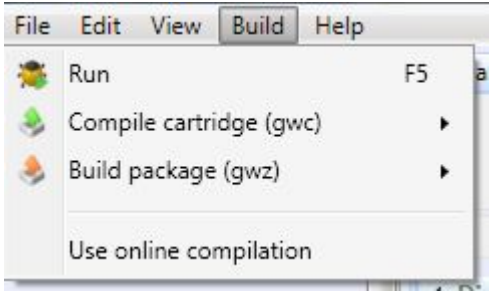
Urwigo y Lua

Documentación



Existen tres tipos de ficheros:

- GWC es el cartucho ya compilado y el que ponemos en nuestro dispositivo para jugar. Se obtiene compilando el GWZ
- GWZ es un fichero ZIP que podemos abrir utilizando un descompresor. En el encontramos un fichero con extensión .lua que es el que contiene el código del cartucho y todos los ficheros auxiliares (imágenes y sonidos) que utilicemos en el juego. Este es el fichero que tenemos que subir a la página de Wherigo.
- GWL es un fichero de log que se genera para usarlo en la depuración de los cartuchos. Para que el fichero se genere, tenemos que tener activada la casilla login del objeto cartucho
- GWS es el fichero que se genera cuando guardamos nuestros cartucho y el que utilizamos para volver al mismo sitio cuando volvemos a ejecutar un cartucho..

Desde Urwigo tenemos 3 opciones compilar el cartucho gwc, esto nos sirve para poder probar nuestro cartucho en un GPS o teléfono antes de subirlo, la compilación la hace Urwigo en nuestro ordenador. Generar el archivo GWZ que es el que tenemos que subir a la página de Wherigo para compartirlo. Usar compilación online genera el GWC pero en lugar de hacerlo en nuestro ordenador lo envía a la página para compilarlo, esta es más real pues es la que posteriormente se distribuirá.

Descompilar un cartucho con gwcd, en linea de comandos: **python gwcd.py --all --output .\salida alerta_zombie.gwc**

¿Qué es Lua?

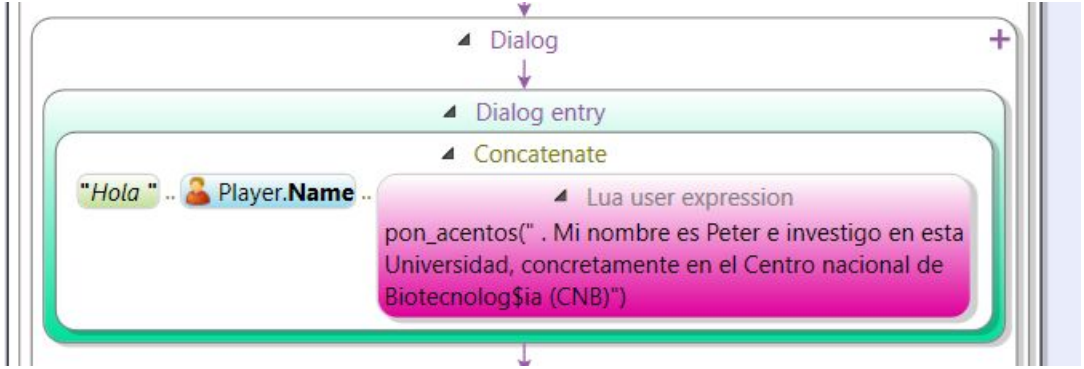
- Lenguaje de programación
- No tipado, las variables pueden contener cualquier cosa, incluso cambiar el tipo de datos que contienen.
- Compilado a un código intermedio llamado Bytecode
- Multiplataforma, si tenemos una máquina virtual para nuestro sistema podemos ejecutar el bytecode correspondiente.
- Aunque no está orientado específicamente a objetos, gracias a sus potentes estructuras de datos, permite crear objetos.
- Robusto y empotrable (se puede integrar como lenguaje de macros dentro de otro programa).
- Todo esto hace que se use en un montón de sitios: ESP8266(esto da para otra charla), Femm, Adobe Lightroom,

