**ВВЕДЕНИЕ**

В наше время игры в жанре Tower Defense (с англ. — «Башенная защита»), сокращённо TD, стали уже классикой. Повсеместно выпускаются сотни игр с этим жанром с различными вариациями. Игры в жанре TD просты в разработке. Им не требуется создание сложной графики и большого её количества. Достаточно сделать игровое поле, башни и монстров. Так же и с программированием. Достаточно создать движение для монстров, атаку по этим монстрам башням, сделать улучшение башен, разнообразить их, и игра готова.

Но сделать просто TD, не оригинальная затея. Следовательно, нужно придумать что-то отличающееся от классических игр этого жанра.

В этой игре будет реализована идея о том, что недостаточно просто защищаться, нужно ещё и нападать. А именно, игрок будет иметь возможность отправить войска на сторону противника, чтобы уничтожить его замок. Так же, ему придётся защищаться от армий врага, который тоже будет пользоваться данной ему возможностью нападать.

В игре будет реализована возможность получения опыта со смерти ваших воинов, отправленных на сторону врага. За него можно будет улучшать воинов и шахту, которая будет приносить монеты в секунду.

Так же, будет реализована возможность строительства башен, улучшать и продавать их.

Следовательно, целью курсового проекта является разработка 3D игры на платформе Unity. Из этой цели следуют следующие задачи:

1. создание игровых объектов и полей в 3D редакторах;
2. создание функционала смоделированным объектам;
3. создание интерфейса приложения;
4. проработка игровой логики.

**1 Проектирование приложения**

**1.1 Описание предметной области**

1) Башня. О ней следует хранить следующую информацию:

1.1) Объект снаряда;

1.2) Позиция точки вылета снаряда;

1.3) Урон;

1.4) Стоимость;

1.5) Цель;

1.6) Скорость стрельбы;

1.7) Установлена ли башня;

1.8) Следующий уровень башни;

1.9) Место расположения башни;

1.10) Отступ для меню изменения башни;

1.11) Тип;

1.12) Противника ли башня;

1.13) Уровень башни;

1.14) Модель башни;

1.15) Стреляет ли башня.

2) Зона выбора цели башни. О ней нужно хранить следующую информацию:

2.1) Башня, к которой принадлежит зона;

2.2) Цель;

2.3) Выбрана ли цель.

3) Зона выбора цели огненной башни. О ней нужно хранить следующую информацию:

3.1) Список целей;

3.2) Башня, к которой принадлежит зона;

3.3) Стреляет ли башня.

4) Зона выбора цели ледяной башни. О ней нужно хранить следующую информацию:

4.1) Сила замедления врагов;

4.2) Список целей;

4.3) Башня, к которой принадлежит зона;

4.4) Стреляет ли башня.

5) Место для башни. О нём нужно хранить следующую информацию:

5.1) Башня, установленная на это место;

5.2) Занято ли место.

6) Снаряд башни. О нём нужно хранить следующую информацию:

6.1) Скорость полёта снаряда;

6.2) Цель;

6.3) Башня, из которой пускается снаряд.

6.4) Позиция новой цели, в случае уничтожение старой.

7) Снаряд молния. О нём нужно хранить следующую информацию:

7.1) Цель;

7.2) Башня, из которой пускается молния;

7.3) Стреляет ли молния.

8) Контроль камеры. О нём нужно хранить следующую информацию:

8.1) Главная камера;

8.2) Скорость поворота;

8.3) Скорость движения;

8.4) Скорость приближения;

8.5) Поворот камеры по X координате;

8.6) Поворот камеры по Y координате;

8.7) Поворот камеры по Z координате;

8.8) Трансформация камеры;

8.9) Позиция камеры по Y координате;

8.10) Старая позиция камеры по X координате;

8.11) Старая позиция камеры по Y координате;

8.12) Старая позиция камеры по Z координате;

8.13) Шаг камеры при прокрутке колёсика мыши.

