Санкт-Петербургский государственный университет

Кафедра системного программирования

Группа 21.Б15-мм

Сетевой симулятор для обучения в форме игры

Бадреева Полина Денисовна

Отчёт по производственной практике
в форме «Решение»

Научный руководитель:
ст. преподаватель кафедры системного программирования Зеленчук И. В.

Санкт-Петербург
2024

Оглавление

Введение

Специалисты в области сетевой инженерии играют ключевую роль в проектировании, установке, настройке и обслуживании компьютерных сетей. Они ответственны за настройку сетевого оборудования, обеспечение безопасности сети и решение различных проблем, связанных с сетевой инфраструктурой. Без их участия было бы невозможно обеспечить эффективное взаимодействие компьютеров и передачу данных. С увеличением важности сетевой инженерии в современном мире возрастает потребность в качественном обучении, что требует значительных временных и ресурсных затрат.

”Сетевой симулятор для обучения в форме игры”, представляет собой новый подход к обучению сетевой инженерии. Этот симулятор ,будет позволять специалистам обучаться, создавая и настраивая свои уникальные топологии, не прибегая к использованию физического оборудования.

В перспективе проекта необходимо доработать симулятор с целью расширения его функциональности. Планируется внедрение новых элементов и возможностей, обогащающих обучающий опыт пользователей. Улучшение текстур и интерфейса также станет важным направлением развития, обеспечивая более интуитивное и удобное взаимодействие с приложением. Эти изменения направлены на повышение эффективности обучения и создание более полноценного инструмента для подготовки к сетевой инженерии.

1. Постановка задачи

Целью является усовершенствование сетевого симулятора, предназначенного для обучения в формате игры. Для достижения данной цели были поставлены следующие задачи:

1. Реализация свитча и роутера, с целью обогащения функциональности симулятора.

2. Добавление разнообразных сетевых устройств различных моделей, для расширения возможностей обучения.

3. Улучшение визуального опыта пользователей путем совершенствования текстур в направлении более реалистичного воспроизведения среды.

2. Обзор

2.1. Игра

Архитектура проекта представляет собой комплексную систему, состоящую из различных компонент, каждая из которых выполняет определенные функции и взаимодействует с другими элементами для обеспечения работы всей системы в целом.

Архитектура включает в себя следующие ключевые элементы:

2.1.1. Объекты в игре

Объекты — это устройства и мебель, которые пользователь может ставить и с которыми он может взаимодействовать.

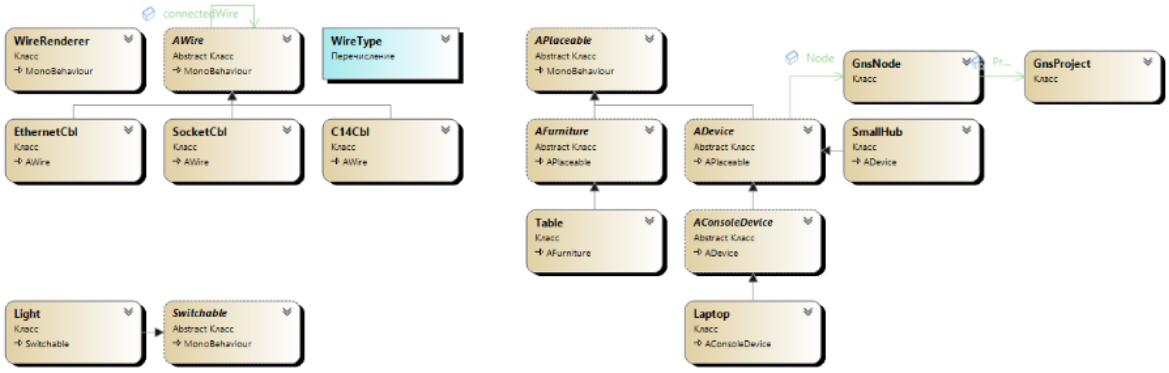


Рис. 1: Alekseev-report[1]

1. **Мебель (AFurniture)**: Абстрактный класс, от которого наследуются объекты, представляющие собой различные предметы мебели в игре. Эти объекты не являются сетевыми интерфейсами и могут использоваться для размещения других объектов.

2. **Устройства (ADevice)**: Абстрактный класс, от которого наследуются объекты, которые представляют собой сетевые интерфейсы. Эти устройства можно создавать и подключать к другим устройствам для обмена информацией.

3. **Консольные устройства (AConsoleDevice)**: Абстрактный класс, наследуемый от ADevice, который предоставляет метод для открытия терминала. Создано для устройств содержащих терминал.

На начальном этапе разработки игры были реализованы следующие объекты:

1. **Ноутбук (Laptop)**: Моделируется с помощью VPCS и представляет собой электронное устройство в игре.
2. **Мини-хаб (MiniHub)**: Эмулируется с использованием Hub и служит для соединения различных устройств в сеть.
3. **Стол (Table)**: Представляет собой объект мебели, на который можно размещать другие объекты.

Кроме того, в системе присутствуют классы для работы с кабелями:

1. **EthernetCbl**: Реализация кабеля для подключения устройства к другому устройству по сети Ethernet.
2. **C14Cbl**: Реализация кабеля для подключения устройства к розетке стандарта C14.
3. **SocketCbl**: Реализация кабеля для подключения устройства к обычной розетке.

А также:

1. **ISwitchable**: Интерфейс, который содержит методы для включения и выключения светового индикатора на объекте.
2. **Light**: Класс, наследуемый от ISwitchable, представляющий собой световой индикатор в игре.

Эти элементы архитектуры обеспечивают основу для дальнейшего расширения функциональности и взаимодействия объектов в игровой среде.

2.1.2. API GNS3

Часть проекта, созданная для взаимодействия с GNS3.

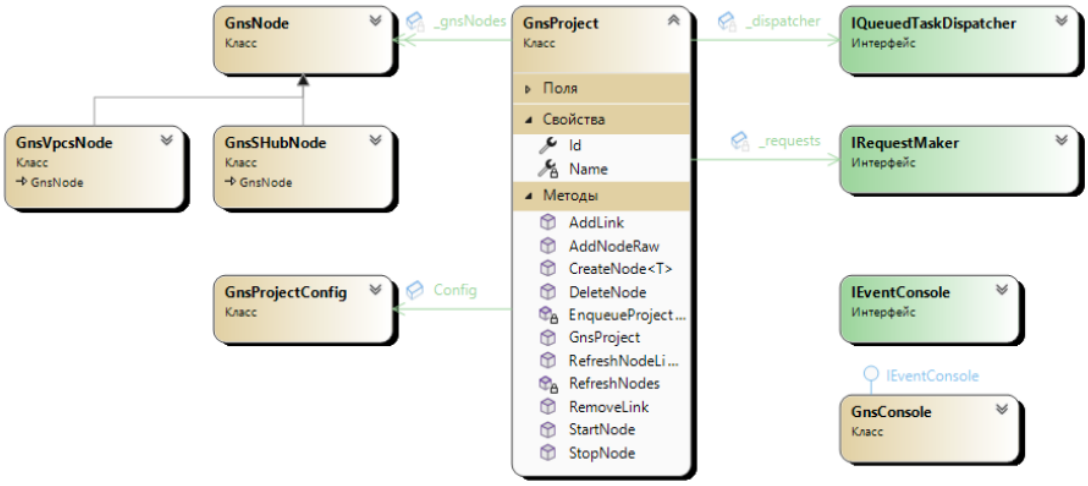


Рис. 2: Alekseev-report[1]

GNSProject: класс, который управляет проектом GNS3. Все запросы к проекту проходят через этот класс.

GnsNode: класс, который обеспечивает взаимодействие с виртуальными устройствами в GNS3. Каждое виртуальное устройство хранит экземпляр этого класса.

Существуют два подкласса GnsNode, каждый из которых хранит в себе объект GnsJ**Name**Node, который используется для сохранения конфигураций виртуальных устройств в формате JSON:

1. **GnsVpcsNode**
2. **GnsSHubNode**

IEventConsole: Интерфейс, определяющий набор методов, служащих основой для функциональности консоли GnsConsole.

GnsConsole: Класс, который предоставляет функциональность взаимодействия с консолью устройства из игрового терминала.

2.1.3. Сохранение и загрузка проекта

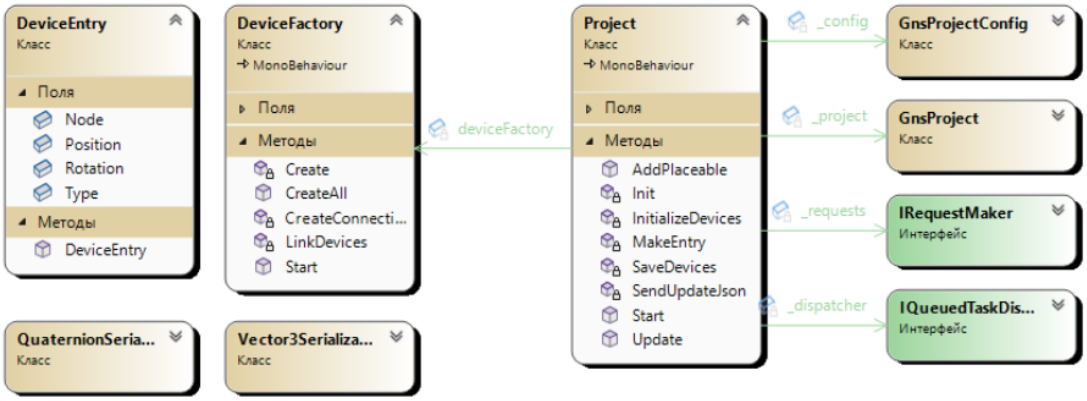


Рис. 3: Alekseev-report[1]

1. **Project**: Класс, представляющий проекты в игре. Инициализируется при авторизации и выборе проекта. Конструктор принимает строку в формате JSON, содержащую описание всей сцены проекта. Далее он использует DeviceFactory, чтобы создать устройства из этой строки.

2. **DeviceFactory**: Класс, ответственный за создание объектов устройств из префабов.

3. **DeviceEntry**: Класс, содержащий данные для создания экземпляров устройств из префабов.

2.1.4. Tasks

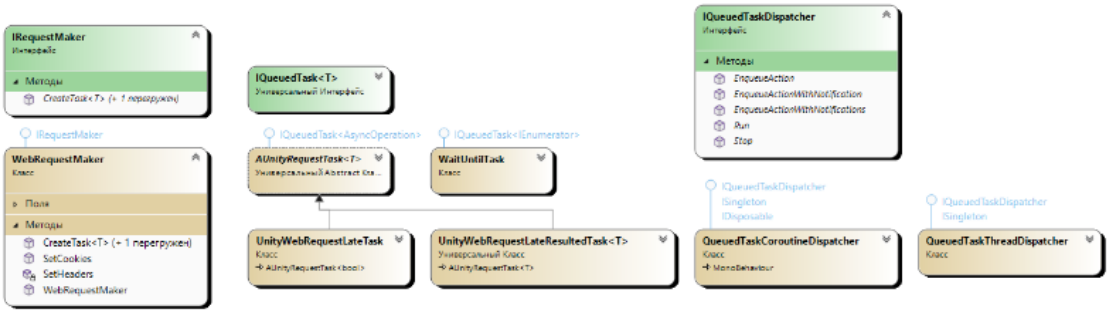


Рис. 4: Alekseev-report[1]

IQueuedTask<T>: Интерфейс для задач, имеющих методы для уведомлений и запуска.

WaitUntilTask: Класс, наследуемый от IQueuedTask, который создается пользователем.

AUnityWebRequestsTask<T>: Абстрактный класс, расширяющий IQueuedTask, для создания и отправки запросов с возвратом результата после выполнения.

UnityWebRequestLateResultedTask<T>: Класс, расширяющий AUnityWebRequestsTask, для создания запросов с ответом в формате JSON.

