**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 2](#_Toc167397571)

[ГЛАВА 1. Исследование рынка видеоигр и выбор инструментов для реализации продукта 5](#_Toc167397572)

[1.1 Обзор рынка 5](#_Toc167397573)

[1.2 Игровой движок 6](#_Toc167397574)

[1.3 Язык программирования Haxe 9](#_Toc167397575)

[1.4 Entity component system 11](#_Toc167397576)

[1.4 Графический и аудио редактор 15](#_Toc167397577)

[ГЛАВА 2. Проектирование игрового процесса и поиск подходящих материалов 19](#_Toc167397578)

[2.1 Проектирование игрового процесса 19](#_Toc167397579)

[2.2 Поиск материалов и ресурсов 20](#_Toc167397580)

[ГЛАВА 3. Разработка проекта 22](#_Toc167397581)

[3.1 Разработка системы отрисовки 22](#_Toc167397582)

[ЗАКЛЮЧЕНИЕ 23](#_Toc167397583)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 24](#_Toc167397584)

[ПРИЛОЖЕНИЯ 26](#_Toc167397585)

# ВВЕДЕНИЕ

Популярность компьютерных игр растет с каждым годом, и все большее число людей увлекается этим видом развлечений. Эта тенденция является прямым следствием технологического прогресса и широкой доступности персональных компьютеров, планшетов и мобильных устройств. Таким образом, растущий рынок развлечений является убедительным основанием для инвестирования в разработку новых игровых проектов.

Значение разработки игр становится все более очевидным, поскольку игровой рынок стремительно расширяется с каждым годом. Самые успешные игры продаются миллионными тиражами. Игровая индустрия зародилась в 1970-х годах, одновременно с появлением первых устройств, ориентированных на потребителя. За последние четыре десятилетия индустрия компьютерных игр значительно выросла, и теперь превосходит своих ближайших конкурентов вместе взятых.

Устоявшиеся программные решения для разработки видеоигр зачастую не имеют открытых лицензий, и лишь немногие дают возможность создать высокопроизводительный продукт. Haxe и Heaps.io обеспечивают оптимизированный подход к созданию производительных кроссплатформенных игр. Высокоуровневый язык программирования Haxe значительно ускоряет разработку программного обеспечения благодаря возможности работать на разных платформах используя еденную кодовую базу. Heaps.io это графический фреймворк, доказавший свою эффективность многими успешными видеоиграми, которые использовали его для разработки. Он даёт базовые инструменты необходимые для разработки мультимедийных приложений, а также возможность использовать различное графическое API на всех основных платформах для видеоигр.

**Актуальность работы** обоснована популярностью у потребителей развлекательного контента и в особенности видеоигр из чего можно заявить, что на сегодняшний день разработка игр может являться актуальным и выгодным проектом.

**Объектом исследования** является проектирование и разработка видеоигр.

**Предмет исследования** является разработка видеоигры с использованием игрового движка Heaps.io на высокоуровневом языке программирования Haxe.

**Практическая значимость** работы заключается в создании прототипа игры, который может использоваться как демонстрация для инвесторов, журналистов, издателей или даже будущих покупателей для того, чтобы заинтересовать их в продукте.

**Цель работы:** разработать демонстрационную версию видеоигры использованием открытого программного обеспечения «Heaps.io».

**Задачи исследования:**

* проанализировать ситуацию на рынке выбранного направления;
* раскрыть целесообразность создания данной игры;
* обосновать использование игрового программного обеспечения «Heaps.io»;
* исследовать возможности языка программирования Haxe;
* изучить доступные графические и аудио редакторы;
* спроектировать игровой процесс;
* спроектировать пользовательский интерфейс;
* поиск подходящих видео и аудио материалов;
* разработать продукт;
* протестировать продукт;
* доработать выявленные недостатки;

Поставленные цели, сформулированные задачи, предмет и объект исследования обусловили структуру выпускной квалификационной работы, которая состоит из введения, трёх глав, заключения, списка использованной литературы и ряда приложений, иллюстрирующих основное содержание выпускной квалификационной работы.

# ГЛАВА 1. Исследование рынка видеоигр и выбор инструментов для реализации продукта

## 1.1 Обзор рынка

Индустрия видеоигр превратилась из нишевого рынка в доминирующий область развлекательного рынка. За последние несколько десятилетий индустрия видеоигр пережила беспрецедентный рост, вызванный развитием технологий и распространением персональных вычислительных устройств. Мировой рынок видеоигр демонстрирует уверенный рост: в 2021 году его доходы достигнут примерно 180 миллиардов долларов, и прогнозы указывают на продолжение роста в ближайшие годы. Стремительное развитие отрасли превратило ее в одного из основных игроков индустрии развлечений, обогнав таких традиционных конкурентов, как кино, музыка и телевидение.