9) Воин. О нём следует хранить следующую информацию:

9.1) Урон;

9.2) Интервал между атаками;

9.3) Текущее здоровье;

9.4) Максимальное здоровье;

9.5) Стоимость в монетах;

9.6) Стоимость в опыте;

9.7) Награда за убийство в опыте;

9.8) Полоска здоровья;

9.9) Замок;

9.10) Деньги;

9.11) Опыт;

9.12) Заморожен ли воин;

9.13) Горит ли воин;

9.14) Время горения;

9.15) Время заморозки;

9.16) Список типов башен, к которым воин неуязвим;

9.17) Урон от огня;

9.18) Сила замедления;

9.19) Ледяная башня;

9.20) Отступ для полоски здоровья;

9.21) Эффект поджога;

9.22) Эффект заморозки;

9.23) Враг ли воин;

9.24) Имя;

9.25) Умер ли воин;

9.26) Атакует ли воин;

9.27) Горит ли воин;

9.28) Замедлен ли воин;

9.29) Скорость анимации;

9.30) Аниматор;

9.31) Движение по точкам.

10) Движение по точкам. О нём нужно хранить следующую информацию:

10.1) Отступ от 0 по Y координате;

10.2) Текущая скорость движения;

10.3) Максимальная скорость движения;

10.4) Максимальная скорость первого уровня;

10.5) Список точек движения;

10.6) Текущий индекс точки;

10.7) Трансформация.

11) Полоска здоровья воина. О ней нужно хранить следующую информацию:

11.1) Отступ от воина по Y координате;

11.2) Воин, к которому принадлежит полоска здоровья.

12) Замок. О нём нужно хранить следующую информацию:

12.1) Врага ли замок;

12.2) Строительство зданий;

12.3) Панель завершения игры;

12.4) Текущее здоровье;

12.5) Максимальное здоровье;

12.6) Текстовое поле с количеством текущего здоровья.

13) Полоска здоровья замка. О ней нужно хранить следующую информацию:

13.1) Отступ от замка по Y координате;

13.2) Замок, к которому принадлежит полоска здоровья;

13.3) Объект замка.

14) Деньги. О них нужно хранить следующую информацию:

14.1) Шахта;

14.2) Количество монет;

14.3) Выдаются ли монеты;

14.4) Количество монет в секунду;

14.5) Текстовое поле, отображающее количество монет.

15) Опыт. О нём нужно хранить следующую информацию:

15.1) Количество;

15.2) Текстовое поле, отображающее количество опыта.

16) Шахта. О ней нужно хранить следующую информацию:

16.1) Уровень;

16.2) Стоимость второго уровня;

16.3) Стоимость третьего уровня;

16.4) Стоимость четвёртого уровня;

16.5) Стоимость пятого уровня;

16.6) Стоимость шестого уровня;

16.7) Количество монет в секунду на первом уровне;

16.8) Количество монет в секунду на втором уровне;

16.9) Количество монет в секунду на третьем уровне;

16.10) Количество монет в секунду на четвёртом уровне;

16.11) Количество монет в секунду на пятом уровне;

16.12) Количество монет в секунду на шестом уровне;

16.13) Опыт;

16.14) Деньги;

16.15) Текстовое поле со стоимость повышения уровня;

16.16) Тестовое поле с названием;

16.17) Цвет максимального уровня;

16.18) Кнопка;

16.19) Максимального ли шахта уровня.

17) Строительство зданий. О нём нужно хранить следующую информацию:

17.1) Деньги;

17.2) Изменение башни;

17.3) Холст;

17.4) Список меню;

17.5) Список панелей;

17.6) Летающая башня;

17.7) Главная камера;

17.8) Открытое изменение башни;

17.9) Последняя башня;

17.10) На паузе ли игра;

17.11) Закончилась ли игра.

18) Изменение башни. О нём нужно хранить следующую информацию:

18.1) Башня, на которой было вызвано;

18.2) Отступ от башни по Y координате;

18.3) Деньги;

18.4) Текстовое поле с уровнем башни;

18.5) Прямоугольное преобразование.

