Populous Reincarnated

A forum for all populous players.
https://www.popre.net/forum/

Scripting in Populous Beta: LUA tutorial

https://www.popre.net/forum/viewtopic.php?f=25&t=11753

Page **1** of **1**

Scripting in Populous Beta: LUA tutorial

Posted: **Thu May 28, 2020 8:31 pm**
by **410172_Chief**
With the large amount of time that went into the creation of the four new beta tribes, I thought it was about time to look into how we should script them.
So here you have it, the first ever LUA scripting tutorial! In this tutorial, you will learn the basics of scripting beta tribes but also learn to do some more complex and fun stuff!

A few notes:

- Quite a part of the LUA code is similar to what you're used to from PopScript, and basics scripts will not be much harder to write. A little experience in PopScript and programming in general is useful, but not required if you're willing to learn.
- Where LUA adds more things we can do in game, it also adds complexity. Try to understand the code you're writing, instead of just copy&paste. Take your time. There is a scripting room in Discord where you can ask questions and share code, and of course we still have the forums.
- This tutorial is still a work in progress. Leave your feedback and I'll be able to do some improvements where needed. If there are any questions, ask!

Having said that, I'd like to present the index of this tutorial:

Tools

Creating 8p in the world editor

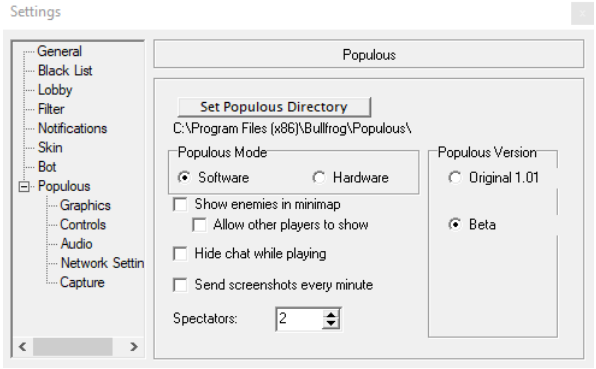
In-game script management

LUA Scripting: The basics
Code to be executed at the beginning of the game
Initiation of the computer player
Setting the AI attributes
Setting attributes in a loop
Setting internal attributes and states
Setting player alliances
Setting spell and marker entries
Using comments in your scripts
Code to be executed during the game
Writing every loops in the LUA script
Basic NAV_CHECK and ATTACK commands
Case: Basic attacks and patrolling

LUA Scripting: Advanced scripting
Create a thing. Anything. (createThing())
AI map analysis (SearchMapCells())
Counting objects during a search (ProcessList())
Using functions
Counting all objects on the map (ProcessGlobalTypeList())
Intelligent conversion of wildmen
Case: Improved shaman-only attacks

Tools
Download the tools from my Drive here (some are too large to attach to this post): https://drive.google.com/file/d/1Q4ULuB ... sp=sharing

- For designing, we will use the 8p world editor.
- For scripting: a code editor of your liking, I use Visual Studio 2019. We will also use a few prewritten LUA scripts to make life easier once we get to scripting.
  Put these scripts in your scripts folder: C:\Program Files (x86)\Bullfrog\Populous\Scripts
- The beta version of Populous!
  If you launch a beta game in MM, the latest version of Populous Beta will automatically be installed on your computer. Select Beta under Populous --> Version in Settings --> Populous.

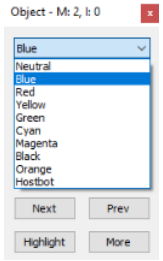


- The tutorial case files.
  During the tutorial, I will work with a case map. If you follow this tutorial from beginning to end, you will have scripted a fully functional AI for the case map, which allows for some challenging gameplay! Rename the "Case.dat" file to "levl2001.dat" and place it in your levels folder: C:\Program Files (x86)\Bullfrog\Populous\levels. Create an empty file "Case.lua" in your scripts folder.

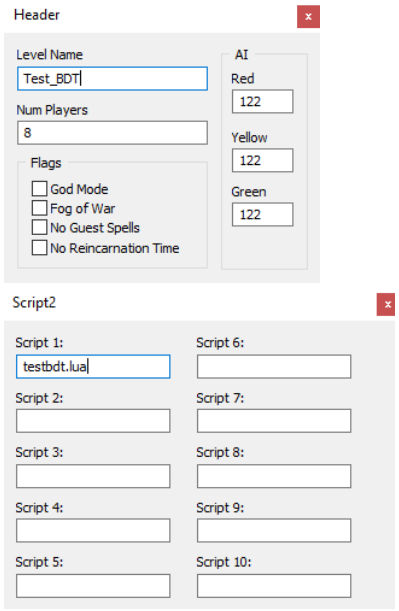
I also submitted the case map as a test map for online play, so you can experience the competition that we are going to script. Give it a spin if you like, be sure to set "Computer Players" to "Yes" and to set the building and spell restrictions for yourself.

Creating 8p in the world editor
The 8-player supported world editor is based off ALACN's famous world editor, with some tweaks. If you're new to mapmaking, you can use another tutorial to get a little bit familiar with the program.
Here, I will discuss the few things that make the 8-player editor special.

You're right. It's the availability of 8 tribes to choose from in the object list. When you add an object, you can choose Cyan, Magenta, Black or Orange as the owner. One thing you may notice is that in the world editor, their masks will be the same ones as for the four original tribes. That don't matter, they will have their own masks in game.
Go ahead and make a simple map with some beta troops in it. Don't forget the Shamans!

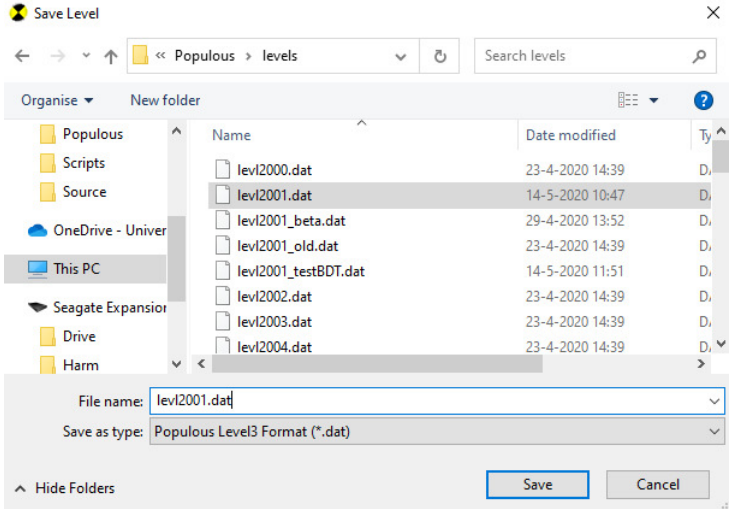


Besides adding the beta tribes in the design, we need to attach the LUA scripts to the map just as we are used to attaching PopScript files to a map. However, if you head to Edit --> Header --> General, there is no entry for any of the beta tribes. These entries are for the PopScript files only, and so only for Red, Yellow and Green. To attach a LUA script, we go to Edit --> Header --> Script2, and enter the name of our LUA script (include the extension). You can attach up to 10 different LUA scripts to your map. Pretty cool right?