¿Qué es Lua? II

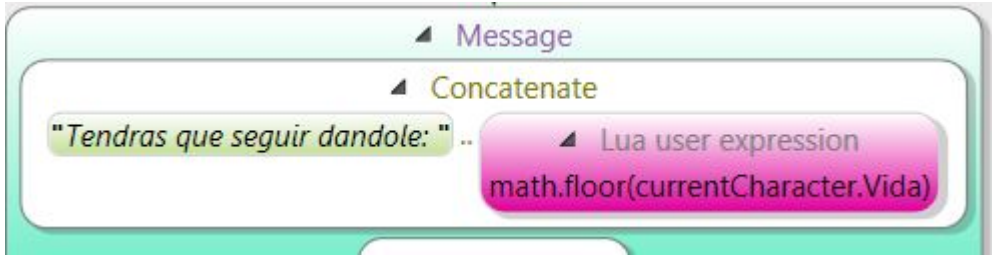
- Para escribir código podemos usar un editor de texto. Notepad++ soporta Lua y separa bloques y colorea el código.
- La ejecución del código la hacemos en urwigo, en el emulador que lleva incorporado.

Integración de Lua y Urwigo

Lua User expression



Lua User expression, es una instrucción de codigo Lua que devuelve un valor. Este lo utilizamos en nuestro programa de Urwigo.
Puede ser el valor retornado por una función, una variable, una operación matemática compleja etc. Es una de las maneras más simples de integrar Lua en Urwigo.



[Ejemplo, partir un cartucho]

Integración de Lua y Urwigo

Lua user code

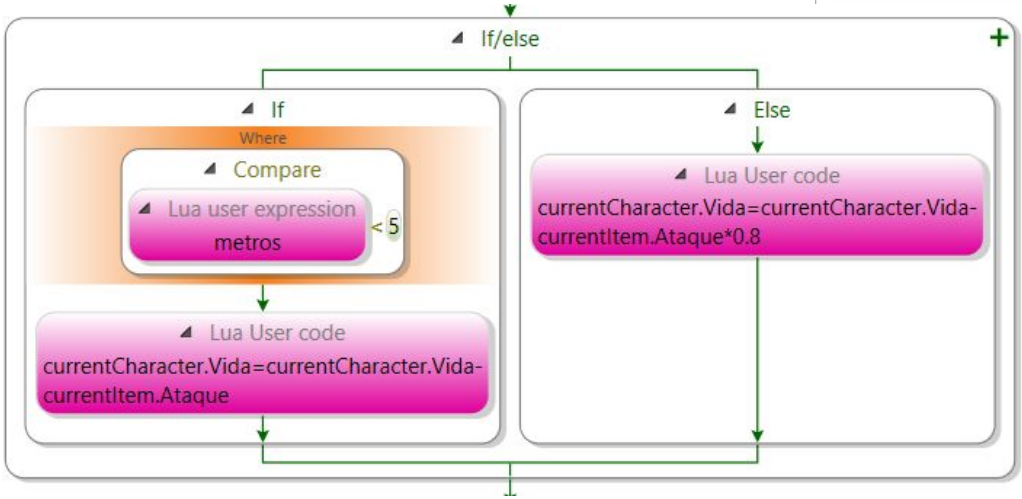
Lua User code

```
--ZOOMBIES TRAS LA EXPLOSION
--INVESTIGADOR ZOMBIE
if objZOna1.Active then
local rumbo=180+objZOna1.CurrentBearing
if  (objProtegido ==true) and
(objZOna1.CurrentDistance:GetValue()<10)
then MueveZona(objZOna1,rumbo-90,1)
else MueveZona(objZOna1,rumbo,1)
end
end
--ZOMBIE ERRANTE
if objZoombie2.Active then
local rumbo=180+objZoombie2.CurrentBearing
if  (objProtegido ==true) and
(objZoombie2.CurrentDistance:GetValue()<10)
then MueveZona(objZoombie2,rumbo-90,1)
else MueveZona(objZoombie2,rumbo,1)
end
end

if objZoombie3.Active then
local rumbo=180+objZoombie3.CurrentBearing
if  (objProtegido ==true) and
(objZoombie3.CurrentDistance:GetValue()<10)
then MueveZona(objZoombie3,rumbo-90,1)
else MueveZona(objZoombie3,rumbo,1)
end
end

if objZoombie4.Active then
local rumbo=180+objZoombie4.CurrentBearing
if  (objProtegido ==true) and
(objZoombie3.CurrentDistance:GetValue()<10)
then MueveZona(objZoombie4,rumbo-90,1)
else MueveZona(objZoombie4,rumbo,1)
end
end
```

En este caso, no se devuelve nada, simplemente se ejecuta el código como si fuera una función. Desde aquí podemos llamar a otras funciones o ejecutar el código que necesitemos.