18.6) Главная камера;

18.7) Текстовое поле с количеством монет, которые вернутся игроку в случае продажи башни;

18.8) Сработало ли условие один раз.

19) Генерация стартовой башни. О ней нужно хранить следующую информацию:

19.1) Список стартовых башен;

19.2) Место башни.

20) Подготовка к строительству. О ней нужно хранить следующую информацию:

20.1) Башня;

20.2) Деньги;

20.3) Текстовое поле со стоимостью башни;

20.4) Кнопка.

21) Отправка воинов. О ней нужно хранить следующую информацию:

21.1) Словарь отправляющихся воинов;

21.2) Текущая задержка между отправкой воинов;

21.3) Интервал между появлениями воинов;

21.4) Задержка перед отправкой войск;

21.5) Преобразование точек появления воинов;

21.6) Преобразование точек движения воинов;

21.7) Полоска здоровья воинов;

21.8) Замок;

21.9) Деньги;

21.10) Опыт;

21.11) Холст;

21.12) Противник ли это;

21.13) Текстовое поле со стоимостью выбранных воинов;

21.14) Текстовое поле с временем до возможности отправить воинов снова;

21.15) Количество всех воинов;

21.16) Список кнопок выбора воинов;

21.17) Стоимость всех воинов;

21.18) Словарь выбранных воинов;

21.19) Предпоследний сгенерированный индекс пути;

21.20) Последний сгенерированный индекс;

21.21) Не совпал ли сгенерированный индекс с предыдущим;

21.22) Воин;

21.23) Список всех позиций точек движения;

21.24) Кнопка отправки;

21.25) Максимальная задержка между отправкой воинов.

22) Улучшения башни. О нём нужно хранить следующую информацию:

22.1) Изменение башни, которое является родительским объектом;

22.2) Текстовое поле со стоимостью улучшения;

22.3) Преобразование кнопки удаления;

22.4) Кнопка;

22.5) Максимального ли уровня башня.

23) Найм воинов. О нём нужно хранить следующую информацию:

23.1) Отправка воинов;

23.2) Опыт;

23.3) Цвет текста при закрытии;

23.4) Цвет текста при открытии;

23.5) Текстовое поле со стоимостью воина;

23.6) Открыт ли воин;

23.7) Количество воинов;

23.8) Текстовое поле с количеством воинов;

23.9) Воин;

23.10) Кнопка.

24) Открытие воина. О нём нужно хранить следующую информацию:

24.1) Текстовое поле с уровнем воина;

24.2) Список объектов воина всех уровней;

24.3) Уровень воина;

24.4) Найм воинов.

25) Улучшение воина. О нём нужно хранить следующую информацию:

25.1) Текстовое поле со стоимостью улучшения;

25.2) Опыт;

25.3) Контроль панелями;

25.4) Открытие воина;

25.5) Уровень;

25.6) Открыт ли этот уровень;

25.7) Стоимость улучшения до этого уровня;

25.8) Воин;

25.9) Кнопка.

26) Отправка воинов врага. О ней нужно хранить следующую информацию:

26.1) Список воинов в волнах в количестве 45 штук;

26.2) Список воинов максимального уровня;

26.3) Список всех волн;

26.4) Количество волн;

26.5) Началась ли волна;

26.6) Отправка воинов.

**1.2 Постановка требований к проекту**

- Процессор: многоядерный Intel или AMD с поддержкой набора инструкций SSE

- Видеокарта: с поддержкой DX10, DX11, или DX12

- Операционная система: Windows 7 (SP1+) / 10 64-разрядная

**1.3 Изучение существующих платформ для разработки игр**

**1.3.1 Unity**

Unity - межплатформенная среда разработки компьютерных игр, разработанная американской компанией Unity Technologies. Unity позволяет создавать приложения, работающие на более чем 25 различных платформах, включающих персональные компьютеры, игровые консоли, мобильные устройства, интернет-приложения и другие. Выпуск Unity состоялся в 2005 году и с того времени идёт постоянное развитие.

Основными преимуществами Unity являются наличие визуальной среды разработки, межплатформенной поддержки и модульной системы компонентов. К недостаткам относят появление сложностей при работе с многокомпонентными схемами и затруднения при подключении внешних библиотек.

На Unity написаны тысячи игр, приложений, визуализации математических моделей, которые охватывают множество платформ и жанров. При этом Unity используется как крупными разработчиками, так и независимыми студиями.

Редактор Unity имеет простой Drag&Drop интерфейс, а также установкой плагинов KALI который легко настраивать, состоящий из различных окон, благодаря чему можно производить отладку игры прямо в редакторе. Движок использует для написания скриптов C#. Ранее поддерживались также Boo (диалект Python, поддержку убрали в 5-й версии) и модификация JavaScript, известная как UnityScript (поддержка прекращена в версии 2017.1). Расчёты физики производит физический движок PhysX от NVIDIA. Графический API — DirectX (на данный момент DX 11, поддерживается DX 12)

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.

К объектам можно применять коллизии (в Unity т. н. коллайдеры — collider), которых существует несколько типов.

Также Unity поддерживает физику твёрдых тел и ткани, а также физику типа Ragdoll (тряпичная кукла). В редакторе имеется система наследования объектов; дочерние объекты будут повторять все изменения позиции, поворота и масштаба родительского объекта. Скрипты в редакторе прикрепляются к объектам в виде отдельных компонентов.

При импорте текстуры в Unity можно сгенерировать alpha-канал, mip-уровни, normal-map, light-map, карту отражений, однако непосредственно на модель текстуру прикрепить нельзя — будет создан материал, которому будет назначен шейдер, и затем материал прикрепится к модели. Редактор Unity поддерживает написание и редактирование шейдеров. Редактор Unity имеет компонент для создания анимации, но также анимацию можно создать предварительно в 3D-редакторе и импортировать вместе с моделью, а затем разбить на файлы.

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. При компиляции проекта создается исполняемый (.exe) файл игры (для Windows), а в отдельной папке — данные игры (включая все игровые уровни и динамически подключаемые библиотеки).

Движок поддерживает множество популярных форматов. Модели, звуки, текстуры, материалы, скрипты можно запаковывать в формат .unitypackage и передавать другим разработчикам, или выкладывать в свободный доступ. Этот же формат используется во внутреннем магазине Unity Asset Store, в котором разработчики могут бесплатно и за деньги выкладывать в общий доступ различные элементы, нужные при создании игр. Чтобы использовать Unity Asset Store, необходимо иметь аккаунт разработчика Unity. Unity имеет все нужные компоненты для создания мультиплеера. UNet был удалён, начиная с версии 2018.4; решение «из коробки» для мультиплеера отсутствует. Также можно использовать подходящий пользователю способ контроля версий. К примеру, Tortoise SVN, Git или Source Gear.

В Unity входит Unity Asset Server — инструментарий для совместной разработки на базе Unity, являющийся дополнением, добавляющим контроль версий и ряд других серверных решений.

**1.3.2 Unreal Engine**

Unreal Engine — игровой движок, разрабатываемый и поддерживаемый компанией Epic Games. Первой игрой на этом движке был шутер от первого лица Unreal, выпущенный в 1998 году. Хотя движок первоначально был предназначен для разработки шутеров от первого лица, его последующие версии успешно применялись в играх самых различных жанров, в том числе стелс-играх, файтингах и массовых многопользовательских ролевых онлайн-играх. В прошлом движок распространялся на условиях оплаты ежемесячной подписки; с 2015 года Unreal Engine бесплатен, но разработчики использующих его приложений обязаны перечислять 5 % роялти от общемирового дохода с некоторыми условиями.

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, Xbox One, PlayStation 2, PlayStation 3, PlayStation 4,PlayStation 5, 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.

Для игры по сети поддерживаются технологии Windows Live, Xbox Live, GameSpy и прочие, включая до 64 игроков (клиентов) одновременно. Таким образом, движок адаптировали и для применения в играх жанра MMORPG.

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

Актёр (actor) (жаргонное выражение, принятое в среде 3D-моделлеров, разработчиков, заменяющее выражение «действующий объект» или «субъект» — являющееся переводом англ. actor — тот, кто действует — англ. action) — родительский класс, содержащий все объекты, которые имеют отношение к игровому процессу и имеют пространственные координаты.

Пешка (pawn) — физическая модель игрока или объекта, управляемого искусственным интеллектом. Название происходит от англ. pawn — тот, кем манипулируют (или пешка, поэтому такой объект без какой-либо модели выглядит как пешка). Метод управления описан специальным объектом, такой объект называется контроллером. Контроллер искусственного интеллекта описывает лишь общее поведение пешки во время игрового процесса, а такие параметры как «здоровье» (количество повреждений, после которых пешка перестаёт функционировать) или, например, расстояние, на котором пешка обращает внимание на звуки, задаются для каждого объекта отдельно.

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

Для работы с простыми и, как правило, неподвижными элементами игрового пространства (например, стены) используется двоичное разбиение пространства — всё пространство делится на «заполненное» и «пустое». В «пустой» части пространства располагаются все объекты, а также только в ней может находиться «точка наблюдения» при отрисовке сцены. Возможность полного или частичного помещения объектов в «заполненную» часть пространства не исключается, однако может привести к неправильной обработке таких объектов (например, расчёт физического взаимодействия) или неправильной отрисовки в случае помещения туда «точки наблюдения» (например, эффект «зала зеркал»). Все пешки, попадающие в «заполненную» часть пространства, сразу «погибают».

Зонирование. В камеру не попадает ни один портал (пунктирная линия) красной зоны, поэтому объекты в ней не обрабатываются вовсе.

Поверхность (surface) является основным элементом двоичного дерева пространства. Эти элементы создаются на грани пересечения между «заполненной» и «пустой» частями пространства. Группа элементов двоичного дерева пространства называется нодом (node, рус. узел). Этот термин, как правило, употребляется в контексте node count — количество нодов на экране или в игровом пространстве вообще. Количество нодов, одновременно видимых на экране, влияет на производительность при прорисовке сцены. Если какой-то нод не попадает на экран или перекрывается целиком другими нодами, он не обсчитывается — это служит для повышения производительности, особенно в закрытых пространствах. Разбиение всего пространства на группы нодов называется зонированием.

Для этого иногда используются порталы — невидимые поверхности, которые служат для того, чтобы вручную разделить крупный нод на два меньших (в версии движка Unreal Engine 3 ввели поддержку аддитивной геометрии, что позволило отказаться от зонирования). Кроме порталов, используются антипорталы.

Описание «заполненных» и «пустых» частей пространства выполняется с помощью набора замкнутых трёхмерных объектов, составленных из непересекающихся поверхностей — брашей (brush, рус. кисть). Этот принцип построения пространства называется конструктивной сплошной геометрией. Геометрия может быть «аддитивной» (всё пространство изначально «пустое») и «вычитательной» (изначально заполненное материей пространство).

Браши делятся на три типа:

- сплошные (solid) — полноценно участвуют в двоичном разбиении пространства.

- аддитивные (additive) — «заполняют» двоичное пространство.

- вычитательные (substractive) — «вырезают» объёмы в пространстве.

- полусплошные (semi-solid) — не влияют напрямую на двоичное дерево пространства, однако влияют на её физическую модель. Могут только «заполнять» пространство. Служат для создания «невидимых» препятствий, а также снижения числа полигонов и нодов.