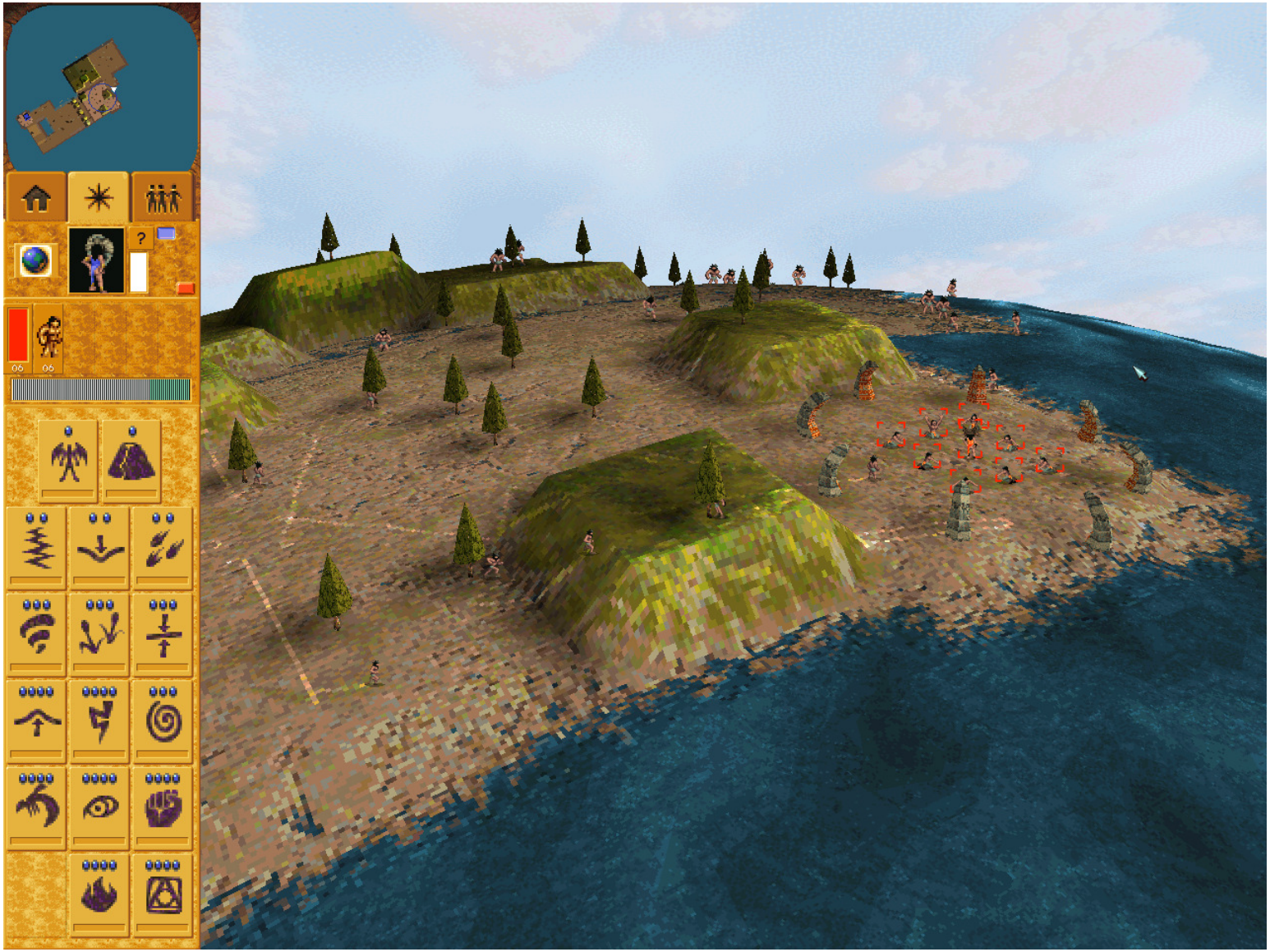


Make sure to put a script with the exact same name that you just entered in your scripts folder.

Finally, let's save our map and put in the levels directory to test it. The first time you save your map, use File --> Save As and save it as a Populous Level3 Format (*.dat). The name of the first map in the game is levl2001.dat. If you opened the world editor as administrator, you can directly save to the level directory.



Now open up Populous Beta and see if you got everything right so far. When starting level 1, you should see your level with some beta AI present. They don't do much yet. If you added braves or wildmen around the AI shaman, you will see them worshipping their leader like this:

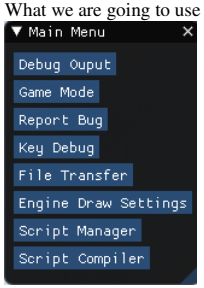


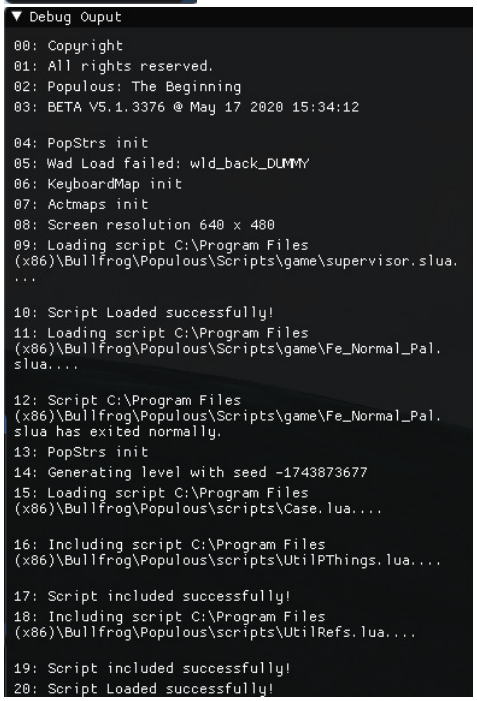
We can also load the script testbdt.lua by starting the Test_BDT level, opening the chat using TAB+F11 and typing “LUA testbdt.lua”. It is easy to do, but not automatic.

Some scripting references that may be useful:

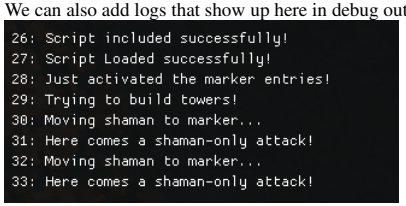
LUA Syntax reference:
http://www.populous3.info/script3_doc/
PopScript Syntax reference: https://ts.popre.net/archive/Downloads/ ... p_File.htm
PopScript Scripting guides: https://www.youtube.com/watch?v=LKVmnB1 ... ulo9kX_yLo and viewtopic.php?f=25&t=11190

In-game script management
There are some in-game features that can help you in script writing and testing. You can access these via the beta game menu (Shift+F1).
What we are going to use the most is the Debug Output. However, the Script Manager and Script Compiler are also things you should check out.



