Туториал: 3D/VR многопользовательский режим

Концепция логических машин платформы Лямбда-Мю 3 позволяет с легкостью решать задачи работы с многими пользователями. При этом клиенты могут по-разному визуализировать данные с платформы, но следуют одному протоколу. Например, 3D или VR вариант одной комнаты.

Разберем взаимодействие платформы с несколькими клиентами.

Пример реализации

Виртуализация при обучении помогает достигнуть целей обучения быстрее, дешевле, безопаснее и при необходимости удаленно. Рассмотрим проект обучения обслуживания серверов: преподаватель и учащиеся подключаются в одну виртуальную комнату - серверную. В режиме 3D или VR. Выполняют сценарии обучения или следят за другими пользователями.

Репозитарий

Файловая структура проекта представлена ниже.

Необходимые скрипты туториала находятся в репозитарии lm3.examples/godot_multiplayer.

Чтобы запустить платформу скачайте бинарный файл и необходимые библиотеки из репозитария lm3.engine.ce и скопируйте в корень проекта.

Исходные данные

- Сцена серверной импортированная в Godot Engine
- Запрограммированное управление виртуальным персонажем в режиме 3D и VR
- Запрограммированный модуль связи lm3 и протокол передачи данных

Цель проекта

Разбор взаимодействия платформы Лямбда-Мю 3 с клиентами в многопользовательских 3D и VR режимах:

- авторизация
- синхронизация данных (положения в пространстве)

Разбор исходных данных

Разработка модуля связи и протокола передачи данных для Godot Engine тема отдельного туториала. Однако, для понимания необходимо отметить общую концепцию обработки данных при приеме от платформы и перед отправкой на платформу.

В Godot Engine доступны сигналы, которые могут исходить от объектов или скриптов. При разработке в паре с платформой lm3 крайне удобно воспользоватся сигналами и перенести ивентную логику.

Модуль связи lm3 (скрипт автозагрузки) выполняет следующие функции: * подключение к серверу, * отслеживание состояния подключения, * получение данных, распаковка, излучение соответствующего сигнала, * буферизация данных на отправку, запаковка, отправка данных.

Каждому объекту взаимодействующему с платформой привязывается скрипт. За каждом скриптом (объекта, сцены или автозагрузки) закрепляется имя - имя объекта. Протокол взаимодействия должен при этом быть картой

```
<Имя Объекта> : <Данные>
```

или

```
<Имя Объекта> : <<Имя Атрибута> : <Данные>>
```

. Для каждого сприпта необходимо описать обработчики прием и отправки данных.

Если придерживаться данной концепции, то при разработки больших проектов всегда можно проследить поток данных и быстро отладить код.

Протокол запросов и ответов

Ответ при подключении - информация о всех пользователях:

```
{ <id:string> : { "type" : <type:string> }, ...}
```

—

Ответ (всем пользователям кроме данного) при отключении:

```
{ <id:string> : { "type" : "NA" } }
```

—

Запрос присвоения типа (3D / VR):

```
{ "clients" = { "type" : <type:string> } }
```

Ответ (всем пользователям кроме данного):

```
{ <id:string> : { "type" : <type:string> } }
```

—

Запрос присвоения положения в пространстве:

```
{ "telemetry" = <data> }
```

Ответ (всем пользователям кроме данного):

```
{ <id:string> = <data> }
```

*<data> различается для 3D и VR

Файловая структура проекта

Взаимосвязи

Скрипты