UnityWebRequestsLateTask: Класс, расширяющий AUnityWebRequestsTask, для создания запросов без ответа.

IQueuedTaskDispatcher: Интерфейс, который предоставляет методы для добавления и запуска задач.

QueuedTaskThreadDispatcher и **QueuedTaskCoroutineDispatcher**: Классы, которые управляют очередью задач, выполняемых в отдельном потоке, и отображают уведомления в пользовательском интерфейсе.

IRequestMaker: Интерфейс, имеющий методы для создания задач типа UnityWebRequestLateResultedTask<T> и UnityWebRequestLateTask.

WebRequestMaker: Класс, реализующий интерфейс IRequestMaker.

2.2. Используемые технологии

2.2.1. GNS3

GNS3 - это мощный сетевой эмулятор, который позволяет создавать виртуальные сетевые топологии и эмулировать работу различных устройств. Он пользуется популярностью среди сетевых инженеров и студентов, благодаря своей простоте использования и богатому набору функций.

С помощью GNS3 пользователь может создавать и настраивать виртуальные маршрутизаторы, коммутаторы и другие сетевые устройства, которые могут взаимодействовать между собой как в реальной сети.

Он поддерживает различные операционные системы, такие как Cisco IOS, Switch, Hub и т.д., что делает его универсальным средством для обучения и тестирования сетевых конфигураций.

GNS3 также обладает расширяемой архитектурой, позволяющей добавлять дополнительные модули и плагины для расширения его возможностей. Это делает его гибким инструментом для воссоздания различных сетевых сценариев и проведения экспериментов с разными протоколами и настройками.

Кроме того, GNS3 имеет интеграцию с виртуальными машинами, что позволяет запускать готовые образы виртуальных машин и взаимодействовать с ними в контексте сетевой топологии. Это облегчает тестирование и развертывание сложных приложений, использующих сетевые услуги.

В целом, GNS3 является мощным и гибким инструментом для обучения сетевой инженерии, позволяющим студентам и профессионалам создавать и изучать различные сетевые сценарии, тестировать конфигурации и разрабатывать инновационные решения в области сетевых технологий.

2.2.2. Unity

Unity - это кросс-платформенный игровой движок и интегрированная среда разработки (IDE), позволяющая создавать 2D и 3D игры, виртуальную и дополненную реальность, а также другие интерактивные мультимедийные приложения. Он предлагает широкий набор инструментов и функций, которые облегчают процесс разработки и позволяют создавать высококачественные игровые проекты.

Благодаря своей популярности в игровой индустрии, экосистема Unity предлагает богатый выбор ресурсов, библиотек, готовых решений и сообщества разработчиков, что упрощает совместную работу и обмен знаниями. Unity позволяет создавать игры и приложения различной сложности, от небольших инди-проектов до крупных коммерческих проектов, делая его популярным инструментом для разработки игр и интерактивной визуализации.

3. Реализация

3.1. Коммутатор

Коммутатор - это сетевое оборудование, используемое в компьютерных сетях для передачи данных между устройствами в локальной сети. Он состоит из портов, которые подключаются к устройствам в сети, и процессора, который управляет передачей данных между этими устройствами.

Для реализации функциональности коммутатора были введены следующие классы и префаб:

Классы:

1. **MiniSwitch**: Потомок класса ADevice.
2. **GnsSwitchNode**: Класс, наследуемый от GnsNode, обеспечивающий взаимодействие с другими устройствами.
3. **GnsJSwitchNode**: Класс, предназначенный для сохранения конфигураций коммутатора [4].

Префаб:

MiniSwitch: Префаб, представляющий коммутатор, состоящий из светового индикатора включения/выключения и корпуса, с восемью портами Ethernet и входом питания C14Cbl.

Префаб интегрирован в класс **DeviceFactory**, который отвечает за создание экземпляров устройств в соответствии с заданными конфигурациями.

3.2. Роутер

Роутер представляет собой коробочное устройство с портами для подключения к другим сетевым устройствам. Порты LAN на роутере выделены белым цветом и предназначены для подключения устройств в локальной сети, а порт WAN используется для подключения к интернету (изображен, но не активирован).

Был загружен образ Cisco Router 7200 [2] для взаимодействия с маршрутизатором в GNS3.

Для реализации функциональности коммутатора были введены следующие классы и префаб:

Классы:

1. **MiniRouter**: Потомок класса AConsoleDevice.
2. **TerminalManagerRouter**: Класс, обеспечивающий взаимодействие с терминалом GNS3.
3. **GnsRouterNode**: Класс, наследуемый от GnsNode, обеспечивающий взаимодействие с другими устройствами.
4. **GnsJRouterNode**: Класс, предназначенный для сохранения конфигураций роутера [3].

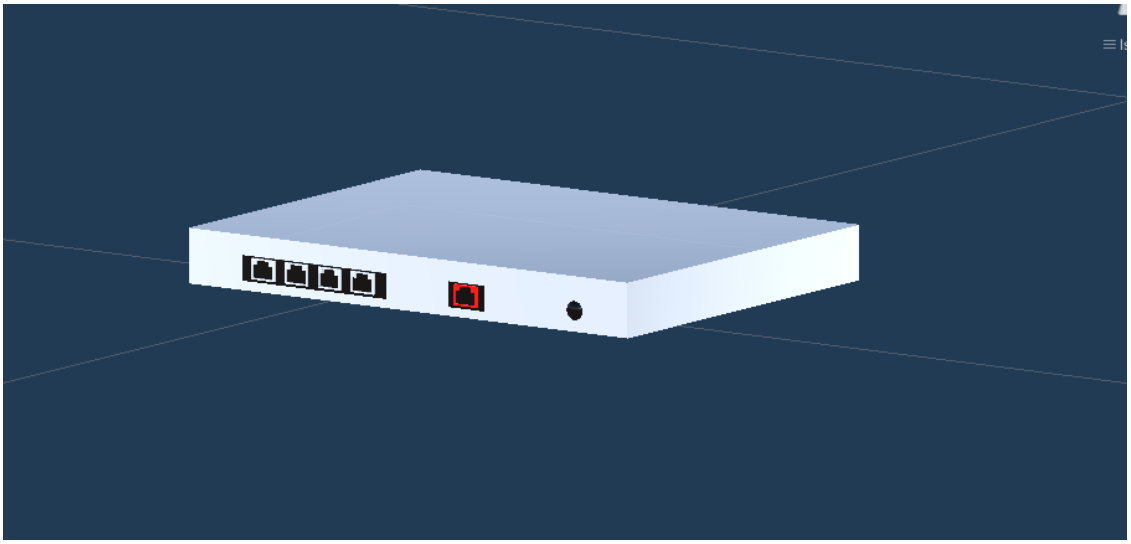
В **GNSProject** добавлены первоначальные настройки для роутера.

В **GnsNode** добавлено поле adapterID, которое обеспечивает соединение роутера на разных слотах.

Префаб:

MiniSwitch: Префаб, представляющий маршрутизатор, состоящий из корпуса, светового индикатора включения/выключения, четырех LAN портов, WAN порта и входа питания C14Cbl.

Префаб интегрирован в класс **DeviceFactory**, который отвечает за создание экземпляров устройств в соответствии с заданными конфигурациями.



Заключение

- Реализация свитча и роутера, с целью обогащения функциональности симулятора.

Проект заморожен, нет ресурсов им заниматься. Сейчас все ресурсы сосредоточены в продукте Miminet.

Список литературы

[1] Alekseev-report. — URL: https://oops.math.spbu.ru/SE/YearlyProjects/vesna-2023/pi-2/Alekseev-report.pdf.

[2] Cisco Router 7200. — URL: https://gns3.com/marketplace/appliances/cisco-7200.

[3] Router configuration. — URL: https://gns3-server.readthedocs.io/en/stable/api/v2/compute/dynamips_vm/projectsprojectiddynamipsnodes.html.

[4] Switch configuration. — URL: https://gns3-server.readthedocs.io/en/stable/api/v2/compute/ethernet_switch/projectsprojectidethernetswitchnodes.html.