|  |
| --- |
| Пермский филиал федерального государственного автономного образовательного учреждения высшего образования  «Национальный исследовательский университет  «Высшая школа экономики»  *Факультет социально-экономических и компьютерных наук* |
| Берсенёв Илья Иванович  Пичеев Андрей Викторович  **ПРОТОТИП КЛОНА ИГРЫ “DOOM”**  *Курсовой проект*  студентов образовательной программы «Программная инженерия» по направлению подготовки *09.03.04 Программная инженерия*  Руководитель,  приглашенный преподаватель  М.Д. Чистогов |

Пермь, 2022 год

**Оглавление**

[1 Введение 3](#_Toc107081790)

[2 Анализ 4](#_Toc107081791)

[2.1 Постановка задачи 4](#_Toc107081792)

[2.2 Изучение игры Doom 4](#_Toc107081793)

[2.3 Изучение знаний о создании 3D приложений 6](#_Toc107081794)

[2.4 Формирование требований для проекта 7](#_Toc107081795)

[2.5 Необходимые средства и условия 8](#_Toc107081796)

[3 Проектирование 9](#_Toc107081797)

[3.1 Общая информация 9](#_Toc107081798)

[3.2 Класс Bullet 10](#_Toc107081799)

[3.3 Класс Enemy1 11](#_Toc107081800)

[3.4 Класс Enemy2 12](#_Toc107081801)

[3.5 Класс Hero 13](#_Toc107081802)

[3.6 Интерфейс Steper 16](#_Toc107081803)

[4 Результаты 17](#_Toc107081804)

[Список литературы 18](#_Toc107081805)

# Введение

Для эффективной и продуктивной работы мы решили поделить наши зоны ответственности: Берсенёв Илья в большей степени отвечал за архитектуру и разработку, а Пичеев Андрей в большей степени за документацию.

Для проектирования Илья изучил документацию языка C#, различные руководства, сайты с ресурсами для разработки, такие как kenney.nl и opengameart.org.

Для создания документации Андрей изучил ГОСТ 19 ЕСПД, правила написания и оформления курсовых работ, проводил анализ требований и функций для создания Проектной заявки и Технического задания, изучал информацию для создания Руководства оператора и Программы и методики испытаний, Текста программы.

# Анализ

**2.1 Постановка задачи**

В современном мире очень популярны компьютерные игры. Но для многих новые игры уже не вызывают такого интереса, как раньше. Люди в нынешнее время всё чаще ищут решения в прошлом, в том числе и компьютерные игры.

Вместе с тем, изучая данный вопрос, наше внимание привлекла известная и очень популярная игра «Doom», а именно оригинальная версия 1993 года.

Во-первых, эта игра имеет очень много поклонников, и сама по себе очень уникальная и интересна. А во-вторых, нас привлекло то, что игра была выпущена почти 30 лет назад, и нам стало интересно, как перенести её на современные средства разработки, доступные в 2022 году.

Исходя из этого, нами было принято решение о выборе в качестве темы Курсового проекта разработку прототипа клона игры «Doom» в версии 1993 года.

Мы поставили следующие задачи:

1. изучить область знаний об игре Doom;
2. проанализировать область знаний о создании приложений с использованием 3D графики;
3. сформировать конкретные требование в создаваемому продукту;
4. спроектировать приложение;
5. протестировать приложение;
6. подготовить документацию по проекту.

**2.2 Изучение игры Doom**

Говоря об игре «Doom», можно процитировать Википедию: «Doom - компьютерная игра в жанре шутера от первого лица, разработанная и выпущенная компанией Id Software в 1993 году. Doom является одной из самых значительных и влиятельных компьютерных игр в истории индустрии; в частности, её популярность во многом определила дальнейшее развитие и распространение жанра шутеров от первого лица».

Из этого понятно, что эта игра действительно очень сильно повлияла на дальнейшие развитие индустрии.