- пустые (non-solid) — только создают поверхности, не влияют на двоичное дерево пространства. Используются преимущественно для создания объёмов (volume) — часть пространства, которая обладает свойствами, отличными от свойств игрового мира. Объёмы имеют приоритет, свойства объёма с большим приоритетом применяются к находящимся в нём актёрам. Игровой мир всегда имеет минимальный приоритет. При помощи объёмов можно изменить гравитацию, вязкость, туман и тому подобное. Объёмы, начиная с версии движка Unreal Engine 2, используются для создания воды (но не водной поверхности).

**1.3.3 CryEngine 3**

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

CryEngine 3 официально анонсирован 11 марта 2009 года, 14 октября 2009 года состоялся официальный релиз (выпуск) движка. Первая игра, для которой было анонсировано использование CryEngine 3 — шутер от первого лица Crysis 2 компании Crytek.

CryEngine 3 является кроссплатформенным движком, он поддерживает IBM PC-совместимые компьютеры и игровые консоли Microsoft Xbox 360 и Sony PlayStation 3.

Шейдеры в CryEngine 3 пишутся единоразово на языке программирования высокого уровня, а затем автоматически компилируются под каждую платформу. Вывод шейдеров оптимизируется через настройки художника и под ту трёхмерную среду, в которой данный шейдер будет использоваться. Благодаря этому становится возможным создание таких эффектов, как: «эффект невидимости», мокрые, грязные и замороженные поверхности, которые могут быть «наслоены» одна на другую и объединены с другими шейдерами, симулирующими такие эффекты, как металл, стекло и другие. CryEngine 3 поддерживает попиксельное освещение реального времени, отражения, преломления, эффекты объёмного жара и анимированные текстуры для симуляции окон, пулевых отверстий, поверхностей с солнечными бликами и много других эффектов. Шейдеры CryEngine 3 используют унифицированную шейдерную архитектуру, которая стала доступна начиная с Direct3D 10.

«Übershader» представляет собой одну шейдерную программу со многими особенностями: от одного до четырёх источников света, типы источников света, кубические карты отражений, туман, детализированные текстуры, карты нормалей, зеркальные текстуры и др. Могут быть сгенерированы миллионы комбинаций Übershader. В Übershader используется динамическое ветвление, разделение на множество проходов, уменьшение комбинаций и принятие вариантов с меньшей функциональностью и меньшей требуемой производительностью. Используется асинхронная компиляция шейдеров и распределённая система заданий (англ. Distributed Job System) для компиляции шейдерного кеша.

CryEngine 3 поддерживает технологию Ati Eyefinity, которая присутствует в графических процессорах Radeon серии 5xxx. Данная способность была продемонстрирована Карлом Джонсом (англ. Carl Jones) 11 сентября 2009 года, сразу после общественной демонстрации технологии Ati Eyefinity. Джонс показал демонстрационное видео CryEngine 3, которое было запущено на видеокарте Radeon 5xxx и на шести дисплеях. При этом суммарное разрешение выводимой «картинки» составляло 5 760 x 2 160 пикселей (шесть дисплеев с разрешением 1 920 x 1 080), хотя теоретически максимальное поддерживаемое разрешение составляет 7 680 x 3 200 пикселей.

**1.4 Выбор технологии для разработки приложений**

Для разработки проекта был выбран движок Unity, так как основным языком является C# и лёгкость в обучении. Так же, игровой движок предоставляет множество функциональных возможностей, позволяющих их задействовать в различных играх, в которые входят моделирование физических сред, карты нормалей, динамические тени и многое другое. В отличие от многих игровых движков, у Unity имеется два основных преимущества: наличие визуальной среды разработки и межплатформенная поддержка. Первый фактор включает не только инструментарий визуального моделирования, но и интегрированную среду, цепочку сборки, что направлено на повышение производительности разработчиков, в частности, этапов создания прототипов и тестирования. Под межплатформенной поддержкой предоставляется не только места развертывания (установка на персональном компьютере, на мобильном устройстве, консоли и т. д.), но и наличие инструментария разработки (интегрированная среда может использоваться под Windows и Mac OS).