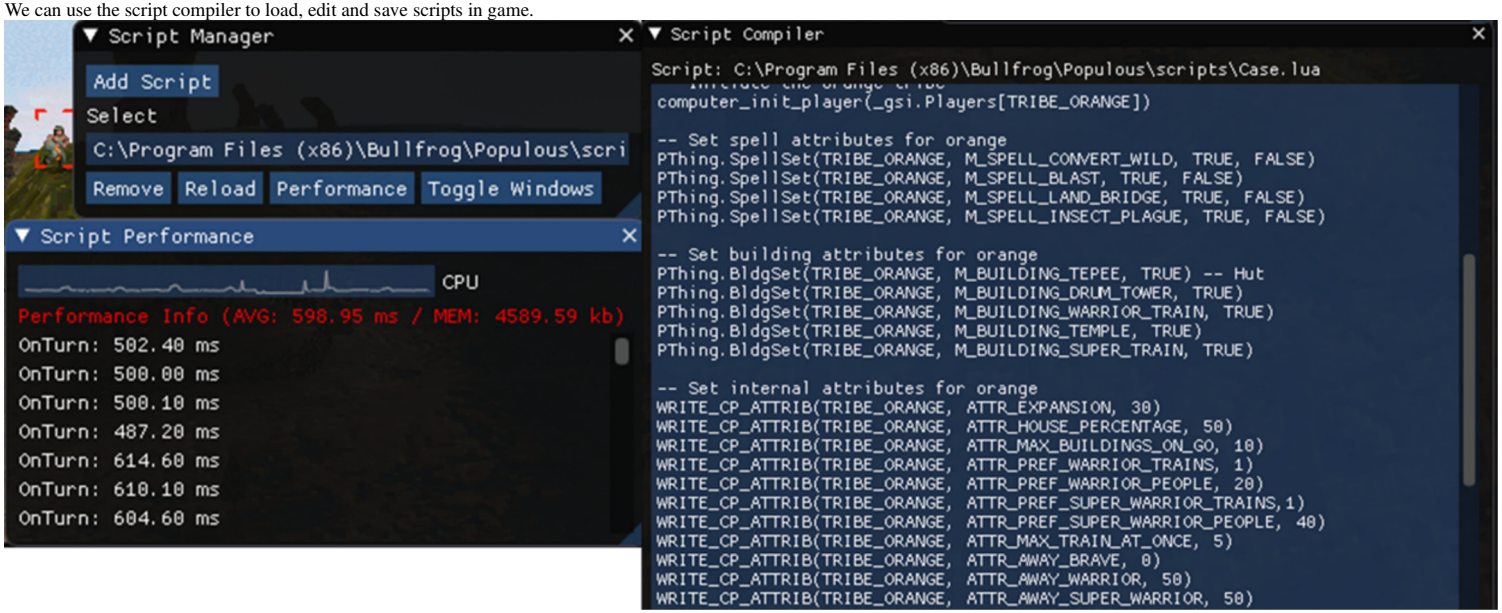


When you open the Debug Output, and resize it a bit to your liking, you will see something like this. This window keeps track of what's going on in the game's mind.
When we load a script, we will get notified about any errors. If there are none, you will see "Script Loaded successfully!"
We can also add logs that show up here in debug output. In that way we can see when a certain part of our script is called.

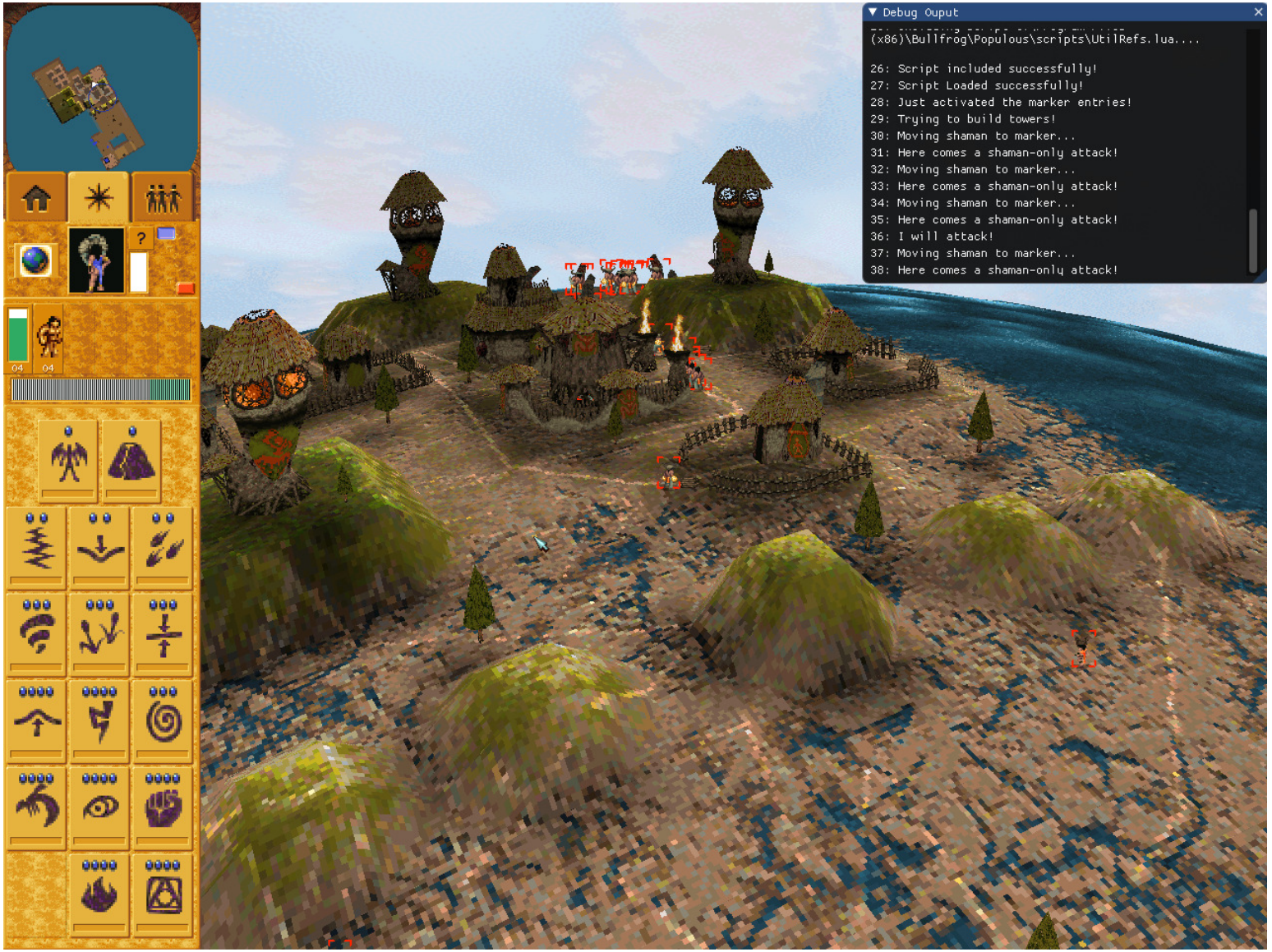


We are going to learn more about logs later on.
Using the script manager, we can load scripts and check their performance.

