации. Maya предоставляет мощные интегрированные инструменты для 3D-анимации, моделирования, создания эффектов и рендеринга в рамках отлаженной расширяемой матрицы процесса создания компьютерной графики. В пакет Maya входит широкий спектр специализированных инструментов для создания персонажей, 3D-монтажа и работы с анимацией по ключевым кадрам, с процедурной и программируемой анимацией. Воплощайте в реальность свои идеи с инструментарием для 3D-анимации, используемым для создания наиболее правдоподобных 3D-персонажей. Программный продукт Maya представляет собой настраиваемое и расширяемое приложение, которое предлагает несколько вариантов интеграции рабочих процессов. Maya предоставляет расширенный набор инструментов по созданию сценариев и API-интерфейса (интерфейса прикладного программирования), оптимизированные рабочие процессы с определенными двумерными и трехмерными приложениями и инструменты для обработки больших наборов данных. Помимо этого, данный трехмерный редактор способен моделировать физику твердых и мягких тел, просчитывать поведение ткани, эмулировать

текучие эффекты, позволяет детально настраивать прическу персонажей, создавать сухой и мокрый мех, анимировать волосы. Визитной карточкой программы является модуль PaintEffects, который дает возможность рисовать виртуальной кистью такие трехмерные объекты, как цветы, траву, объемные узоры и прочее [7].

Cinema 4D

CINEMA 4D – это универсальная комплексная 3D программа, которая позволяет создавать и редактировать трёхмерные объекты и эффекты. Поддерживает как высококачественный рендеринг, так и анимацию. Инструменты скульптурного моделирования предоставляют пользователям высокий уровень контроля над деталями моделей. Этими инструментами художники могут царапать, шпаклевать, вытягивать, надувать, сжимать, а также производить другие действия над трёхмерными объектами. Кроме этого можно использовать в работе функции масок, штампов и симметрии. Многофункциональная система работы со слоями добавляет гибкости. Сложные объекты можно запечь и тем самым перевести миллионы полигонов в карты смещения и нормалей, которые применяются к низкополигональному объекту и тем самым экономят память и время на просчёт, что позволяет использовать объекты для анимации. Уникальная функция MoDynamics предоставляет возможность создавать фотореалистичные имитации большинства физических эффектов, таких как сила тяжести, трение, столкновение. К одним из самых значимых преимуществ Cinema 4D можно отнести достаточно простой интерфейс по сравнению с аналогичными программами. Его архитектура очень логична и интуитивно понятна даже новичку. Инструментарий программы постепенно совершенствовался и расширялся очень полезными дополнениями. Сегодня в Cinema 4D можно найти средства для создания персонажной анимации, удобную среду для

работы с частицами, мощную систему фотореалистичной визуализации и, конечно же, удобные инструменты моделирования. В последних версиях Cinema 4D существенно переработан алгоритм визуализации и расширены возможности обработки трехмерных сцен. Программа позволяет просчитывать эффекты глобальной освещенности, каустику и учитывает подповерхностное рассеивание света, которое можно наблюдать, например, при просвечивании воска свечи [8].

Blender

Blender распространяется в виде бесплатного пакета 3D инструментов, который практически не уступает по функционалу платным приложениям. Blender является кроссплатформенным, потребляет относительно малое количество памяти и требует меньше места на диске, по сравнению с другими программами для 3D-моделирования. Для визуализации своего интерфейса Blender использует OpenGL, что позволяет сохранять внешний вил пользовательского интерфейса одинаковым для всех поддерживаемых платформах. Важным недостатком пакета является отсутствие развёрнутой документации и большого количества демонстрационных сцен. Функции пакета: инструменты анимации, инверсная кинематика, скелетная анимация и сеточная деформация, анимация по ключевым кадрам, нелинейная анимация, редактирование весовых коэффициентов вершин, ограничители [9].

Итог

Сравнение конечных стоимостей лицензий сравниваемых редакторов за минимальный временной период, предоставляемый разработчиком, представлено в таблице 1.

Таблица 1 – Лицензии на пакеты 3D редакторов

Название	Лицензия	Цена, руб./месяц
«3Ds Max»	проприетарная	6 251,20
«Maya»	проприетарная	6 188,20
«Cinema 4D»	проприетарная	3 832,75
«Blender»	свободная	-

«Blender» обладает всем необходимым функционалом создания трехмерных моделей и анимации для разрабатываемого проекта. Главное преимущество выбранного пакета – свободная модель распространения.

1.4.2 Среда разработки

Unity – это кроссплатформенная движок для разработки двухмерных и трехмерных приложений и игр под различные платформы. У Unity распространяется в двух версиях: бесплатная и платная. Отличаются они рядом возможностей, которые могут сильно понадобиться при разработке игры. Во-первых, бесплатная версия Unity поддерживает только Android, Web Player, PC-платформы. Полная версия позволяет разработчикам выкладывать своё творение под все самые известные платформы, такие как: PC, Linux, Mac, Windows Store, IOS, Android, Windows Phone 10 Store, Blackberry 10, Wii U, PS3, Xbox 360, PS4, Xbox One. Есть возможность делать софт для VR(Virtual Reality), т.е. под очки и шлемы виртуальной реальности: Hololens, Oculus Rift, StarVR и прочие, а также писать программы для Kinect 2.0, LeapMotion. Полную версию Unity можно адаптировать под свои нужды: например, если вас интересует разработка софта под Kinect 2.0, вы можете убрать некоторые элементы меню в интерфейсе и дополнить его своими надстройками, которые реально облегчат разработку.

