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

Институт Космических и информационных технологий
Кафедра Вычислительная техника

УТВЕРЖДАЮ
Заведующий кафедрой
_____ А.И. Легалов
подпись инициалы, фамилия
«___» _____ 20__ г.

БАКАЛАВАРСКАЯ РАБОТА

09.03.01 Информатика и вычислительная техника
код – наименование направления

Игра «Танки 2016» для ОС Android
тема

Руководитель _____ _____ А.Ю. Сидоров
подпись, дата должность, ученая степень инициалы, фамилия

Выпускник _____ А.В. Термер
подпись, дата инициалы, фамилия

Нормоконтролер _____ В. И. Иванов
подпись, дата инициалы, фамилия

Красноярск 2016

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Индустрия видеоигр зародилась в середине 1970-х годов как движение энтузиастов и за несколько десятилетий выросла из небольшого рынка в отрасль с годовой прибылью в 88 миллиарда долларов 2015 году (согласно ежегодным отчётам ESA). На рынке работают как крупные игроки, так и небольшие фирмы и стартапы, а также независимые разработчики и сообщества.

Современные игры — одни из самых требовательных приложений. Многие мощные устройства покупаются геймерами, которые требуются для запуска новейших игр, в которых используются самые передовые технологии. Таким образом, игровая индустрия тесно связана с индустрией производства центральных процессоров и другие компонентов, так как игры зачастую требуют более высоких аппаратных мощностей, чем бизнес-приложения.

В настоящее время видеоигры вносят значительный вклад в мировую экономику ввиду большого успеха продаж основных игровых систем и игр вроде Call of Duty: Black Ops, заработавшая в течение первых 5 дней продаж более $600 млн, что стало мировым рекордом пятидневных продаж среди фильмов, книг и видеоигр [1].

С появлением мобильных операционных систем, открылся новый рынок для разработчиков видеоигр и в настоящий момент мобильный рынок является одним из самых быстроразвивающихся в области разработки видеоигр. Так количество приложений в магазине мобильных приложений Google Play превысило полтора миллиона активных приложений (активные – приложения которые можно загрузить в данный момент, за все время существования магазина было опубликовано на много больше продуктов) в магазине Itunes насчитывалось более одного миллиона активных приложений из которых примерно 50% — игры. Также категория видеоигр являлась самой быстрорастущей в 2014 году в Google Play [2].

Видеоигры являются одной из драматических форм, а их интерактивность — это вопрос степени участия, но не формы. Поэтому, как и другие формы, видеоигра имеет пять ключевых элементов: стиль, фабула, герой, декорации и тема. Все хорошие игры должны обладать некоторым развлекательным потенциалом, и в большинстве их он основан на классических законах драмы.

По сообщению сайта, 3DNews, в 2011 году видеоигры были официально признаны правительством США и американским Национальным фондом искусств отдельным видом искусства, наряду с театром, кино и другими. После этого разработчики получили право, наравне с представителями кинематографа, музыки, живописи и литературы, рассчитывать на государственные гранты в размере от 10 до 200 тыс. долларов. Данная финансовая поддержка позволит независимым специалистам и компаниям значительно активней реализовать концептуальные проекты.

Видеоигры оказали столь существенное влияние на общество, что в информационных технологиях отмечена устойчивая тенденция к геймификации для неигрового прикладного программного обеспечения [3].

1 Анализ предметной области

1.1 Анализ на проектирование

Цель проекта: разработка игры для мобильной платформы Android.

Задачи проекта:

– Анализ игры, созданной в 1985 году «Battle city»;

– Реализовать модель движения танков по двум плоскостям;

– Реализовать возможность стрельбы;

– Реализовать возможность временной приостановки игры;

– Реализовать систему, позволяющую генерировать большое количество разнообразных уровней;

– Реализовать возможность игры в offline против ИИ.

1.2 Техническое задание на проект

В процессе выполнения работы должен быть проведен анализ игры «Battle City» и на основе идеи и сеттинга данной игры, разработать собственную игру под мобильные устройства для ОС Android. Физика стрельбы и модель передвижения танков должна осуществляться по двум плоскостям. Игра предназначена для одиночной игры, танками противниками должен управлять ИИ. Также должна быть реализована система, позволяющая генерировать большое количество разнообразных уровней.

Приложение разрабатывается под мобильные устройства под управлением ОС Android с наличием сенсорного экрана, минимальная поддерживаемая версия – 4.0, также должно иметь размер менее 50 МБ и стабильно работать на устройствах, имеющих ЦП с частотой 1 ГГц и ОЗУ в объеме 1 ГБ.

1.3 Игра «Battle city»

Battle City— видеоигра для игровых приставок Famicom и Game Boy, интерфейс которой представлен на рисунке 1. Судя по содержимому ПЗУ игры, программировали её три человека: Дзюнко Одзава, Рёити Окубо и Такэфуми Хёдо (их имена встречаются в самом начале PRG-ROM'а игры и в конце в виде ASCII-графики иероглифов ['RYOUITI OOKUBO TAKEFUMI HYOUDOU JUNKO OZAWA']).

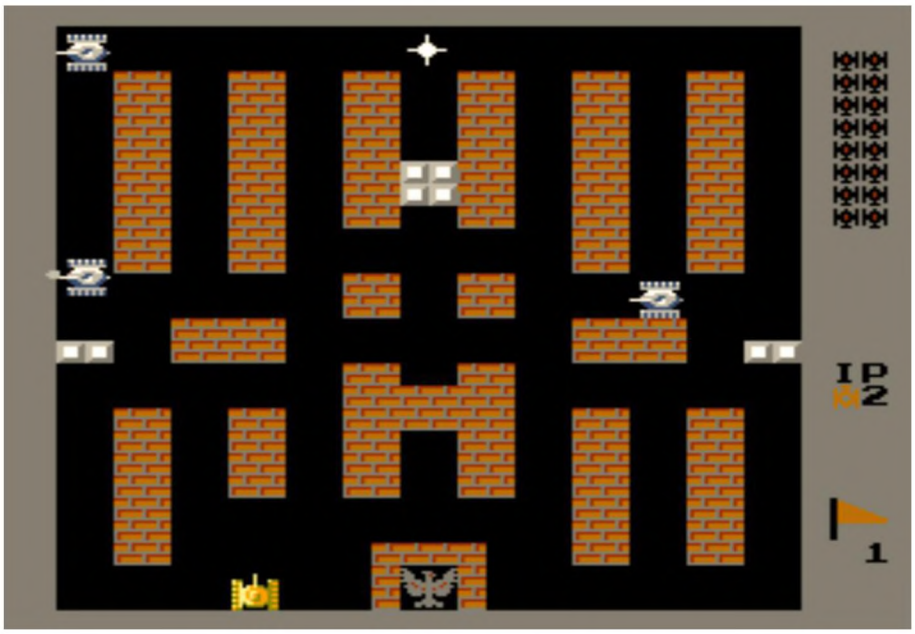

Рисунок 1 – игра «Battle City»

Игра состоит из 35 основных уровней. Уровень — квадратное поле из 169 секторов (13×13). Как и во многих ранних играх NES, в Battle City нет никакого сюжета и логической концовки. После 35-го уровня (в версии для Game Boy — после 100-го) игра продолжается снова с первого уровня, однако изменяется набор врагов для каждого уровня (набор последнего уровня) и время между генерацией врагов уменьшается до минимального. Если уровни проходить по третьему кругу, то вышеперечисленные изменения исчезают.

Правила таковы, что игрок должен, управляя своим танком, уничтожить все вражеские танки на уровне, которые постепенно появляются вверху игрового поля. Враги пытаются уничтожить базу игрока (внизу игрового поля в виде орла) и его танк. На каждом уровне нужно уничтожить около двадцати единиц бронетехники противника разных видов. Если противник сможет разрушить базу или лишит игрока всех жизней — игра окончена, если же игроку удастся уничтожить все танки противника – загружается следующий уровень.

«Battle City» — очень необычная игра для своего времени. Она сильно выделяется среди игр «первой волны Nintendo». Не платформер, не драка, а стрелялка на двоих. Простой, но очень увлекательный геймплей.

Контент

Несмотря на то, что игра была разработана более 30 лет назад, в ней имеются различные типы преград и ландшафта, улучшения танков и различные бонусы, которые можно активировать во время игры.

Существует всего 5 видов препятствий:

– Обычная стена – стенка, которая может быть разрушена танком с одного или нескольких выстрелов, в зависимости от своего уровня;

– Бетонная стена – данное препятствие невозможно разрушить обычным выстрелом, но ее можно уничтожить улучшенным танком;

– Кусты – препятствие через которое игрок может проехать и стрелять, но снижающее видимость;

– Лед – через это препятствие также можно проехать и стрелять, но управляемость танка будет снижена;

– Вода – блокирует передвижение, но пропускает снаряды.

Всего 5 видов препятствий, но каждое из них способно изменить тактику, которую игрок будет применять для того чтобы выиграть в бою.

Другой частью игры является возможность улучшения танка игрока, для чего необходимо собирать звезды, которые появляются в случайном месте, всего существует 4 варианта модификации танка:

- Без звезд – малый танк с минимальной скоростью полета снаряда;
- 1 звезда – легкий танк с высокой скоростью полета снаряда;
- 2 звезды – средний танк с возможностью стрелять очередями;
- 3 звезды – тяжелый танк, способный пробивать бетонные стены.

Как видно из этих бонусов, они влияют только на ударную силу танка, а уничтожить его можно с 1 выстрела, после чего танк перерождается около базы в виде малого танка.

В игре имеется четыре типа вражеских танков, за уничтожение которых игроку начисляются очки, танки противника отличаются скоростью и прочностью:

- Обычный танк – имеет стандартную скорость и уничтожается с одного выстрела (100 очков);
- Бронетранспортер – танк, имеющий повышенную скорость хода, но уничтожается также с одного выстрела (200 очков);
- Скорострельный танк – стреляет очередями (300 очков);
- Тяжелый танк – танк с повышенной прочностью, уничтожается после 4 попаданий игрока (400 очков).

На каждой карте может быть не более 20 танков, но одновременное их количество ограничено 4 для режима однопользовательской игры и 6 для игры вдвоем.

Кроме того, имеются бонусы:

- Дополнительная жизнь;
- Звезда – для улучшения танка;
- Бомба – взрывает все танки противника;
- Часы – на 10 секунд останавливает врагов;

– Лопата – на 10 секунд делает кирпичную стену штаба игрока бетонной, что защищает его от вражеских снарядов. Если до активации бонусы, кирпичная стены вокруг штаба была уничтожена, после прекращения бонуса она будет восстановлена.

– Каска – на 10 секунд делает танк игрока неуязвимым.

Всего за игру возможно взять 3 бонуса, которые появляются после уничтожения 4, 11 и 18 танков [4].

Система генерации контента

Battle city – это игра написанная более 30 лет назад, в то время перед разработчиками стояли проблемы с использованием памяти из-за ее малых объемов, что несомненно наложило отпечаток на систему хранения и генерации контента в данной игре, но благодаря тому, что исходные коды игры были декомпилированы, есть возможность исследовать то, как игра хранит и использует ресурсы.

Карты в Battle City состоят из тайлов — блоков размером 8x8 пикселей. Весь фон, видимый на экране, является картой.

Однако, в сериализованном виде уровни хранятся более компактно: для этого тайлы компонуются в блоки размером 2x2 тайла. Всего имеется 16 разновидностей блоков, которые приведены в таблице ниже. Хранится только основная часть карты (13x14 блоков), по которой и ездят танки — нет смысла сохранять статичные стены и вспомогательную информацию. Для записи одного блока используется 4 бита, таким образом, вся карта занимает 91 байт. Виды блоков и их коды представлены на рисунке 2.