Поведение потребителей также изменилось, и игры стали основным видом деятельности среди различных демографических групп. Стереотипное представление о геймере как о молодом мужчине изменилось, и значительный рост наблюдается среди женщин-геймеров и более старших возрастных групп. Такая диверсификация игровой аудитории открыла перед разработчиками новые возможности для создания контента, отвечающего более широкому кругу предпочтений и интересов.

Прибыльность разработки видеоигр подтверждается впечатляющими финансовыми показателями индустрии. Зачастую громкие релизы игр приносят существенный доход уже через несколько дней после их выхода, что подчеркивает прибыльный потенциал успешных игр.

Прибыльность разработки видеоигр обусловлена несколькими факторами:

* Масштабируемость: в отличие от физических товаров, цифровые игры можно распространять по всему миру с минимальными дополнительными затратами, что позволяет разработчикам охватить огромную аудиторию.
* Постоянный доход: Появление внутриигровых покупок, загружаемого контента (DLC) и моделей подписки обеспечивает постоянные потоки доходов после первоначальной продажи игры.
* Лояльность к бренду и франшизы: Успешные игры часто порождают франшизы, а сиквелы и спин-оффы получают выгоду от устоявшихся фанатских баз и узнаваемости бренда.
* Разнообразные стратегии монетизации: Разработчики могут использовать различные стратегии монетизации, включая рекламу, партнерство и спонсорство в эсспорте, для получения максимального дохода.

Индустрия видеоигр представляет собой динамичный и быстрорастущий рынок со значительным потенциалом. Технологический прогресс, меняющееся поведение потребителей и разнообразные модели получения прибыли способствуют устойчивым финансовым показателям отрасли. Для инвесторов и разработчиков сектор видеоигр открывает широкие возможности, делая его стратегически выгодным предприятием в современном развлекательной индустрии. По мере того как индустрия будет продолжать внедрять инновации и расширяться, разработка видеоигр останется перспективным направлением для экономического роста и творческого самовыражения.

## 1.2 Игровой движок

Игровой движок - это программная основа, предназначенная для создания и разработки видеоигр. Как правило, он включает в себя набор инструментов и библиотек, обеспечивающих такие функции, как рендеринг графики, обработка физики, обработка ввода и управление игровыми активами. Игровые движки абстрагируют большую часть технических сложностей, позволяя разработчикам сосредоточиться на игровом процессе и дизайне.

Концепция игровых движков возникла в конце XX века, а ранними примерами стали движки id Tech от id Software, на которых были созданы такие знаковые игры, как Doom и Quake. В этих движках была реализована идея многоразовых кодовых баз, которые можно было лицензировать для использования в нескольких играх, что положило начало современному рынку игровых движков.

На текущий момент существует множество различных игровых движков, каждый из которых имеет свои сильные и слабые стороны. Был проведён сравнительный анализ наиболее популярных игровых движков таких как Unity, Unreal Engine, Godot и Heaps.io, чтобы подчеркнуть их возможности и ограничения.

1. Unity

Сильными сторонами Unity являются универсальность и простота использования. Он поддерживает широкий спектр платформ, от мобильных до VR, и предлагает большое количество готовых решений. Скрипты Unity основаны на языке программирования C#, который прост в изучении и использовании. Так же Unity является движком с самым крупным сообществом, что помогает в решении возникших проблем.

Несмотря на свои сильные стороны, Unity может страдать от проблем с производительностью, особенно в сложных графических проектах, а факт того, что код движка закрыт мешает решению возникших трудностей и модификации движка. Так же стоимость лицензирования может быть непомерно высокой для небольших команд и одиночных разработчиков.

1. Unreal Engine

Unreal Engine славится своей реалистичной графикой и надежным набором инструментов. Он особенно популярен при разработке высокобюджетных игр и предлагает мощные возможности, как система визуальных сценариев Blueprint и продвинутое моделирование физики.

Сложность и комплексность Unreal Engine может стать препятствием для новичков. Высокая требовательность движка к ресурсам также может затруднить его оптимизацию для менее мощных платформ, а его ориентация на реалистичную графику делает его неподходящим вариантом для небольших проектов.

1. Godot

Godot - это движок с открытым исходным кодом, известный своим богатым функционалом и простотой использования. Он поддерживает разработку 2D- и 3D-игр и имеет простой скриптовый язык GDScript, который похож на Python. Проекты на этом движке являються очень легковесными, что критично на мобильных и веб платформах.

Несмотря на богатый функционал Godot, он менее развит, чем Unity и Unreal Engine, активная стадия разработки началась не так давно относительно конкурентов, поэтому движок критично не хватает оптимизации и базового функционала для рендеринга, которые есть у его конкурентов.