Третьим преимуществом называется модульная система компонентов Unity, с помощью которой происходит конструирование игровых объектов, когда последние представляют собой комбинируемые пакеты функциональных элементов. В отличие от механизмов наследования, объекты в Unity создаются посредством объединения функциональных блоков, а не помещения в узлы дерева наследования. Такой подход облегчает создание прототипов, что актуально при разработке игр.

**2 Разработка приложения**

**2.1 Разработка префабов для игры**

Основным для создания префаба является создание модели.

Для создания модели карты использовалась программа MagicaVoxel, которая позволяет создавать модели при мощи 3D пикселей. Благодаря этой программе были созданы пещера, замок, дорога от пещеры до замка с окружающими её полями для башен.

Для создания самих башен и воинов использовалась программа Blender, позволяющая создавать оптимизированные модели с малым количеством полигонов и с возможностью использовать одну текстуру для всех моделей. В этой программе были созданы четыре вида башен, три вида воинов с пятью уровнями каждого из них.

После создания моделей и при перенесении их в игровой движок, создаётся пустой объект, куда и помещается модель. При том, пустой объект должен иметь стандартные параметры, к примеру, координаты, размер и т.д. На пустой объект, ставший родителем, вешаются все необходимые компоненты. Это делается для того, чтобы при проигрывании анимаций, взаимодействия с объектами и т.д. префаб не выходил из строя и логику производил именно он.

В завершении, объект перекидывается в нужную папку в проекте, после чего и становится префабом.

**2.2 Структура уровня**

Стартовая сцена состоит лишь из зелёного поля и случайно генерируемой башни первого уровня.

Игровой уровень состоит из карты, на которой расположены две территории (союзническая и вражеская), на каждой из которых располагаются замок, пещера, дорога связывающая эти два строения и окружающие её две платформы с местами под башни. На стороне противника уже расположены башни различных видов и уровней. Также, всю карту окружают деревья.

**2.3 Описание игровой логики**

При старте игры показывается главное меню слева и случайная башня первого уровня справа. Игрок может начать игру, настроить игру или выйти из игры.

При выборе начать игру, отображается игровая сцена, где и находится основной игровой уровень. Игрок начинает игру с пятьюстами монет, тысячью очков здоровья замка и нулём опыта.

Противник же не имеет таких параметров, кроме количества здоровья своего замка. Так же, противник начинает игру уже с выстроенными башнями различных видов и уровней, которые будут убивать войска, отправленные игроком. За неимением денег и опыта, противник будет отправлять каждые двадцать секунд первые сорок пять волн войска, строго заданными самой игрой. После сорок пятой волны войска отправляются случайным образом из трёх видов воинов пятого уровня. После пятидесятой волны войска начинают отправляться без перерывов.

Игрок может строить башни на полях своей территории, улучшать и продавать их. При том, строить и улучшать башни он может только, если у него хватит на это монет, так как, при строительстве и улучшении башен, стоимость списывается из средств игрока. При продаже башни, возвращается стоимость, но в два раза меньше.

Строительство башни осуществляется при помощи выбора её в панели, после чего она зацепляется за курсор игрока. Если башню невозможно установить на место, указанное курсором, башня будет подсвечиваться красным, если же можно, то зелёным. Так же, при установке башни, вокруг неё будет нарисован круг, показывающий дальность атаки башни. При наведении курсора на незанятое поле игрока, башня закрепится в центре это поля, но после передвижения курсора в сторону этого поля, отцепится от него.

Игрок может ставить башни при помощи панели, но, если он уже установил хотя бы одну башню, он может установить ещё одну или несколько таких же башен при помощи сочетания клавиш SHIFT + ЛКМ по полям территории игрока. Также можно и улучшать башни при помощи сочетания клавиш SHIFT + ПКМ по уже установленным башням. Но строительство и улучшение башен сработает лишь в том случае, если на эти действия будет хватать монет.

**2.4 Описание интерфейса игры**

**ЗАКЛЮЧЕНИЕ**

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**