Туториал: Ментор [Лямбда-Мю 3 + Godot Engine]

В многопользовательском проекте у пользователей бывают разные роли. Платформа Лямбда-Мю может одновременно взаимодействовать с пользователями, имеющими разные роли. Рассмотрим пример такого проекта.

Пример реализации

Рассмотрим проект обучения обслуживания серверов: преподаватель (ментор) и учащиеся подключаются в одну виртуальную комнату - серверную. Учащиеся в режиме 3D или VR. Преподаватель подключается в режиме 3D. Ему доступно переключение между разными видами: от первого лица с управлением, от лица учащихся.

Дополнительно в проекте реализована доставка текстового контента на VR шлем (текстовый чат).

Репозитарий

Файловая структура проекта представлена ниже.

Необходимые скрипты туториала находятся в репозитарии lm3.examples/godot.mentor.

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

Реализация режима преподавателя. Преподаватель может переключаться между режимами свободного перемещения и просмотра глазами учащихся. Дополнительно, реализация доставки текстовых данных в шлем (текстовый чат).

Разбор исходных данных

Со стороны lm3 нужно добавить атрибут видимости клиента (модель ментора должна исчезать, если он в режиме просмотра учащихся); добавить объект MsgSender, который будет организовывать передачу текстовых данных. Со стороны Godot необходимо добавить элементы управления переключением и отправку / прием текстовых данных.

Файловая структура проекта

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")

init_MsgSender("msgSender")

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")
msgSender:Connect("OnSend", "packer", "DoPackTagged")

packer:AddHandler("OnUnpack", function(o, ev)
    -- lm3:Log("Incoming:"..lm3:DataToString(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} )
                elseif attr_name == "Visability" then
                    clients:Visability( { Tag = ev.Tag, Visability =
attr_value} )
                end
            end
        elseif obj_name == "telemetry" then
            clients:Telemetry( {Tag = ev.Tag, Data = obj_data} )
        elseif obj_name == "msgSender" then
            for attr_name, attr_value in pairs(obj_data) do
                if attr_name == "Msg" then
                    msgSender:Msg( {Tag = ev.Tag, Msg = attr_value} )
                end
            end
        end
    end
end)
```

main / mp_lib.lua

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

    obj.ClientList = {}
    for i = 1, lmconf.MaxClientCount do
        obj.ClientList[i] = {
            isConnected = false,
            Type = ""
        }
```

```
    end

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

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

        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)

    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)

    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)
```

```
    obj:AddInEvent("Visability", "table[Tag:int,Visability:bool]",
function(o, ev)
        if obj.ClientList[ev.Tag].isConnected == true then
            obj.ClientList[ev.Tag].Visability = ev.Visability
            local sendData = { }
            sendData["Player"..tostring(ev.Tag)] = { Visability =
ev.Visability  }
            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)

    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

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


    obj:AddInEvent("Msg", "table[Tag:int,Msg:string]", function(o, ev)
        lm3:Log("MsgSender: Msg:"..lm3:DataToString(ev.Msg))
        -- todo
        if clients.ClientList[ev.Tag].isConnected == true then
            local sendData = { }
            sendData["console"] = { Msg = ev.Msg }
            for i = 1, lmconf.MaxClientCount do
                -- todo
                if clients.ClientList[i].isConnected == true then
```

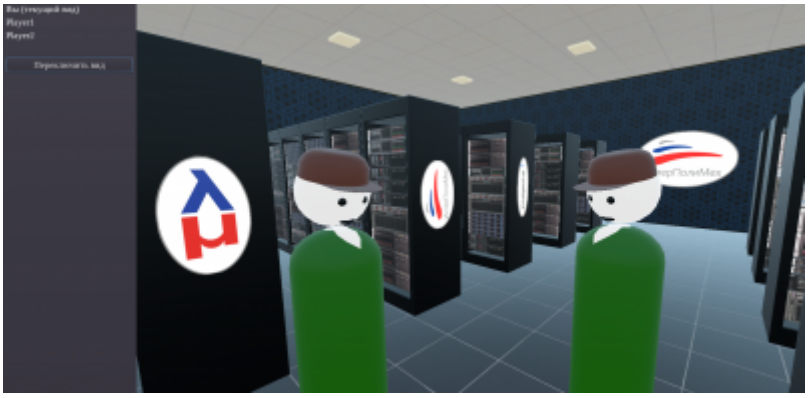
```
                --if i ~= ev.Tag and obj.ClientList[i].isConnected == true
then
                    obj:OnSend( { Tag = i, Data = sendData } )
                end
            end
        end
    end)

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

    return obj
end
```

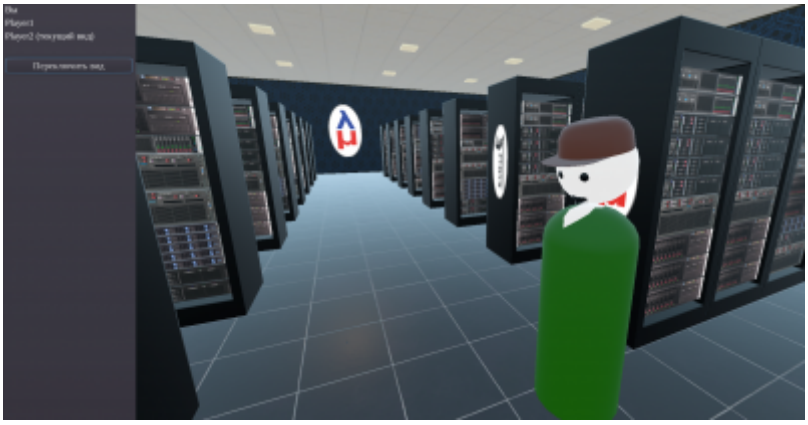
Скриншоты

Ментор, свободное движение



Ментор, от Player1

Ментор, от Player2



VR учащийся