1. Heaps.io

Heaps.io - это высокопроизводительный игровой движок с открытым исходным кодом, предназначенный для разработки кроссплатформенных игр. Он отличается эффективностью и гибкостью, предлагая легкий фреймворк, способный обрабатывать сложные игры с меньшим потреблением ресурсов. Heaps.io построен на языке программирования Haxe, что позволяет без проблем создавать кросс-компиляции для различных платформ.

Основным ограничением Heaps.io является недостаток документации, небольшое сообщество и меньшее количество готовых решений по сравнению с более известными движками. Однако для опытных разработчиков его сильные стороны - производительность и гибкость - часто перевешивают эти недостатки.

После проведения сравнительного анализа было принято решение выбрать Heaps.io для разработки видеоигры. Преимущества Heaps.io делают его хорошим выбором для опытных разработчиков игр, особенно для тех, кто ориентирован на производительность и кроссплатформенность. Благодаря своей легковесности игры эффективно работают на различных устройствах, от мобильных телефонов до высокопроизводительных ПК, без ущерба для качества. Использование Haxe обеспечивает мощную кроссплатформенную поддержку, что позволяет переносить игры на различные платформы с минимальными изменениями.

Более того, открытый исходный код Heaps.io позволяет разработчикам изменять и расширять движок в соответствии со своими потребностями, способствуя инновациям и кастомизации.

Игровые движки незаменимы в современном мире разработки игр, предоставляя важные инструменты и фреймворки, которые упрощают процесс создания. Хотя Unity, Unreal Engine и Godot предлагают уникальные преимущества, Heaps.io выделяется своей производительностью, гибкостью и кроссплатформенными возможностями. Несмотря на небольшое сообщество и меньшее количество готовых решений, сильные стороны Heaps.io делают ее оптимальным выбором для опытных разработчиков, стремящихся создавать качественные и эффективные игры на разных платформах. Поскольку игровая индустрия продолжает развиваться, Heaps.io представляет собой перспективное, адаптируемое решение, которое отвечает разнообразным потребностям современных разработчиков игр.

## 1.3 Язык программирования Haxe

Haxe - это статически типизированный язык с открытым исходным кодом, предназначенный для работы на нескольких платформах с одной кодовой базой. Он компилируется на различные платформы, включая JavaScript, C++, Java, C#, PHP и Python, что делает его идеальным выбором для кроссплатформенной разработки. Способность Haxe генерировать исходный код для нескольких языков позволяет разработчикам писать код один раз и разворачивать его на разных платформах с минимальными изменениями. Эта особенность особенно ценна в контексте разработки игр, где игры часто должны запускаться на различных устройствах, таких как ПК, консоли и мобильные телефоны.

Синергия между Haxe и Heaps.io приводит к оптимизации процесса разработки, где повторное использование кода и его согласованность максимальны, что сокращает время и усилия, необходимые для переноса игр на разные платформы.

Одним из значительных преимуществ использования Haxe с Heaps.io является оптимизация производительности. Процесс компиляции Haxe позволяет разработчикам создавать высокооптимизированный код для каждой целевой платформы. Например, Haxe может генерировать нативный код на C++ для критически важных приложений, обеспечивая плавную работу игр даже на устройствах с ограниченными ресурсами. Heaps.io использует эту возможность для создания легкого и эффективного игрового движка, способного обрабатывать сложную игровую логику и графику высокой четкости без значительных накладных расходов.

Еще одним преимуществом использования Haxe в сочетании с Heaps.io является гибкость и выразительность языка. Haxe сочетает в себе черты различных парадигм программирования, включая объектно-ориентированное, функциональное и императивное программирование, что позволяет разработчикам выбирать наилучший подход для своих конкретных нужд. Эта гибкость отражена в дизайне Heaps.io, который предоставляет полный набор инструментов и библиотек для разработки игр, от рендеринга и аудио до физики и сетевого взаимодействия. Легкость интеграции пользовательских модулей и расширения функциональности Heaps.io свидетельствует об универсальности Haxe.

Экосистема Haxe еще больше расширяет возможности разработчиков, предлагая богатый набор библиотек и фреймворков. Стандартная библиотека Haxe включает в себя основные утилиты и функции, а сторонние библиотеки предоставляют дополнительные возможности, такие как компоненты графического интерфейса, структуры данных и алгоритмы. Эта обширная экосистема дополняет возможности Heaps.io, позволяя разработчикам быстро создавать сложные игры с расширенными возможностями. Наличие множества библиотек также способствует быстрому созданию прототипов и проведению экспериментов.

Открытый исходный код - дополнительные преимущества Haxe и Heaps.io. Открытый исходный код Haxe и Heaps.io означает, что разработчики могут проверять, изменять и улучшать исходный код.

