Министерство Образования и Исследований Республики Молдова

Технический Университет Молдовы

Факультет Вычислительной Техники, Информатики и Микроэлектроники

Департамент Программной Инженерии и Автоматики

**КУРСОВАЯ РАБОТА**

по предмету

методы и механизмы проектирования программного обеспечения

на тему:

**Реализация системы экипирования в онлайн шутере**

**Implementation of the equipment system in an online shooter**

**Implementarea sistemului de echipamente într-un shooter online**

Студент: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Коваль Роман

подпись

Проверил: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ унив. ассистент Булдумак Олег

подпись

Кишинэy, 2022

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc104769927)

[1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 4](#_Toc104769928)

[1.1 Значимость темы 4](#_Toc104769929)

[1.2 Схожие системы и их специфика 5](#_Toc104769930)

[1.3 Цели создания системы 7](#_Toc104769931)

[1.4 Требования к системе и ее спецификация 8](#_Toc104769932)

[1.2 МОДЕЛИРОВАНИЕ СИСТЕМЫ 9](#_Toc104769933)

[2.1 Иерархия файлов 9](#_Toc104769934)

# ВВЕДЕНИЕ

Шу́тер (от англ. shooter) — жанр компьютерных игр. На момент зарождения жанра за рубежом укрепилось слово «шутер», как вариант описания игрового процесса и перевод для слова shooter.

Обычно, в играх такого жанра игрок находится в трёхмерном пространстве и имеет некоторую свободу передвижения. Уровни, как правило, являют собой ограниченный лабиринт, в котором расположены враги, союзники и нейтрально настроенные объекты. Действия большинства шутеров разворачивается в анизотропном пространстве (помещения имеют очевидные пол и потолок, в них действует гравитация), хотя существуют и исключения, в которых пространство изотропно.

Для данной работы, в качестве основы взяты принципы многопользовательского шутера, для которого применимы все ранее перечисленные особенности, но он существенно отличается по своей сути и принципам игровых действий.

Массовый многопользовательский шутер от первого лица (от англ. Massively multiplayer online first-person shooter, или MMOFPS) — жанр компьютерных игр, сочетающий в себе особенности шутеров от первого лица и массовых многопользовательских онлайн-игр, в том числе и в формате браузерной игры. В этих играх большое количество пользователей взаимодействует друг с другом внутри виртуального мира. Можно сказать, что MMOFPS — это онлайн-игра в реальном времени, происходящая внутри крупной игровой области, с большим количеством одновременно участвующих игроков в виде шутера от первого лица. В этих играх представлены крупномасштабные, иногда командные сражения.

Ввиду быстротечного, стратегического характера игр в этом жанре, игроки должны рассчитывать в первую очередь на свои знания и физическую координацию, что придает большее значение индивидуальному мастерству игроков, нежели их обобщенным характеристикам, поскольку никакое количество игровых бонусов или подобного не способно восполнить отсутствие у игрока умения прицеливаться или тактического мышления.

Во многих MMOFPS функционируют экономические системы. Игровая валюта и предметы обычно зарабатываются в процессе игры и имеют для игроков определенную ценность. Подобные виртуальные экономики могут быть исследованы (если осуществляется журналирование игровых данных), что имеет значение в экономических исследованиях. Более значительным является то, что эти «виртуальные» экономики могут оказывать влияние на экономики реального мира.

# 1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

Выбор темы должен опираться на разные аспекты, которые связаны с этой предметной областью. Такими аспектами выступают: значимость темы, описание и наличие плюсов этой реализации перед схожими системами, цели сотворения системы и требования к системе. Любой из этих качеств должен быть пристально рассмотрен перед созданием системы, так как процесс изучения перечисленного выше обязательно скажется на качестве выполнения данной работы.

Так, например, для разработки компьютерной игры в нынешних реалиях в команду могут одновременно входить: продюсер, издатель, геймдизайнер, художник, программист, звукорежиссер, тестировщик.

## Значимость темы

В начале 1980-х годов после появления первых домашних компьютеров и игровых приставок один программист мог управлять почти всеми задачами, связанными с разработкой игры. Разработка современных игр предполагает наличие широкого круга навыков и персонала поддержки. Для работы над одним проектом требуются целые команды, в состав которых обычно входят представители ряда специализаций.

Разработкой компьютерных игр может заниматься как один человек, так и фирма (коллектив разработчиков). Коммерческие игры создаются командами разработчиков, нанятыми одной фирмой. Фирмы могут специализироваться на производстве игр для персональных компьютеров, игровых приставок или планшетных компьютеров. Разработка может финансироваться другой, более крупной фирмой — издателем. Фирма-издатель по окончании разработки занимается распространением игры и берёт на себя связанные с этим затраты. Противоположным подходом является такая разработка, когда фирма самостоятельно (без участия издателей) распространяет копии игр, например, средствами цифровой дистрибуции.

Разработка наиболее крупнобюджетных игр может стоить десятки миллионов долларов США, причём в последние 2 десятилетия эти бюджеты непрерывно росли, как и численность команд разработчиков и сроки разработки. Так, в конце девяностых игру для консоли PlayStation для конечного покупателя — могла сделать команда из 10 человек за год, для PlayStation 2 (первая половина 2000-х годов) необходима была команда из 30-50 человек и два года разработки, к 2012 году речь шла уже о командах из свыше чем 100 разработчиков и срок порядка трёх лет. По утверждению Алекса Мура, геймдизайнера из компании Sumo Digital, если бы цена игры для конечного потребителя росла в той же пропорции, игры в 2012 году стоили бы по 1800 долларов США; иными словами, чтобы окупить возросшие бюджеты при сохранении тех же цен в магазинах, компании-издатели должны продавать намного больше копий игр.

Крупнобюджетная игра для двух платформ — Xbox 360 и PlayStation 3 — обходилась в 2012 году в среднем в 20 миллионов долларов, и для того, чтобы она окупилась, нужно было продать около двух миллионов копий.

Таким образом, разработка разнообразных механизмов, реализующих различную логику и поведение в играх может стать отличным опытом для любого разработчика и несомненно улучшить его умения моделирования, логистики и технические навыки. В будущем эти навыки, возможно, помогут ему стать частью большой команды разработчиков компьютерных, или мобильных игр, а может быть и даст толчок к чему-то самостоятельному, что заимеет огромный успех на просторах игрового пространства.

## 1.2 Схожие системы и их специфика

В настоящее время, практически в каждом онлайн шутере присутствует, своего рода, система экипирования. Эта система предоставляет пользователю выбор экипировки, необходимой его персонажу по мере игры. Как правило, в соревновательных шутерах, такая система представляет собой некий “магазин”, в котором игрок может получить необходимую экипировку заплатив за это внутриигровую валюту, полученную в виде регулярных начислений в течении матча, а также в качестве бонусов за хорошие игровые показатели или результаты его команды.

В качестве примера реализации данной системы можно привести популярные в нынешнее время онлайн шутеры.

В первом шутере, на рисунке 1.1, система экипирования представляет собой интерактивное меню, вызываемое пользователем по нажатию клавиши B на клавиатуре. Далее в меню пользователь может использовать для навигации как мышь, так и клавиатуру. Кроме того, в самом меню есть множество подсказок и дополнительных функций, обеспечивающих улучшение удобства пользования для игрока. Интерфейс такого меню интуитивно понятен и не вызывает трудностей в пользовании, однако концепция игра не предусматривает продажу оружия. Так, если пользователь необдуманно выбрал какую-либо экипировку и приобрел её, вернуть валюту за неё будет невозможно и раунд придется играть с ней.

Рисунок 1.1 – Интерактивное меню экипирования в Сounter Strike

Во втором случае, представлена система экипирования которая реализована в шутере под названием Valorant, интерфейс которой представлен на рисунке 1.2.

Главным отличием этой системы от предыдущей является способность возвращать экипировку в отведенное количество времени после её покупки, а также более узкий выбор экипирования. Однако здесь это нельзя назвать недостатком, так как концепция данной игры предполагает, что пользователь так же может приобрести не только экипировку, но и специальные заряды для необходимых в процессе игры умений, которые уникальны для каждого персонажа.