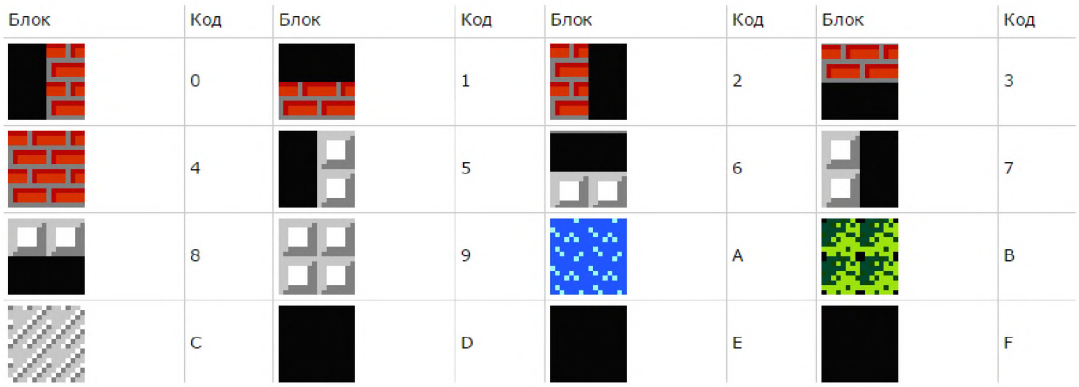

Рисунок 2 – Виды блоков и их коды в игре «Battle City»

В отличии от блоков, разновидностей самих тайлов гораздо больше, а именно — 256, т.е. ровно столько, сколько вмещает одна страница знакогенератора — участок видеопамяти, где каждому тайлу соответствует индекс от 0 до 255. Тайлы используются как для создания окружения уровня, так и для отображения информации о количестве вражеских танков, жизней и т.п. Но непосредственно в элементах уровня их используется всего 22 — из них 6 для формирования перечисленных блоков, остальные 15 — дополнительные тайлы кирпичных блоков.

На каждой карте имеются фиксированные точки появления танков и орла, который охраняется игроком, из-за чего объекты будут появляется на своих местах независимо от того, что находится в местах их появления.

Как уже говорилось выше, в оригинальной игре имеется всего 6 бонусов, однако перед генерацией бонуса игра определяет какой из них должен появится и происходит это делением по модулю случайного числа на 8, в результате чего получается номер от 0 до 7, которое является индексом в таблице бонусов, где место свободных бонусов повторно заняли звезда и граната. В итоге вероятность их выпадения равна ¼, в то же время вероятность выпадения 4 других бонусов равна 1/8 [5].

Так в игре существовало 35 различных уровней, но кроме того, Battle City – одна из первых игр, в которой появился встроенный редактор карт, так игрок мог создать свой уровень, расставить различные блоки препятствий, но точки

появления танков противника, место появления игроков и базы изменить было нельзя. Также в связи с ограничением памяти, не было возможности сохранить созданную карту.

Интеллект противников

Как уже говорилось выше, при совместной игре или при повторном прохождении игра усложняется, также сложность увеличивается с каждый следующим уровнем – это происходит путем уменьшения задержки между появлением врагов. Время респауна в кадрах для заданного уровня сложности и количества игроков можно вычислить по следующей формуле: 190 – уровень * 4 – (количество игроков – 1) * 20. Считая, что игра происходит при 60 кадрах в секунду, чтобы получить время появления врагов в секундах, нужно умножить результат на 60.

Существует три периода поведения танков: сначала они движутся хаотично, затем они преследуют игроков, и, в конце концов, начинают атаковать штаб. Длительность первых двух периодов одинакова и равна восьми периодам респауна. Т.е., поделив время респауна на 8, мы получим длительность в секундах (или, умножив на 8, получим то же самое время в кадрах) — например, для первого уровня и одного игрока это будет 23 секунды. Третий же период будет длиться до тех пор, пока счётчик секунд не обнулится (достигнув 256), и цикл периодов не начнётся заново.

Вся вражеская тактика основана на системе команд, для чего выделено 8 байтов: два под танки игроков и шесть под вражеские танки. Четыре старших бита такого байта используются под команду, оставшиеся четыре младших — под её аргументы (как правило это направление движения). В итоге получается 16 команд, под каждую из которых существует обработчик. Обработчики вызываются раз в кадр для каждого танка после обработки движения.

Рассмотрим существующие команды:

- 0 — NOP (танк отсутствует);
- 1..7 — обработать взрыв танка (по команде на кадр анимации);
- 8 — обработать статус (скольжение по льду и т.п.);
- 9 — изменить направление;
- A — проверить на пересечение границы тайла;
- B — двигаться к штабу;
- C — двигаться к танку второго игрока;
- D — двигаться к танку первого игрока;
- E..F — команды управления респауном.

Существует функция смены направления, которая при вызове в первом периоде поведения меняет направление танка случайным образом, во втором периоде даёт танкам с чётными номерами команду двигаться к первому игроку, с нечётными — ко второму, а в третьем даёт команду двигаться в сторону штаба.

При пересечении вражеским танком границы тайла (когда обе координаты становятся кратны восьми), существует вероятность 1/16, что для танка будет вызвана эта функция. Если же координаты танка не были кратны восьми, либо же не была вызвана функция смены направления, и при всём этом танк упирается в препятствие, то с вероятностью 1/4 произойдёт следующее: танк сменит направление на противоположное, если хотя бы одна из координат не кратна восьми, в ином случае танку будет дана команда смены направления.

При выполнении команды смены направления происходит несколько другое: с вероятностью 1/2 вызывается описанная выше функция, иначе с равными вероятностями циклически берётся либо предыдущее, либо следующее направление из списка: вверх, влево, вниз, вправо (т.е. танк поворачивается либо по часовой, либо против часовой стрелки).

Если танк упирается в препятствию или стену, даже если он не повернул, одна из координат постоянно кратна восьми и перед ним существует тайл-

препятствие, то вероятность поворота возрастает в разы. Поэтому танки практически не застревают на места: даже попадая в угол или нишу, они довольно быстро оттуда выезжают.

Обработка коллизий

Расчет коллизий и в настоящее время является довольной трудоемкой задачей для физического «Движка» в видеоиграх, что соответственно было критично для такого процессора, как у NES.

Просчёт взаимодействия между танками с реализацией напрямую требовал бы сравнить каждую пару танков, что, несмотря на их небольшое количество, всё же довольно дорогостоящая процедура для процессора с тактовой частотой 1.76 МГц. Но не стоит забывать, что кроме танков по карте передвигаются и выпущенные снаряды. Поэтому разработчики пошли на довольно интересную хитрость.

Дело в том, что каждый танк создает под собой невидимую стену или коллайдер. Таким образом, обнаружение коллизии между танками происходит на этапе обнаружения коллизий между танками и элементами уровня, для чего достаточно лишь определить, имеется ли по определённой координате препятствие. Имеет это и побочные эффекты: если пытаться «въехать» сзади в движущийся вперёд танк, движение игрока блокируется, пока танк не отъедет на некоторое расстояние — как раз в этот момент он переезжает на следующий тайл, «рисуя» там новую стенку и «стирая» старую. Тот же эффект можно заметить, когда в танк попадает снаряд. Визуально соприкосновение возникает на разной дистанции от границы спрайта танка (как внутрь, так и наружу) — от 0 до 7 пикселей. Более того, на расчёт коллизий между снарядами уходит большая часть времени кадра. И в особых случаях его может даже не хватить, тогда все эти операции перенесутся на следующий кадр, а текущий не будет изменён. Отображение коллайдеров изображено на рисунке 3.

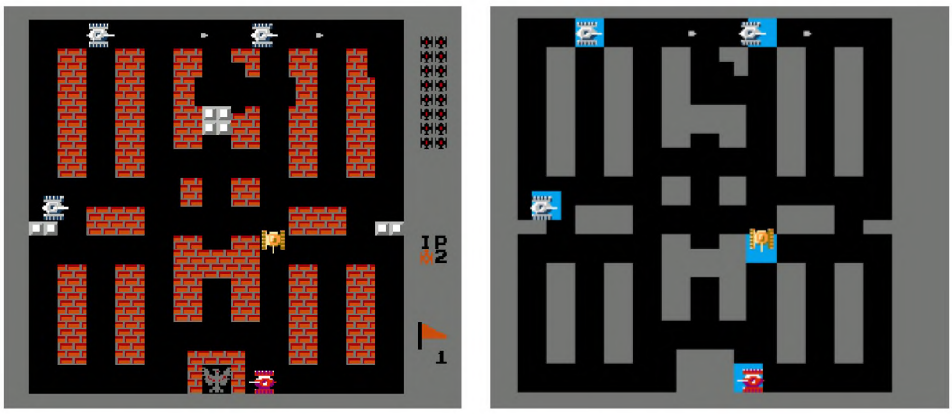

Рисунок 3 – На рисунке слева изображена игра, а на рисунке справа добавлены синие квадраты – коллайдеры, которые показывают то, как представляется физика игры и обработка коллизий изнутри

Передвижение

Все передвижение, как и скорость игры привязаны к количеству кадров в секунду, которые отображаются пользователю на экран, будем полагать что FPS всегда находится на уровне 60 кадров в секунду.

Скорость в игре обуславливается количеством пропускаемых кадров при изменении координат. Так, у игрока координата меняется каждые три кадра из четырёх, у самого быстрого врага в игре — каждый кадр. То же самое касается и снарядов: быстрые, летящие со скоростью 4 пикселя в кадр, и медленные, имеющие скорость 2 пикселя в кадр.

Поскольку при движении также необходимо обрабатывать коллизии, а обрабатывать их каждый кадр на всех танках довольно затратно, используется операция XOR (сложение по модулю 2) между номером кадра и номером танка, затем проверяется чётность/нечетность результата. В итоге получается, что в четных кадрах обрабатывается одна половина медленных танков, а в нечетных – другая.

Также была разработана механика движения танка по льду: после того, как игрок отпустит кнопку управления на льду, танк проедет еще 28 кадров или пока не окажется на земле [5].

1.4 Особенности разработки под мобильные устройства

С появлением мобильных устройств, на них начали появляется и мобильные игры, но выпуск игр, для таких устройств осложнялся различными факторами по сравнению с играми для ПК и консолей:

– Во время игры телефон должен выполнять свою основную функцию – поддерживать связь с базовой станцией и иметь возможность принимать звонки.

– Слабый процессор устройства и маленький объем оперативной памяти – со временем эта проблема все меньше беспокоит разработчиков, но в сравнении с ПК и консолями разница все еще очень большая и данных фактор все равно приходится принимать во внимание;

– Маленькое разрешение экрана – данный фактор также все меньше требует к себе внимания, но при появлении первых мобильных устройств, было довольно сложно уместить игру на экране.

– Вертикальная ориентация экрана – с появлением мобильных устройств появился вертикальный формат игр, если ранее экран компьютера или телевизора находился в горизонтальном положении, то с появлением мобильных устройств, пользователи получили возможность пользоваться устройством и просматривать контент в вертикальном положении устройства.

– Одним из главных ограничений мобильных устройств, которое сложно обойти, является устройство ввода, так на первых устройствах были кнопочные клавиатуры, на которых может быть легко играть в простые аркадные игры или головоломки, но невозможно будет играть в более сложные игры, в которых требуется большая скорость действий и выполнение нескольких задач двумя руками одновременно. Современные устройства частично обходят эту проблему,

благодаря помещению контроллеров ввода любых форм на экран – это позволяет разработчику создать нужный ему контроллер под любое устройство, но все равно это возможно не для всех разновидностей игр.

Для мобильных, также, как и для компьютерных игр остро стоит проблема поддержки большого числа устройств, которые различаются операционной системой, размером экрана, соотношением сторон, максимально возможным размером приложения или объемом ОЗУ, что необходимо учитывать при разработке как технической части игры, так и интерфейсов пользователя.

1.5 ОС Android и магазин приложений Google Play

В соответствии с заданием, разработанная игра, должна будет поддерживать мобильную ОС Android.

Android — операционная система для смартфонов, интернет-планшетов, электронных книг, цифровых проигрывателей, наручных часов, игровых приставок, нетбуков, смартбуков, очков Google, телевизоров и других устройств. Основана на ядре Linux и собственной реализации виртуальной машины Java от Google. Изначально разрабатывалась компанией Android Inc., которую затем купила Google.