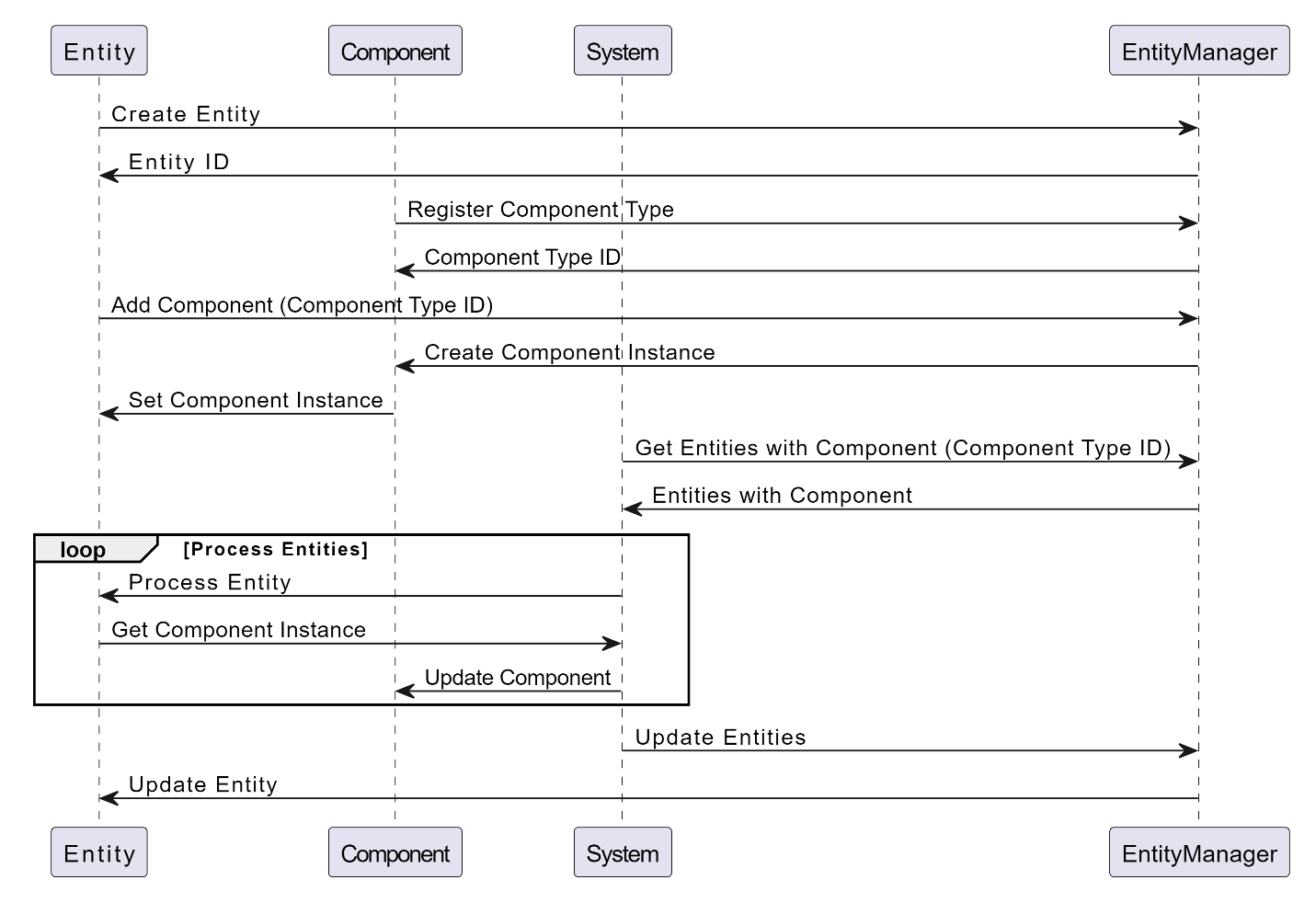
Сочетание Haxe и Heaps.io также решает проблему поддержки и масштабируемости при разработке игр. Сильная типизация и модульная конструкция Haxe способствуют созданию чистых, удобных для обслуживания кодовых баз. Разработчики могут определять четкие интерфейсы и инкапсулировать функциональность в модулях многократного использования, сокращая дублирование кода и улучшая читабельность. Архитектура Heaps.io поддерживает масштабируемую разработку игр, позволяя командам эффективно управлять проектами.

## 1.4 Entity component system

В качестве основополагающей архитектуры приложения был выбран архитектурный паттерн Entity component system.

Entity component system (ECS) — это архитектурный паттерн программного обеспечения, который в основном используется в разработке видеоигр для представления объектов игрового мира. ECS включает в себя сущности, составленные из компонентов данных, и системы, которые работают с этими компонентами.

ECS следует принципу композиции над наследованием, что означает, что каждая сущность определяется не иерархией типов, а компонентами, которые с ней связаны. Системы действуют глобально над всеми сущностями, которые имеют необходимые компоненты.