Меню здесь имеет совершенно другой вид, но дизайн и конструкция так же интуитивно понятны и не вызывают никаких трудностей в пользовании даже для новых игроков.

Возможно, её преимуществом можно назвать то, что здесь весь арсенал представлен пользователю сразу и нет необходимости преодолевать несколько вкладок меню, для того чтобы добраться до необходимого предмета экипировки. Однако и здесь не все так просто – вероятнее всего такой подход допускается ввиду не очень большого количества экипировки, в отличии от первого варианта, где предметов экипировки для персонажа куда больше.

Рисунок 1.2 – Интерактивное меню экипирования в Valorant

## 1.3 Цели создания системы

Главной целью данной работы является улучшение навыков в области разработки различных логических систем для разнообразных целей. В данном случае – подобие движка для системы экипирования в игре. Выполнив поставленные задачи, разработчик оттачивает существующие навыки и приобретает новые умения в области моделирования и построения логистических систем для взаимодействия человека с системой. Кроме того, задачи включают в себя создание чего-то нового, без сомнений, похожего на существующие и зарекомендовавшие себя системы, но с различными обновлениями и изменениями в угоду личных предпочтений разработчика, на основе его опыта взаимодействия с данными системами.

## 1.4 Требования к системе и ее спецификация

Система не должна целиком повторять существующие аналоги и представлять пользователю новые, расширенные функции. Системы должна быть понятна клиенту и в отношении пользователя не нести никакой нагрузки. То есть, пользователь должен как можно меньше взаимодействовать с логикой системы, а лишь использовать интуитивно понятные, настроенные операции.

Система должна быть открыта к расширению настолько, насколько это возможно. То есть, подразумевается использование множества модулей, расширение которых, ни в коем случае не нарушит работу и целостность существующих связей в данной системе.

Система должна реализовывать поставленные логистические задачи, которые должны соответствовать тем задачам, которые реализуют системы такого рода в контексте компьютерных игр.

Система должна придерживаться основных принципов программирования и принципов распределения ответственности и соблюдать четкий, или наиболее определенный баланс между функционалом, оформлением, читабельностью, доступностью и другими аспектами проектирования систем.

# МОДЕЛИРОВАНИЕ СИСТЕМЫ

Для упрощения рабочего процесса и улучшения общего понимания концепции системы необходимо проработать модель системы и взаимодействие её компонентов, начиная с аппаратных средств и заканчивая программными модулями. Далее рассматривается некоторые базовые принципы построения данного проекта и очень обобщенные взаимодействия модулей в контексте программных средств, используемых в разработке.

## 2.1 Иерархия файлов

Проект представляет множество TypeScript модулей, взаимодействующих между собой. Средой для разработки проекта будет Visual Studio Code, а платформой для взаимодействия наших компонентов – Node JS, основанная на движке V8, превращающая JavaScript из узкоспециализированного языка в язык общего назначения. Node.js добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через свой API, написанный на C++, подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к ним из JavaScript-кода. Node.js применяется преимущественно на сервере, выполняя роль веб-сервера, но есть возможность разрабатывать на Node.js и десктопные оконные приложения и даже программировать микроконтроллеры.

Так иерархия компонентов в проекте выглядит так, как показано на рисунке 2.1.

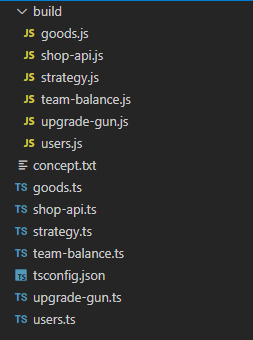


Рисунок 2.1 – Иерархия файлов в проекте.

Здесь, каждый из файлов выполняет свою конкретную роль в целой системе, а в директории build находятся дистрибутивы, компилируемые компилятором из TypeScript в JavaScript, который в последствии преобразует исходный JavaScript код в собственный машинный.