Мы решили агрегировать нужные нам для работы над проектом характеристики и свойства «Doom»:



***Рисунок 1 – Скриншот из игры “Doom” 1993 года***

Во-первых, оружие. В игре пользователь стреляет из разных видов оружия, таких как пистолет, пулемёт, дробовик и так далее, которые обладают разной эффективностью и разными характеристиками.

Во-вторых, предметы и артефакты. В игре есть несколько типов предметов, таких как патроны, аптечки, броня и прочее, и артефакты, такие как очки ночного виденья, неуязвимость и так далее, которые помогают игроку выживать и побеждать монстров.

В-третьих, о монстрах. В игре присутствуют несколько типов монстров: зомби, собственно, непосредственно монстры, каждый из которых включает в себя несколько типов противников. Кроме того, в игре есть боссы, которые отличаются большим количеством здоровья, сильной защитой и атакой.

В-четвертых, уровни. В игре было несколько эпизодов, в каждом из которых было в среднем десять уровней, каждый из которых обладал своим оформлением и сюжетом.

Для понимания сути интересно узнать, а как и в каких условиях создавали “Doom” разработчики оригинальной игры? Оказывается, изначально задумка была о бесшовном мире, но от этой идеи отказались, так как на тот момент она была слишком сложной в реализации. Был сделан выбор в пользу отказа от реализма в угоду динамике.

Для получения текстур противников дизайнеры создавали модели монстров из глины и подручных материалов, а потом их снимали на видео и фотографировали с разных сторон, после оцифровывали, цветность снижали до стандартных на то время 256 бит.

Для оружия также использовались реальные, игрушечные и самодельные образцы, а анимация удара была записана одним из членов команды разработчиков.

Для “Doom” был создан абсолютно новый движок “Doom Engine”, который сильно превосходил предыдущий движок, который использовался для разработки Wolfenstein 3D.

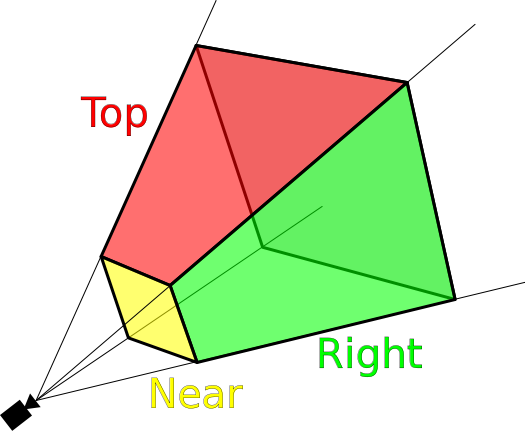
Из-за открытого исходного кода игры было разработано огромное множество модификаций игры.

**2.3 Изучение знаний о создании 3D приложений**

Исходя из выбранной темы, нам необходимо было изучить информацию о создании и разработке 3D игр. Начался поиск нужной нам информации.

Стало понятно, что сначала обязательно нужен 3D движок. Что это включает?

Положение игрока, трассировку лучей, отрисовку окружающего пространства, обзор:



***Рисунок 2 – Обзор игрока***

Выделим основные действия, которые нужно сделать конкретнее по этапам по порядку:

1. Необходима карта нашего мира;
2. Далее нужно создать координаты для местоположения игрока;
3. Нужен угол обзора игрока;
4. После формируем функцию обзора;
5. Далее создаём луч для подсчёта расстояния до препятствия методом дальномера;
6. Формируем сектор обзора с помощью этих лучей;
7. Формируем вертикальны обзор на основании расстояния до препятствия;
8. Исправляем эффект “рыбьего глаза”;
9. Загружаем текстуры для препятствий;
10. Определяем текстуры на сами препятствия;
11. Определяем места для монстров;
12. Загружаем текстуры для монстров;
13. Определяем размер монстров по горизонтали и по вертикали;
14. Загружаем текстуры для монстров;
15. Добавляем действия на клавиши и кнопки мыши.

**2.4 Формирование требований для проекта**

Исходя из реальной ситуации и практической оценки наших потенциальных возможностей, мы решили сформировать список таких требований, которые реализуемы и достижимы, не нарушая суть проекта.