Сущности в ECS являются базовыми единицами идентификации и служат контейнерами для компонентов. Сама сущность часто представляет собой простой идентификатор, например целое число или уникальную строку, лишенную каких-либо данных или поведения. Такая простота позволяет эффективно управлять и обрабатывать сущности в игровом движке.

Компоненты — это контейнеры данных, в которых хранятся атрибуты сущностей. Каждый тип компонента содержит определенные данные, такие как позиция, скорость, здоровье или текстура, но не содержит никакого поведения или логики. Разлагая данные сущности на отдельные компоненты, ECS способствует модульности и возможности повторного использования. Например, компонент положения может быть использован в различных типах сущностей, от персонажей игрока до внутриигровых предметов, без изменений.

Системы отвечают за реализацию логики и поведения сущностей. Система работает с сущностями, обладающими определенными компонентами, обновляя их данные на основе игровых правил и механики. Системы перебирают соответствующие компоненты и применяют необходимые вычисления или преобразования. Например, система физики может обновлять компоненты положения и скорости всех сущностей, основываясь на физическом моделировании, а система рендеринга может рисовать сущности на экране, используя их компоненты положения и текстуры.

Истоки ECS можно проследить в ранних попытках разработки игр управлять растущей сложностью игровых объектов и их взаимодействий. Традиционные подходы ООП, которые часто включают в себя глубокие иерархии наследования и тесно связанные классы, становились все более сложными для поддержки и расширения. ECS появился как решение этих проблем, делая акцент на композиции, а не на наследовании, что позволяет создавать более гибкие и удобные в обслуживании кодовые базы.

Одно из главных преимуществ ECS - возможность повысить производительность за счет ориентированного на данные проектирования. Организуя компоненты в смежные блоки памяти, системы ECS могут использовать современные архитектуры кэш-памяти процессора для более эффективной обработки данных. Такой подход, ориентированный на кэш, минимизирует время доступа к памяти и максимизирует пропускную способность, что особенно полезно для приложений реального времени, таких как видеоигры.

Еще одно существенное преимущество ECS - поддержка параллельной обработки. Поскольку системы работают с отдельными наборами компонентов, они часто могут выполняться параллельно без конфликтов. Такой параллелизм позволяет игровым движкам в полной мере использовать преимущества многоядерных процессоров, распределяя рабочую нагрузку между несколькими потоками и повышая общую производительность. Разделение данных и поведения в ECS еще больше упрощает задачу распараллеливания игровой логики.

ECS также повышает гибкость и масштабируемость разработки игр. Используя композицию вместо наследования, разработчики могут создавать сложные объекты, комбинируя простые компоненты многократного использования. Такой модульный подход позволяет добавлять, удалять или изменять функции, не затрагивая при этом несвязанные части кодовой базы. В результате ECS способствует более итеративному и гибкому процессу разработки, позволяя вносить изменения и дополнения на протяжении всего цикла разработки.

Язык программирования Haxe имеет множество различных фреймворков реализующий архитектурный паттерн ECS, но, к сожалению, большинство из них перестали разеваться годы назад. На основании этого был выбран фреймворк Echoes который продолжает активно поддерживаться и адаптироваться под новые версии Haxe.

Echoes – это Entity Component System фреймворк на основе макросов, ориентированная на простоту использования. Этот фреймворк был разработан и реализован пользователем deepcake и сейчас поддерживается player-03. Данный фреймворк является отличной реализацией паттерна и используя возможности макросов языка Haxe.

В заключение следует отметить, что парадигма Entity Component System представляет собой значительный прогресс в архитектуре разработки игр, устраняя ограничения традиционных объектно-ориентированных подходов. Разделяя сущности, компоненты и системы, ECS способствует модульности, повторному использованию и удобству обслуживания. Его дизайн, ориентированный на данные, и поддержка параллельной обработки повышают производительность, что делает его хорошо подходящим для приложений реального времени. Широкое распространение ECS в современных игровых движках подчеркивает его эффективность и универсальность, укрепляя его роль в качестве основополагающей технологии в эволюции разработки игр.

## 1.4 Графический и аудио редактор

Выбор графических и аудиоредакторов — важнейший аспект разработки видеоигр, существенно влияющий на качество и эффективность создания ресурсов.

Пиксель-арт — это особый и популярный стиль графики видеоигр, часто характеризующийся ретро-эстетикой и точным дизайном на основе сетки. Выбор подходящего графического редактора имеет важное значение для достижения желаемого визуального качества и эффективности рабочего процесса. Среди различных доступных инструментов наиболее заметными являются Aseprite, Photoshop и Piskel.