Приложения под операционную систему Android являются программами в нестандартном байт-коде для виртуальной машины Dalvik, для них был разработан формат установочных пакетов .APK. Для работы над приложениями доступно множество библиотек: Bionic (библиотека стандартных функций, несовместимая с glibc); мультимедийные библиотеки на базе PacketVideo OpenCORE (поддерживают такие форматы, как MPEG-4, H.264, MP3, AAC, AMR, JPEG и PNG); SGL («Движок» двухмерной графики); OpenGL ES 1.0 ES 2.0 («Движок» трёхмерной графики); Surface Manager (обеспечивает для приложений доступ к 2D/3D); WebKit (готовый «Движок» для веб-браузера; обрабатывает HTML, JavaScript); FreeType («Движок» обработки шрифтов);

SQLite (легковесная СУБД, доступная для всех приложений); SSL (протокол, обеспечивающий безопасную передачу данных по сети). По сравнению с обычными приложениями Linux приложения Android подчиняются дополнительным правилам: Content Providers — обмен данными между приложениями; Resource Manager — доступ к таким ресурсам, как файлы XML, PNG, JPEG; Notification Manager — доступ к строке состояния; Activity Manager — управление активными приложениями.

Некоторые обозреватели отмечают, что Android проявляет себя лучше одного из своих конкурентов, Apple iOS, в ряде особенностей, таких как веб-сёрфинг, интеграция с сервисами Google и прочих. Также Android, в отличие от iOS, является открытой платформой, что позволяет реализовать на ней больше функций [6].

– Несмотря на изначальный запрет на установку программ из «непроверенных источников» (например, с карты памяти), это ограничение отключается штатными средствами в настройках аппарата, что позволяет устанавливать программы на телефоны и планшеты без интернет-подключения (например, пользователям, не имеющим Wi-Fi-точки доступа и не желающим тратить деньги на мобильный интернет, который обычно стоит дорого), а также позволяет всем желающим бесплатно писать приложения для Android и тестировать на своём аппарате. Кроме того, возможность установки программ из «непроверенных источников» способствует пиратству на платформе Android.

– Android доступен для различных аппаратных платформ, таких как ARM, MIPS, x86.

– Существуют альтернативные Google Play магазины приложений: Amazon Appstore (англ.), Opera Mobile Store, Yandex.Store, GetUpps!,Mobogenie, F-Droid, 1Mobile Market.

– В версии 4.3 введена поддержка многопользовательского режима.

– Для распространения своих приложений на мобильной операционной системе Android, необходимо загрузить разработанное приложение в магазин Google Play или в любой из аналогов [6].

Google Play (предыдущее название — Android Market) — магазин приложений, игр, книг, музыки и фильмов компании Google и других компаний, позволяющий владельцам устройств с операционной системой Android устанавливать и приобретать различные приложения (владельцам Android-устройств из Соединённых Штатов и России также доступно приобретение на Google Play книжных изданий, музыки, фильмов).

Регистрация в магазине приложений стоит $25 и взимается один раз, но после этого каждый разработчик имеет право загружать любое количество приложений от своего имени [7].

1.6 Технологии разработки игр

При выборе технологии разработки игры, одним из решающих факторов становится выбор «Движка», на котором и будет строиться весь игровой процесс. «Движок» состоит из подсистем, позволяющих контролировать различные части игры и является набором систем, которые упрощают наиболее часто используемые функции. Рассмотрим основные подсистемы, которые используются в современных играх.

Графическая подсистема

Основной задачей является визуализация (рендеринг) двухмерной или трехмерной компьютерной графики. Основное и важнейшее отличие «игровых» графических движков от неигровых состоит в том, что первые должны обязательно работать в режиме реального времени, тогда как вторые могут тратить по несколько десятков часов на вывод одного изображения [11].

Физический «Движок»

Производит компьютерное моделирование физических законов реального мира в виртуальном мире, с той или иной степенью аппроксимации. Физический «Движок» позволяет создать некое виртуальное пространство, которое можно наполнить телами (виртуальными статическими и динамическими объектами), и указать для него некие общие законы взаимодействия тел и среды, в той или иной мере приближенные к физическим, задавая при этом характер и степень взаимодействий (импульсы, силы и т. д). Собственно, расчёт взаимодействия тел «Движок» и берёт на себя. Когда простого набора объектов, взаимодействующих по определённым законам в виртуальном пространстве, недостаточно в силу неполного приближения физической модели к реальной, возможно добавлять (к телам) связи. Рассчитывая взаимодействие тел между собой и со средой, физический «Движок» приближает физическую модель получаемой системы к реальной, передавая уточнённые геометрические данные средству отображения (рендереру) [10].

Подсистема ввода

Отвечает за обработку команд, подаваемых с контроллеров устройства.

Звуковая подсистема

Программный компонент игрового «Движка», отвечающий за воспроизведение звука в игре или любом другом приложении. Звуковой «Движок» также часто отвечает за имитацию определённых акустических условий, воспроизведению звука согласно местоположению, эхо и т.д. [12].

Системное ядро

Отвечает за координированный доступ к данным, памяти, координирует различные части и подсистемы игры.

Некоторые игры могут иметь больше подсистем в зависимости от их потребностей. Например, дополнительные подсистемы могут работать с сетью, анимацией, игровым искусственным интеллектом и многим другим. В основном все, что может быть сгруппировано в категории, может стать подсистемами [9].

Такой подход имеет множество достоинств:

– Упрощает процесс разработки игр. Вместо вызова множества библиотечных функций для такой простой задачи, как вывод изображения на экран, можно использовать «Движок», который сделает это с помощью одной единственной функции;

– Делает игру переносимой. Хорошо спроектированный игровой «Движок» упрощает перенос игры на другую библиотеку или даже на другую платформу. Если бы вы использовали только вызов библиотечных функций напрямую, вам бы пришлось изменить все части игры и, возможно, переделать всю ее структуру. В противном случае, мы могли бы просто адаптировать определенные подсистемы «Движка»;

– Делает код более организованным и более управляемым;

– Позволяет работать абстрактно, а не иметь дело с низкоуровневыми представлениями о том, как работает та или иная функция;

– Также хорошо спроектированный игровой «Движок» может быть повторно использован в множестве других проектов.

Стоит заметить, что перед разработчиками всегда стоит выбор: использовать готовый «Движок» или написать собственный, каждый из подходов имеет свои достоинства и недостатки, поэтому стоит рассмотреть их более подробно. Также стоит заметить, что предлагаемые далее решения рассматриваются с точки зрения разработки игры для ОС Android.

Для разработки собственного «Движка» под ОС Android, существуют инструменты от Google, который находятся в свободном доступе и предназначены для таких ОС как: Linux, Mac OS и Windows. Разработку приложений для Android можно вести на языке Java (не ниже Java 1.5), также в 2009 году был опубликован Android Native Development Kit (NDK) — пакет инструментариев и библиотек, позволяющий реализовать часть приложения на языке C/C++. NDK рекомендуется использовать для разработки участков кода, критичных к скорости.

К достоинствам разработки собственного игрового «Движка», следует отнести следующее:

- Возможность реализации всех задуманных идей;
- Полное понимание структуры «Движка» и принципов его работы;
- Возможность доработки и адаптации «Движка» под любую задачу;
- Возможность самостоятельно исправлять ошибки его работы.

Все перечисленные достоинства могут быть важны, если разработчику необходимо иметь максимальный контроль над движком, но разработка следует понимать, что собственная разработка требует специальных знаний и глубокого понимания того, как должен работать каждый компонент системы.

Все достоинства использования готовых движков, можно отнести к недостаткам разработки собственного:

- Экономия времени – один из основных факторов для большинства разработчиков, так как разработка собственного игрового «Движка» требует больших временных затрат и в основном все достоинства готовых решений сводятся именно к тому, что можно сэкономить время и силы на реализацию и тестирование «Движка», а сосредоточится на создании игры;
- Сформировавшиеся сообщества разработчиков – у большинства больших игровых движков, есть уже сформированные сообщества разработчиков, который могут помочь советом по реализации любого элемента игры;

– Быстрое расширение – благодаря тому, что у готовых игровых движков уже сформированы сообщества, кроме советов, разработчики могут поделится различными кусками кода, которые могут быть вам полезны. Например, это часто бывает полезно при интеграции какого-либо стороннего сервиса в вашу игру (показ рекламы, аналитика и другое) – при использовании собственного «Движка» будет необходимо самостоятельно встраивать поддержку каждого сервиса в ваше приложение и своевременно обновлять, при использовании готовых решений, весь необходимый код для работы с сервисом, будут предоставлять разработчики самого сервиса – что также позволяет экономить уйму времени.

Достоинств готовых игровых движков очень много, и они изменяются в зависимости от того, какой будет использован, но все достоинства относятся к первому пункту, а именно к экономии времени и средств.

2 Разработка игры

2.1 Выбор технологии разработки

Так как темой моей работы является разработка игры, а не игрового «Движка», была решено использовать готовое решение, что позволит сохранить время для разработки самой игры. Так как приложение должно быть предназначено для мобильных устройств, игровой «Движок» должен соответствовать следующим требованиям:

– Разработка приложений для мобильной ОС Android – данное требование обусловлено заданием на проект;

– Возможность разработки на ОС Windows – требование обусловлено тем, что разработка проекта ведется на данной ОС;

– Возможность разработки на ЯП C++ или C# – не обязательное требование, но данные ЯП в приоритете, в виду их распространенности и наличия опыта работы с ними у разработчика;

– Встроенной графический и физический «Движок», система звука;

– Наличие встроенной системы тестирования;

– Наличие встроенной системы оптимизации производительности;

– Возможность разработки игр в жанре стрельбы;

– Низкая стоимость или возможность бесплатного использования при разработке и выпуске продукта на Android.

Сейчас, когда требования сформированы, необходимо рассмотреть существующие решения и выбрать то, которое максимально соотносится со списком требований, те решения, которые заранее не подходят по какому-либо из пунктов были убраны заранее и не рассматриваются.

Unity3d

Unity — это инструмент для разработки двух- и трёхмерных приложений и игр, работающий под операционными системами Windows, OS X. Созданные с помощью Unity приложения работают под операционными системами Windows, OS X, Windows Phone, Android, Apple iOS, Linux, а также на игровых приставках Wii, PlayStation 3, PlayStation 4, Xbox 360, Xbox One. Есть возможность создавать приложения для запуска в браузерах с помощью специального подключаемого модуля Unity (Unity Web Player), а также с помощью реализации технологии WebGL.

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

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

Unity 3D поддерживает систему Level Of Detail (сокр. LOD), суть которой заключается в том, что на дальнем расстоянии от игрока высоко

детализированные модели заменяются на менее детализированные, и наоборот, а также систему Occlusion culling, суть которой в том, что у объектов, не попадающих в поле зрения камеры не визуализируется геометрия и коллизия, что снижает нагрузку на центральный процессор и позволяет оптимизировать проект [13].

Вывод для Unity3d: в целом, «Движок» подходит ко всем выставленным требованиям и может быть использован при разработке проекта, он поддерживает весь необходимый функционал и содержит бесплатную версию, которую можно использовать до тех пор, пока доход игры не превысит $100 000 в год.

Unreal Engine

Unreal Engine — игровой «Движок», разрабатываемый и поддерживаемый компанией Epic Games. Первая игра, созданная на этом движке — Unreal — появилась в 1998 году.

Написанный на языке C++, «Движок» позволяет создавать игры для большинства операционных систем и платформ: Microsoft Windows, Linux, Mac OS и Mac OS X; консолей Xbox, Xbox 360, PlayStation 2, PlayStation 3, PSP, PS Vita, Wii, Dreamcast, GameCube и др., а также на различных портативных устройствах, например, устройствах Apple (iPad, iPhone), управляемых системой iOS и прочих. (Впервые работа с iOS была представлена в 2009 году, в 2010 году продемонстрирована работа «Движка» на устройстве с системой webOS).

Для упрощения портирования «Движок» использует модульную систему зависимых компонентов; поддерживает различные системы рендеринга (Direct3D, OpenGL, Pixomatic; в ранних версиях: Glide, S3, PowerVR), воспроизведения звука (EAX, OpenAL, DirectSound3D; ранее: A3D), средства голосового воспроизведения текста, распознавание речи), модули для работы с сетью и поддержки различных устройств ввода [14].

Вывод для Unreal Engine: также, как и Unity3d, «Движок» соответствует всем требованиям и является бесплатным пока доход от игры составляет менее $3000 за квартал.