Unity имеет очень простой Drag and Drop интерфейс. Весь движок только на английском языке. Разделен на несколько окон: Hierarchy, где находятся

названия всех объектов на сцене, которые можно группировать и легко переходить по ним, Scene, где можно рассмотреть определенную сцену под нужным вам ракурсом, Inspector, который поможет с настройкой выделенного объекта, Project, где видны все материалы проекта, Toolbar (или меню с инструментами).

Unity поддерживает два языка: C#(наиболее используемый) и Javascipt. Некоторые моменты Unity делает только на одном из двух языков, или это делается намного труднее, чем на другом языке программирования. Предпоследняя версия Unity, а именно Unity 4, поддерживала язык программирования Boo(диалект Phyton), но его убрали из 5-ой версии, так как им практически никто не пользовался, да и документации, если честно, на официальном сайте Unity особо не было. Расчеты физики в Unity 5 производит та же NVIDIA PhysX.

Объекты в Unity могут быть пустыми, (чтобы объединить несколько объектов в одну группу и сделать их дочерними GameObject), содержать компоненты, с которыми взаимодействуют скрипты, могут быть названы одним и тем же именем, могут быть присвоены теги, которые служат для того, чтобы скрипт нашел нужный нам объект.

Unity поддерживает множество форматов для хранения трехмерных сцен, что позволяет интегрировать модели и анимации из различных редокторов: 3Ds Max, Maya, Blender и прочие. Также Unity обладает собственным редактором для работы с анимацией. Материалы в Unity 5 играют важную роль. Импортированные текстуры в Unity прикрепить к объекту нельзя, необходимо создать материал, который можно присваивать игровому объекту. К назначенным материалу шейдерам будут присвоены текстуры. Шейдеры можно редактировать прямо в Unity3d. Unity 5 позволяет генерировать нормал-мапы(normal-map), лайт-мапы(light-map), различные альфа-каналы и mip-уровни. В полной версии Unity 5 возможно полное настраивание шейдеров, а в бесплатной - нет.

Unity 5 имеет две очень важные функции: Occlusion Culling и Level Of Detail. Обе функции позволяют сильно снизить нагрузку на центральный процессор, благодаря высокой детализации. Например, в играх жанра 2D и 3D Runner при преодолении определенной дистанции все, что было позади вас, удаляется, а то, что впереди вас, генерируется. Occlusion Culling не визуализирует геометрию и коллайдеры объектов, находящихся не в поле зрении камеры, а Level Of Detail заменяет детализированные объекты, находящиеся далеко от игрока, на менее детализированные, причем разработчик сам настраивает эту систему.

Unity 5 обладает огромным количеством преимуществ перед другими игровыми движками. На сегодняшний момент сообщество Unity является самым большим в мире. На официальном сайте Unity есть специальный раздел, в котором отображена статистика по игровым движкам. По этим данным Unity 5 используют более 50% разработчиков видеоигр. 20% принадлежат Unreal Engine, а остальные игровые движки - 30%. Для разработки 2D или 3D инди-игр Unity 5 подходит по всем параметрам. Unity 5 позволяет создать один проект под множество платформ, что очень сильно облегчает процесс разработки.

2.Реализация проекта

2.1 Интеграция трехмерных моделей и анимации в Unity

Главный персонаж и противники имеют общую модель персонажа, созданную при помощи стороннего пакета ПО MakeHuman. Для визуального отличия моделей главного персонажа присваивается зеленый цвет, для противников – красный. Общая модель персонажей представлена на рисунке 1.

Рисунок 1 – Трехмерная модель персонажа

Оснастка

Оснастка – это процесс подготовки персонажа к анимации, включающий создание и размещение внутри трёхмерной модели рига, виртуального скелета – набора костей или суставов, установления иерархической зависимости между ними и значений возможных трансформаций для каждой из этих костей.

Скелетная анимация, для которой и применяется оснастка, эффективна прежде всего тем, что позволяет манипулировать большим количеством составных элементов анимируемой фигуры (конечности, глаза, мышцы лица, губы и т.д.) с помощью относительно малого количества управляющих элементов: костей и их регулируемых характеристик.

Структура сочленений виртуального скелета внешне очень похожа на сочленения скелета человека. От сложности этой структуры напрямую зависит, насколько она будет гибкой, и насколько реалистичной получится анимация. С другой стороны, чем больше костей в структуре рига, тем сложнее будет с ним работать.

Чтобы созданный скелет мог быть интегрирован в Unity, необходимо соблюдать определенную иерархию костей скелета. Такое требование

обусловлено созданием человеческого аватара оснастки в инструменте Mecanim, отвечающего за работу с анимационными клипами в Unity. В дальнейшем созданный аватар можно применить для других интегрированных моделей с аналогичной оснасткой, что также позволит привязать ранее интегрированную анимацию к новым моделям. Иерархия скелета должна состоять из 15 костей с корневой тазовой костью. Помимо основных 15 костей существует опция добавления дополнительных суставов для работы с пальцами рук, мышцами лица и глазами. Структура иерархии основных костей представлена на рисунке 2.