Aseprite широко известен как лучший графический редактор для пиксельной графики благодаря своим специализированным функциям и удобному интерфейсу. Он предлагает полный набор инструментов, специально разработанных для пиксельной графики, включая наложение сетки, луковую очистку для анимации и систему управления палитрой. Интуитивно понятный интерфейс Aseprite позволяет сосредоточиться на творческом процессе, не перегружая ненужными функциями. Кроме того, поддержка слоев и покадровой анимации делает его идеальным выбором для создания детализированной и анимированной пиксельной графики. Однако Aseprite — платное программное обеспечение, но благодаря своей лицензии его можно собрать из исходного кода и использовать бесплатно.

Photoshop, хорошо зарекомендовавший себя графический редактор, также поддерживает создание пиксельной графики, но менее специализирован по сравнению с Aseprite. Photoshop предлагает широкий спектр функций, включая расширенные инструменты манипулирования изображениями, обширные возможности кистей и надежное управление слоями. Эти функции могут быть полезны для более сложных проектов, требующих сочетания пиксельной графики и других графических стилей. Однако сложность и высокая стоимость Photoshop могут оказаться ошеломляющими для новичков и тех, кто сосредоточен исключительно на пиксельной графике. Его универсальный дизайн может также привести к менее оптимизированному рабочему процессу для задач, специфичных для пикселей.

Piskel, веб-редактор пиксельной графики с открытым исходным кодом, предоставляет более доступную альтернативу. Piskel предлагает необходимые инструменты для создания пиксельной графики, такие как наложение сетки, поддержка анимации и управление палитрой. Простота и удобство использования делают его отличным выбором для новичков или тех, кто работает над небольшими проектами. Будучи веб-инструментом, Piskel не зависит от платформы и не требует установки, что делает его очень доступным. Однако его набор функций ограничен по сравнению с Aseprite и Photoshop, что может ограничивать более сложные или крупномасштабные проекты.

В области редактирования звука при разработке игр выбор правильного инструмента имеет решающее значение для достижения высококачественного звукового дизайна и эффективного рабочего процесса. Reaper, Audacity и FL Studio входят в число ведущих аудиоредакторов, каждый из которых имеет свои преимущества и недостатки.

Reaper считается лучшим аудиоредактором для разработки игр благодаря своим мощным функциям, гибкости и доступности. Он поддерживает широкий спектр аудиоформатов и предлагает расширенные возможности редактирования. Настраиваемый интерфейс Reaper и возможности создания сценариев позволяют пользователям адаптировать программное обеспечение к своим конкретным потребностям, повышая производительность. Кроме того, модель лицензирования Reaper экономически эффективна, что делает ее доступной как для любителей, так и для профессиональных разработчиков. Однако крутая кривая обучения Reaper может быть недостатком для новичков, незнакомых с программным обеспечением для редактирования аудио.

Audacity, популярный аудиоредактор с открытым исходным кодом, известен своей простотой и удобством использования. Он предоставляет основные инструменты редактирования аудио, такие как вырезание, копирование, вставка, а также базовые эффекты, такие как реверберация и эквалайзер. Будучи бесплатным и открытым исходным кодом, он легко доступен для всех разработчиков. Однако Audacity не хватает расширенных функций и возможностей профессионального уровня Reaper, что может ограничивать сложное звуковое оформление и высококачественное производство.

FL Studio также предлагает надежные возможности редактирования звука. Он включает в себя полный набор инструментов для создания, аранжировки и редактирования звука, включая широкий спектр виртуальных инструментов и эффектов. Рабочий процесс FL Studio, основанный на шаблонах, особенно подходит для создания сложных аудиокомпозиций и их интеграции в игровые проекты. Однако его высокая стоимость и ориентация на производство музыки могут сделать его менее подходящим для разработчиков, которым нужен специальный аудиоредактор для звукового оформления игр.

Выбор графических и аудиоредакторов может сыграть ключевую роль при разработке видеоигр. Aseprite и Reaper являются лучшим выбором в своих категориях благодаря своим специализированным функциям, гибкости и экономической эффективности. Акцент Aseprite на пиксельной графике и удобный интерфейс делают его оптимальным выбором для редактирования графики, а мощные возможности Reaper и настраиваемый рабочий процесс делают его лучшим аудиоредактором для разработки игр. Хотя Photoshop, Piskel, Audacity и FL Studio предлагают уникальные преимущества, они в конечном итоге менее подходят для конкретных потребностей создания пиксельной графики и редактирования игрового звука по сравнению с Aseprite и Reaper.

# ГЛАВА 2. Проектирование игрового процесса и поиск подходящих материалов

## 2.1 Проектирование игрового процесса

Важной частью разработки является проектирование игрового процесса, поскольку он служит основой всего процесса создания игры. Эффективный дизайн игрового процесса включает в себя концептуализацию и детализацию механики, правил и взаимодействий, которые будут определять опыт игрока.