We can use the script compiler to load, edit and save scripts in game.



When I script a level, I mostly have Visual Studio open along with the Populous Beta. When you run Visual Studio as administrator, you can directly edit your scripts in the Scripts folder. Scripts run fine in Populous even when they are also opened by Visual Studio. If you make changes to a script in Visual Studio while it is running in Populous, make sure to save and then reload the level or script in game. Also, keep Debug Output open in a small window to keep track of errors and logs.



Ready to start with your first LUA scripts? Let's continue with the scripting basics straight away!

LUA Scripting: The basics
The basic scripting syntax in LUA is quite similar to that of PopScript. In this part of the tutorial we will make the AI build and do some basic attacks. Using LUA, of course.
If you are not familiar with PopScript, don't worry. Pay close attention. Some basic PopScript knowledge will come in handy though. Also some basic programming knowledge is useful.
If you'd like to educate yourself, there are many decent tutorials out there.

Let's get started. Remember the old Populous script? It started pretty simple.

```
{
```

```
}
SCRIPT_END
```

In most of the cases, we would add an if-statement so that we could add code to be executed at the beginning of the game, or during the game:

```
{
        IF (INT_GAME_TURN == 0)
{
                Code to be executed at the beginning of the game
}
ELSE
{
        Code to be executed during the game
}
ENDIF
}
SCRIPT_END
```

In LUA, we kinda do the same thing using a function OnTurn():

```
function OnTurn()
  Code here
end
```

The code within this function is executed every gameturn (1/12 seconds).

In practice, most of our scripts will look something like this:

```
Code to be executed at the beginning of the game
function OnTurn()
        Code to be executed during the game
end
```

Not so difficult, right? We only need to know how to write all our statements in correct LUA syntax.
Yes. That's basically it. However, we do need to allow the script to access some references. The beta programmers did the most amount of work here, and we are just going to make use of that.

Code to be executed at the beginning of the game
There is a lot of code written that helps the game to interpret our LUA scripts. This is all written in so called modules. We're not going to look at them. You do need to import those into your script using:

```
import(Module_System)
import(Module_Players)
import(Module_Defines)
import(Module_PopScript)
import(Module_Game)
import(Module_Objects)
import(Module_Map)
```

We also include two prewritten scripts that contain the definition of functions we're going to use:

```
include("UtilPThings.lua")
include("UtilRefs.lua")
```

So as a rule, every script that we will write starts with those lines.
There are more modules out there, but we won't need them for now. What we do need are the two scripts we included. Make sure you place the UtilPThings.lua and UtilRefs.lua scripts in your Scripts folder.

That is the start of your script. Step by step we will add more functionality to it.

SPOILER: HIDE

```
import(Module_System)
import(Module_Players)
import(Module_Defines)
import(Module_PopScript)
import(Module_Game)
import(Module_Objects)
import(Module_Map)
include("UtilPThings.lua")
include("UtilRefs.lua")

function OnTurn()

end
```

Initiation of the computer player
One of the most important lines of your script. Without this line, the AI will remain absent no matter how much lines of code your script for them.

```
computer_init_player(_gsi.Players[MY_TRIBE])
```

MY_TRIBE is the TRIBE indentifier, which is the computer tribe. Choose from TRIBE_CYAN, TRIBE_PINK, TRIBE_BLACK, TRIBE_ORANGE for beta tribes. (And there's TRIBE_RED, TRIBE_YELLOW, TRIBE_GREEN as well.)

Setting the AI attributes
Instead of using a separate game file listing the AI attributes, we now define the attributes for beta tribes in our script. We can set the attributes by using the SpellSet and BldgSet functions, which are defined in the UtilPThings.lua script.

Say we want the AI to use a spell, we can use the syntax:

```
PThing.SpellSet(MY_TRIBE, M_SPELL, TRUE, FALSE)
```

M_SPELL is the spell identifier which can be different for each spell. You may remember them from PopScript. If not, here's a list of all the spell identifiers:

SPOILER: HIDE

```
M_SPELL_GHOST_ARMY
M_SPELL_BLAST
M_SPELL_CONVERT_WILD
M_SPELL_INSECT_PLAGUE
M_SPELL_INVISIBILITY
M_SPELL_SHIELD
M_SPELL_LAND_BRIDGE
M_SPELL_LIGHTNING_BOLT
M_SPELL_HYPNOTISM
M_SPELL_WHIRLWIND
M_SPELL_SWAMP
M_SPELL_FLATTEN
M_SPELL_EARTHQUAKE
M_SPELL_EROSION
M_SPELL_FIRESTORM
M_SPELL_ANGEL_OF_DEATH
M_SPELL_VOLCANO
```

Say we want to give the orange tribe the knowledge of blast spells, then we use:

```
PThing.SpellSet(TRIBE_ORANGE, M_SPELL_BLAST, TRUE, FALSE)
```

For buildings, we use the syntax:

```
PThing.BldgSet(MY_TRIBE, M_BUILDING, TRUE)
```

Which works in the same way. Say we want the orange tribe to be able to build huts:

```
PThing.BldgSet(TRIBE_ORANGE, M_BUILDING_TEPEE, TRUE)
```

Here are some more building identifiers you can choose from:

SPOILER: HIDE

```
M_BUILDING_TEPEE
M_BUILDING_DRUM_TOWER
M_BUILDING_WARRIOR_TRAIN
M_BUILDING_TEMPLE
M_BUILDING_SUPER_TRAIN
M_BUILDING_SPY_TRAIN
M_BUILDING_BOAT_HUT_1
M_BUILDING_AIRSHIP_HUT_1
```

As an example: below is the code we would use to attribute Orange with 4 spells and 4 buildings:

```
PThing.SpellSet(TRIBE_ORANGE, M_SPELL_CONVERT_WILD, TRUE, FALSE)
PThing.SpellSet(TRIBE_ORANGE, M_SPELL_BLAST, TRUE, FALSE)
PThing.SpellSet(TRIBE_ORANGE, M_SPELL_LAND_BRIDGE, TRUE, FALSE)
PThing.SpellSet(TRIBE_ORANGE, M_SPELL_INSECT_PLAGUE, TRUE, FALSE)
PThing.BldgSet(TRIBE_ORANGE, M_BUILDING_TEPEE, TRUE)
PThing.BldgSet(TRIBE_ORANGE, M_BUILDING_DRUM_TOWER, TRUE)
PThing.BldgSet(TRIBE_ORANGE, M_BUILDING_WARRIOR_TRAIN, TRUE)
PThing.BldgSet(TRIBE_ORANGE, M_BUILDING_SUPER_TRAIN, TRUE)
```

Setting attributes in a loop
If there are a lot of spells and buildings to be used, we can also but those in array and use a for loop to do the trick. Below is the code that does the same as the 8 lines we saw above.

```
botSpells = {M_SPELL_CONVERT_WILD,
                    M_SPELL_BLAST,
                    M_SPELL_LAND_BRIDGE,
                    M_SPELL_INSECT_PLAGUE
}
botBldgs = {M_BUILDING_TEPEE,
                   M_BUILDING_DRUM_TOWER,
                   M_BUILDING_WARRIOR_TRAIN,
                   M_BUILDING_SUPER_TRAIN
}

for u,v in ipairs(botSpells) do
    PThing.SpellSet(TRIBE_ORANGE, v, TRUE, FALSE)
end

for y,v in ipairs(botBldgs) do
    PThing.BldgSet(TRIBE_ORANGE, v, TRUE)
end
```

In either way the orange tribe will now be able to build huts, towers, warrior training huts and fw huts. The orange Shaman will be able to use the spells Convert, Blast, LandBridge and Swarm.