Lua User code

```
local newZp = Player.ObjectLocation
local dist = Wherigo.Distance(5, 'm')--separacion desde
centro
local rumbo= math.random(1,360)
--Wherigo.TranslatePoint(newZp, dist, rumbo)
objZoombie3.OriginalPoint=Wherigo.TranslatePoint
(newZp, dist, rumbo)
rumbo= math.random(1,360)
objZoombie4.OriginalPoint=Wherigo.TranslatePoint
(newZp, dist, rumbo)
```



Integración de Lua y Urwigo

Lua user directives

Abrimos la ventana del editor pulsando en View/Lua User Directives.
Todo lo que pongamos en este editor irá colocado al principio del fichero Lua que se genera al compilar.
Sirve fundamentalmente para crear e iniciar variables globales o hacer los require.

Integración de Lua y Urwigo

Lua user functions

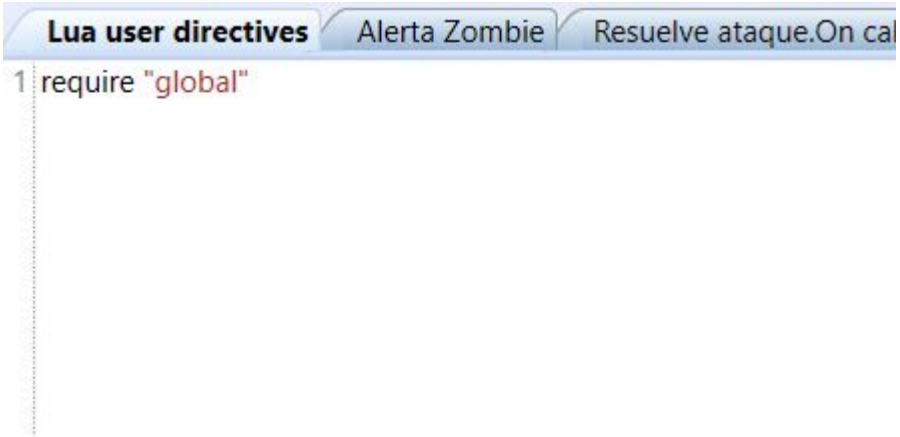
Alerta Zombie | **Lua user functions**

```lua
nombres ={}
nombres["AlfonsoyRosa"]="Alfonso"
nombres["MiTesoro"]="Mariano"
nombres["Mr. Smith"]="Mariano"

diametro=5
Velocidad_Explosion=20

function MueveZona(zZone,rumbo,distancia)
--zZone.Visible = false
zZone.Active = false
local point=zZone.OriginalPoint
local dist = Wherigo.Distance(distancia, 'm')
local newZp = Wherigo.TranslatePoint(point, dist, rumbo)   -- new center point for the new zone
dist = Wherigo.Distance(2, 'm')--separacion desde centro
local pts = {
      Wherigo.TranslatePoint(newZp, dist, 45),
      Wherigo.TranslatePoint(newZp, dist, 135),
      Wherigo.TranslatePoint(newZp, dist, 225),

 }
--ver como esta la zona y dejarla igual

zZone.OriginalPoint=newZp
zZone.Points = pts
zZone.Active = true
--zZone.Visible = true
end

function CreaZona(zZone,diametro)
--zZone.Visible = false
zZone.Active = false
local point=zZone.OriginalPoint --centro sigue igual
local dist = Wherigo.Distance(diametro, 'm') --creo una variable distancia
```