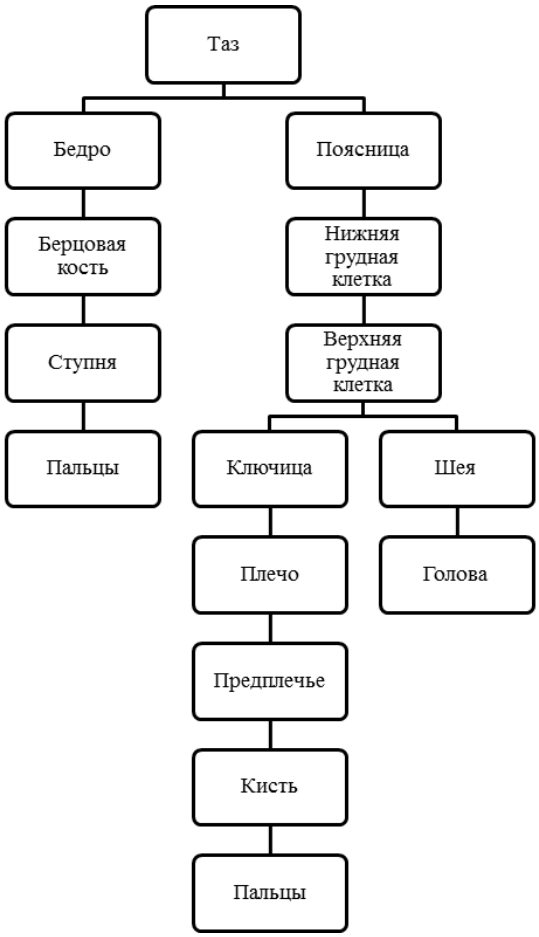

Рисунок 2 – Иерархия костей для интеграции в Unity

Скелет, созданный согласно иерархии приведенной на рисунке 3, изначально поддерживает прямую кинематику (FK). Принцип прямой кинематики состоит в передаче воздействия на элемент иерархической цепочки сверху вниз, то есть дочерние сегменты движутся относительно родительских. Сначала положение и ориентацию меняет родительский сегмент, вследствие чего происходит изменение положения и ориентации всех дочерних сегментов затронутого сегмента. Далее изменяется положение следующего сегмента в цепочке, при этом изменяется положение всех последующих дочерних к нему сегментов, в то время как родительские сегменты остаются неподвижными.

Альтернативой прямой кинематики является более эффективный механизм - Инверсная кинематика (IK). В инверсной кинематике дочерний сегмент, движение которого изменяет положения и ориентацию других объектов и который расположен в середине отдельной иерархической цепочке сегментов, называется эффектором. Если эффектор является конечным объектом рассматриваемой иерархической цепочки, то он называется конечным эффектором. Именно через эффектор осуществляется манипулирование всей иерархической цепочкой. Изменение положения и ориентации конечного эффектора приводит к изменению положения и ориентации всех сегментов иерархической цепочки по законам инверсной кинематики. Изменение положения и ориентации не конечного эффектора приводит к изменению положения объектов, стоящих по иерархии ниже его, меняется по законам прямой кинематики, а объектов с более высокой иерархией — по законам инверсной кинематики.

Для организации инверсивной кинематики создаются дополнительные кости: корневая, кистевая цель, локтевая цель, ступневая цель, коленная цель. Всем целевым костям и тазовой кости в качестве родителя присваивается корневая кость. По итогу тазовая кость, локтевая и коленная цели становятся простыми эффекторами, а ступневая и кистевая цели – конечными. Реализованный скелет представлен на рисунке 3.

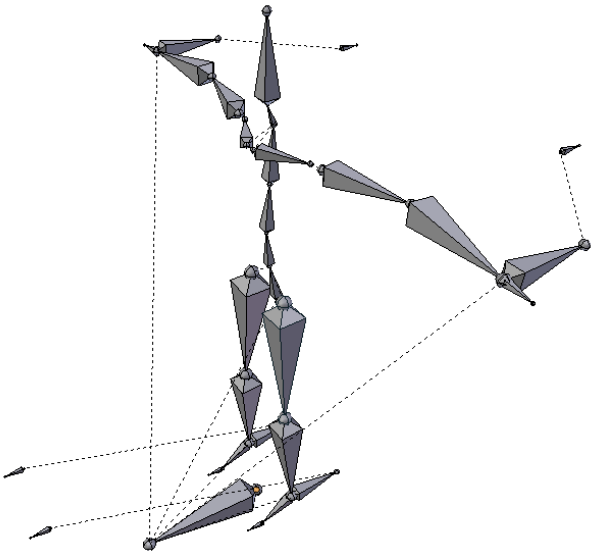

Рисунок 3 – Скелет персонажа

Скиннинг

Это процесс привязки вершин модели к скелету. Делается это для того, чтобы при движении скелета двигалась и сама модель персонажа. Вершина может быть связана с костью напрямую (жёсткая привязка) или с несколькими костями, используя на них смешанные воздействия (мягкая привязка). Изначальная настройка автоматизирована. Дальнейшая настройка производиться с помощью тепловой карты (heatmap). Работа с тепловой картой представлена на рисунке 4.

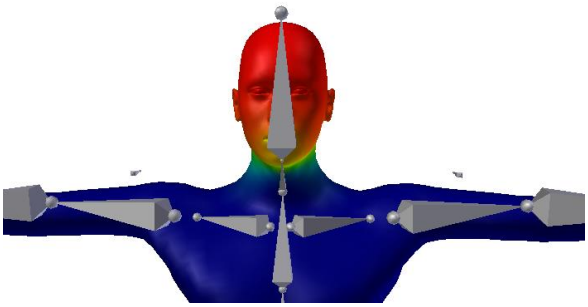

Рисунок 4 – Скиннинг персонажа

Далее с помощью элементов управления скелета создаются анимации персонажей и импортируются в Unity. Список интегрированных анимаций представлен на рисунке 5.

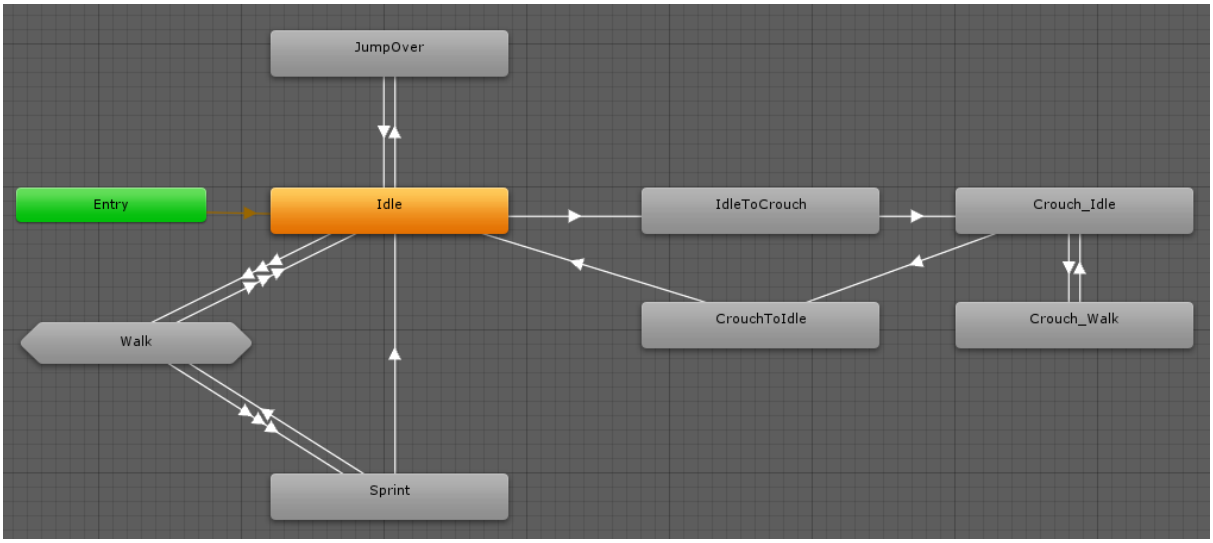

Рисунок 5 – Все возможные состояния аниматора персонажа

3.2 Искусственный интеллект

С самых первых дней зарождения игровой индустрии искусственный интеллект стал неотъемлемой частью практически любой компьютерной игры. В связи с тем, что сегодня многие аспекты современных игр достигли удивительной степени совершенства, внимание разработчиков все больше и больше сосредоточивается на усовершенствовании средств искусственного интеллекта. На простейшем уровне «искусственный интеллект» заключается в моделировании или имитации поведения других игроков или объектов, то есть всех элементов игры, которые могут действовать или с которыми может действовать игрок. Основной принцип состоит в том, что это поведение имитируется. Другими словами, ИИ для игр является более «искусственным», нежели «интеллектом». Система ИИ может быть крайне проста и представлять

собой набор правил или же может быть довольно сложной и выполнять роль командующего армии противника, с которой предстоит сражаться игроку.

С точки зрения игр реалистичный ИИ далеко выходит за рамки требований развлекательного программного проекта. Игровой ИИ не должен быть наделен чувствами и самосознанием, ему нет необходимости обучаться чему-либо за пределами рамок игрового процесса. Главной целью ИИ в играх является имитация разумного поведения и в предоставлении игроку убедительной, правдоподобной задачи.

При создании компьютерного оппонента в играх возникает необходимость имитации разумного поведения противника для создания более убедительной и правдоподобной игровой ситуации. Основа искусственного интеллекта в играх состоит из базового восприятия и принятия решений. Существуют различные подходы к реализации системы искусственного интеллекта в играх, из которых можно выделить основные: система на основе правил, автомат в качестве искусственного интеллекта. Они в свою очередь различаются по сложности реализации и разумности интеллекта.

Одной из главных проблем при создании компьютерных игр является разработка искусственного интеллекта, способного противостоять человеческому противнику. Компьютер должен принимать осмысленные решения на основе восприятия и оценки обстановки в игре на данный момент. Для решения этой задачи существуют различные методы, один из которых – это метод определения разности координат между объектом, управляемым искусственным интеллектом, и его целью [10].

2.2.1 Состояния ИИ

Искусственный интеллект с конечным числом состояний является способом моделирования и реализации объекта, обладающего различными состояниями в течение своей жизни. Каждое состояние может представлять

физические условия, в которых находится объект, или, например, набор эмоций, выражаемых объектом. Здесь эмоциональные состояния не имеют никакого отношения к эмоциям ИИ, они относятся к заранее заданным поведенческим моделям, вписывающимся в контекст игры.

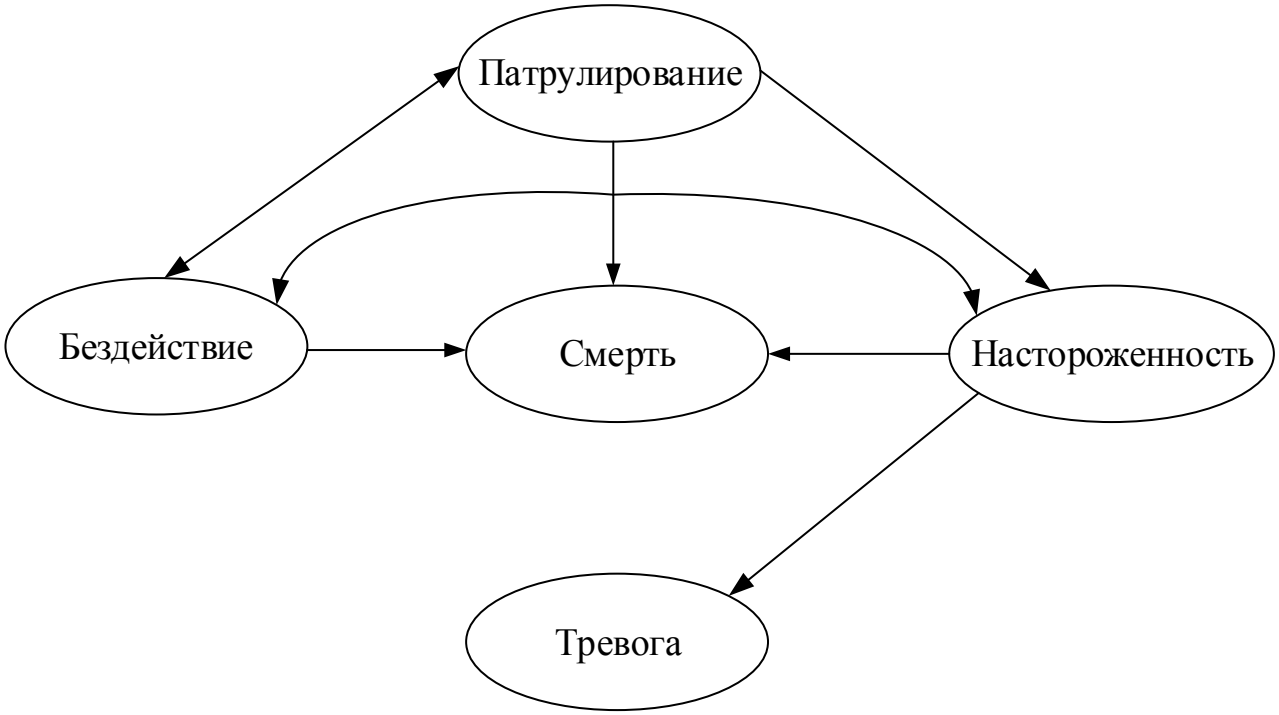

Рисунок 6 – Состояния ИИ

Реализованный в этом проекте искусственный интеллект имеет 5 различных состояний. Без каких-либо вмешательств со стороны игрока ИИ будет выполнять алгоритм охранника, основанный на последовательном переключении между состояниями бездействие и патрулирование. Процесс патрулирования заключается в перемещении между точками на уровне и высматривании неприятеля (игрока) на пути. По достижению точки ИИ выжидает несколько секунд в состоянии бездействия, осматриваясь при этом по сторонам.

Переключение на состояние настороженности обусловлено обнаружением следов присутствия в окрестностях уровня игрока.

Переключаясь в состояние настороженности, ИИ начинает проверять окрестности на предмет наличия игрока. Если игрок не обнаружен, то происходит переключение в состояние бездействия, а затем продолжается процесс патрулирования окрестностей по прежнему маршруту.

В состояние тревоги ИИ переходит после обнаружения игрока. Обнаружение игрока патрульным влечет за собой провал задания, что объясняет отсутствие переходов из состояния тревоги. Состояние тревоги служит сообщением об окончании уровня для игрового менеджера, отвечающего за организацию игрового процесса.

2.2.2 Ориентация ИИ

Ориентация искусственного интеллекта на уровне реализована при помощи неориентированного графа, содержащего основные точки перемещения охранников по уровню. Изображение графа для ориентации на уровне данного проекта представлено на рисунке 7.

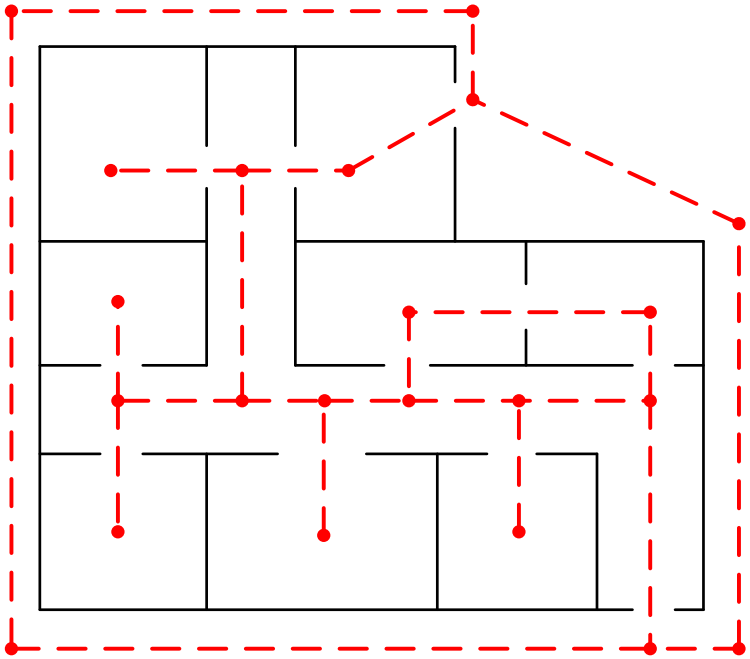

Рисунок 7 – Граф точек передвижения ИИ по уровню

Каждому экземпляру реализованного ИИ выдается коллекция точек, передвигаясь по которым ИИ осуществляет патрулирование уровня. Каждая точка такой коллекции принадлежит графу, описывающему возможные пути передвижений по уровню (рельсы). Граф хранится в статичном классе, содержащим список вершин и матрица смежностей, где номерами вершин служат индексы из списка вершин. Нахождения пути в графе осуществляется при помощи алгоритма поиска в глубину.

2.2.3 Обнаружение игрока

Зрение охранников реализовано в виде сектора круга с фиксированным угловым значением в 140 градусов с максимальной дистанцией обнаружения – десять метров.

Каждый противник обладает информацией о текущем местоположении игрока. Сначала проверяется угол между вектором прямой видимости охранника и вектором, направленным от охранника непосредственно к игроку. Если измеренный угол меньше 70 градусов, то игрок попадает в угол видимости данного охранника. Далее проверяется расстояние от охранника до игрока. Если игрок расположен менее чем в восьми метрах от охранника, то игрок попадает в зону прямой видимости охранника. Затем осуществляется проверка на нахождение препятствий между охранником и игроком. С помощью физического движка прокладывается луч от охранника до игрока, и, если на пути луча не встречается ни один физический коллайдер, то игра окончена – игрок обнаружен.

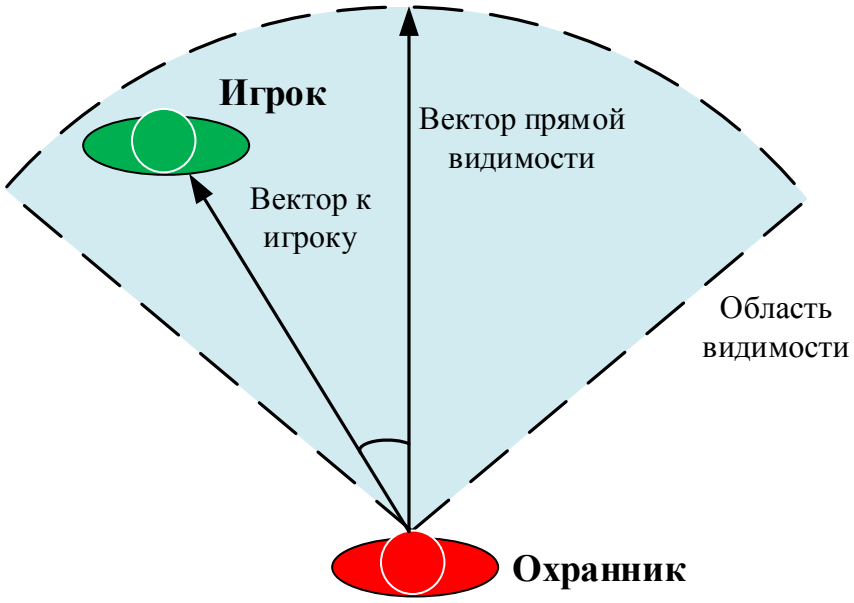

Рисунок 8 – область видимости главного персонажа

3 Архитектура классов

Основными составляющими игровой механики являются следующие объекты:

– главный персонаж, находящийся под управление игрока;

– противники, находящиеся под управлением искусственного интеллекта;

– портфель с документами;

– двери;

– окна.

Весь функционал вышеописанных объектов реализуют классы, представленные на рисунке 9.