Are you still with me? Great. Let's see what our total script looks like now:

SPOILER: HIDE

```
import(Module_System)
import(Module_Players)
import(Module_Defines)
import(Module_PopScript)
import(Module_Game)
import(Module_Objects)
import(Module_Map)
include("UtilPThings.lua")
include("UtilRefs.lua")

computer_init_player(_gsi.Players[TRIBE_ORANGE])

PThing.SpellSet(TRIBE_ORANGE, M_SPELL_CONVERT_WILD, TRUE, FALSE)
PThing.SpellSet(TRIBE_ORANGE, M_SPELL_BLAST, TRUE, FALSE)
PThing.SpellSet(TRIBE_ORANGE, M_SPELL_LAND_BRIDGE, TRUE, FALSE)
PThing.SpellSet(TRIBE_ORANGE, M_SPELL_INSECT_PLAGUE, TRUE, FALSE)
PThing.BldgSet(TRIBE_ORANGE, M_BUILDING_TEPEE, TRUE)
PThing.BldgSet(TRIBE_ORANGE, M_BUILDING_DRUM_TOWER, TRUE)
PThing.BldgSet(TRIBE_ORANGE, M_BUILDING_WARRIOR_TRAIN, TRUE)
PThing.BldgSet(TRIBE_ORANGE, M_BUILDING_SUPER_TRAIN, TRUE)

function OnTurn()

end
```

Setting internal attributes and states
We can set internal attributes and states for AI in LUA in a similar way as were used to from PopScript. Take a look at a few examples shown below.

PopScript:

```
SET INT_ATTR_HOUSE_PERCENTAGE X
```

LUA script:

```
WRITE_CP_ATTRIB(TRIBE, ATTR_HOUSE_PERCENTAGE, X)
```

PopScript:

```
DO STATE_CONSTRUCT_BUILDING ON
```

LUA Script:

```
STATE_SET(TRIBE, TRUE, CP_AT_TYPE_CONSTRUCT_BUILDING)
```

The functions that in PopScript start with “SET”, have their own function in LUA:
PopScript:

```
DO SET_BUCKET_USAGE ON
```

LUA Script:

```
SET_BUCKET_USAGE(TRIBE, TRUE)

SET_BUCKET_COUNT_FOR_SPELL(TRIBE, M_SPELL_BLAST, X)
SET_DRUM_TOWER_POS(TRIBE, X, Z)
```

Of course, there are many, many more internal attributes and states that you will need for building, training and attacking. I will show you a few we are going to use in our script for the orange tribe.

SPOILER: HIDE

```
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_EXPANSION, 30)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_HOUSE_PERCENTAGE, 50)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_MAX_BUILDINGS_ON_GO, 10)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_PREF_WARRIOR_TRAINS, 1)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_PREF_WARRIOR_PEOPLE, 20)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_PREF_SUPER_WARRIOR_TRAINS,1)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_PREF_SUPER_WARRIOR_PEOPLE, 40)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_MAX_TRAIN_AT_ONCE, 5)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_AWAY_BRAVE, 0)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_AWAY_WARRIOR, 50)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_AWAY_SUPER_WARRIOR, 50)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_AWAY_RELIGIOUS, 0)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_AWAY_SPY, 0)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_AWAY_MEDICINE_MAN, 100)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_MAX_ATTACKS, 999)
STATE_SET(TRIBE_ORANGE, TRUE, CP_AT_TYPE_FETCH_WOOD)
STATE_SET(TRIBE_ORANGE, TRUE, CP_AT_TYPE_CONSTRUCT_BUILDING)
STATE_SET(TRIBE_ORANGE, TRUE, CP_AT_TYPE_TRAIN_PEOPLE)
STATE_SET(TRIBE_ORANGE, TRUE, CP_AT_TYPE_AUTO_ATTACK)
STATE_SET(TRIBE_ORANGE, TRUE, CP_AT_TYPE_POPULATE_DRUM_TOWER)
STATE_SET(TRIBE_ORANGE,TRUE, CP_AT_TYPE_MED_MAN_GET_WILD_PEEPS)
```

Do you still remember what all these lines do?
If not, take a look at the PopScript tutorials and get familiar with these lines. You’ll need to.

I’m not feeling that main drum tower today. Let’s not build one.

```
DELAY_MAIN_DRUM_TOWER(TRUE,TRIBE_ORANGE)
```

Before I show you the code so far, let’s discuss two last important things we want to set at the beginning of the game.

Setting player alliances
The piece of code below sets the alliances of the players for the level. Are you all ready for an offline dictator game against AI?

```
set_players_allied(TRIBE_1,TRIBE_2)
```

Make sure you ally both ways, so that you don’t get betrayed by your ally:

```
set_players_allied(TRIBE_1,TRIBE_2)
set_players_allied(TRIBE_2,TRIBE_1)
```

We’re not going to use this in our case today, but you can try it out on your own.
Lastly, we will look at how spell and marker entries are defined in LUA.

Setting spell and marker entries
Setting spell and marker entries in LUA is done in a very similar way compared to PopScript:

PopScript:

```
DO SET_SPELL_ENTRY ENTRY_NO M_SPELL M_SPELL_COST FREQ NUM_PEOPLE BASE
```

LUA Script:

```
SET_SPELL_ENTRY(MY_TRIBE, ENTRY_NO, M_SPELL, M_SPELL_COST, FREQ, NUM_PEOPLE, BASE)
```

PopScript:

```
 DO SET_MARKER_ENTRY ENTRY_NO MK1 MK2 NO_BRAVES NO_WARRIORS NO_FW NO_PREACHERS
```

LUA Script:

```
SET_MARKER_ENTRY(MY_TRIBE, ENTRY_NO, MK1, MK2, NO_BRAVES, NO_WARRIORS, NO_FW, NO_PREACHERS)
```

Let's use a few of these in our script for the orange tribe, just so you can see that they work.

```
SET_SPELL_ENTRY(TRIBE_ORANGE, 0, M_SPELL_INSECT_PLAGUE, 25000, 64, 3, 0)
SET_SPELL_ENTRY(TRIBE_ORANGE, 1, M_SPELL_INSECT_PLAGUE, 25000, 64, 3, 1)
```

These spell entries allow the orange shaman to use the swarm spell both offensively and defensively.

```
SET_MARKER_ENTRY(TRIBE_ORANGE, 0, 0, 0, 0, 0, 2, 0)
SET_MARKER_ENTRY(TRIBE_ORANGE, 1, 1, 1, 0, 0, 2, 0)
```

These marker entries will allow two firewarriors to patrol at marker 0 and two at marker 1.
Remember we do still need to activate them using:

```
MARKER_ENTRIES(TRIBE_ORANGE, 0, 1, -1, -1)
```

Using comments in your scripts
Before we move on, lets learn how to add comments to our script. This way, we can keep everything a little organized. We can add a comment line by using "- - ". For example, I start our script for the orange tribe with:

```
-- LUA Test script for orange tribe in SP.
```

Add comments to your liking to prevent yourself from getting lost in your own writing.