Project Anarchy

Project Anarchy самый молодой из рассматриваемых игровых движков, он был выпущен в 2013 году компанией Havok, которая ранее была известна своим физическим движком, используемым во многих играх. Project Anarchy — мощный инструмент, который включает в себя широкие возможности по созданию графики, физики и AI. Он позволяет бесплатно создавать игры для платформ Android, iOS и Tizen и при покупке лицензии появляется возможность выпуска игр на PS, Xbox, PC [15].

Вывод для Project Anarchy: данный «Движок» содержит множество ограничений при разработке для ПК или консолей, но отлично подходит при разработке под Android, благодаря отсутствию любых платежей при выпуске на данной платформе. Так же как предыдущие 2 «Движка» Project Anarchy имеет большое сообщество разработчиков и удобные инструменты.

Вывод

В результате более глубокого изучения и анализа данных игровых движков, было решено разрабатывать проект на бесплатной версии Unity3d, так как соответствуя всем требованиям проекта, он имеет субъективно более удобный интерфейс и позволяет быстро начать разработку проекта.

2.2 Общий алгоритм функционирования

Общий алгоритм функционирование приложения можно разделить на 2 части: построение карты и игровой процесс, который заключается в сражении танков, на том поле, что было создано. На рисунке 4 представлена UML диаграмма последовательностей, описывающая данную схему.

Рисунок 4 – UML диаграмма последовательностей, описывающая общую схему работы приложения

Рассмотрим подробнее каждый блок алгоритма:

– Построение карты – способ реализации данного блока рассматривается в пункте 2.3 настоящего документа. В общем случае данный блок отвечает за то, чтобы получить карту, на которой будет происходит игра, как только карта получена – программа ожидает действия пользователя, который может

сохранить данную карту и начать игру, либо начать построение карты заново, для этого существуют блоки «Играть» и «Создать другую карту»;

– Играть – данный блок переводит игру из состояния выбора карты, в состояние «Игровой процесс», в результате чего происходит выбор начальной позиции для танка игрока, а также запускается алгоритм генерации волн врагов;

– Создать другую карту – данный блок возобновляет построение карты, но перед этим также служит для удаления старой, которая уже не нужна;

– Игровой процесс – в этом блоке происходят все основные действия, связанные с игровым процессом, так здесь контролируется появление врагов, подсчет очков, передвижение игрока и противников, в любой момент времени, игрок может приостановить игру и поставить ее на «паузу», после чего возобновить бой с того же момента. Также проверяется событие разрушения танка игрока, которое приводит к переходу в блок «Вывести результат»;

– Вывести результат – финальный блок, который отвечает за вывод результатов игры и предоставляет возможности для выбора новой карты и возобновления игрового процесса.

2.3 Структура программы

Из общего алгоритма функционирования, рассмотренного в предыдущей главе, видно, что игра содержит различные компоненты, на рисунке 5 изображена UML диаграмма классов, описывающая отношения между основными классами в программе, к которым относятся:

– Game Manager;
– Map Generator;
– Tank;
– Bullet;
– Enemy With AI;
– Player.

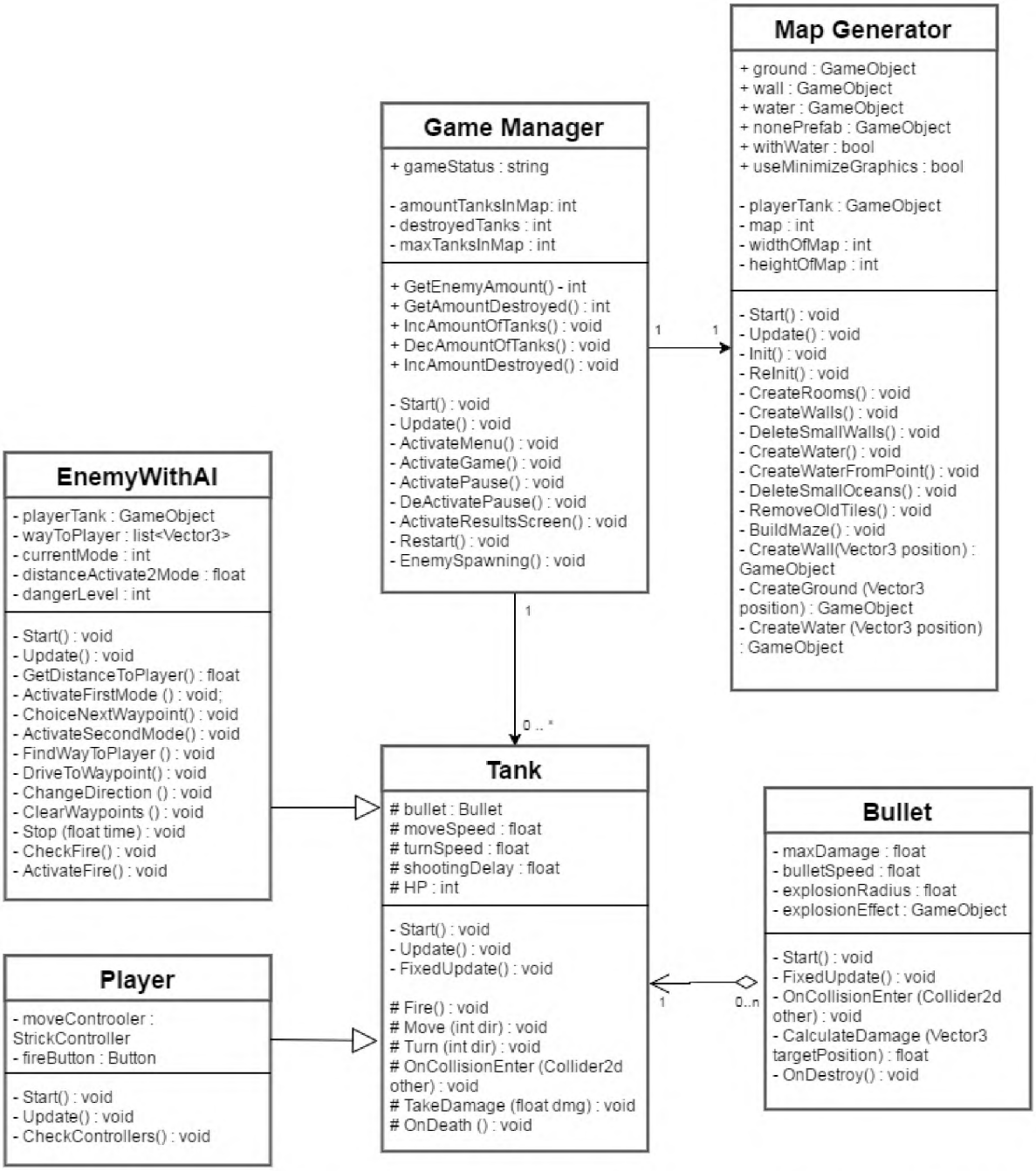

Рисунок 5 – UML диаграмма классов, описывающая данную программу

На диаграмме изображены 6 основных классов и взаимосвязи между ними, перед тем как рассказывать о каждом классе более подробно, стоит рассмотреть структуру, которой руководствуется Unity при работе программы:

– Функция Start() – данная функция есть во всех классах программы, и она вызывается движком автоматически при первом запуске, в основном в данной функции происходит инициализация и настройка начальных параметров.

– Функция Update() – данная функция также присутствует во всех классах программы, и она вызывается на каждом такте работы (т.е. при значении FPS = 60 – функция будет вызвана 60 раз за секунду);

– Функция FixedUpdate() – данная функция вызывается движком автоматически, но только для тех объектов, для которых необходимо просчитывать физику, обычно в данную функцию разработчики вносят работу с физикой и количество вызовов данной функции можно ограничивать.

– Функция OnCollisionEnter (Collider2d name) – функция вызывается автоматически движком, когда происходит столкновение между объектами.

Сейчас, когда мы рассмотрели общие функции, имеет смысл рассмотреть подробнее каждый из основных классов программы:

– Game Manager – один из управляющих классов в программе, содержит данные о текущем статусе игры, количестве танков на карте и количестве уничтоженных танков, имеет функции переключения сцен, начала и конца боя, вывода результата, а также контролирует количество танков противника;

– Map Generator – данный класс отвечает за процедурную генерацию и хранение карт, содержит множество методов, для реализации данной генерации. Именно данный класс контролирует то, на каком уровне будет сражаться игрок, форму этого уровня, количество полей, стен и воды, а также отвечает за то, чтобы не было замкнутых или отрезанных путей.

– Tank – основной класс отвечающий за модель поведения танков в игре, хранит информацию о скорости, скорости поворота, уровне жизни танков и многое другое. К методам класса относятся активация стрельбы, передвижение, поворот и события получения урона, либо полного разрушения танка. Данная модель позволяет передвигать танк по двум плоскостям и совершать стрельбу под любым углом. Данный класс расширяется двумя дочерними классами: EnemyWithAI и Player;

– EnemyWithAI – класс отвечающий за управление танком противника с использованием искусственного интеллекта. Экземпляр данного класса

управляет одним конкретным танком и содержит методы для двух режимов управления, метод поиска пути до танка игрока и тактические методы.

– Player – второй дочерний класс от Tank, экземпляр которого используется для танка игрока и содержит методы для управления танком через пользовательский интерфейс, который отображается на экране пользователя;

– Bullet – класс отвечающий за снаряды, которыми стреляют танки, снаряд не может менять направление и наследует его от танка, который совершил выстрел, но данный класс содержит информацию о максимальном уроне, скорости снаряда, радиусе взрыва, а также методы позволяющие передвигать снаряд, зафиксировать попадание и высчитать нанесенный урон.

В приложении А подробно рассмотрен класс Tank, со всеми его функциями и переменными.

2.4 Создание карт

Сражения танков происходят на различных картах, которые могут иметь как проходимые элементы, такие как земля, так и не проходимые, такие как вода или стены. В игре «Battle City» разработчиками всего было создано 35 различных карт, все они были заготовлены заранее и повторялись при каждой новой игре, нам бы хотелось, чтобы карт было много больше, а лучше если они вообще никогда не будут повторятся.

Методы создания карт

Учитывая, что необходимо разработать большое количество уровней, необходимо использовать те методы, которые позволят сделать это наиболее рационально и быстро, к таким можно отнести:

– Создание и последующее использование редактора уровней;

– Процедурная генерация уровней.

Оба подхода довольно эффективны и часто применяются в современной индустрии игр, поэтому стоит рассмотреть их подробнее.

Редактор уровней

Редактор уровней (также известен как игровой редактор, редактор карт, кампаний или сценариев; англ. level editor) — прикладное программное обеспечение, которое используется для проектирования и создания уровней в компьютерных играх.

Редакторы уровней в большинстве случаев используются для создания уровней лишь для определённого игрового «Движка». Разработка игрового редактора занимает много времени и средств, поэтому намного выгодней выпустить несколько игр на основе одного игрового «Движка» и предлагающемуся к нему редакторе, чем для каждой новой игры заново создавать новый отдельный редактор [16].

Редактор уровней позволяет значительно ускорить процесс создания карт для игры и сделать процесс более наглядным, кроме того, разработчики могут не только сами пользоваться созданным редактором, но и выпустить его для бесплатного использования – что позволит фанатам игры самостоятельно создавать различные карты. Такой подход используется во многих известных играх и проявляет себя положительно, благодаря увеличению времени вовлеченности игроков.

Процедурная генерация

Данный термин относится к методу создания контента посредством какого-либо алгоритма, а не вручную. В большей степени это связано с уменьшением финансовых затрат на ручное создание этого контента, так посредством процедурной генерации в играх могут создаваться как полноценные

ландшафты мира, разнообразные подземелья, лабиринты, острова, так и новые существа, оружие, различные предметы, характеры персонажей и даже квесты.

Для того чтобы создать интересный и разнообразный контент необходимо определить правила генерации – это трудоемкая, но интересная задача, ведь по сути разработчик пишет алгоритм генератора, который может создать что угодно, но полученный результат должен удовлетворять некоторым рамкам, которые выставляются самой игрой или платформой (например, для того, чтобы уровень был проходим или чтобы игра была сбалансирована).

