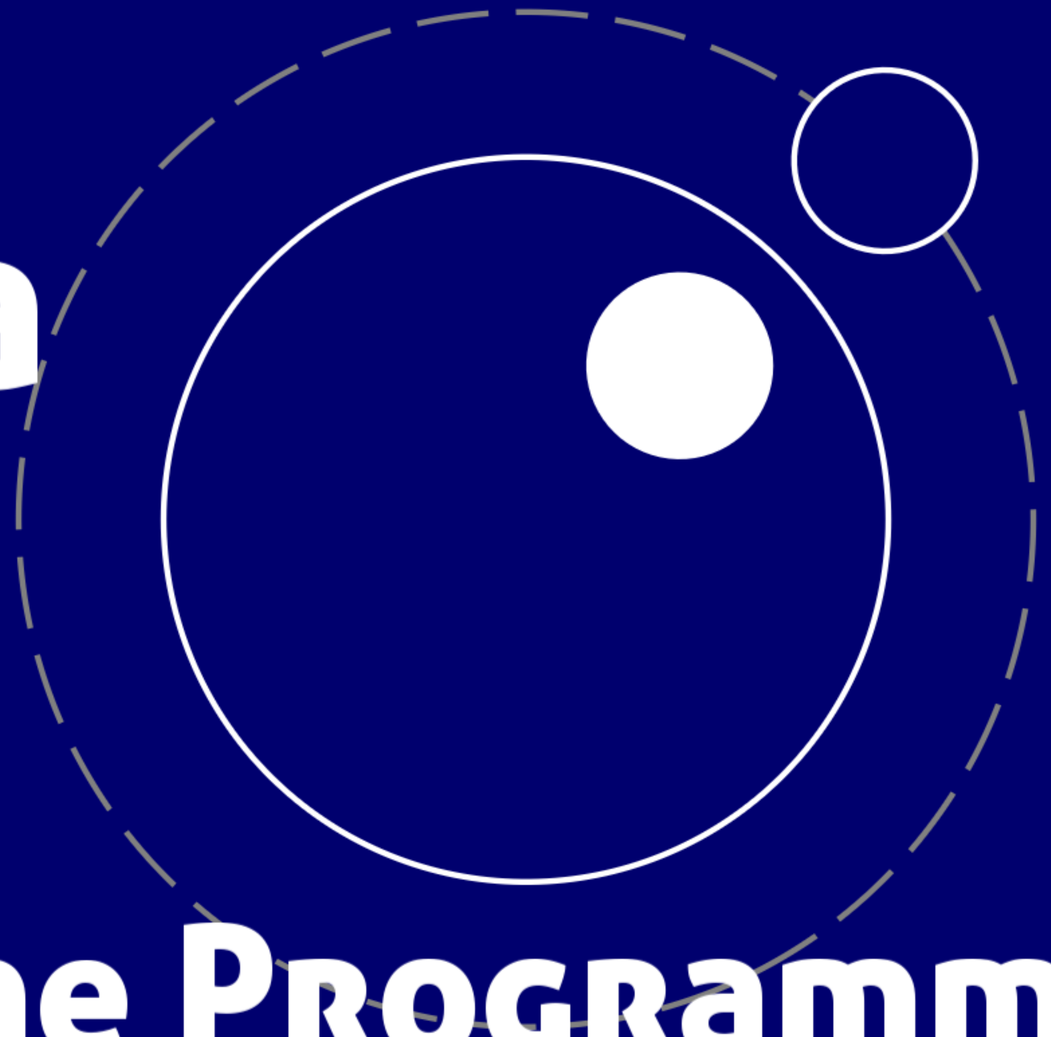
Lua



For

Game Programming

What is Lua?

very small (~200kb)
fast (even faster with LuaJIT)
dynamically typed
readable
easy to embed (written in ANSI C)
very flexible (great for DSL)

Why script at all?