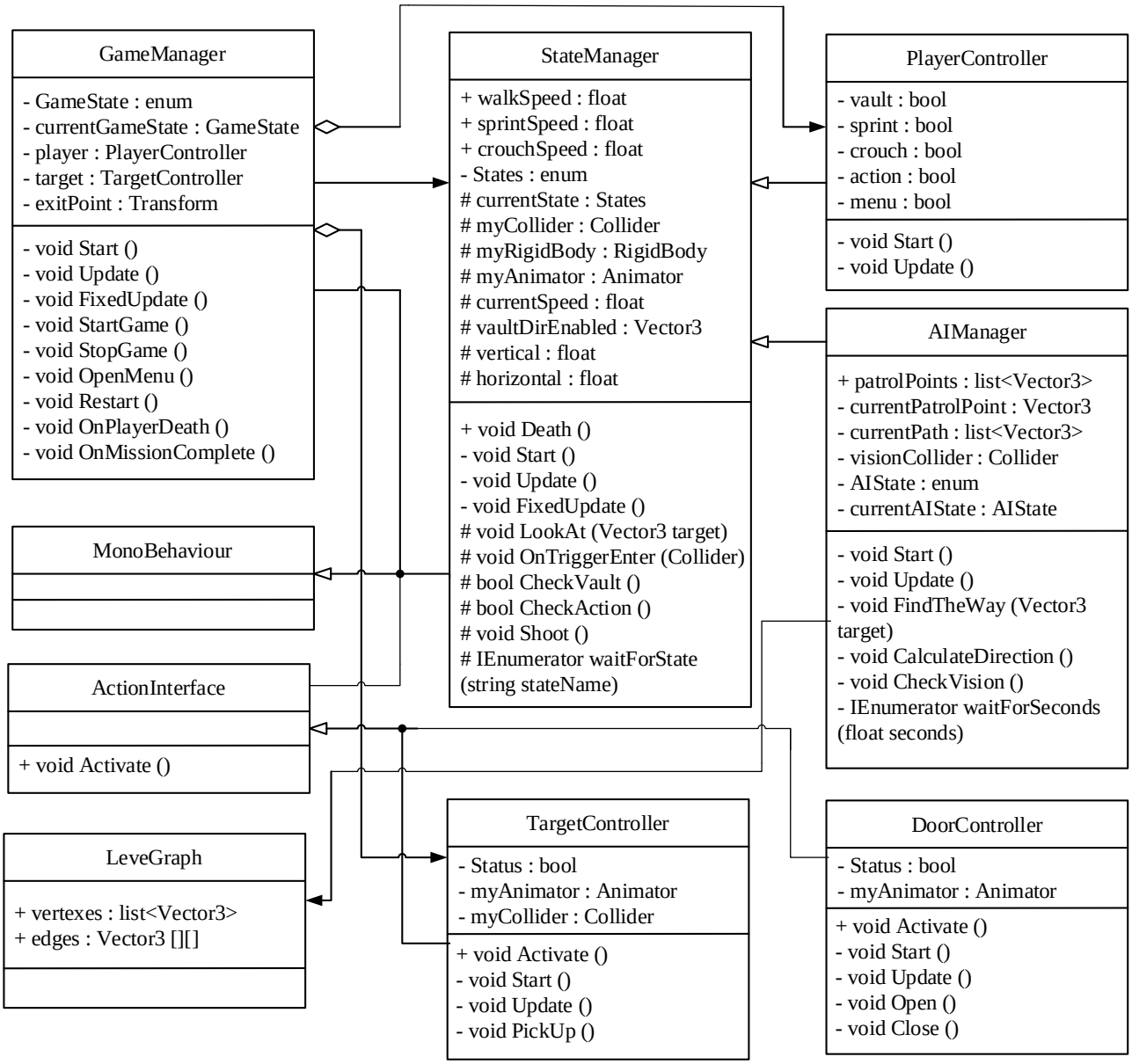

Рисунок 9 – UML диаграмма классов

Как видно из диаграммы, 3 класса наследуются от *MonoBehaviour*. *MonoBehaviour* – это базовый класс, от которого наследуются все скрипты в среде разработки Unity. *MonoBehaviour* предоставляет доступ к работе с базовыми функциями и компонентам игрового и физического движков. Все используемые в данном проекте функции базового класса *MonoBehaviour* и описание их функционала указаны в таблице 2.

Таблица 2 – Используемые методы базового класса Monobehaviour

Метод	Назначение
Start ()	Функция *Start ()* вызывается при активации экземпляра *MonoBehaviour* перед первым вызовом функции *Uppdate ()*.
Update ()	Функция *Update ()* вызывается каждый кадр, если экземпляр *MonoBehaviour* активен.
FixedUpdate ()	Функция *FixedUpdate ()* вызывается с фиксированной частотой кадров, синхронизируясь с частотой работы физического движка.
OnTriggerEnter (Collider)	Функция вызывается *OnTriggerEnter (Collider)*, когда привязанный к текущему экземпляру *MonoBehaviour* физический коллайдер соприкасается со сторонним коллайдером-триггером.
StartCoroutine (IEnumerator routine)	Функция *StartCoroutine (IEnumerator routine)* выполняет запуск сопрограммы (корутины).
StopCoroutine (string methodName)	Функция *StopCoroutine (string methodName)* останавливает все сопрограммы (корутины) с именем *methodName*, запущенные на этом экземпляре *MonoBehaviour*.

3.1 Функциональное описание классов

3.1.1 GameManager

Выполняет общую организацию игрового процесса. Открывает главное меню по запуску приложения. По нажатию кнопки «*Start Game*» подгружает сцену с уровнем и запоминает ссылки на экземпляры классов *PlayerController* и *TargetController*. Отслеживает состояния экземпляра *PlayerController*. В случае смерти игрока выдает сообщение об окончании игры с возможностью перезапуска (*OnPlayerDaeth()*). Отслеживает нажатие игроком кнопки вызова

главного меню и включения паузы (*StopGame ()*). *GameManager* статус экземпляра *TargetController* и выводит на экран сообщение об успешном завершении уровня по достижению игрока точки выхода (*ExitPoint).*

3.1.2 StateManager

StateManager является абстрактным классом, реализующим функции персонажей по передвижению в пространстве, взаимодействию с интерактивным окружением и воспроизведению анимации.

Как передвижение, так и обнаружение интерактивных объектов реализовано посредством физического движка. Скорость и направление движение персонажа задается при помощи свойства *myRigidBody*. С помощью физического коллайдера (*myCollider*) производится обнаружение коллизий с коллайдерами-триггерами интерактивных объектов (окна, двери, портфель с документами, спины охранников) и дальнейшее взаимодействие с этими объектами. В случае обнаружения интерактивного объекта и получения команды на начало взаимодействия, происходит вызов метода *Activate()* класса ActionInterFace, если обнаруженный объект – дверь, окно или портфель. Если же обнаруженный триггер – это спина противника, то происходит вызов метоа *Death()* класса *StateManager*. Помимо обнаружения триггеров, коллайдер играет роль физического тела персонажа, что не позволяет ему проваливаться через пол и проходить сквозь стены.

Воспроизведение анимации осуществляется посредством изменения состояний аниматора (*myAnimator*). Параллельно с изменением состояния аниматора изменяется состояние (*currentState*) данного экземпляра *StateManager*. Список возможных состояний включает в себя: *Idle* (бездействие), *Walk* (ходьба), *Sprint* (бег), *Crouch* (передвижение крадучись), *Vault* (вскарабкивание), *Dead* (смерть). Механизм состояний позволяет избежать конфликтов при взаимодействии с окружением. Так, например, при

воспроизведении анимации вскарабкивания в окно блокируются какие-либо другие действия. Для контроля окончания процесса вскарабкивания запускается сопрограмма, которая не блокирует основной поток и ожидает переключения аниматора в следующее состояние, а затем переключает состояние *StateManager*. Состояние *Dead* устанавливается в функции *Death()* и влечет за собой блокировку данного экземпляра *MonoBehaviour*.

3.1.3 PlayerController

Является потомком класса *StateManager*. Выполняет обработку нажатых игроком клавиш, а затем вызывает переключение состояний *StateManager* на основе полученных от игрока команд. В проекте создается максимум один экземпляр класса *PlayerController*.

3.1.4 AIManager

Является потомком класса *StateManager*. Выполняет функцию патрульного – охранника портфеля с документами. Главная задача патрульного – не допустить проникновения нарушителя (игрока) на охраняемую территорию и предотвратить похищение портфеля с документами.

В качестве инструмента для ориентации на уровне используется статический класс *LevelGraph*, который содержит в себе неориентированный граф. С помощью графа в методе *FindTheWay(Vector3 target)* прокладывается маршрут (коллекция точек *currentPath*), по которому персонаж следует до необходимой точки. Перечень точек, по которым патрульный совершает свой обход территории, содержится в коллекции *patrolPoints*.

AIManager имеет 4 состояния: *Idle* (бездействие), *Patrol* (патруль), *Watch* (настороженность) и *Alert* (тревога). В режиме патруля охранник проходит поочередно по точкам из коллекции *patrolPoints*. По достижении каждой из

патрульных точек охранник переходит на несколько секунд в состояние бездействия и осматривается по сторонам. Задержка осуществляется при помощи сопрограммы *waitForSeconds(float seconds)*. В состояние настороженности осуществляется переход в случае нахождения триггеров: открытого окна, открытой двери или охранника без сознания. После обнаружения охранником триггера выбираются несколько соседних точек из графа *LevelGraph* и по выбранным точкам осуществляется обход аналогичный режиму патрулирования. Если в режиме настороженности не был обнаружен игрок, то охранник переходит в режим патрулирования. В случае обнаружения игрока, охранник переходит в состояние тревоги и открывает огонь по игроку. Выстрел происходит при помощи метода *Shoot()* класса *StateManager*. В методе *Shoot()* в направлении игрока от охранника прокладывается луч (*raycast*). Далее происходит вызов метода *Death()* класса *StateManager* для экземпляра класса *PlayerController*.

Обнаружение игрока и триггеров перехода в состояние настороженности производится при помощи физического движка. С каждым тактом проверяется вхождение в прикрепленный к охраннику физический коллайдер иного коллайдера с тегом *Player*.

3.1.5 ActionInterface

ActionInterface – это интерфейс, описывающий метод взаимодействия с интерактивными объектами (окна, двери, портфель). Потомками *ActionInterface* являются классы *DoorController* и *TargetController*.

Класс *DoorController* выполняет функции окна и двери. Открытие и закрытие осуществляется с помощью методов *Open ()* и *Close ()*. В функции открытии двери/окна происходит смена свойства-статуса *Status* и смена состояния аниматора *myAnimator*, воспроизводящего анимацию открытия и

закрытия двери/окна. Метод *Activate ()* вызывается персонажем, взаимодействующим с экземпляром *DoorController ()*.

Класс *TargetController* выполняет функцию взаимодействия с портфелем. Метод *PickUp ()* меняет значение свойства *Status* на противоположное и запускает анимацию исчезновения портфеля, изменяя состояние аниматора *myAnimator*. Вызов функции *PickUp ()* производится из метода *Activate ()* по запросу игрока, чей коллайдер контактирует с коллайдером-триггером портфеля.

ЗАКЛЮЧЕНИЕ

В ходе выполнения проекта была разработана одиночная трехмерная игра в жанре стэлс-экшен на платформе «Unity» для персональных компьютеров на базе операционных систем Windows. Также для игры был разработан искусственный интеллект для управления противником.

СПИСОК СОКРАЩЕНИЙ

NPC – Non-Player character

ИИ – Искусственный интеллект

IDE - Integrated development environment

FK – Forward kinematics

IK - Inverse kinematics

ПО – Программное обеспечение

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Mike Gnade, What Exactly is an Indie Game? [Электронный ресурс] - Режим доступа: http://www.indiegamemag.com/what-is-an-indie-game/
2. Laura Parker, The Rise of the Indie Developer. [Электронный ресурс] - Режим доступа: https://www.gamespot.com/articles/the-rise-of-the-indie-developer/1100-6298425/
3. Clive Thompson, Hide and Go Sneak. [Электронный ресурс] - Режим доступа: http://www.slate.com/articles/technology/gaming/2004/07/hide-and-go-sneak.html
4. Gaming's most important evolutions. [Электронный ресурс] - Режим доступа: https://www.gamesradar.com/gamings-most-important-evolutions/
5. Keith Stuart, Jordan Erica Webber, 16 trends that will define the future of videogames. [Электронный ресурс] - Режим доступа: http://www.theguardian.com/technology/2015/jul/23/16-trends-that-will-change-the-games-industry
6. Возможности трехмерного редактора 3DS Max [Электронный ресурс] - Режим доступа: https://www.autodesk.ru/products/3ds-max/overview
7. Возможности трехмерного редактора Maya [Электронный ресурс] - Режим доступа: https://www.autodesk.ru/products/maya/overview
8. Cinema 4D Overview [Электронный ресурс] - Режим доступа: https://reviews.financesonline.com/p/cinema-4d/
9. Ключевые особенности трехмерного редактора Blender [Электронный ресурс] - Режим доступа: https://docs.blender.org/manual/ru/dev/index.html
10. Donald Kehoe, Designing Artificial Intelligence for Games. [Электронный ресурс] - Режим доступа: https://software.intel.com/en-us/articles/designing-artificial-intelligence-for-games-part-1