Данный прием также очень популярен в разработке игр, но уже давно вышел за данную индустрию и кроме игр используется в такой индустрии как кинематограф, например, для создания различных визуальных эффектов. Процедурная генерация позволяет создавать совершенно уникальный контент и его разнообразие ограничено лишь исходными данными [17].

Итак, оба метода являются эффективными, в зависимости от ситуации в которой будут использоваться, в данном проекте для создания карт, мы будем использовать процедурную генерацию, так как для нашего проекта, данный метод обладает следующими преимуществами:

– Разработка редактора уровней, требует довольно много времени, но когда редактор будет закончен, также необходимо разработать сами карты, что для данной игры займет больше времени, чем процедурная генерация;

– Практически бесконечное количество карт. При каждой новой игре генерируется новая уникальная карта и игрок не имеет возможности узнать в заранее какая из них будет сгенерирована, т.к. правила генерации могут изменятся постоянно.

Данный метод позволяет создавать довольно интересный игровой опыт, но он все же содержит свои недостатки, из-за которых разработчики могут использовать первый метод – разработку редактора карт:

– Во-первых, в редакторе карт проще контролировать результат;

– Во-вторых, для некоторых игр очень сложно сформировать правила для процедурной генерации, такие, чтобы игра была одновременно интересной и разнообразной;

– В-третьих, сложно использовать процедурную генерацию для игр, которые глубоко привязаны к сюжету игры.

Но для данного проекта, процедурная генерация подходит идеально, поэтому было решено использовать данный метод для генерации карт в игре.

Рассмотрение методов процедурной генерации для создания карт

Перед разработкой метода процедурной генерации, необходимо определится с видом карт, которые должны получаться на выходе. Различных алгоритмов процедурной генерации карт довольно много, рассмотрим несколько из них:

– Наивный алгоритм генерации лабиринтов – суть алгоритма в том, что карта создается по комнатам, которые находятся в случайных местах поля, имеют случайные размеры и в итоге соединяются коридорами [18].

– Алгоритм Эллера для генерации лабиринтов – позволяет создавать лабиринты, имеющие только один путь между двумя точками. Алгоритм работает построчно, двигаясь сверху вниз, слева направо, и позволяет создавать «идеальный лабиринт», в котором нет циклов (между двумя точками всегда существует один единственный путь) и изолированных частей (ячейки, которые не связаны с другими частями лабиринта) [19].

– Генерация лабиринтов с использованием клеточного автомата – за основу работы алгоритма, взят клеточный автомат «Жизнь», придуманный математиком Конвэем. Суть предложенного алгоритма состоит в реализации всего двух шагов: сначала все поле заполняется случайным образом стенами — т.е. для каждой клетки случайным образом определяется, будет ли она свободной или непроходимой — а затем несколько раз происходит обновление состояния

карты в соответствии с условиями, похожими на условия рождения/смерти в «Жизни» [20].

Реализация метода процедурной генерации

Рассмотрев различные методы генерации карт, было решено скомбинировать их и в результате были выделены следующие правила:

– Карта может строится из 3 различных блоков: земля, вода и стены. Танки могут проехать только по земле, могут стрелять сквозь землю и воду, но не сквозь стены;

– Отсутствие изолированных частей – при генерации не должно остаться изолированных участков карты, которые могут образовываться путем ограждения водой или стенами, т.е. каждый танк на игровом поле, должен иметь возможность добраться и уничтожить любой другой танк;

– Так как танк игрока может быть уничтожен с одного выстрела – танки противника всегда появляются на некотором, достаточно большом расстоянии от игрока;

– Для создания разнообразных карт перед работой алгоритма постоянно меняются различные значения, отвечающие за минимальный и максимальный размер комнат, наличие воды, количество стен и другое.

В результате выполнения всех этих правил, всегда будет генерироваться уникальный проходимый уровень.

Работа алгоритма в общем случае, показана на рисунке 6.

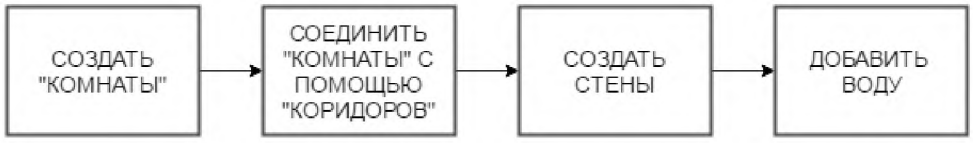

Рисунок 6 – последовательность генерации уровня

Рассмотрим работу алгоритма более подробно:

– В начале работы алгоритма создается массив для хранения карты и задается множество входных параметров, которые определяют максимальные размеры, минимальные размеры, количество «комнат», наличие воды на карте и ее количество – данный шаг определяет будущую карту, но отобразить его работу на экране в этот момент невозможно;

– Вторым шагом создаются «комнаты», их размеры уже ограничены в первом пункте, но все еще остаются случайными в некотором диапазоне. Позиция каждой комнаты определяется не случайно, она также ограничена некоторыми правилами, для того, чтобы не было больших пустых зазоров. Работу алгоритма на данном этапе можно увидеть на рисунке 7.

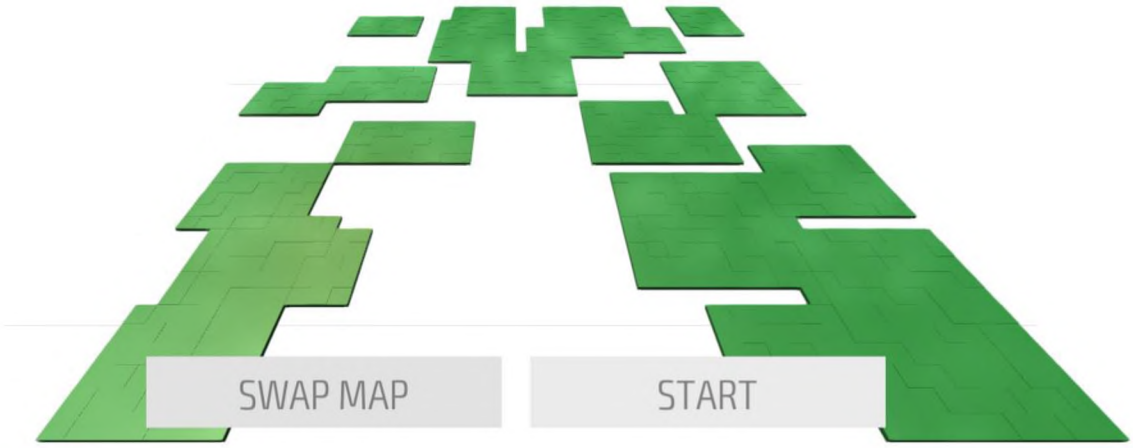

Рисунок 7 – Процедурная генерация – шаг 2 – создание «комнат»

Как видно из рисунка 7, после выполнения шага 2, множество комнат остались изолированными друг от друга и переход между ними не возможен.

– Третьим шагом является создание «коридоров», что позволит создать не изолированную карту. Работу алгоритма на данном этапе можно увидеть на рисунке 8 – на рисунке также видно, что сформированная карта сильно отличается от предыдущей.

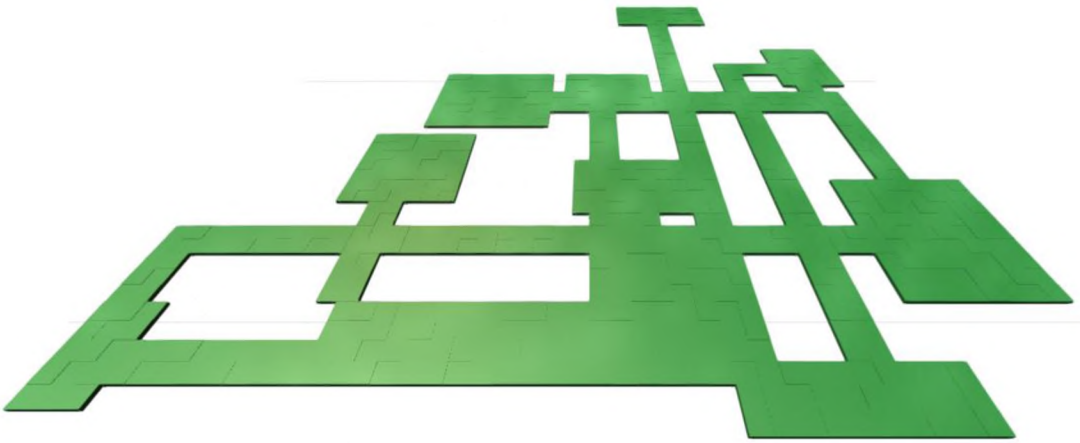

Рисунок 8 – Процедурная генерация – шаг 3 – создание «коридоров»

– На четвертом шаге создаются стены, через которые игрок и танки противника не имеют возможности стрелять или проезжать. Работа алгоритма представлена на рисунке 8 – на данном этапе уже хорошо видно сформированную карту, на которой можно играть.

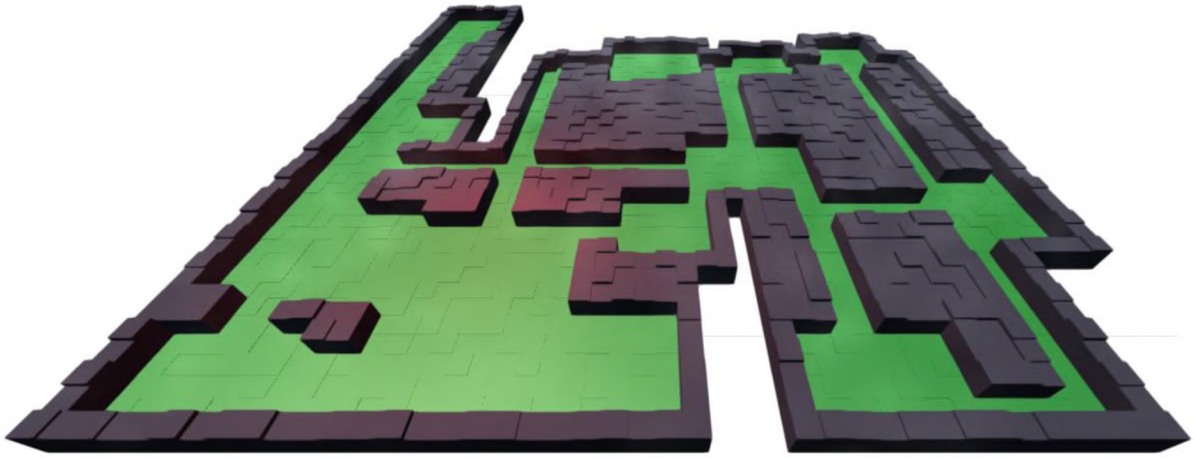

Рисунок 9 – Процедурная генераций – шаг 4 – создание стен

– На пятом шаге в игру добавляется вода, количество которой было определено еще в первом шаге. Вода служит не проходимым препятствием, но с возможностью стрельбы через нее. Алгоритм работает таким образом, что заменяет некоторые блоки стен на воду, но правилами генерации внесены

различные ограничения на виды этих блоков и форму, работу алгоритма можно увидеть на рисунке 10.

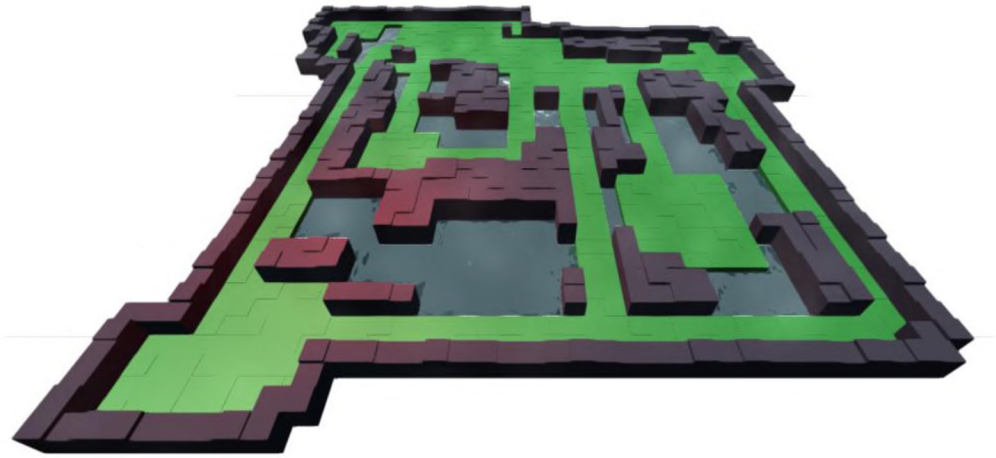

Рисунок 10 – Процедурная генерация – шаг 5 – добавление воды

– Следующий шаг, служит для того, чтобы исправить различные шероховатости, созданные в предыдущих шагах, так, например, на рисунке 10 видно совсем маленькие и незначительные стенки и воду – на данном шаге, все маленькие и незначительные объекты будут переработаны в более крупные или заменены на землю, что позволяет создать более качественные уровни. На этом шаге можно считать, что генерация уровня закончена.

После того, как уровень сгенерирован, пользователь может удалить его и начать генерацию уровня с первого шага, либо начать играть на данном уровне, в этом случае, на карте появятся танки игрока и противников, при этом танк игрока появляется первым, а все танки противника появляются минимум в 30 метрах (по данным игрового поля) от танка игрока.

2.5 Искусственный интеллект противника

В традиционных исследованиях в области ИИ целью является создание настоящего интеллекта, хотя и искусственными средствами. В таких проектах, как Kismet Массачусетского технологического института (МТИ) делается

попытка создать ИИ, способный к обучению и к социальному взаимодействию, к проявлению эмоций.

С точки зрения игр подлинный ИИ далеко выходит за рамки требований развлекательного программного проекта. В играх такая мощь не нужна. Игровой ИИ не должен быть наделен чувствами и самосознанием, ему нет необходимости обучаться чему-либо за пределами рамок игрового процесса. Подлинная цель ИИ в играх состоит в имитации разумного поведения и в предоставлении игроку убедительной, правдоподобной задачи.

Основным принципом, лежащим в основе работы ИИ, является принятие решений. Для выбора при принятии решений система должна влиять на объекты с помощью ИИ. При этом такое воздействие может быть организовано в виде «вещания ИИ» или «обращений объектов».

В системах с «вещанием ИИ» система ИИ обычно изолирована в виде отдельного элемента игровой архитектуры. Такая стратегия зачастую принимает форму отдельного потока или нескольких потоков, в которых ИИ вычисляет наилучшее решение для заданных параметров игры. Когда ИИ принимает решение, оно передается всем участвующим объектам. Такой подход лучше всего работает в стратегиях реального времени, где ИИ анализирует общий ход событий во всей игре.

Системы с «обращениями объектов» лучше подходят для игр с простыми объектами. В таких играх объекты обращаются к системе ИИ каждый раз, когда объект «думает» или обновляет себя. Такой подход отлично подходит для систем с большим количеством объектов, которым не нужно «думать» слишком часто, например, в шутерах. Такая система также может воспользоваться преимуществами многопоточной архитектуры, но для нее требуется более сложное планирование

Чтобы искусственный интеллект мог принимать осмысленные решения, ему необходимо каким-либо образом воспринимать среду, в которой он находится. В простых системах такое восприятие может ограничиваться простой

проверкой положения объекта игрока. В более сложных системах требуется определять основные характеристики и свойства игрового мира, например, возможные маршруты для передвижения, наличие естественных укрытий на местности, области конфликтов.

Системы на основе правил

Простейшей формой искусственного интеллекта является система на основе правил. Такая система дальше всего стоит от настоящего искусственного интеллекта. Набор заранее заданных алгоритмов определяет поведение игровых объектов. С учетом разнообразия действий конечный результат может быть неявной поведенческой системой, хотя такая система на самом деле вовсе не будет «интеллектуальной».

В более сложных и разумных системах в качестве основы используются последовательности условных правил. В тактических играх правила управляют выбором используемой тактики. В стратегических играх правила управляют последовательностью строящихся объектов и реакцией на конфликты. Системы на основе правил являются фундаментом ИИ.

Адаптивный искусственный интеллект

Если же в игре требуется большее разнообразие, если у игрока должен быть более сильный и динамичный противник, то ИИ должен обладать способностью развиваться, приспосабливаться и адаптироваться.

Адаптивный ИИ часто используется в боевых и стратегических играх со сложной механикой и огромным количеством разнообразных возможностей в игровом процессе.

Способность точно предугадывать следующий ход противника крайне важна для адаптивной системы. Для выбора следующего действия можно

использовать различные методы, например, распознавание закономерностей прошлых ходов или случайные догадки.

Одним из простейших способов адаптации является отслеживание решений, принятых ранее, и оценка их успешности. Система ИИ регистрирует выбор, сделанный игроком в прошлом. Все принятые в прошлом решения нужно каким-то образом оценивать (например, в боевых играх в качестве меры успешности можно использовать полученное или утраченное преимущество, потерянное здоровье или преимущество по времени). Можно собирать дополнительные сведения о ситуации, чтобы образовать контекст для решений, например, относительный уровень здоровья, прежние действия и положение на уровне (люди играют по-другому, когда им уже некуда отступать).

Можно оценивать историю для определения успешности прежних действий и принятия решения о том, нужно ли изменять тактику. До создания списка прежних действий объект может использовать стандартную тактику или действовать произвольно. Эту систему можно увязать с системами на основе правил и с различными состояниями.

В тактической игре история прошлых боев поможет выбрать наилучшую тактику для использования против команды игрока, например, ИИ может играть от обороны, выбрать наступательную тактику, атаковать всеми силами невзирая на потери или же избрать сбалансированный подход. В стратегической игре можно для каждого игрока подбирать оптимальный набор различных боевых единиц в армии. В играх, где ИИ управляет персонажами, поддерживающими игрока, адаптивный ИИ сможет лучше приспособиться к естественному стилю игрока, изучая его действия.

Навигация искусственного интеллекта

Приняв решение, интеллектуальному агенту нужно понять, как двигаться из точки А в точку Б. Для этого можно использовать разные подходы, выбрав

оптимальный в зависимости от характера игры и от нужного уровня производительности.

Алгоритм, условно называемый «Столкнуться и повернуть», является одним из простейших способов формирования маршрута движения объекта. Вот как это работает.

– Двигайтесь в направлении цели.

– Если столкнетесь со стеной, повернитесь в направлении, при котором вы окажетесь ближе всего к цели. Если ни один из доступных для выбора вариантов не имеет очевидных преимуществ, выбор делается произвольно.

Такой подход неплохо работает для простых игр, но при большом количестве сложных стен и игровых объектов показывает себя много хуже, так как объект управляемый таким ИИ, может оказаться запертым и не сможет найти выхода.

Если же агенты в игре должны действовать не столь бестолково, можно расширить простое событие столкновения и снабдить агентов памятью. Если агенты способны запоминать, где они уже побывали, то они смогут принимать более грамотные решения относительно того, куда поворачивать дальше. Если же повороты во всех возможных направлениях не привели к удаче, агенты смогут вернуться назад и выбрать другой маршрут движения. Таким образом, агенты будут производить систематический поиск пути к цели.

Преимущество этого метода состоит в невысокой нагрузке на вычислительные ресурсы. Это означает, что можно поддерживать большое количество перемещающихся агентов без замедления игры. Этот метод также может использовать преимущества многопоточной архитектуры. Недостатком является расход огромного объема памяти впустую, поскольку каждый агент может отслеживать целую карту возможных путей.

Кроме того, для поиска оптимального пути может использоваться огромное множество различных алгоритмов, позволяющих вычислить и найти минимальный путь до цели.

Реализация искусственного интеллекта в данном проекте

Проанализировав различные алгоритмы и подходы к реализации ИИ для игр, было решено разделить работу ИИ на 2 части, алгоритм их взаимодействия представлен на рисунке 11.

Рисунок 11 – Алгоритм взаимодействия режимов ИИ

Первый режим работает до того момента, как танк игрока, войдет в область видимости ИИ (в игровых координатах это 50 м). В первой части ИИ управляет танком следуя алгоритму, который можно назвать «Поверни если столкнулся», т.е. танк движется до того момента как упрется в стену или в воду, после чего поворачивает в случайном направлении, также для большей реалистичности добавлены события остановки танка и поворота даже без столкновения.

Второй режим работы ИИ включается при условии, что игрок находится в области видимости ИИ. В этом случае работа алгоритма также делится на 2 части: если есть вероятность попадания в танк игрока – ИИ начинает стрельбу, если же попасть в танк игрока невозможно из-за препятствий в виде стен, ИИ активирует функцию поиска кратчайшего пути до танка игрока и двигается в его сторону, до того момента, как появится возможность стрельбы, либо заданная точка будет достигнута. Если заданная точка достигнута, а стрельба по танку игрока не возможна – алгоритм начинает свою работу сначала.

2.6 Адаптация игры для мобильных устройств

При разработке игр или приложений на «движках», которые поддерживают кроссплатформенность – адаптация для мобильных устройств заключается в том, что необходимо создать контроллеры управления, которые будет возможно использовать на устройстве и оптимизировать производительность игры, а также адаптировать приложение для работы с различными разрешениями и соотношениями сторон экранов. Оптимизация будет рассматриваться в следующем пункте, а сейчас необходимо рассмотреть способ адаптации игры для управления на мобильном устройстве.

Игра ориентирована на ОС Android, поддержка старых устройств не рассматривается, поэтому можно считать, что все устройства имеют экран с touch-screen. В действительности, так как игровая физика строится в 2Д, нам не требуется продумывать сложное управление, какие-либо системы прицеливания и многое другое, что может быть трудно на экране мобильного устройства, в данном же случае, нам необходим один контроллер для управления передвижением, который также отвечает за поворот танка и одна кнопка для стрельбы.

2.7 Оптимизация производительности

При тестировании игры на мобильном устройстве, было обнаружено несколько проблем, связанных с производительностью. Перед началом любой оптимизации необходимо найти основные причины проблем с производительностью, для этого в Unity есть такой инструмент как профайлер.

Профайлер Unity помогает оптимизировать игру. Он сообщает о том, как много времени тратится в различных областях игры. Например, профайлер может сообщить процент времени, потраченный на рендеринг, анимацию или внутри игровую логику.

Профайлер работает при включенном редакторе, т.е. в тот момент пока разработчик тестирует различные моменты игры с включенным профайлером, им будут записываться данные о производительности. Записанные данные отображаются на временной шкале в окне профайлера, позволяя обнаружить зашкаливающие (занимающие больше времени) по сравнению с другими кадры или области. Кликнув по любому кадру на временной шкале можно открыть подробную информацию для этого кадра [21, 22].

Основные проблемы с производительностью игры появляются в тот момент, когда на уже сгенерированной карте начинается бой между танками, так на рисунке 12 отображены данные профайлера, в момент игры.

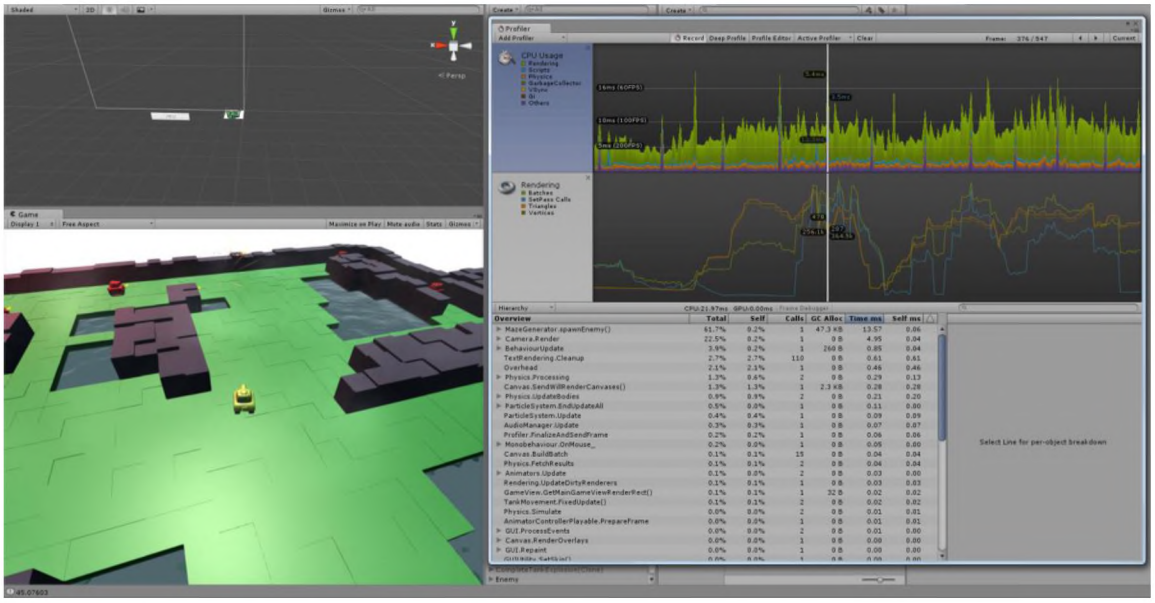

Рисунок 12 – Данные о потреблении ресурсов приложением из профайлера

По данным профайлера можно судить о проблемах с производительностью приложения и определить способы их решения, так проанализировав данные, можно определить следующие проблемы:

– Большая часть ресурсов отводится на рендеринг, т.е. отображение всей графики на экране. Если уменьшить объем ресурсов отводимый на рендеринг, можно поднять средний ФПС в игре.

– Второй большой проблемой являются пики нагрузки, которые также видно на графике – эта проблема также может быть очень заметна в игре и проявляется в виде кратковременных задержек. Пики нагрузки возникают в результате одновременного появления нескольких новых танков, такое возможно, как в начале игры, что не особо существенно, так и в процессе, когда было уничтожено сразу несколько танков и вместо них появляются новые.

Другие проблемы связанные с расчетом физики, сбором мусора и скриптами влияют не так существенно на конечную производительность, поэтому рассматривать их на данный момент не имеет смысла.

Оптимизация рендеринга

Как говорилось выше, оптимизация рендеринга направлена на повышение среднего значения ФПС в игре и для достижения нужного результата будем использовать разные методы:

– В Unity доступны дополнительные параметры рендеринга для игр, которые можно настроить с помощью Quality Manager, который представлен на рисунке 13.

Рисунок 13 – значения качества графики в Quality Manager

Как видно из рисунка 13, настройки качества для всех платформ находятся в максимальных значениях, настройки качества влияют на размерность текстур, методы и глубину сглаживания, обработку теней и света, качество теней и

дальность их прорисовки, а также некоторые другие настройки, влияющие на отображение картинки на экране. Для данного проекта настройка значений графики свелась к уменьшению размера текстур и уменьшению степени сглаживания, в результате оптимизации мы экономим ресурсы системы, повышая производительность, а качество изображения ухудшается не значительно.

– Вторым методом является уменьшение количества Draw Calls – вызовов команд, направленных графическому API. Графический API производит значительную работу для каждого Draw Call, что сильно влияет на производительность центрального процессора [23].

Для экономии ресурсов Unity3d может использовать системы статического или динамического батчинга объектов, смысл которой заключается в том, что несколько объектов, использующих один и тот же материал отрисовываются за 1 draw call. Это означает, что 3 одинаковых здания будут отрисованы за 1 проход, а 3 разных за 3 прохода, что существенно увеличивает нагрузку на ЦП. У батчинга есть множество ограничений, при которых он не будет работать, поэтому, для того чтобы использовать батчинг, необходимо оптимизировать проект в соответствии с условиями работы данной системы.

После оптимизации игровой сцены, удалось вдвое уменьшить количество Draw calls, что дало ощутимую прибавку к ФПС.

Пики нагрузки

Изучив данные профайлера, было обнаружено, что пики нагрузки возникают в момент уничтожения нескольких танков, т.к. в данном случае в короткий промежуток времени происходят такие действия как: обработка коллизий сразу нескольких объектов, удаление разрушенных танков с поля боя, создание новых танков. Так для разрушения и создания новых танков используются дорогие операции в плане использования времени процессора,

было решено создавать танки только в начале игры – сразу после генерации карты, а при уничтожении танка не удалять его с поля боя, а временно отключать, далее необходимо изменить позицию этого танка, восстановить жизни и включить его отображение в новом месте – данный подход позволяет существенно сократить процессорное время, затрачиваемое на разрушение и создание объекта, т.к. не приходится сначала выгружать все ресурсы, а потом загружать их заново.

Результат

В результате выполнения оптимизации было улучшено среднее значение ФПС и на целевом устройстве составляет постоянные 60 кадров в секунду, также удалось побороть проблему кратковременных зависаний, вызванную пиками нагрузки. Дальнейшая оптимизация не требуется.

В таблице 1 приведены данные производительности до оптимизации и после. Данные взяты с смартфона Sony Xperia Acro S, под управлением мобильной ОС Android 4.1.2 Jelly Bean.

Таблица 1 – Производительность до оптимизации и после

Состояние	Среднее значение FPS	FPS в момент пика
До оптимизации	30 - 40	10-20
После оптимизации	60	>50

3 Результаты работы

3.1 Требования к аппаратному и программному обеспечению

Разработанный проект, является мобильным приложением и предназначен только для мобильных устройств на базе платформы Android.

Требования к аппаратному обеспечению:

- Сенсорный экран;
- 512 МБ ОЗУ или более;
- Процессор с частотой не менее 1 ГГц;
- 23 МБ свободной памяти.

Требования к операционной системе мобильного устройства:

Версия Android 2.3 или выше.

3.2 Установка

После публикации приложения, оно будет доступно бесплатно для скачивания в официальном магазине приложений google – http://play.google.com

3.3 Адаптация к разным экранам

Часто при разработке приложений под мобильные устройства встает проблема адаптации под разные разрешения экрана, соотношения сторон, плотность пикселей и другое. В данном случае, игра будет запускаться на устройстве строго в горизонтальном режиме и адаптирована под всевозможные устройства. На рисунке 14 изображен момент игры в момент генерации карты при разрешении экрана 1920 на 1080 пикселей (что соответствует соотношению сторон 16:9).

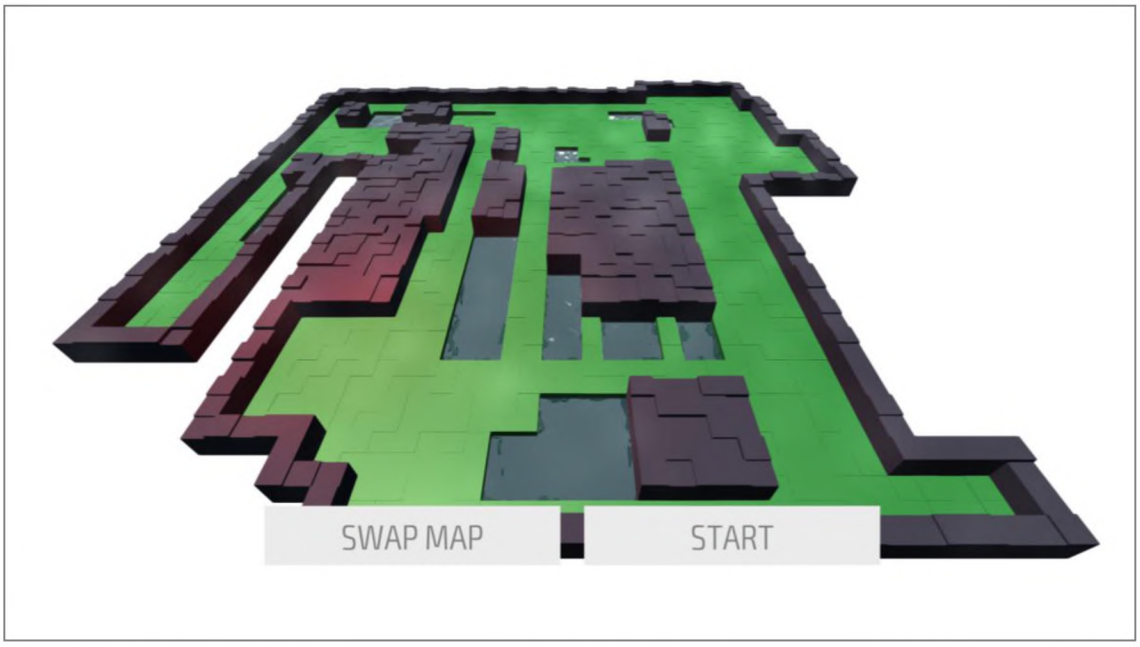

Рисунок 14 – генерация карты при разрешении 1920 на 1080

На рисунке 15 изображен тот же момент игры, только для разрешения 1600 на 1200 (соотношение сторон 4:3)

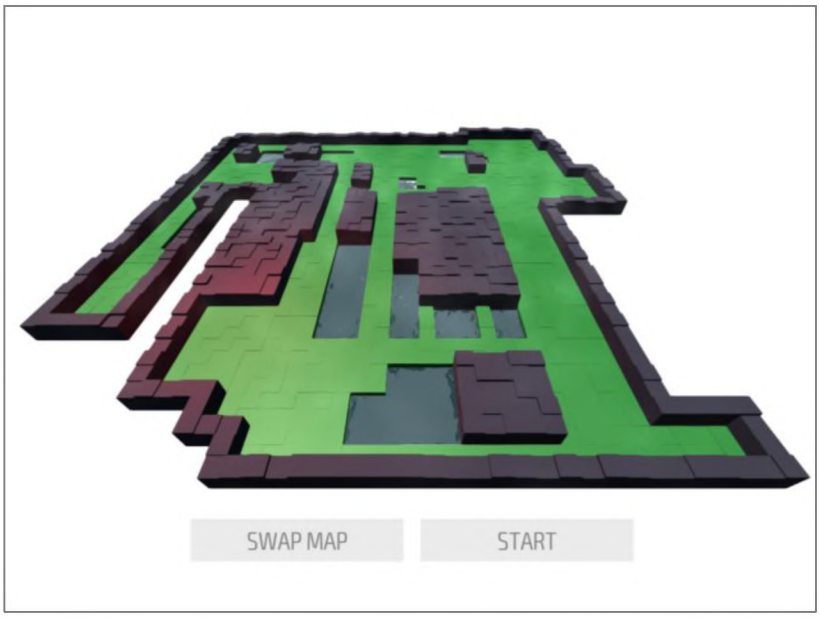

Рисунок 15 – генерация карты при разрешении 1600 на 1200

На рисунке 16 изображен момент игры, в который игрок сражается с танками противника при разрешении экрана 1920 на 1080.

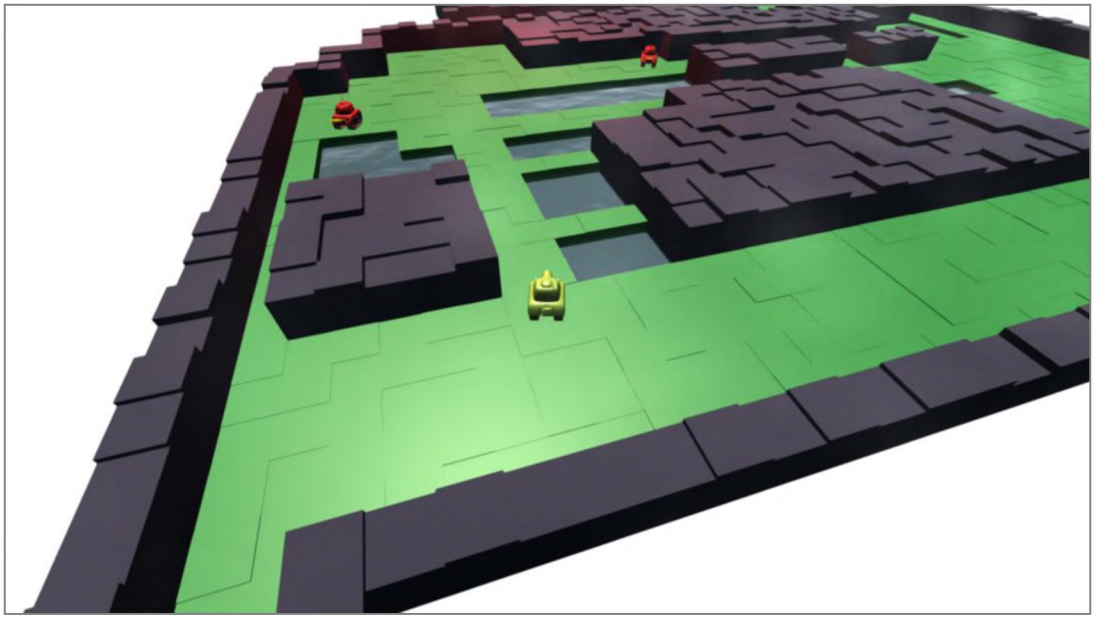

Рисунок 16 – сражение при разрешении 1920 на 1080

На рисунке 17 изображен тот же момент игры, но при разрешении экрана 1600 на 1200.

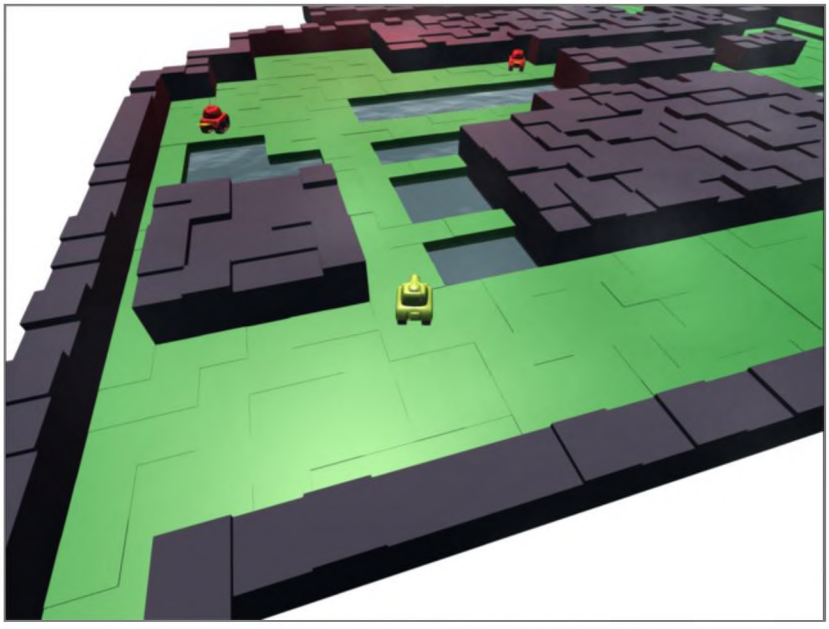

Рисунок 17 – сражение при разрешении 1600 на 1200

Из рисунков 14 – 17 видно, что приложение хорошо адаптируется под разные разрешения и соотношения сторон экранов, что позволяет запускать игру на широком спектре устройств.

ЗАКЛЮЧЕНИЕ

По мнению аналитиков, первое место по инвестициям и стремительной скорости роста в ближайшее время займет рынок игровых услуг. В настоящее время у более чем двух миллиардов людей есть смартфоны, а мобильные игры составляют приблизительно 40% от общего числа загрузок приложений. Ни одна другая область творчества не распространяется так широко.

В результате выполнения данной работы, была разработана 2Д игра про танки, для мобильных устройств на базе ОС Android. Реализована система процедурной генерации карт, которая позволяет создавать бесконечное количество разнообразных уровней. Также для одиночной игры, была разработана система, основанная на ИИ для управления танками противника.

В процессе разработки приложения, было проведено тестирование и оптимизация, в результате чего, игра стабильно работает на большом спектре мобильных устройств на базе ОС Android и имеет сравнительно не высокие требования к аппаратному обеспечению.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Интернет-энциклопедия Wikipedia [Электронный курс]. Индустрия компьютерных игр. Режим доступа: http://ru.wikipedia.org/wiki/Индустрия_компьютерных_игр

2 Блог компании AppFigures [Электронный курс]. App stores growth accelerates in 2014. Режим доступа: http://blog.appfigures.com/app-stores-growth-accelerates-in-2014/

3 M. Silbiger Press Play to Grow! Designing Video Games as «Trojan Horses» to Catalyze Human Development through the Conveyor Belt of Growth; John F. Kennedy University, Pleasant Hill, CA, 2008. – 43 с.

4 Интернет-энциклопедия Wikipedia [Электронный курс]. Компьютерная игра. Режим доступа: http://wikipedia.org/wiki/Computer_game

5 Интернет-энциклопедия Wikipedia [Электронный курс]. Видеоигра Battle City. Режим доступа: http://wikipedia.org/wiki/Battle_City

6 Социальное СМИ – Хабрахабр [Электронный курс]. Разработка. Ностальгия: роемся у «Танчиков» под капотом. Режим доступа: http://habrahabr.ru/post/142126/

7 Интернет-энциклопедия Wikipedia [Электронный курс]. Android. Режим доступа: http://wikipedia.org/wiki/Android

8 Интернет-энциклопедия Wikipedia [Электронный курс]. Google Play. Режим доступа: http://wikipedia.org/wiki/Google_Play

9 Интернет-энциклопедия Wikipedia [Электронный курс]. Game engine. Режим доступа: http://wikipedia.org/wiki/Game_Engine

10 Интернет-энциклопедия Wikipedia [Электронный курс]. Physic engine. Режим доступа: http://wikipedia.org/wiki/Physic_Engine

11 Интернет-энциклопедия Wikipedia [Электронный курс]. Graphic engine. Режим доступа: http://wikipedia.org/wiki/Graphic_Engine

12 Интернет-энциклопедия Wikipedia [Электронный курс]. Audio engine. Режим доступа: http://wikipedia.org/wiki/Audio_Engine

13 Интернет-энциклопедия Wikipedia [Электронный курс]. Unity Game Engine. Режим доступа: http://wikipedia.org/wiki/Unity_(Game_Engine)

14 Интернет-энциклопедия Wikipedia [Электронный курс]. Unreal Engine. Режим доступа: http://wikipedia.org/wiki/Unreal_Engine

15 Сайт компании Havok [Электронный курс]. Project Anarchy Engine. Режим доступа: http://havok.com/

16 Интернет-энциклопедия Wikipedia [Электронный курс]. Level editor. Режим доступа: http://wikipedia.org/wiki/Level_Editor

17 Интернет-энциклопедия Wikipedia [Электронный курс]. Процедурная генерация. Режим доступа: http://wikipedia.org/wiki/Procedural_generation

18 Tutorials & Free Online Courses [Электронный курс]. Create a procedurally generated dungeon cave system. Режим доступа: http://gamedevelopment.tutsplus.com/tutorials/create-a-procedurally-dungeon

19 Блог NeoComputer [Электронный курс]. Eller's Algorithm. Режим доступа: http://www.neocomputer.org/projects/eller.html

20 Tutorials & Free Online Courses [Электронный курс]. Generate Random Cave Levels Using Cellular Automata. Режим доступа: http://gamedevelopment.tutsplus.com/tutorials/generate-random-cave-levels-using-cellular-automata--gamedev-9664

21 Официальная документация «Движка» Unity3d [Электронный курс]. Unity Profiler. Режим доступа: http://docs.unity3d.com/ru/current/Manual/Profiler.html

22 Официальная документация «Движка» Unity3d [Электронный курс]. Unity Profiler Window. Режим доступа: http://docs.unity3d.com/ru/current/Manual/ProfilerWindow.html

23 Официальная документация «Движка» Unity3d [Электронный курс]. Draw call batching. Режим доступа: http://docs.unity3d.com/ru/current/Manual/DrawCallBatching.html

ПРИЛОЖЕНИЕ А

Класс Tank

Перед тем как рассматривать класс подробнее, стоит ознакомиться с основным его кодом:

```
using UnityEngine;

public class Tank : MonoBehaviour
{
    protected float speed = 12f;
    protected float turnSpeed = 180f
    protected int HP = 100;
    protected float shootingDelay = 0.5f;

    protected Rigidbody bulletRB;
    protected Bullet bullet;
    protected Transform fireTransform;
    private Rigidbody rigidbody;
    private bool m_Dead;

    private float timeOfLastShoot = 0;

    private void Start ()
    {
        rigidbody = GetComponent<Rigidbody> ();

        m_Dead = false;
        timeOfLastShoot = 0;
    }

    protected void Move (int dir)
    {
        Vector3 movement = transform.forward * speed * Time.deltaTime *
dir;
        rigidbody.MovePosition(rigidbody.position + movement);
    }

    protected void Turn (int dir)
    {
        float turn = turnSpeed * Time.deltaTime * dir;

        Quaternion turnRotation = Quaternion.Euler (0f, turn, 0f);
        rigidbody.MoveRotation (rigidbody.rotation * turnRotation);
    }
```

```
39
40    protected void Fire ()
41    {
42        if (Time.timeSinceLevelLoad-shootingDelay < timeOfLastShoot())
43         {
44            Rigidbody newBullet = Instantiate (byllet, fireTrans
form.position, fireTransform.rotation) as Rigidbody;
45            timeOfLastShoot = Time.timeSinceLevelLoad;
46        }
47    }
48
49    protected void TakeDamage (float dmg)
50    {
51        HP -= dmg;
52
53        if (HP <= 0f && !m_Dead)
54            OnDeath ();
55    }
56
57    protected void OnDeath ()
58    {
59        gameObject.SetActive (false);
60    }
61 }
```

Так в 3 строке программы мы видим, что класс Tank наследуется от класса MonoBehavior, который является базовым для множества классов в Unity3d и содержит множество функций для получения данных об объекте, например, позволяет получить имя созданного объекта, его позицию в системе координат, угол поворота и многое другое.

Далее в строках 5 – 8 находятся переменные, которые определяют характеристики танка, такие как: скорость движения, скорость поворота, прочность и время перезарядки.

Строки 10 – 12 содержат переменные, определяющие параметры снаряда которым танк будет стрелять, а также устанавливают позицию, в которой снаряд появляется при активации стрельбы. Переменная, которая находится в 13 строке хранит ссылку на компонент, отвечающий за физику танка, а переменная в 14 строке хранит значение состояния танка (разрушен или нет). Переменная в 16

строке необходимо для того, чтобы сохранить время выстрела и при последующем выстреле проверить, успел ли танк перезарядится.

В строках 18 – 24 находится функция Start, которая вызывается «движком» автоматически при создании объекта в данном случае она необходима для того, чтобы задать начальные значения и определить компонент отвечающий за физику танка.

В строках 26 – 30 находится функция Move, которая отвечает за передвижение танка, она вызывается при управлении танком с помощью интерфейса пользователи, если это танк игрока или данную функцию вызывает класс, отвечающий за искусственный интеллект. В любом случае, данная функция позволяет переместить танк лишь в направлении его гусениц (т.е. только назад или вперед по направлению).

В строках 32 – 38 находится функция Turn, которая отвечает за вращение танка вокруг его оси, она вызывается при управлении танком с помощью интерфейса пользователи, если это танк игрока или данную функцию вызывает класс, отвечающий за искусственный интеллект. В любом случае, данная функция позволяет вращать танк на 360 градусов вокруг его оси.

Вместе функции Move и Turn позволяют перемещать танк в любую точку карты и настраивать дуло танка под любым углом – т.к. его направление зависит от направления гусениц, которым управляет функция Turn.

В строках 40 – 47 находится функция Fire, которая отвечает за активацию выстрела танка, она вызывается при управлении танком с помощью интерфейса пользователи, если это танк игрока или данную функцию вызывает класс, отвечающий за искусственный интеллект. В любом случае, данная функция активирует стрельбу, создавая новый объект снаряда, добавляя ему компонент отвечающий за физику и выстраивая направление в котором полетит снаряд.

В строках 49 – 55 находится функция TakeDamage, которая активируется при попадании в танк снаряда, при том не важно является ли снаряд вражеским

или игрок просто был задет взрывом от собственного заряда – в любом случае танк получит урон.

В строках 57-60 находится функция OnDeath() и вызывается она когда уровень прочности танка становится меньше или равным нулю, в данном классе она лишь отключает уничтоженный танк, но также данная функция расширяется дочерними классами и позволяет завершить игру и вывести результаты, если разрушенный танк был танком игрока, либо просто обратившись к классу Game Manager добавить очки и количество уничтоженных танков, если уничтоженный танк был танком противника.

Данный класс является родительским для классов Player и EnemyWithAI и содержит в себе лишь основные функции.