much higher expressiveness

memory safety

faster feedback loop

no recompiling

changing behaviour at runtime

Functions

can return multiple values

can return values of different types

can take a variable amount of arguments

are first class values

have lexical scoping

Functions

returning multiple values of different types:

```
function foo()
    return 1, "two", someOtherFunction
end
```

Functions

taking a variable amount of arguments:

```
function pickUpItems( item, ... )
    if not item then  return  end
    addToInventory( item )
    return pickUpItems( ... )
end
```

Functions

lexical scoping:

```
function makeTween( a, b, easing )
    return function( t )
        return a + (b-a)*easing( t )
    end
end

local tween = makeTween( 0, screen:getHeight(), function( t )
    return t^2
end )
```

Tables

```
t = {}
t[1] = 8
t[2] = "foo"
t[3] = function()
    print"bar"
end
t[4] = { 1, 2, 3 }
```

or

```
t = {
    8,
    "foo",
    function()
        print"bar"
    end,
    { 1, 2, 3 },
}
```

Tables

```
description = {
    [8]      = "the number eight",
    ["foo"] = "the string 'foo'",
    [t[3]]  = "a function that prints 'bar'",
    [t[4]]  = "a table containing 1, 2 and 3",
}
```

Tables

```
description = {
    [8]     = "the number eight",
    foo     = "the string 'foo'",
    [t[3]]  = "a function that prints 'bar'",
    [t[4]]  = "a table containing 1, 2 and 3",
}
```

Tables

```
if key=="escape" then
    <...>
elseif key=="left" then
    <...>
elseif key=="right" then
    <...>
elseif key=="up" then
    <...>
elseif key=="down" then
    <...>
end
```

Tables

```
if key=="escape" then
    <...>
elseif key=="left" then
    <...>
elseif key=="right" then
    <...>
elseif key=="up" then
    <...>
elseif key=="down" then
    <...>
end
```

=>

```
keys = {
    escape = function() <...> end,
    left   = function() <...> end,
    right  = function() <...> end,
    up     = function() <...> end,
    down   = function() <...> end,
}

local action = keys[key]
if action then  action()  end
```

Environments

Easy way to find all global variables:

```
for name, value in pairs( _G ) do
    print( name, value )
end
```

Metatables

override default behaviour

can overload operators

can make tables read-only

can make arbitrary OOP systems

and lots more...

Metatables

making a table read-only:

```
metatable = {
      __index = t,
      __newindex = function()  error"r/o"  end,
      __metatable = false,           -- permanent
}
t = setmetatable( {}, metatable )
```

Metatables

adding descriptions to tables:

```
metatable = { __tostring = function( t )  return t.desc  end }

spoon = { desc = "a shiny spoon" }
spoon = setmetatable( spoon, metatable )

fork = setmetatable( { desc = "a shiny fork" }, metatable )
```

Metatables

creating a basic class:

```
Monster = {}
Monster.__index = Monster
function Monster.new( img, x, y, hp )
    newMonster = { img = img, x = x, y = y, hp = hp }
    return setmetatable( newMonster, Monster )
end
function Monster:draw()
    drawImage( self.img, self.x, self.y )
end
```

Metatables

environments with metatables:

```
function loadBoxed( filename )
    local newEnv = setmetatable( {}, __index = _G )
    loadfile( filename, "t", newEnv )()
    return newEnv
end
```

LuaJIT

the fastest JIT compiler out there

great FFI to C

supports many architechtures

is stuck on Lua 5.1 (with some 5.2 features)

used in LÖVE

What is LÖVE?

baby don't hurt me

don't hurt me

no more

What is LÖVE?

fast, lightweight 2D framework

multiplatform

extensible

nice community

well documented

free

```
function love.load()
    logo = love.graphics.newImage"love.png"
    love.graphics.setBackgroundColor( 0, 0, 110 )
end

function love.draw()
    love.graphics.draw( logo, 400, 100 )
end
```

Thanks for listening!

Twitter: @undefdev