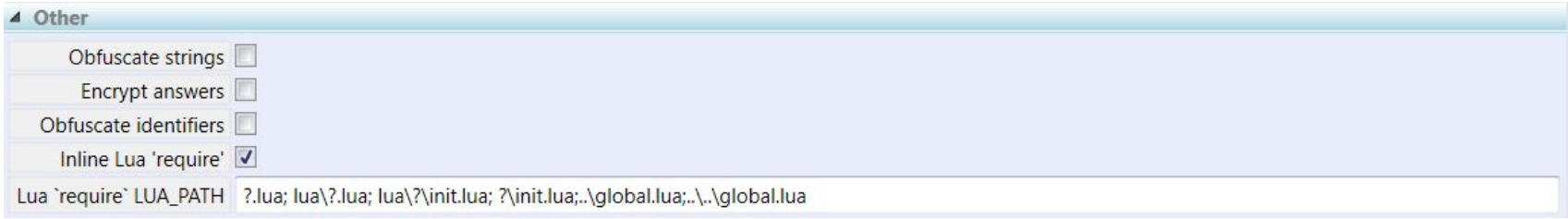
Abrimos el editor en view/lua user functions.
Aquí podemos escribir nuestras propias funciones y objetos que luego podemos llamar utilizando Lua User Expresion o Lua User Code

Integración de Lua y Urwigo

Require

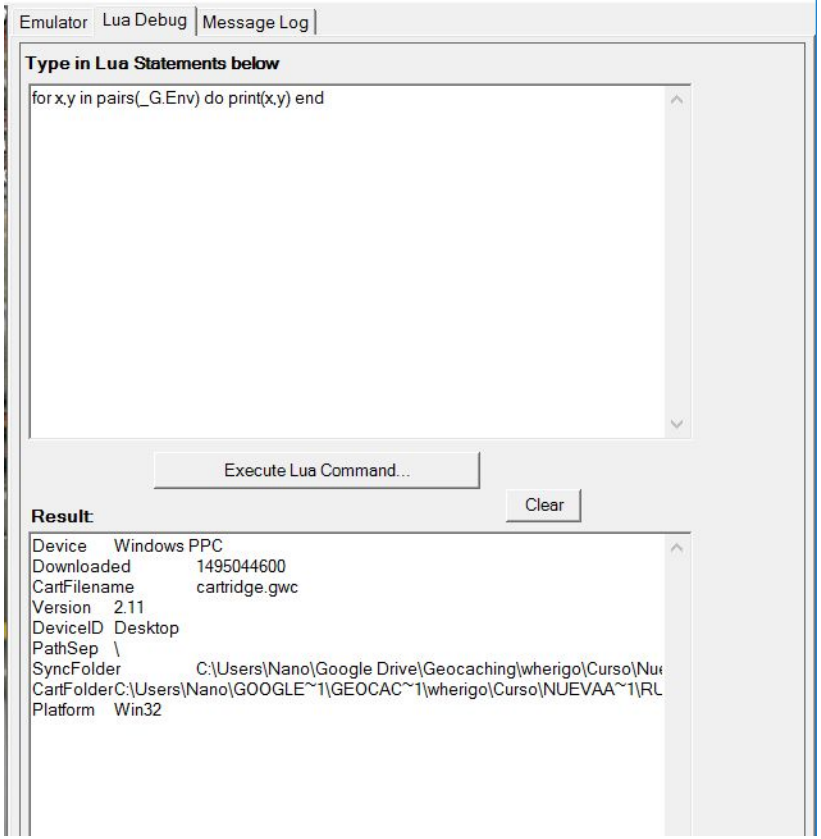


En este caso, se trata de incluir un fichero externo en el cual tendremos funciones u objetos que queremos incluir en nuestro código y que queremos reutilizar en varior cartuchos.
Tenemos que definir correctamente el path a nuestro fichero.
Podemos incluir el fichero directamente aquí.
O podemos hacer un require independiente usando Lua User Require.



[Ejemplo Acentos]

Lua Debug




```
print("Hola Mundo")
print(objEnergia)
for x,y in pairs(_G) do print(x,y) end
for x,y in pairs(Wherigo) do print(x,y) end
```

```
local o = cartridge;
print(o);
print(getmetatable(o));
for k,v in pairs(getmetatable(o)) do print(k,v)
end;
print(getmetatable(o)._self);
for k,v in pairs(getmetatable(o)._self) do
print(k,v) end;
```

Objetos Globales

- cartridge Objeto de tipo ZCartridge cotiene os datos del cartucho que estamos jugando.
- Player Objeto de tipo ZCharacter, contiene los datos del jugador actual.
- Wherigo contine funciones que se utilizan en programación
- Env contine datos del entorno en el que se está ejecutando.
- WIGInternal Contiene funciones para programar.

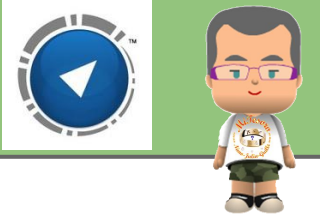
ENV

Device Windows PPC
Downloaded 1495044600
CartFilename cartridge.gwc
Version 2.11
DeviceID Desktop
PathSep \
SyncFolder C:\Users\Nano\Google Drive\Geocaching\wherigo\Curso\Nueva AlertaZombie\run-cf281744-3037-4d38-b078-b654e849447f\
CartFolder C:\Users\Nano\GOOGLE~1\GEOCAC~1\wherigo\Curso\NUEVAA~1\RUN-CF~1\
Platform Win32

Objeto Wherigo

Dialog function: 0D799A80
PlayAudio function: 0D7999C0
ShowStatusText function: 0D7999E0
Player a ZCharacter instance
VectorToZone function: 0D79A780
TranslatePoint function: 0D79A740
NoCaseEquals function: 0D79A6C0
_NAME Wherigo
Distance a Distance class

Objetos WIGInternal



GetInput function: 0D79A7E0
MediaEvent function: 0D79A820
ShowStatusText function: 0D79A7C0
LogMessage function: 0D79A840
TimerEvent function: 0D799D00
CommandChangedEvent function: 0D799A40
EnginePrint function: 0D799B60
IsPointInZone function: 0D79A680
InventoryEvent function: 0D799BC0
NotifyOS function: 0D799CE0
AttributeChangedEvent function: 0D799A60
VectorToPoint function: 0D79A720
CartridgeEvent function: 0D799C60
MessageBox function: 0D79A760
TranslatePoint function: 0D79A740

Funciones Wherigo

MessageBox(table)
Dialog(table)
PlayAudio(media)
ShowStatusText(text)
VectorToZone(point, zone)
VectorToSegment(point, p1, p2)
IsPointInZone(point, zone)
TranslatePoint(point, distance, bearing)
VectorToPoint(p1, p2)
NoCaseEquals(s1,s2)
Command(command) SaveClose, DriveTo, Stop-Sound, Alert
LogMessage(table)
GetInput(input)
ShowScreen(screen, item)

Ejemplos:

```
dist = Wherigo.Distance(2, 'm')
local newZp = Wherigo.TranslatePoint(point, dist, rumbo)
```

Objetos

Bearing(value) Objeto que repesenta un azimut
Distance(value, units) Objeto para distancias
ZCommand(table) Comandos
ZReciprocalCommand
ZonePoint(lat, lon, alt) objeto para puntos
ZObject Objeto del cual heredan los demás
Zone Objeto zona, está compuesto de puntos
ZCartridge Objeto Cartucho
ZMedia Objeto para ficheros de imagen y sonido
ZItem Objeto para items
ZTask Objeto para tareas
ZTimer Objeto para timers
ZInput Objeto para entradas
ZCharacter Objeto para personajes

ZonePoint

A ZonePoint is probably intended to represent a vertex of a zone. It is also used to represent arbitrary points. It contains latitude, longitude and altitude. Two ZonePoints could also be compeared if latitude, longitude and altitude are equal.

An INVALID_ZONEPOINT is defined as ZonePoint(360,360,360).

Properties

altitude	Distance. The altitude of the pointin meters.
latitude	Number. The latitude of the point in decimal degrees.
longtitude	Number. The longitude of the point in decimal degrees.

Methods

Clone	Returns ZonePoint. Creates a copy of the ZonePoint object.
Constructor	Wherigo.ZonePoint(latitude, longitude, altitude)

Cartridge

An instance of the ZCartridge object represents a cartridge. There is only one instance of ZCartridge, and it represents the current cartridge. The .lua file should return the single Cartridge instance as its final statement.
Properties are used for development with the Wherigo Builder, for in game informations and for updating the website.
ZCartridge has all Properties, events and functions, which ZObject has.

Properties
ZVariables Table. Association of user-defined variables created in Builder, linking variable name to initial values. All variables, which are in this table will be saved, if the game is saved. It is enough to make ZVariables[<variable name>] = true to save the variable with name <variable name>.

Methods
table GetAllOfType(ZCartridge self, string type)
Returns a table of all ZObject of the given type.

No insertar --[[ ]] no compila luego http://jakuje.dta3.com/webwig/luadoc/index.html#Tables