Концепция игры заключается в создании аркадного top-down шутера. В игре игроку предстоит как можно дольше сохранять показатели здоровья выше нуля, сражаясь с волнами противников, в попытке набрать наибольшее количество очков.

Были выделены следующие основные концепции проекта:

1. Процедурная генерация фона: для визуального разнообразия игрового процесса было решено генерировать задний фон на основе шума Перлина. Данный шум используется в компьютерной графике для увеличения реализма или графической сложности поверхности геометрических объектов. Шум Перлина даёт реалистичное представление ландшафта, что улучшит визуальную составляющею игры и избавит от необходимости создавать фон вручную.
2. Динамическая сложность: для интересного игрового процесса требуется соответствующий уровень сложности. С течением времени сложность будет пропорционально расти. Данная механика будет достигаться количеством и разнообразием противников, которые будут появляться в процессе игры.
3. Разнообразие противников: для достижения определённого уровня сложности необходимо разнообразие противников, которые вынуждают игрока на различные модели поведения. Комбинирование нескольких противников, с различным искусственным интеллектом генерируют новые игровые ситуации, что обогащает игровой процесс.
4. Предметы: в процессе игры при уничтожении противника игроку может выпасть один из двух предметов: первый предмет позволяет восстановить часть здоровья игрока, второй усиливает его.
5. Пользовательский интерфейс: на экране должна содержаться вся информация необходимая игроку. Ключевыми показателями являются здоровье, количество заработанных очков и время с начала игры.

Сформулировав основные концепции игрового процесса, появляется конкретное представление о графических и аудио аспектах, которые должны присутствовать в проекте.

## 2.2 Поиск материалов и ресурсов

Выбор правильных игровых ресурсов имеет решающее значение для определения визуального и звукового стиля игры, улучшения погружения и обеспечения последовательности. Высококачественные ресурсы вносят значительный вклад в общий опыт игроков и могут выделить игру на переполненном рынке. Использование ресурсов по лицензии Creative Commons Zero (CC0) особенно выгодно, поскольку позволяет разработчикам свободно использовать, изменять и распространять ресурсы без юридических ограничений или требований об указании авторства. Такая гибкость ускоряет разработку, снижает затраты и облегчает творческий подход, позволяя разработчикам больше сосредоточиться на инновациях и игровом процессе, а не на юридических проблемах или создании ресурсов с нуля.

Были найдены и использованы следующие ресурсы с лицензией CC0:

1. основой графической части проекта стал набор Pixel Shmup за авторством kenney (см. приложение А);
2. пользовательский интерфейс был создан с использованием Pixel UI Pack за авторством kenney (см. приложение Б);
3. в качестве шрифта был использован Monogram за авторством datagoblin
4. для создания аудио эффектов были использованы наборы Impact Sounds, Sci-fi Sounds, Digital Audio за авторством kenney.

Собрав необходимые ресурсы, была начата разработка проекта.

# ГЛАВА 3. Разработка проекта

## 3.1 Разработка системы отрисовки

Установив необходимые компоненты и создав проект, первым шагом в разработке является отрисовка графики. Heaps.io предоставляет несколько способов для достижения данного функционала.

1. Bitmap — это простой контейнер для рисования одного тайла. Это самый простой тип, но его следует использовать с осторожностью, поскольку каждый Bitmap вызывает отдельный вызов отрисовки.
2. TileGroup позволяет пакетно рисовать плитки, принадлежащие одной и той же текстуре, с возможностью применять индивидуальный оттенок, преобразование и прозрачность к каждой плитке. Этот класс можно использовать для рисования статических тайлов, но он не очень хорошо подходит для динамического перемещения, так как его необходимо очищать и назначать данные каждый раз, когда что-то нужно изменить.
3. SpriteBatch позволяет упаковывать в пакет данных для отрисовки тайлы, принадлежащие одной и той же текстуре, но, в отличие от TileGroup, он обновляет данные в каждом кадре и позволяет управлять каждым элементом SpriteElement индивидуально.

Чтоб добиться эффективной отрисовки множества динамических объектов, принадлежащих к одной текстуре, было решено выбрать SpriteBatch. Данный класс позволит рисовать множество различных тайлов с минимальным количеством вызовов отрисовки.

Для удобства использования данного функционала была создана система и несколько компонентов.

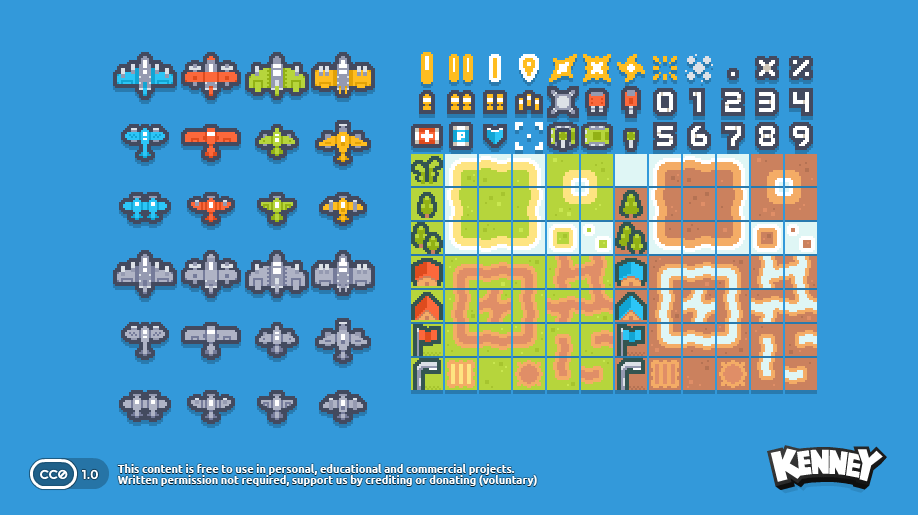
# ЗАКЛЮЧЕНИЕ

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Уэйд М. Доигрались! Как поколение геймеров навсегда меняет бизнес-среду / Уэйд Митчел ­– Москва: Претекст, 2008. – 252 c.
2. Фримен Э. Head First. Паттерны проектирования. Обновленное юбилейное издание / Фримен Эрик, Робсон Элизабет – СПб: Питер, 2018. – 656 c.
3. Каманкина М. В. Видеоигры: общая проблематика, страницы истории, опыт интерпретации / Каманкина М. В. — Москва: ГИИ, 2016. – 338 c.
4. Jason G.. Game Engine Architecture / Jason Gregory — NW: CRC Press, 2018. – 830 c.
5. Haggard G., Discrete Mathematics for Computer Science / Haggard Gary [и д.р.] ­– США: Cengage Learning. 2005. ­– 624 c.
6. Shaker N. Procedural Content Generation in Games / Noor Shaker [и д.р.] – Режим доступа: <http://pcgbook.com/> (Дата обращения: 25.04.2024 г.)
7. Онлайн-сервис для размещения, продажи и загрузки инди-игр Itch.io «Бесплатные ассеты»– Режим доступа: <https://itch.io/> (Дата обращения: 25.04.2024 г.)
8. Официальный сайт Aseprite «Документация Aseprite» – Режим доступа: <https://www.aseprite.org/docs/> (Дата обращения: 25.04.2024 г.)
9. Официальный сайт Godot «Особенности» – Режим доступа: <https://godotengine.org/> (Дата обращения: 25.04.2024 г.)
10. Официальный сайт Haxe «Документация Haxe» – Режим доступа: <https://haxe.org/manual/introduction.html> (Дата обращения: 25.04.2024 г.)
11. Официальный сайт Heaps.io «Документация Heaps.io» – Режим доступа: <https://heaps.io/documentation/home.html> (Дата обращения: 25.04.2024 г.)
12. Официальный сайт Unity «О продукте» – Режим доступа: <https://unity.com/> (Дата обращения: 25.04.2024 г.)
13. Официальный сайт Unreal Engine «Что такое Unreal Engine» – Режим доступа: <https://www.unrealengine.com/> (Дата обращения: 25.04.2024 г.)
14. Ресурс для игровых разработчиков GamesIsArt «О жанрах видеоигр» – Режим доступа: <http://gamesisart.ru/> (Дата обращения: 25.04.2024 г.)
15. Ресурс для игровых разработчиков GameDesigning «The Top 10 Video Game Engines» – Режим доступа: <https://www.gamedesigning.org/career/video-game-engines/> (Дата обращения: 25.04.2024 г.)
16. Algorithms for Procedural Content Generation – Режим доступа: <http://pcg.wikidot.com/category-pcg-algorithms> (Дата обращения: 25.04.2024 г.)
17. FinancesOnline «20 Best Graphic Design Software Solutions of 2020» – Режим доступа: <https://financesonline.com/top-20-graphic-design-software-solutions/> (Дата обращения: 25.04.2024 г.)
18. Matt Barton, The History of Computer Role-Playing Games Part 1: The Early Years (1980-1983) – Режим доступа: <https://www.gamasutra.com/view/feature/132024/the_history_of_computer_.php> (Дата обращения: 25.04.2024 г.)
19. Musician on a mission «The best free audio editor 2020» – Режим доступа: <https://www.musicianonamission.com/best-free-audio-editor/> (Дата обращения: 25.04.2024 г.)

# ПРИЛОЖЕНИЯ

**ПРИЛОЖЕНИЕ А**



**ПРИЛОЖЕНИЕ Б**

