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

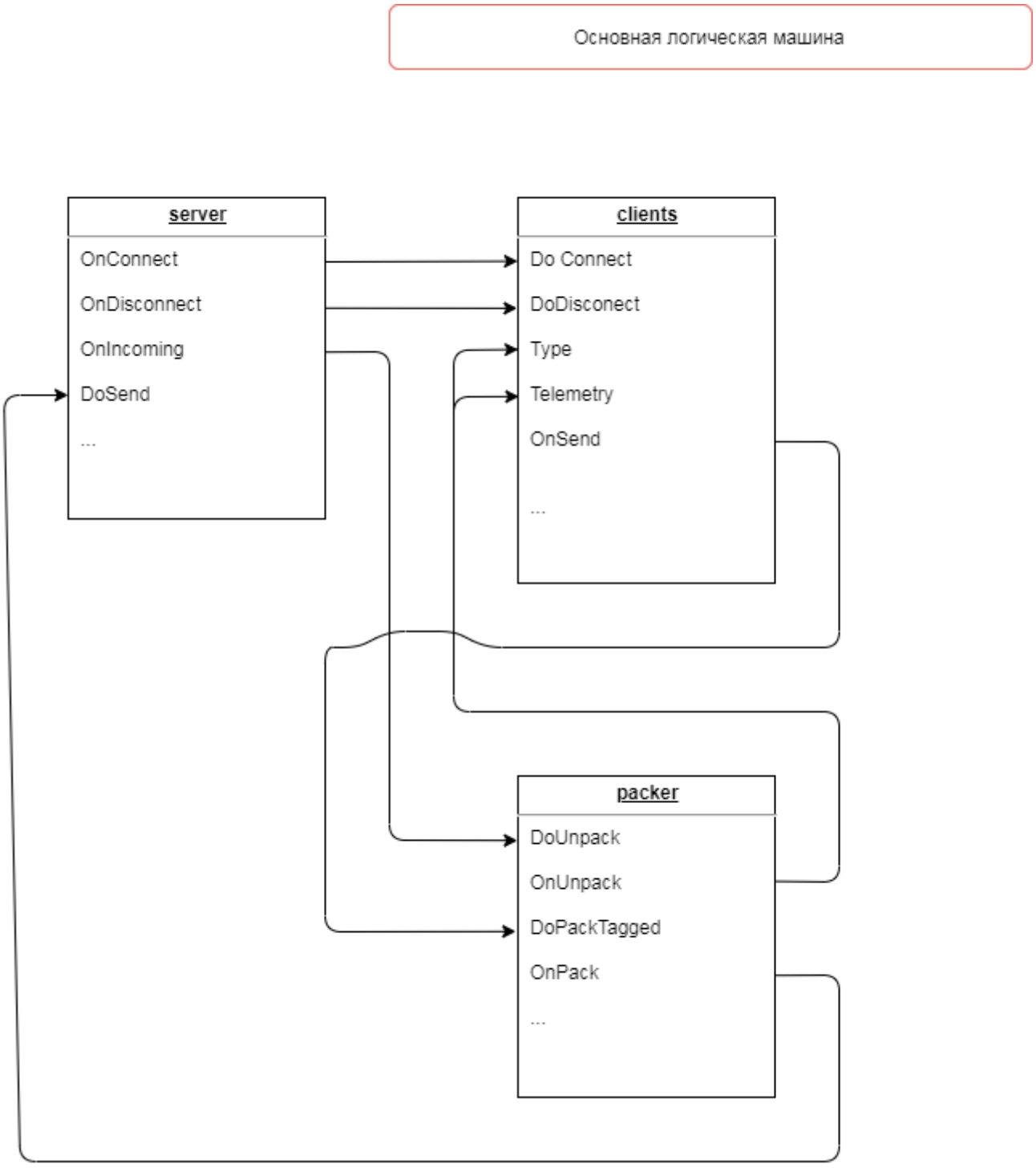
Файловая структура проекта

Взаимосвязи

Объекты основной логической машины:

- TCP-server server: прием и передача данных клиенту
- DataPack packer: распаковка и запаковка данных клиенту
- Clients clients: взаимодействие клиентов

Диаграмма связей:



Скрипты

main / start.lua

```
lm3:Include("mp_lib.lua")
lm3:LoadLibrary( {Alias = "sys", FileName = "lm3system"} )

lm3:CreateLObject( { LibAlias = "sys", Type = "TCPServer", Name = "server" }
)
```

```
server:SetAttr("SocCount", lmconf.MaxClientCount)
server:SetAttr("Port", lmconf.TCPVRPort)

lm3:CreateLObject( { LibAlias = "sys", Type = "DataPack", Name = "packer" }
)
packer:SetAttr("BufCount", lmconf.MaxClientCount)

init_Clients("clients")

init_Telemetry("telemetry")

-- связи объектов
server:Connect("OnConnect", "clients", "DoConnect")
server:Connect("OnDisconnect", "clients", "DoDisconnect")
server:Connect("OnIncoming", "packer", "DoUnpack")
packer:Connect("OnPack", "server", "DoSend")
clients:Connect("OnSend", "packer", "DoPackTagged")

-- распределим данные
packer:AddHandler("OnUnpack", function(o, ev)
    for obj_name, obj_data in pairs(ev.Data) do
        if obj_name == "clients" then
            for attr_name, attr_value in pairs(obj_data) do
                if attr_name == "Type" then
                    clients:Type( { Tag = ev.Tag, Type = attr_value} )
                end
            end
        elseif obj_name == "telemetry" then
            clients:Telemetry( {Tag = ev.Tag, Data = obj_data} )
        end
    end
end)
```

main / mp_lib.lua

```
function init_Clients(_name)
    local obj = lm3:CreateObject( { Type = "Base", Name = _name } )

    -- структура данных о клиентах
    obj.ClientList = {}
    for i = 1, lmconf.MaxClientCount do
        obj.ClientList[i] = {
            isConnected = false,
            Type = ""
        }
    end

    -- подключение клиента
    obj:AddInEvent("DoConnect", "int", function(o, ev)
        lm3:Log("New client connected:"..tostring(ev))
```

```
        obj.ClientList[ev].isConnected = true
        obj.ClientList[ev].Type = "NA"

        local sendData = { }
        local needSend = false
        for i = 1, lmconf.MaxClientCount do
            if i ~= ev and obj.ClientList[i].isConnected == true then
                sendData["Player"..tostring(i)] = { Type =
obj.ClientList[i].Type  }
                needSend = true
            end
        end

        if needSend == true then
            obj:OnSend( { Tag = ev, Data = sendData } )
        end
    end)

    -- отключение клиента
    obj:AddInEvent("DoDisconnect", "int", function(o,ev)
        lm3:Log("Client disconnected:"..tostring(ev))
        obj.ClientList[ev].isConnected = false
        local sendData = { }
        sendData["Player"..tostring(ev)] = { Type = "NA"  }
        for i = 1, lmconf.MaxClientCount do
            if obj.ClientList[i].isConnected == true then
                obj:OnSend( { Tag = i, Data = sendData } )
            end
        end
    end)

    -- присвоение типа
    obj:AddInEvent("Type", "table[Tag:int,Type:string]", function(o, ev)
        if obj.ClientList[ev.Tag].isConnected == true then
            obj.ClientList[ev.Tag].Type = ev.Type
            local sendData = { }
            sendData["Player"..tostring(ev.Tag)] = { Type = ev.Type  }
            -- lm3:Log("Set player type:"..lm3:DataToString(sendData))
            for i = 1, lmconf.MaxClientCount do
                if i ~= ev.Tag and obj.ClientList[i].isConnected == true
then
                    obj:OnSend( { Tag = i, Data = sendData } )
                end
            end
        end
    end)

    -- присвоение положения
    obj:AddInEvent("Telemetry", "table[Tag:int,Data:any]", function(o, ev)
        local sendData = {}
        sendData["Player"..ev.Tag] = ev.Data
```

```
        for i = 1, lmconf.MaxClientCount do
            if i ~= ev.Tag and obj.ClientList[i].isConnected == true then
                obj:OnSend( { Tag = i, Data = sendData } )
            end
        end
    end)

    obj:AddOutEvent("OnSend", "table[Tag:int,Data:any]")

    return obj
end

function init_Telemetry(_name)
    local obj = lm3:CreateObject( { Type = "Base", Name = _name } )


    return obj
end
```

Скриншоты

С пультом управления:

Без: