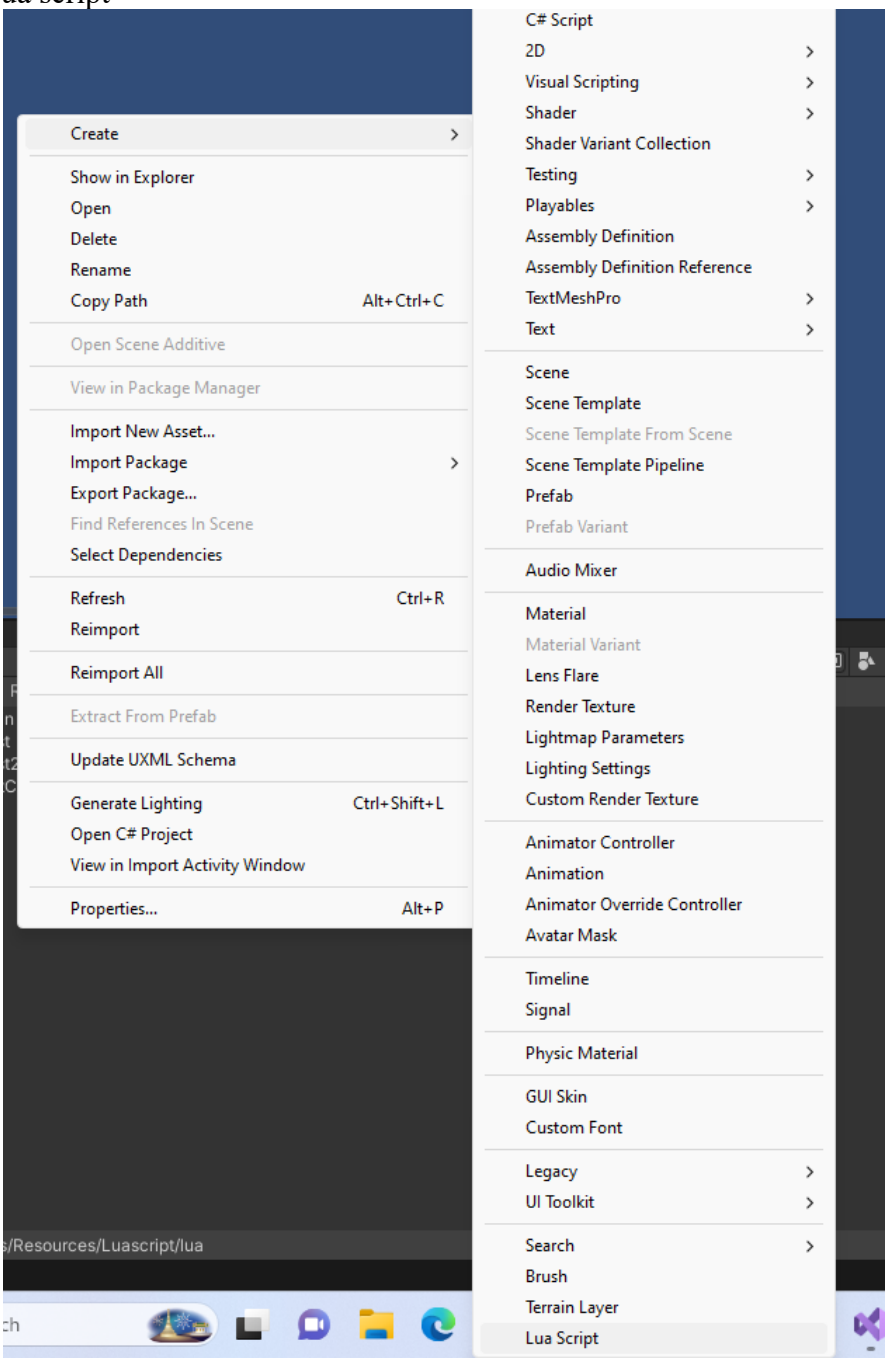
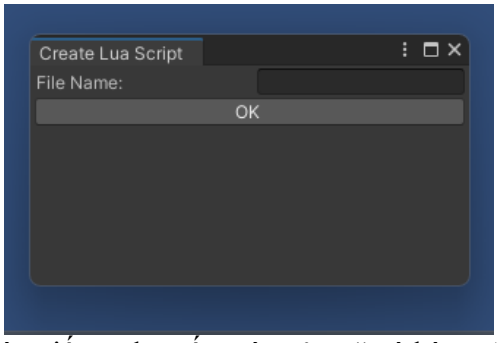
API Lua script MyFish

Hướng dẫn tạo script Component

Trong Unity bạn hãy vào thư mục “Assets/Resources/Luascript/lua” sau đó nhấp chuột bấm vào Create chọn Lua script



Sau khi nhấp vào sẽ hiện cửa sổ đặt tên file, file này cũng sẽ tạo các cấu trúc cơ bản của lua component



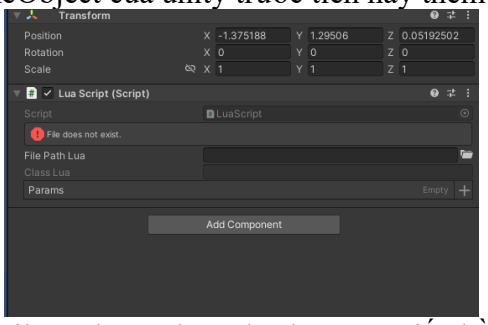
Trong Lua Component cấu trúc giống như cấu trúc của c# có hàm, Awake, Start, Update,….
Lưu ý: bắt buộc phải đưa đường dẫn file vào thuộc tính __path, điều này nhằm mục đích debug, xác định vị trí của file đó nằm ở đâu

```
---@class hello : MonoBehaviour
local hello = class("hello", MonoBehaviour)
hello.__path = "lua/hello.lua" -- bắt buộc phải có
function hello:Start()

end

function hello:Update()

end

return hello
```

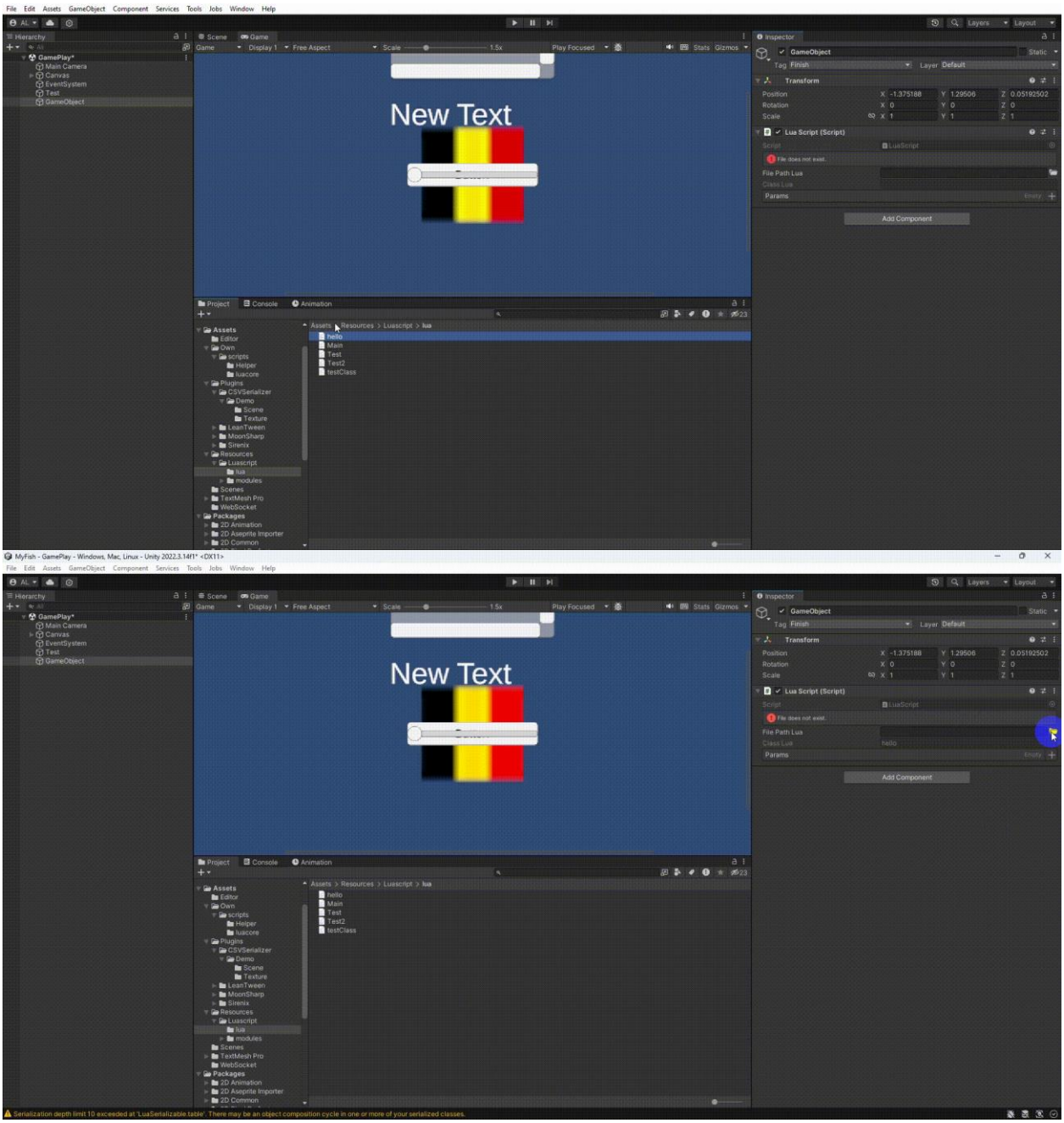
Sau khi tạo xong hãy nhớ vào file Main.lua require file đó, file nào được require sẽ được biên dịch

```
require "Test"
require "Test2"
require "hello" -- require ở đây
```

Để thêm Component vào gameObject của unity trước tiên hãy them Component LuaScript



Sau đó kéo file lua vào trong File Path Lua hoặc là chọn trực tiếp bằng cách nhấn vào thư mục ở bên gốc phải

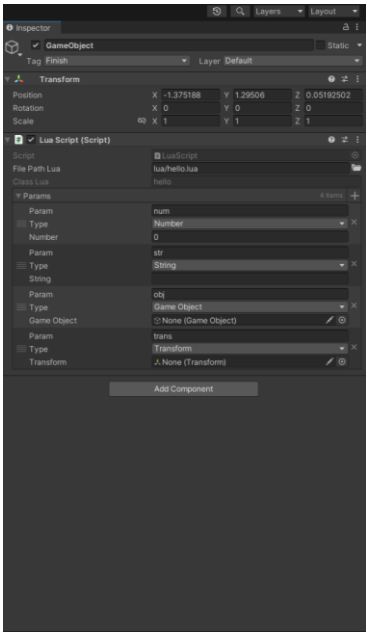


Thuộc tính

Để hiển thị được các thuộc tính trên Inspector ta hãy điền các giá trị field vào trong lua script

```
---@class hello : MonoBehaviour
---@field num number
---@field str string
---@field obj GameObject
---@field trans Transform
```

Kết quả


Layers
Layout
Inspector
GameObject
Static
Tag Finish
Layer Default
Transform
Position X -1.375188 Y 1.29506 Z 0.05192502
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Lua Script (Script)
Script LuaScript
File Path Lua lua/hello.lua
Class Lua hello
Params 4 items
Param num
Type Number
Number 0
Param str
Type String
String
Param obj
Type Game Object
Game Object None (Game Object)
Param trans
Type Transform
Transform None (Transform)
Add Component

Các API

class

Tham số

- className: kiểu string, tên của class bắt buộc phải có
- super: kế thừa một class khác

Để khai báo một class nào đó trong Lua ta phải tạo một biến bằng với phương thức class, Ví dụ:

```
local helloClass = class("helloClass")
```

Lưu ý phải định danh class bằng phương pháp định danh dữ liệu của Lua để IDE giợi ý, đồng thời cũng để trình biên dịch C# hiểu được đây là class nào

```
---@class helloClass
local helloClass = class("helloClass")
```

Kế thừa:

Ta có thể kế thừa class một class khác Ví dụ:

```
local helloClass,base = class("helloClass", MonoBehaviour)
```

Ví dụ cho thấy lớp helloClass đang kế thừa lớp MonoBehaviour

Base: chính là các hàm của lớp cha

MonoBehaviour

MonoBehaviour là môt class dùng để kế thừa, có các chức năng tương tự như bên c#,
Các thuộc tính bao gồm

Awake:
Khởi chạy 1 lần khi một gameObject được tạo
Ví dụ:

```
function hello:Awake()
    print("hello")
end
```

OnEnable:
Khởi chạy sau Awake khi gameObject được bật lên
Ví dụ:

```
function hello:OnEnable()
    print("hello OnEnable")
end
```

Start:
Khởi chạy 1 lần sau 1 frame khi gameObject được tạo
Ví dụ:

```
function hello:Start()
    print("hello start")
end
```

Update:
Khởi chạy sau mỗi 0.05 giây
Ví dụ:

```
function hello:Update()
    print("hello Update")
end
```

OnDisable:
Khởi chạy trước khi gameObject ẩn
Ví dụ:

```
function hello:OnDisable()
    print("hello OnDisable")
end
```

OnDestroy:
Khởi chạy trước khi gameObject bị xóa
Ví dụ:

```
function hello:OnDestroy()
    print("hello OnDestroy")
end
```

OnMouseDown:
Khởi chạy khi click xuống vào gameObject
Ví dụ:

```
function hello:OnMouseDown()
    print("hello OnMouseDown")
end
```

OnMouseUp

Khởi chạy khi click lên vào gameObject

Ví dụ:

```
function hello:OnMouseUp()
    print("hello OnMouseUp")
end
```

OnMouseEnter

Khởi chạy khi đưa chuột vào gameObject

Ví dụ:

```
function hello:OnMouseEnter()
    print("hello OnMouseUp")
end
```

OnMouseExit

Khởi chạy khi đưa chuột ra khỏi gameObject

Ví dụ:

```
function hello:OnMouseExit()
    print("hello OnMouseUp")
end
```

GetComponent:

Tham số:

- classname: kiểu string là tên của class mà mình muốn lấy

Trả về:

- Component: một object nếu có trên component

Ví dụ:

```
function hello:OnEnable()
    self:GetComponent("Hi")
end
```

GetInstanceID:

Trả về:

- Number: Một số duy nhất của Component

Ví dụ

```
function hello:OnEnable()
    local InstanceID = self:GetInstanceID()
end
```

GetEnable:

Trả về:

- Boolean: Trạng thái ẩn hoặc hiện của Component

Ví dụ:

```
function hello:OnEnable()
    local enable = self:GetEnable()
end
```

SetEnable:

Tham số:

- enable:Kiểu bool trạng thái ẩn hoặc hiện

Ví dụ:

```
function hello:OnEnable()
    self:SetEnable(false)
end
```

Destroy:

Tham số:

- Object: là Component hoặc là GameObject

Ví dụ:

```
function hello:OnEnable()
    self:Destroy(self.gameObject)
end
```

Instantiate<T>

Tham số:

- Object: là Component hoặc là GameObject

Trả về

- T: Tùy thuộc vào tham số nào truyền vào

Ví dụ:

```
function hello:OnEnable()
    self:Instantiate(self.obj)
end
```

transform

Là biến Transform

Ví dụ:

```
function hello:OnEnable()
    local transform = self.transform
end
```

gameObject

Là biến GameObject

```
function hello:OnEnable()
    local gameObject = self.gameObject
end
```

GameObject

GameObject là một class dùng để điều khiển object trong unity
Các thuộc tính bao gồm:

AddComponent

Tham số:

- Classname: Kiểu string Tên của Component

Trả về:

- Component: trả về Component

Ví dụ:

```
function hello:OnEnable()
    self.gameObject:AddComponent("Hi")
end
```

SetActive

Tham số:

- Active: Kiểu bool trạng ẩn hoặc hiện

Ví dụ:

```
function hello:OnEnable()
    self.gameObject:SetActive(false)
end
```

GetActive

Trả về:

- Bool: trạng thái ẩn hoặc hiện

Ví dụ:

```
function hello:OnEnable()
    local active = self.gameObject:GetActive()
end
```

Transform

Transform là một class điều khiển transform của unity
Các thuộc tính bao gồm:

GetPosition

Trả về:

- Vector3: vị trí hiện tại của transform

Ví dụ:

```
function hello:OnEnable()
    local position = self.transform:GetPosition()
end
```

SetPosition

Tham số:

- Vector3: Kiểu Vector3 đặt vị trí hiện tại của Transfrom

Ví dụ:

```
function hello:OnEnable()
    self.transform:SetPosition(Vector3.new(10,10,10))
end
```

Move

Di chuyển mượt và liên tục theo Vector3
Tham số:

- Vector3: Kiểu Vector3

Ví dụ

```
function hello:OnEnable()
    -- di chuyển lên trên liên tục
    self.transform:Move(Vector3.new(0,1,0))
end
```

StopMove

Dừng lại khi đang di chuyển
Ví dụ:

```
function hello:OnEnable()
    self.transform:StopMove()
end
```

GetRotation

Trả về:

- Quaternion: góc xoay hiện tại

Ví dụ

```
function hello:OnEnable()
    local rotation = self.transform:GetRotation()
end
```

SetRotation

- Tham Số: Kiểu Quaternion chỉnh góc xoay hiện tại

Ví dụ

```
function hello:OnEnable()
    self.transform:SetRotation(Quaternion.identity)
end
```

SmoothRotate

Xoay mượt transform liên tục theo góc Euler

Tham số:

- Vector3: Kiểu Vector3 là góc Euler

Ví dụ

```
function hello:OnEnable()
    self.transform:SmoothRotate(Vector3.new(0,0,30))
end
```

StopRotate

Dừng lại transform đang xoay

Ví dụ:

```
function hello:OnEnable()
    self.transform:StopRotate()
end
```

GetLocalPosition

Trả về:

- Vector3: vị trí local hiện tại của transform

Ví dụ:

```
function hello:OnEnable()
    local localPosition = self.transform:GetLocalPosition()
end
```

SetLocalPosition

Tham số:

- Vector3: Kiểu Vector3 đặt vị trí local của transform

Ví dụ

```
function hello:OnEnable()
    self.transform:SetLocalPosition(Vector3.new(10,10,10))
end
```

GetChildCount

Trả về:

- Number: Số lượng con của transform

Ví dụ:

```
function hello:OnEnable()
    local childCount = self.transform:GetChildCount()
end
```

GetChild

Tham số:

- Index: kiểu number là vị trí của con, bắt đầu từ 1

Trả về:

- Transform

Ví dụ

```
function hello:OnEnable()
    local child = self.transform:GetChild(1) -- lấy vị trí đầu tiên của con
end
```

GetAllChild

Trả về:

- Transform[]: Một mảng transform là con của cha

Ví dụ

```
function hello:OnEnable()
    local childs = self.transform:GetAllChild()
    for index, value in ipairs(childs) do
        childs:SetPosition(Vector3.new(10,10,10))
    end
end
```

Vector3

Vector3 là một class trong Lua dùng để biểu diễn một vector 3 chiều. Nó có các thuộc tính và phương thức sau:

Thuộc tính

x: Tọa độ x của vector.
y: Tọa độ y của vector.
z: Tọa độ z của vector.

Phương thức

new(x, y, z): Tạo một vector mới với các tọa độ x, y, z cho trước.
set(x, y, z): Thay đổi các tọa độ của vector.
get(x, y, z): Lấy giá trị của các tọa độ x, y, z.
length(): Trả về độ dài của vector.
normalized(): Trả về một vector được chuẩn hóa.
dot(v): Trả về tích vô hướng của vector với v.
cross(v): Trả về vector chéo với v.

Ví dụ

```lua
local v1 = Vector3.new(1, 2, 3)
local v2 = Vector3.new(4, 5, 6)

-- Lấy giá trị của các tọa độ
print(v1.x) -- 1
print(v1.y) -- 2
print(v1.z) -- 3

-- Thay đổi các tọa độ
v1.x = 10
v1.y = 20
v1.z = 30

-- Trả về độ dài của vector
print(v1.length()) -- 31.62277660168379

-- Trả về một vector được chuẩn hóa
local v1n = v1.normalized()
print(v1n.x) -- 0.942809041582063
print(v1n.y) -- 0.3333333333333333
print(v1n.z) -- 0.07692307692307692

-- Trả về tích vô hướng của vector với v
print(v1.dot(v2)) -- 70

-- Trả về vector chéo với v
local v1xv2 = v1.cross(v2)
print(v1xv2.x) -- 6
print(v1xv2.y) -- -8
```

```
print(v1xv2.z) -- 2
```

LeanTween

LeanTween là một thư viện dùng để hỗ trợ việc di chuyển mượt mà theo đường Tween

LeanTweenType

Các kiểu đường tween easing , có thể tham khảo tại đây

LTDescr

Là lớp điều khiển đường Tween đang chạy, có thể điều chỉnh đường easing hoặc bắt sự kiện, Nó có các phương thức như sau:

setOnComplete

Tham số:

- Func: kiểu function, sự kiện trả về khi kết thực 1 tween

Ví dụ:

```
function hello:OnEnable()
    LeanTween:move(self.gameObject, Vector3.up * 10, 10):setOnComplete(function
()
        -- sau khi chạy xong Tween sẽ in ra Complete
        print("Complete")
    end)
end
```

setEase

Cài đường Easing

Tham số:

- LeanTweenType

Ví dụ

```
function hello:OnEnable()
    LeanTween:move(self.gameObject, Vector3.up * 10, 10)
        :setEase(LeanTweenType.easeInOutSine) -- cài đường Easing InOutSine
end
```

pause

Dừng lại 1 Tween

Ví dụ:

```
function hello:OnEnable()
    local tween = LeanTween:move(self.gameObject, Vector3.up * 10, 10)
    Time:startTimer(5, function () -- tween chạy sau 5 giây sẽ dừng lại
        tween:pause()
    end)
end
```

resume

Chạy tiếp Tween khi đã dừng

Ví dụ:

```
function hello:OnEnable()
    local tween = LeanTween:move(self.gameObject, Vector3.up * 10, 10)
```

```
    Time:startTimer(5, function () -- tween chạy sau 5 giây sẽ dừng lại
        tween:pause()
    end)
    Time:startTimer(7, function () -- sau khi dừng lại đợi 2 giây tiếp tục chạy
        tween:resume()()
    end)
end
```

cancel

Xóa hẳn 1 tween và không thể resume

Ví dụ:

```
function hello:OnEnable()
    local tween = LeanTween:move(self.gameObject, Vector3.up * 10, 10)
    Time:startTimer(5, function () -- tween chạy sau 5 giây sẽ dừng lại xóa tween
        tween:cancel()
    end)
end
```

LeanTween

Đây là lớp static để điều khiển di chuyển position hay xoay rotation hay phóng to, thu nhỏ scale gồm các phương thức:

move

Di chuyển position theo đường tween

Tham số:

- gameObject: kiểu GameObject là object nào mình muốn di chuyển
- vector3: Kiểu Vector3 là vị trí mình muốn tới
- time: Kiểu number là thời gian di chuyển

Trả về:

- LTDescr

Ví dụ

```
function hello:OnEnable()
    local tween = LeanTween:move(self.gameObject, Vector3.up * 10, 10)
end
```

moveLocal

Di chuyển localPosition theo đường tween

Tham số:

- gameObject: kiểu GameObject là object nào mình muốn di chuyển
- vector3: Kiểu Vector3
- time: Kiểu number là thời gian di chuyển

Trả về:

- LTDescr

Ví dụ

```
function hello:OnEnable()
```

```
    local tween = LeanTween:moveLocal(self.gameObject, Vector3.up * 10, 10)
end
```

scale

Phóng to hoặc thu nhỏ theo đường Tween

Tham số:

- gameObject: kiểu GameObject là object nào mình muốn di chuyển
- vector3: Kiểu Vector3
- time: Kiểu number là thời gian di chuyển

Trả về:

- LTDescr

Ví dụ

```
function hello:OnEnable()
    local tween = LeanTween:scale(self.gameObject, Vector3.up * 10, 10)
end
```

rotate

Xoay theo đường Tween

Tham số:

- gameObject: kiểu GameObject là object nào mình muốn di chuyển
- vector3: Kiểu Vector3
- time: Kiểu number là thời gian di chuyển

Trả về:

- LTDescr

Ví dụ

```
function hello:OnEnable()
    local tween = LeanTween:rotate(self.gameObject, Vector3.up * 10, 10)
end
```

Event

Event là một lớp gửi và nhận sự kiện từ một hệ thống khác
Các phương thức bao gồm

Register

Đăng ký 1 sự kiện
Tham số

- eventName: Kiểu string là tên event muốn đăng ký
- func: Kiểu function là hàm thực thi khi sự kiện được gọi

Trả về

- number: là eventID dùng để hủy sự kiện

Ví dụ

```
function hello:OnEnable()
    local eventID = Event:Register("OnEvent", Lib.handler(self, self.OnEvent)) --
thực hiện hàm OnEvent khi sự kiện OnEvent được gọi
end

function hello:OnEvent()
    print("Hello event")
end
```

UnRegister

Hủy một sự kiện
Tham số

- eventName: Tên sự kiện muốn hủy
- eventID: Kiểu number là ID của Event đăng ký

Ví dụ

```
function hello:OnEnable()
    self.eventID = Event:Register(“OnEvent”, Lib.handler(self, self.OnEvent)) -
thực hiện hàm OnEvent khi sự kiện OnEvent được gọi
end

function hello:OnEvent()
    print(“Hello event”)
end

-- sau khi object bị ẩn thì hủy sự kiện đi
function hello:OnDisable()
```

```
    Event:UnRegister(“OnEvent”, self.eventID)
end
```

Emit

Thực hiện một sự kiện

Tham số:

- eventName: Kiểu string tên sự kiện muốn thực thi
- ...: Các tham số muốn truyền vào

Ví dụ

```
function hello:OnEnable()
    self.eventID = Event:Emit("OnEvent") -- gọi sự kiện OnEvent
    self.eventID2 = Event:Emit("OnEvent", 1) -- gọi sự kiện OnEvent và truyền vào
tham số là 1
end
```

RegisterRequestData

Đăng ký một yêu cầu dữ liệu và bắt buộc phải trả về dữ liệu khi được yêu cầu

Lưu ý: RegisterRequestData chỉ có thể đăng ký 1 lần duy nhất, nếu tên đăng ký bị trùng thì nó sẽ ghi đè lên tên cũ

Tham số:

- requestName: Kiểu string là tên request muốn đăng ký
- func: Kiểu function là hàm thực thi khi sự kiện được gọi

Ví dụ

```
function hello:OnEnable()
    Event:RegisterRequestData("OnRequest", Lib.handler(self, self.OnRequest))
end

-- hàm request bắt buộc phải có param và callBack
-- param là tham số sẽ được truyền vào từ nơi request
-- callBack là một function để trả dữ liệu cho nơi request
function hello:OnRequest(param, callBack)
    callBack(2)
end
```

RequestData

Yêu cầu dữ liệu từ một hệ thông khác

Tham số:

- requestName: Kiểu string là tên request muốn yêu cầu
- param: Kiểu any (đa số dùng table) là tham số truyền vào khi request
- callback: Kiểu function là hàm nhận dữ liệu request

Ví dụ

```
function hello:OnEnable()
    Event:RequestData("OnRequest", 1 ,Lib.handler(self, self.OnRequest))
end

-- data là dữ liệu nhận được từ RequestData
function hello:OnRequest(data)
    print(data)
end
```

ServerHandler

Là một lớp kết nối với server, yêu cầu dữ liệu, đăng ký sự kiện khi được server gọi xuống
Các phương thức bao gồm:

On

Đăng ký sự kiện từ server
Tham số:

- event: Kiểu string là sự kiện của server
- func: Kiểu function là hàm thực thi khi sự kiện được gọi

Ví dụ

```
function hello:OnEnable()
    ServerHandler:On("OnGameStart", Lib.handler(self, self.OnGameStart))
end

-- data kiểu table là dữ liệu server truyền vào
function hello:OnGameStart(data)
    print(data)
end
```

Off

Hủy một sự kiện từ server
Tham số:

- event: Kiểu string sự kiện muốn hủy

Ví dụ

```
function hello:OnEnable()
    ServerHandler:On("OnGameStart", Lib.handler(self, self.OnGameStart))
end

-- data kiểu table là dữ liệu server truyền vào
function hello:OnGameStart(data)
    print(data)
end

-- sau khi object bị ẩn thì hủy sự kiện đi
function hello:OnDisable()
    ServerHandler:Off("OnGameStart")
end
```

SendMesseage

Gửi một tin nhắn lên server và nhận về dữ liệu (nếu có)
Tham số

- event: Kiểu string tên sự kiện của server
- message: Kiểu table, dữ liệu muốn gửi lên server
- callback: Kiểu function (tùy chọn) server sẽ gọi lại hàm này sau khi thực hiện yêu cầu

Ví dụ

```
function hello:OnEnable()
    ServerHandler:SendMesseage("sendServer", {msg = "hello"}, Lib.handler(self,
self.CallBack))
end

-- data kiểu table là dữ liệu server truyền vào
function hello:CallBack(data)
    print(data)
end
```

SendHTTP

Gửi HTTP lên server

Tham số:

- url: Kiểu string là đường dẫn của server (không bao gồm tên miền)
- data: kiểu table là dữ liệu gửi lên server
- callback: kiểu function là dữ liệu được trả về

Ví dụ

```
function hello:Start()
    ServerHandler:SendHTTP("user/login", {
        email = "admin@gmail.com",
        password = "admin"
    },
    function (data)
        Lib.pv(data)
    end)
end
```

Time

Lớp đếm thời gian, dùng để delay khoản thời gian và thực hiện tác vụ
Gồm những phương thức sau

startTimer

Bắt đầu thực hiện đếm thời gian
Tham số:

- time: Kiểu number là thời gian đếm
- func: Kiểu function là hàm gọi lại sau khi đếm xong

Trả về:

- number: Là id thời gian dùng để dừng đếm

Ví dụ:

```
function hello:OnEnable()
    Time:startTimer(10, function ()
        print("Start delay 10s") -- sau 10 giây thực hiện hàm
    end)
end
```

stopTimer

Dừng đếm thời gian
Tham số:

- id: id của thời gian bắt đầu

Ví dụ

```
function hello:OnEnable()
    self.timeID = Time:startTimer(10, function ()
        print("Start delay 10s") -- sau 10 giây thực hiện hàm
    end)
end

function hello:OnDisable()
    Time:stopFramer(self.timeID)
end
```

startFramer

Bắt đầu đếm theo frame
Tham số:

- frame: Kiểu number là số frame muốn đếm
- func: Kiểu function là hàm gọi lại sau khi đếm xong

Trả về:

- number: Là id dùng để dừng đếm

Ví dụ:

```
function hello:OnEnable()
    self. frameD= Time:startFramer(10, function ()
        print("Start delay 10 frame") -- sau 10 frame thực hiện hàm
    end)
```

```
end
```

stopFramer

Dừng đếm frame

Tham số:

- id: id của frame

Ví dụ

```
function hello:OnEnable()
    self.frameD = Time:startFramer(10, function ()
        print("Start delay 10 frame") -- sau 10 frame thực hiện hàm
    end)
end

function hello:OnDisable()
    Time:stopFramer(self.timeID)
end
```

Các Event

Các sự kiện phần cứng được tryền từ c#

Sự kiện	Mô tả	Tham số
KEY_DOWN	Sự kiện khi một phím bất kỳ nhấn xuống	KeyCode
KEY_PRESSED	Sự kiện khi một phím bất kỳ nhấn giữ	KeyCode
KEY_UP	Sự kiện khi một phím bất kỳ nhấn lên	KeyCode
MOUSE_DOWN	Sự kiện khi chuột nhấp vào màn hình	{code: number, position: Vector3}
MOUSE_UP	Sự kiện khi chuột nhả nhấp vào màn hình	{code: number, position: Vector3}

Ví dụ:

```
Event:Register("KEY_DOWN", function (keyCode)
    if keyCode == KeyCode.A then -- kiểm tra có phải phím A không
        print("phím a được nhấn xuống")
    end
end)
```

```
Event:Register("KEY_UP", function (keyCode)
    if keyCode == KeyCode.A then -- kiểm tra có phải phím A không
        print("phím a được nhấn lên")
    end
end)
```

```
Event:Register("KEY_PRESSED", function (keyCode)
    if keyCode == KeyCode.A then -- kiểm tra có phải phím A không
        print("phím a được nhấn giữ")
    end
end)
```

```
Event:Register("MOUSE_DOWN", function (param)
    local code = param.code -- 0 là chuột trái, 1 là chuột phải, 2 là chuột
giữa
    print("Chuột "..code.."được nhấn tại vị trí "..param.position:tostring())
end)
```

```
Event:Register("MOUSE_UP", function (param)
    local code = param.code -- 0 là chuột trái, 1 là chuột phải, 2 là chuột
giữa
    print("Chuột "..code.."được nhấn tại vị trí "..param.position:tostring())
```

```
end)
```

DataLocalManager

Là data được lưu vào dưới máy client, thường là những data không quan trọng

GetValue(key)

Hàm dùng để lấy data với tham số key là một chuỗi string

SetValue(key, value)

Hàm dùng để đặt và lưu trữ data theo key và value

Save()

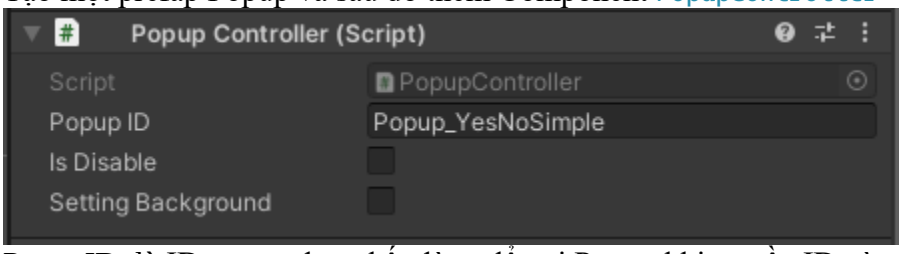
Hàm dùng để lưu data ngay lặp tức

Plugin Popup

Plugin Popup là một Plugin hỗ trợ tạo những Popup UI một cách nhanh chóng và dễ dàng sử dụng

Hướng dẫn

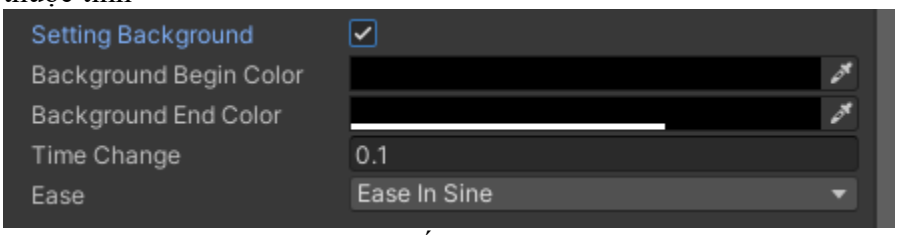
Tạo một prefap Popup và sau đó thêm Component `PopupController`



PopupID: là ID popup duy nhất dùng để gọi Popup khi truyền ID vào

Is Disable: Nghĩa là có sử dụng Popup này trong project không

Setting Background: Cài đặt background bên ngoài popup, khi nhấp vào sẽ hiển thị thêm mội số thuộc tính

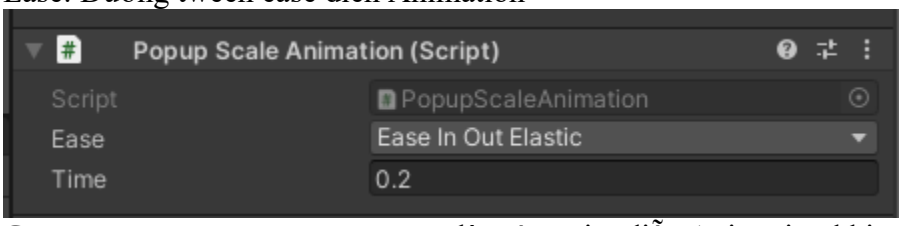


Background Begin Color: Màu sắc mà Background trước khi mở

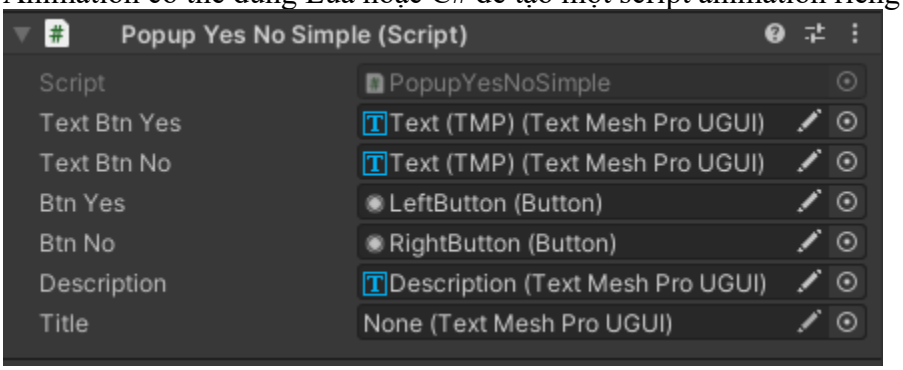
Background End Color: Màu sắt mà Backgrond sau khi mở

Time Change: Thời gian diễn Animation từ Background Begin Color sang Background End Color

Ease: Đường tween ease diễn Animation



Component `PopupScaleAnimation` là một script diễn Animation khi popup đóng hoặc mở. Animation có thể dùng Lua hoặc C# để tạo một script animation riêng mình



Component `PopupScript` là component viết logic cho Popup ví dụ như, click vào nút button nào thì thực hiện hành động nào. Ví dụ `PopupYesNoSimple` là component dành cho Popup thông báo lựa cho Yes hoặc No và thực hiện một hành động của người truyền Popup

PopupManager

PopupManager là một lớp quản lý tất cả popup, có nhiệm vụ mở popup và đóng popup khi được yêu cầu

show

Đây là hàm dùng để mở Popup với các tham số:

- popupID: là popupID
- param: là tham số muốn truyền vào Popup đó (Lưu ý tham số truyền vào phải là kiểu table)

Ví dụ

```
PopupManager:show("Popup_Notification", {
    title = "THÔNG BÁO",
    desrciption = "Bạn có muốn bán cá",
    btnYes = {
        text = "Xác nhận",
        onClick = function()
                self:SellFish(fish)
            end
    },
    btnNo = {
        text = "Hủy Bỏ",
        onClick = function ()
            PopupManager:hide("Popup_Notification")
        end
    }
})
```

hide

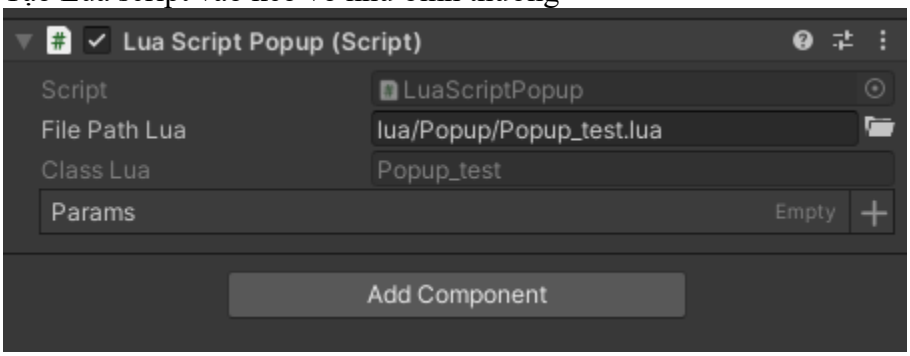
Dùng để đóng Popup theo PopupID

Hướng dẫn tạo Popup bằng Lua

Trước tiên thêm `PopupController` vào prefab nào mình muốn nó thành Popup và thêm `PopupAniation` nếu muốn

Thêm Componnet `LuaScriptPopup` Thay vì `LuaScript`

Tạo Lua script vào kéo vô như bình thường



Trong Popup Lua sẽ có những thuộc tính và sự kiện như sau

- OnBeginShow(): là một sự kiện được gọi trước khi mở
- OnEndShow(): là một sự kiện được gọi sau khi mở
- OnBeginHide(): là một sự kiện được gọi trước khi đóng
- OnEndHide(): là một sự kiện được gọi sau khi đóng
- PopupID: là poupID
- param: là tham số được truyền vào

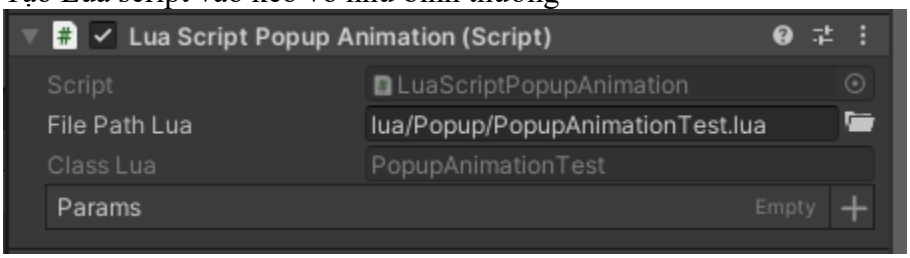
Ví dụ

```lua
---@class Popup_test : MonoBehaviour
local Popup_test = class("Popup_test", MonoBehaviour)
Popup_test.__path = __path

function Popup_test:OnBeginShow()
    print(self.PopupID)
    Lib.pv(self.param)
end

function Popup_test:OnEndShow()

end
function Popup_test:OnBeginHide()

end
function Popup_test:OnEndHide()

end

function Popup_test:close()
    PopupManager:hide(self.PopupID)
end
```

```
_G.Popup_test = Popup_test
```

Hướng dẫn custom Animation Popup bằng Lua

Trước tiên thêm `PopupController` vào prefab nào mình muốn nó thành Popup
Thêm Componnet `LuaScriptPopupAnimation` Thay vì `LuaScript`
Tạo Lua script vào kéo vô như bình thường



Trong Popup Lua sẽ có những thuộc tính và sự kiện như sau

- OnShow(onComplete): Sự kiện khi popup được mở, (onComplete là tham số trả về khi kết thực animation)
- PopupAnimationTest:OnHide(onComplete): Sự kiện khi popup được đóng

Lưu ý:
onComplete là một Action của C# nên bạn không thể truyền nó ngược về C# để gọi lại nên hay bao bọc nó trong một hàm rồi truyền về C#
Ví dụ

```lua
---@class PopupAnimationTest : MonoBehaviour
local PopupAnimationTest = class("PopupAnimationTest", MonoBehaviour)
PopupAnimationTest.__path = __path

function PopupAnimationTest:OnShow(onComplete)
    self.transform:SetLocalPosition(Vector3.new(0,1000,0))
    LeanTween:moveLocal(self.gameObject, Vector3.zero,
0.5):setOnComplete(function ()
        onComplete()
    end)
end

function PopupAnimationTest:OnHide(onComplete)
    LeanTween:moveLocal(self.gameObject, Vector3.new(0,1000,0),
0.5):setOnComplete(function ()
        onComplete()
    end)
end

_G.PopupAnimationTest = PopupAnimationTest
```