В итоге мы приняли к работе следующие требования и условия:

* прототип должен иметь один уровень;
* в игре должно быть два типа противников;
* каждый противник имеет здоровье на одно попадание;
* у пользователя один изначальный тип оружия;
* у игрока бесконечный боекомплект;
* должна иметься возможность осматриваться;
* должна присутствовать возможность двигаться по территории;
* противники стреляют по игроку, но убить не могут;

**2.5 Необходимые средства и условия**

Исходя из полученной информации, нам надо было составить предварительный список необходимых элементов, которые должны быть к началу проектирования и разработки проекта.

Список необходимого:

* текстуры противников;
* текстуры стен и препятствий;
* текстура оружия для игрока;
* текстура пули;

Также для работы необходим следующий набор программных средств и компонентов:

* система управления версиями Git;
* Microsoft Visual Studio 2019;
* установленный .Net Framework 4.8

# Проектирование

**3.1 Общая информация**

Переходя к проектированию, началась работа по плану. Для разработки сразу было начато использование Microsoft Visual Studio с установленным .Net Framework 4.8.

В Visual Studio для героя, противников и пули был создан для каждого свой класс.

Для пули было решено сделать единственную функцию, в которой будут происходить все события, такие как передвижение и уничтожение.

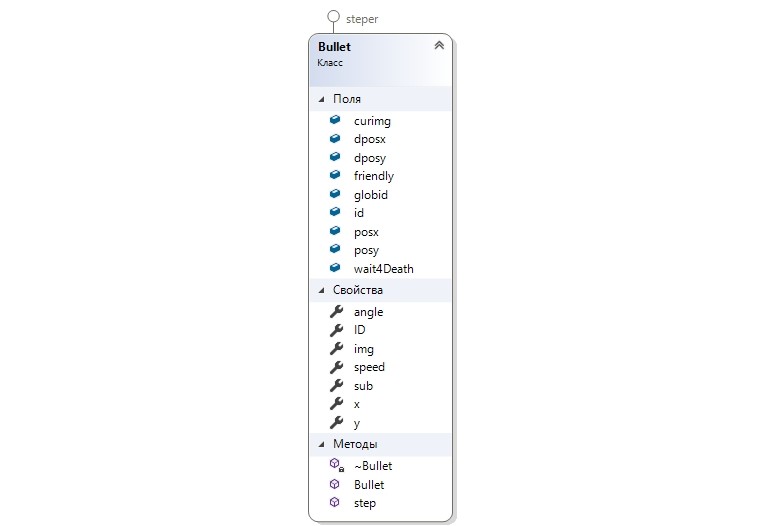
Для первого типа противника было решено создать такого, который стреляет с близкого расстояния и стремится сократить дистанцию. Все действия было решено оставить в функции, используя машину состояний, получающую на вход расстояние до игрока.

Для второго типа противника было решено создать такого, который стреляет с дальнего расстояния и стремится разорвать увеличить. Все действия было решено также оставить в функции, используя машину состояний, получающую на вход расстояние до игрока.

Для класса героя было принято решение добавить в него обработку нажатий клавиш. Для реализации передвижения добавлена отдельная функция. Ещё одна функция - для события рендеринга.

Разберём все классы подробнее.

**3.2 Класс Bullet**

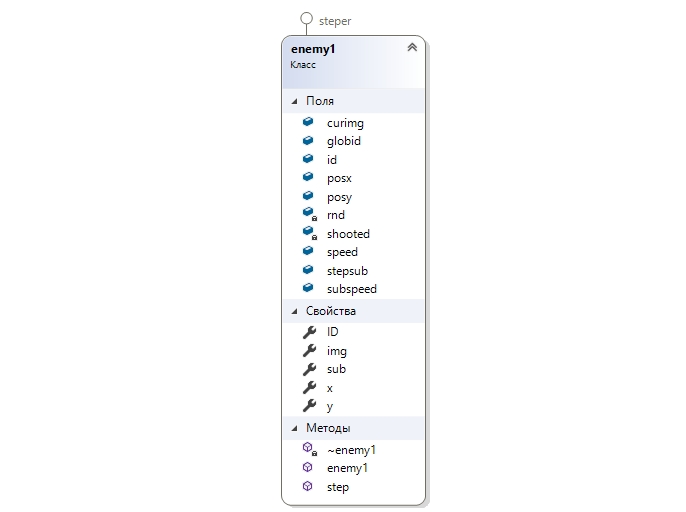
Рассмотрим класс Bullet:

***Рисунок 3 – Класс Bullet***

Класс Bullet имеет единственную значимую функцию под названием Step, в которой происходит передвижение по карте. В случае, если спрайт переключается на спрайт взрыва, экземпляр останавливается, а через 5 секунд уничтожается.

**3.3 Класс Enemy1**

Рассмотрим класс Enemy1:

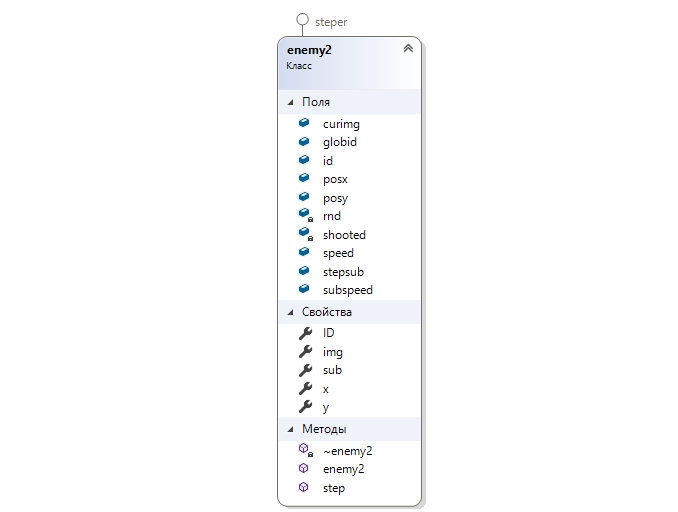


***Рисунок 4 – Класс Enemy1***

Класс Enemy1 обладает единственной важной функцией под названием Step. Функция Step – своего рода машина состояний, где на вход подаётся расстояние до игрока. В случае, если расстояние до игрока среднее и на пути к игроку не стоит объект, экземпляр начинает движение к игроку и меняет свой спрайт на спрайт движения. В случае, если расстояние до игрока низкое, экземпляр меняет спрайт на спрайт атаки и раз в несколько секунд создаёт экземпляр класса Bullet, движущийся в сторону игрока. В случае, если расстояние до игрока слишком велико, объект бездействует, спрайт заменяется на спрайт стрельбы, частота смены спрайтов снижается до нуля.

**3.4 Класс Enemy2**

Рассмотрим класс Enemy2:

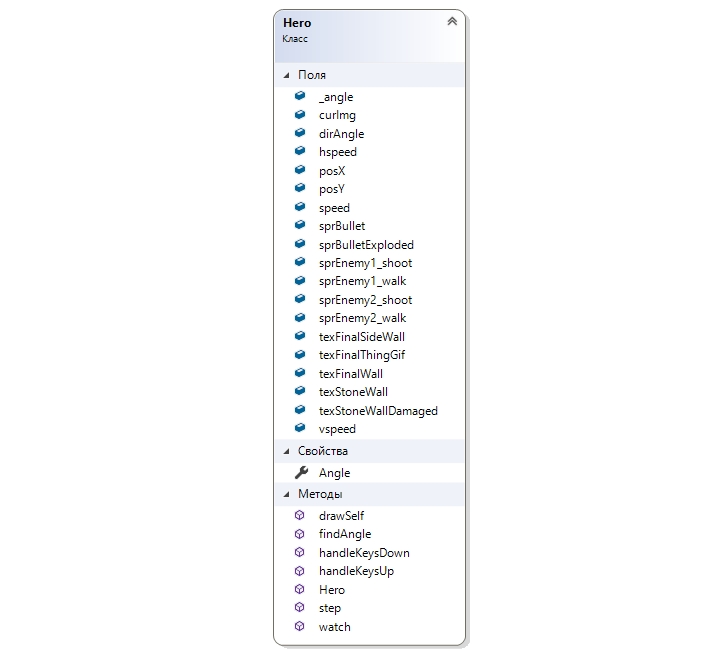


***Рисунок 5 – Класс Enemy2***

Класс Enemy2 также обладает единственной важной функцией под названием Step. Step – своего рода машина состояний, где на вход подаётся расстояние до игрока. В случае, если расстояние до игрока среднее или низкое и на пути к игроку не стоит объект, экземпляр начинает движение в сторону, противоположную игроку и меняет свой спрайт на спрайт движения. В случае, если расстояние до игрока высокое, экземпляр меняет спрайт на спрайт атаки и раз в несколько секунд создаёт экземпляр класса Bullet, движущийся в сторону игрока.

**3.5 Класс Hero**

Рассмотрим класс Hero:



***Рисунок 6 – Класс Hero***

Hero – класс, в экземпляре которого проводится рендеринг от лица игрока, это также класс, отвечающий за игрока. За рендеринг отвечает функция Watch, получающая на вход объект Graphics. В начале функции инициализируется вложенная функция getCollision, о которой речь пойдёт дальше, а также 2 объекта Graphics: на первом отрисовываются стены – псевдо-3д объекты, а на втором – плоские объекты: враги и пули.

Далее идёт цикл от нуля до 720, т.к. окно программы – 720 пикселей в ширину, и соответственно в каждой итерации цикла отрисовывается соответствующая вертикальная полоса на экране. В этом цикле с помощью функции getCollision определяется расстояние до объекта и его ID.

Функция getCollision реализована следующим образом: инициализируется массив целых чисел с координатами всех врагов и пуль, которые находятся в массиве stepers. Далее берётся переданный угол, который вычисляется по формуле:

*(1)*

где X – необходимый угол,

X0 – угол игрока,

C – переменная цикла.

Таким образом, у нас получается полное поле зрения в 90 градусов, так как

; . После того, как определён угол, запускается цикл от 0 до 12 800 с шагом один.

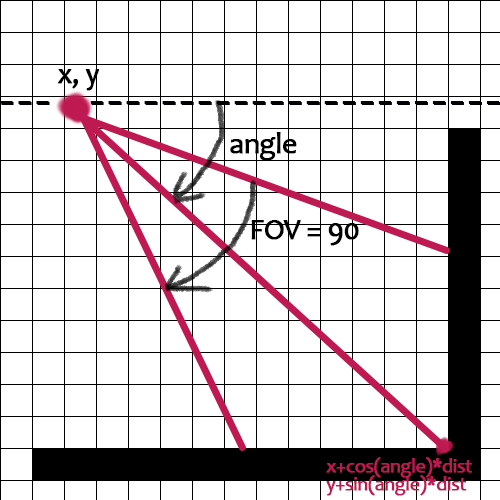
В каждом шаге идёт проверка, вычисляется координата. X-координата определяется суммой X0 - координатой персонажа и косинуса текущего угла, умноженного на переменную текущего цикла t:

(2)

Y-координата определяется аналогично, но с помощью синуса:

(3)

Проверяется: присутствует ли на текущей координате враг, пуля либо стена:



***Рисунок 7 – Проверка на препятствие***

В случае, если обнаружен враг, возвращаются его координаты и идентификатор, который является целым числом в интервале от 100 до 150 и от 200 до 250 включительно.

В случае, если обнаружена стена, возвращаются координаты и число от 1 до 10 включительно, так как у стен нет идентификатора. После определения координат стен или врага рассчитывается длина до объекта и на базе длины рассчитывается высота той области, которая будет выведена на экран, высота области обратно пропорциональна расстоянию до объекта.

В случае, если была определена стена, рассчитывается Х-координата текстуры, которая будет выведена на экран.

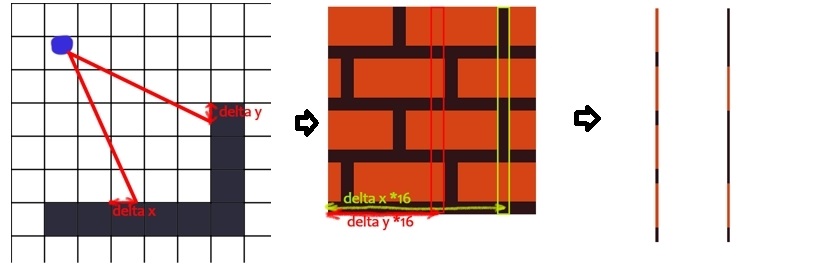
Она определяется следующей формулой:

*(4)*

где X – координата по X,

Y – координата по Y.

В данной формуле определяется максимум из разницы между координатой стены по Х и координатой по Х из функции getCollision и разницы между координатой стены по Y и координатой из функции getCollision по Y, затем эта разница (целое число от 0 до 1) умножается на 16, что является шириной текстуры стены:



***Рисунок 8 – Формирование текстуры***

Деление на 64 происходит из-за того, что стены расставлены по сетке в 64, то есть в игре одна стена занимает 64 квадратные единицы

В случае, если был обнаружен враг или пуля, его ID и расстояние до него заносятся в массив длинной в 720 элементов. Далее идёт отрисовка врагов и пуль в порядке убывания расстояния до врага или пули. Враг или пуля отрисовываются в цикле, каждая вертикальная полоса рисуется только если ничего её не перекрывает.

В классе Hero есть и другие, не столь значительные функции. Функции HandleKeysDown и HandleKeysUр контролируют нажатие клавиш W, A, S, D, Escape. Нажатие клавиш движения задаёт игроку направление.

В функции Step происходит передвижение игрока, координата X изменяется на косинус направления r, умноженный на скорость sp:

(5)

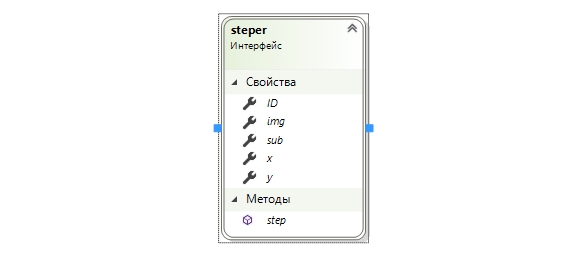
Координата Y изменяется аналогично, но с синусом направления:

(6)

Функция FindAngle – простейшая функция, получающая на вход 2 пары координат и возвращающая угол между ними.

**3.6 Интерфейс Steper**

Рассмотрим интерфейс Steper:



***Рисунок 9 –Интерфейс Steper***

Интерфейс steper необходим, чтобы обеспечить полиморфизм для объектов, которые будут отрисовываться на 2 уровне, то есть для всего кроме стен.

# Результаты

По итогу выполнения проекта были получены навыки создания программных продуктов, включая само приложение и документацию к нему.

Говоря по отдельности, Илья в большей степени изучил основы 3D-графики, принцип работы рей-кастеров в различных играх и разработал большую часть проекта

Андрей в большей степени изучил требования и правила для оформления документации, включая ГОСТы и другие нормативные документы, которые были использованы в Техническом задании и других документах.

# Список литературы

1. Введение в программирование: простой 3D-шутер с нуля– [Электронный ресурс]. – URL: https://habr.com/ru/post/439698/
2. Сайт с элементами для разработки– [Электронный ресурс]. – URL: https://www.kenney.nl/
3. Сайт с текстурами для разработки– [Электронный ресурс]. – URL: https://opengameart.org/
4. Информация об игре «Doom» – [Электронный ресурс]. – URL: https://ru.wikipedia.org/wiki/Doom\_(игра,\_1993)
5. Официальная документация C#– [Электронный ресурс]. – URL: https://docs.microsoft.com/ru-ru/dotnet/csharp/