Our script so far:

SPOILER: HIDE

```
-- LUA Test script for orange tribe in SP.

-- Import modules and include files
import(Module_System)
import(Module_Players)
import(Module_Defines)
import(Module_PopScript)
import(Module_Game)
import(Module_Objects)
import(Module_Map)
include("UtilPThings.lua")
include("UtilRefs.lua")

-- Initiate the orange tribe
computer_init_player(_gsi.Players[TRIBE_ORANGE])

-- Set spell attributes for orange
PThing.SpellSet(TRIBE_ORANGE, M_SPELL_CONVERT_WILD, TRUE, FALSE)
PThing.SpellSet(TRIBE_ORANGE, M_SPELL_BLAST, TRUE, FALSE)
PThing.SpellSet(TRIBE_ORANGE, M_SPELL_LAND_BRIDGE, TRUE, FALSE)
PThing.SpellSet(TRIBE_ORANGE, M_SPELL_INSECT_PLAGUE, TRUE, FALSE)

-- Set building attributes for orange
PThing.BldgSet(TRIBE_ORANGE, M_BUILDING_TEPEE, TRUE) -- Hut
PThing.BldgSet(TRIBE_ORANGE, M_BUILDING_DRUM_TOWER, TRUE)
PThing.BldgSet(TRIBE_ORANGE, M_BUILDING_WARRIOR_TRAIN, TRUE)
PThing.BldgSet(TRIBE_ORANGE, M_BUILDING_SUPER_TRAIN, TRUE) -- FW Hut

-- Set internal attributes for orange
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_EXPANSION, 30)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_HOUSE_PERCENTAGE, 50)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_MAX_BUILDINGS_ON_GO, 10)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_PREF_WARRIOR_TRAINS, 1)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_PREF_WARRIOR_PEOPLE, 20)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_PREF_SUPER_WARRIOR_TRAINS,1)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_PREF_SUPER_WARRIOR_PEOPLE, 40)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_MAX_TRAIN_AT_ONCE, 5)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_AWAY_BRAVE, 0)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_AWAY_WARRIOR, 50)
```

```
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_AWAY_SUPER_WARRIOR, 50)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_AWAY_RELIGIOUS, 0)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_AWAY_SPY, 0)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_AWAY_MEDICINE_MAN, 100)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_MAX_ATTACKS, 999)

-- Set states for orange
STATE_SET(TRIBE_ORANGE, TRUE, CP_AT_TYPE_FETCH_WOOD)
STATE_SET(TRIBE_ORANGE, TRUE, CP_AT_TYPE_CONSTRUCT_BUILDING)
STATE_SET(TRIBE_ORANGE, TRUE, CP_AT_TYPE_TRAIN_PEOPLE)
STATE_SET(TRIBE_ORANGE, TRUE, CP_AT_TYPE_AUTO_ATTACK)
STATE_SET(TRIBE_ORANGE, TRUE, CP_AT_TYPE_POPULATE_DRUM_TOWER)
STATE_SET(TRIBE_ORANGE,TRUE, CP_AT_TYPE_MED_MAN_GET_WILD_PEEPS)

-- Don't build the drum tower
DELAY_MAIN_DRUM_TOWER(TRUE,TRIBE_ORANGE)

-- Spell entries
SET_SPELL_ENTRY(TRIBE_ORANGE, 0, M_SPELL_INSECT_PLAGUE, 25000, 64, 3, 0)
SET_SPELL_ENTRY(TRIBE_ORANGE, 1, M_SPELL_INSECT_PLAGUE, 25000, 64, 3, 1)

-- Marker entries
SET_MARKER_ENTRY(TRIBE_ORANGE, 0, 0, 0, 0, 0, 2, 0)
SET_MARKER_ENTRY(TRIBE_ORANGE, 1, 1, 1, 0, 0, 2, 0)

function OnTurn()

end
```

Code to be executed during the game
Now that we have covered the attributes and building mechanics, let's move on to attacks, spell use, and all other kinds of things that we want the AI to do during the game. Remember that we are going to write this all in the function OnTurn(), which is executed every gameturn (1/12 seconds):

```
function OnTurn()
  Code to be executed during the game
end
```

Writing every loops in the LUA script: the every2Pow() function
In correspondence with the original PopScript, most in-game actions are executed when the gameturn equals some power of 2. In PopScript, this was denoted as:

```
every a
{
Code to be executed during the game
}
```

In which a is a power of 2. The same structure can be applied in LUA by using the function every2Pow(a), which is written in the UtilRefs.lua script. We write the statement as follows:

```
if (every2Pow(10)) then
   Code
End
```

If the current gameturn equals 2^10, the Code written in the if statement is executed.
Therewith it would have the same effect as "every 1024" (as 2^10 = 1024).

Of course we can also use other statements, including variables. Some examples:

```
if (spell_delay > 0) then
    spell_delay = spell_delay - 1
end
```

using such a spell delay variable is a great way to prevent the AI from spamming the same spells over and over. We'll discuss this later.

We can make it as complicated as we want:

```
if (every2Pow(10) and spell_delay > 0 and some_random_variable > 100) then
    Do something cool
end
```

Endless possibilities. This is where your programming skills come in. For loops and while loops may also be useful in translating your creative ideas to a script. Check out some very nice and short LUA tutorials on the web.

Back to Pop, here are a few Populous-related example for conditions we can use in our statements:
Every loop:

```
every2Pow(X)
```

Population count:

```
 _gsi.Players[TRIBE_ORANGE].NumPeople > X
```

Troop count:

```
PLAYERS_PEOPLE_OF_TYPE(TRIBE_ORANGE, M_PERSON_WARRIOR) > X
```

Shaman availability:

```
IS_SHAMAN_AVAILABLE_FOR_ATTACK(TRIBE_ORANGE) > 0
```

Basic NAV_CHECK and ATTACK commands
Nav-Check command:
PopScript:

```
DO NAV_CHECK ENEMY_TRIBE TARGET_TYPE MODEL REMEMBER VARIABLE
```

LUA Script:

```
NAV_CHECK(MY_TRIBE,ENEMY_TRIBE,ATTACK_BUILDING,MODEL,0)
```

Attack command:
PopScript:

```
DO ATTACK ENEMY_TRIBE NUM_PEOPLE TARGET_TYPE ATTACK MODEL DAMAGE M_SPELL1 M_SPELL2 M_SPELL3 ATTACK_TYPE BRING_BACK_VEHICLE MK1 MK2 MK3
```

LUA Script:

```
ATTACK(MY_TRIBE, ENEMY_TRIBE, NUM_PEOPLE, TARGET_TYPE, ATTACK_MODEL, DAMAGE, M_SPELL1, M_SPELL2, M_SPELL3, ATTACK_TYPE, BRING_BACK_VEHICLE, MK1, MK2, MK3)
```

Case: Basic attacks and patrolling
Let’s expand the code we have so far with a few attack commands, using if statements to check for conditions. Download the case map file and place it as levl2001.dat in your levels directory.
Take the code that you have so far and place it in a file called “Case.lua”, that you save to your Scripts directory. Start level 1 in Populous Beta to check if it’s working. You should see the orange tribe building huts, a warrior training hut, and a firewarrior training hut. You’ll also see them training people. If you’re patient, you’ll also see the orange shaman converting wildmen.

That is what we had written so far. In this case we will add the following things:

- There are a few markers in the map design. Let two firewarriors patrol at each of the markers 3 till 7.
  Here is how you should activate a marker entry:
  PopScript:

  ```
  DO MARKER_ENTRIES ENTRY1 ENTRY2 ENTRY3 ENTRY4
  ```

  LUA Script:

  ```
  MARKER_ENTRIES(MY_TRIBE, ENTRY1, ENTRY2, ENTRY3, ENTRY4)
  ```

- Script an attack that happens every 1024 game turns, if the orange population is higher than 30. Let them attack with 10 warriors and/or firewarriors. Let the shaman cast tornado and lightning during the attack.

- AI is still pretty much defenseless when we attack them. Let them build a few drum towers at strategic positions and fill them. The code below will get you started.

  Here is how you can build a drum tower:
  PopScript:

  ```
  DO BUILD_DRUM_TOWER X Z
  ```

  LUA Script:

  ```
  BUILD_DRUM_TOWER(MY_TRIBE, X, Z)
  ```

  Here is how you can populate a drum tower:
  PopScript:

  ```
  DO STATE POPULATE_DRUM_TOWER ON
  ```

  LUA Script:

  ```
  STATE_SET(TRIBE_ORANGE, TRUE, CP_AT_TYPE_POPULATE_DRUM_TOWER)
  ```

Case solution

SPOILER: HIDE

```
-- Populous the Beginning: LUA Scripting Case: The basics

-- Import modules and include files
import(Module_System)
import(Module_Players)
import(Module_Defines)
import(Module_PopScript)
import(Module_Game)
import(Module_Objects)
import(Module_Map)
import(Module_Math)
include("UtilPThings.lua")
include("UtilRefs.lua")

-- Initiate the orange tribe
computer_init_player(_gsi.Players[TRIBE_ORANGE])

-- Set spell attributes for orange
PThing.SpellSet(TRIBE_ORANGE, M_SPELL_CONVERT_WILD, TRUE, FALSE)
PThing.SpellSet(TRIBE_ORANGE, M_SPELL_BLAST, TRUE, FALSE)
PThing.SpellSet(TRIBE_ORANGE, M_SPELL_LAND_BRIDGE, TRUE, FALSE)
PThing.SpellSet(TRIBE_ORANGE, M_SPELL_INSECT_PLAGUE, TRUE, FALSE)

-- Set building attributes for orange
PThing.BldgSet(TRIBE_ORANGE, M_BUILDING_TEPEE, TRUE) -- Hut
PThing.BldgSet(TRIBE_ORANGE, M_BUILDING_DRUM_TOWER, TRUE)
PThing.BldgSet(TRIBE_ORANGE, M_BUILDING_WARRIOR_TRAIN, TRUE)
PThing.BldgSet(TRIBE_ORANGE, M_BUILDING_TEMPLE, TRUE)
PThing.BldgSet(TRIBE_ORANGE, M_BUILDING_SUPER_TRAIN, TRUE)

-- Set internal attributes for orange
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_EXPANSION, 30)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_HOUSE_PERCENTAGE, 50)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_MAX_BUILDINGS_ON_GO, 10)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_PREF_WARRIOR_TRAINS, 1)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_PREF_WARRIOR_PEOPLE, 20)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_PREF_SUPER_WARRIOR_TRAINS,1)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_PREF_SUPER_WARRIOR_PEOPLE, 40)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_MAX_TRAIN_AT_ONCE, 5)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_AWAY_BRAVE, 0)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_AWAY_WARRIOR, 50)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_AWAY_SUPER_WARRIOR, 50)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_AWAY_RELIGIOUS, 0)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_AWAY_SPY, 0)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_AWAY_MEDICINE_MAN, 100)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_MAX_ATTACKS, 999)

-- Set states for orange
STATE_SET(TRIBE_ORANGE, TRUE, CP_AT_TYPE_FETCH_WOOD)
STATE_SET(TRIBE_ORANGE, TRUE, CP_AT_TYPE_CONSTRUCT_BUILDING)
STATE_SET(TRIBE_ORANGE, TRUE, CP_AT_TYPE_TRAIN_PEOPLE)
STATE_SET(TRIBE_ORANGE, TRUE, CP_AT_TYPE_AUTO_ATTACK)
```

```
STATE_SET(TRIBE_ORANGE, TRUE, CP_AT_TYPE_POPULATE_DRUM_TOWER)
STATE_SET(TRIBE_ORANGE,TRUE, CP_AT_TYPE_MED_MAN_GET_WILD_PEEPS)

-- Don't build the drum tower
DELAY_MAIN_DRUM_TOWER(TRUE,TRIBE_ORANGE)

-- Spell entries
SET_SPELL_ENTRY(TRIBE_ORANGE, 0, M_SPELL_INSECT_PLAGUE, 25000, 64, 3, 0)
SET_SPELL_ENTRY(TRIBE_ORANGE, 1, M_SPELL_INSECT_PLAGUE, 25000, 64, 3, 1)

-- Marker entries
for i = 0,7 do
SET_MARKER_ENTRY(TRIBE_ORANGE, i, i, i, 0, 0, 2, 0)
end

function OnTurn()

-- Activating the marker entries
if (_gsi.Counts.GameTurn == 512) then
for i = 3,7 do
MARKER_ENTRIES(TRIBE_ORANGE, i, -1, -1, -1)
end
end

-- Attacking the blue tribe
if (every2Pow(10) and _gsi.Players[TRIBE_ORANGE].NumPeople > 30 and PLAYERS_PEOPLE_OF_TYPE(TRIBE_ORANGE, M_PERSON_WARRIOR) > 5 and PLAYERS_PEOPLE_OF_TYPE(TRIBE_ORANGE, M_PERSON_SUPER_WARRIOR) > 5) then
ATTACK(TRIBE_ORANGE, TRIBE_BLUE, 10, ATTACK_BUILDING, -1, 999, M_SPELL_WHIRLWIND, M_SPELL_LIGHTNING_BOLT, M_SPELL_LIGHTNING_BOLT, ATTACK_NORMAL, 0, NO_MARKER, NO_MARKER, NO_MARKER)
end

-- Placing tower plans
if (_gsi.Counts.GameTurn == 512) then
BUILD_DRUM_TOWER(TRIBE_ORANGE, 0, 10)
BUILD_DRUM_TOWER(TRIBE_ORANGE, 10, 0)
BUILD_DRUM_TOWER(TRIBE_ORANGE, 240, 240)
BUILD_DRUM_TOWER(TRIBE_ORANGE, 230, 0)
end

end
```

Re: Scripting in Populous Beta: LUA tutorial

Posted: **Thu May 28, 2020 8:33 pm**
by **410172_Chief**
LUA Scripting: Advanced scripting
Now that we understand the basics of LUA scripting and can write most of the PopScript stuff in LUA, let's move on to some advanced scripts. We will now discuss functions that are LUA-only: new things that PopScript can't do, but LUA can. Read closely, use your brains, good luck and have fun!
We use the case solution for the Basics part as a starting point for this part of the tutorial.

Create a thing. Anything.
The createThing() function is used to, literally, create a thing on the map. This makes it one of the most versatile functions available in the LUA language. It's syntax:

```
createThing(TYPE, MODEL, CREATOR, COORDINATE, false, false)
```

We can create effects:

```
createThing(T_EFFECT, M_EFFECT_RISE, TRIBE_HOSTBOT, c3d, false, false)
```

Pieces of scenery:

```
createThing(T_SCENERY,M_SCENERY_ROCK,TRIBE_HOSTBOT,c3d,false,false)
```

We can create wildmen:

```
createThing(T_PERSON,M_PERSON_WILD,TRIBE_NEUTRAL,c3d,false,false)
```

Or use the function to let a shaman cast a spell:

```
createThing(T_SPELL, M_SPELL_LAND_BRIDGE, shaman.Owner, c3d, false, false)
```

We can almost create any kind of Populous object you can think of. And we get to determine the conditions at which the object is created:

```
if (condition) then
createThing(TYPE, MODEL, CREATOR, COORDINATE, false, false)
end
```

Now that I've convinced you that the createThing() function is absolutely great, lets take a closer look at the syntax.

- The first thing we need to give in is the TYPE of thing we are creating: T_PERSON, T_SPELL, T_SCENERY, T_EFFECT, and there probably is more.
- Then, we need to specify the MODEL. So: the type of effect, type of scenery, type of person, or type of spell. We have already seen some spell types: M_SPELL_BLAST, M_SPELL_CONVERT_WILD, M_SPELL_LAND_BRIDGE, etc. Those kind of things.
- Thirdly, there is the CREATOR, or the owner of the thing. If we are creating an effect or scenery object, there will be no owner and we write TRIBE_HOSTBOT. For wildmen, we use TRIBE_NEUTRAL. This is, however, not the case if we want a shaman to cast a spell.

Let's give that example a little more detail before we move to the COORDINATE c3d.

```
createThing(T_SPELL, M_SPELL_LAND_BRIDGE, shaman.Owner, c3d, false, false)
```

In this line, the TYPE is a spell and the MODEL, in this case the type of spell, is land bridge.
As CREATOR, we have shaman.Owner, which is an object property. Using this will actually make a shaman cast the spell in game.

Just using shaman.Owner does not tell the game which shaman it should use. That's something we need to define first. As we were scripting for the orange tribe, we say:

```
local shaman = getShaman(TRIBE_ORANGE)
```

so the total code required to let the orange shaman cast a land bridge spell is:

```
local shaman = getShaman(TRIBE_ORANGE)
createThing(T_SPELL, M_SPELL_LAND_BRIDGE, shaman.Owner, c3d, false, false)
```

Finally, there is the COORDINATE, c3d. c3d is a 3D-coordinate that contains the position where the thing is created, or in this case the position where the orange shaman will cast the land bridge spell to.
A 3D-coordinate contains an XPos, ZPos, and YPos, where YPos is height.
To define a 3D-coordinate from an (X,Z) position on the map, we use the syntax:

```
local c3d = MAP_XZ_2_WORLD_XYZ(X,Z)
```

Now let's say that we want the orange Shaman to cast a land bridge spell at a certain point in time. For example, to connect to the land of the player so she can do an attack. We could write:

```
if (_gsi.Counts.GameTurn == 1024) then
local c3d = MAP_XZ_2_WORLD_XYZ(36,74)
local shaman = getShaman(TRIBE_ORANGE)
createThing(T_SPELL, M_SPELL_LAND_BRIDGE, shaman.Owner, c3d, false, false)
end
```

Then of course the position (36,74) should make sense. We don't want to waste spells.

AI map analysis
We have discussed how we can change the TYPE, MODEL and OWNER of the thing we're creating with createThing(), and how to provide a COORDINATE. But what if you don't know the COORDINATE you want to create a thing? What if the COORDINATE changes over time?

You're right! We're going to discuss a way for the AI to do map analysis and combine this with casting a spell. For map analysis we will use the SearchMapCells() function. This function searches a specific area of the map to find a position that meets certain conditions. The syntax:

```
SearchMapCells(FORM, 0, 0, RADIUS, MAP_COORDINATE, function(me)
if (condition) then
return false
end
return true
end)
```

The FORM is either CIRCULAR (for searching in a circle around MAP_COORDINATE) or SQUARE (for searching in a square around MAP_COORDINATE). The RADIUS is the radius of this circle, or the length of the square. MAP_COORDINATE is a 3D coordinate. This coordinate however is a different coordinate type than the c3d coordinate we have seen before. That is because the game uses different ways to describe a position.

Luckily, there is a function for anything. We just need to add an extra line to convert the 3D-COORDINATE into a MAP_COORDINATE:

```
local c3dLookPosition = MAP_XZ_2_WORLD_XYZ(214,220)
map_coordinate = world_coord3d_to_map_idx(c3dLookPosition)
```

That's just the way it works, folks. We gotta live with it.

Function(me) is the mapelement function that describes the conditions. It should contain at least this:

```
if (condition) then
return false
end
return true
```

I'll explain why. SearchMapCells() actually is a loop. Using FORM, RADIUS and COORDINATE, it comes up with a list of positions to check and then loops through every position of that list, one by one. The "return true" statement is required to keep the loop running. If we find a cell that meets our conditions, we must say "return false" to make the loop stop.

Let's illustrate this with a quick example:

```
local c3d = MAP_XZ_2_WORLD_XYZ(214,220)
SearchMapCells(CIRCULAR, 0, 0, 6, world_coord3d_to_map_idx(c3d), function(me)
local exampleVariable = 0
if (exampleVariable == 3) then
return false
end
exampleVariable = exampleVariable + 1
return true
end)
```

Yes, that is one unless piece of code. All it does is change the variable exampleVariable and stops once it obtains a value of 3. But I hope you now have an understanding of how that looping works.
Now let's make it do something useful, shall we? Remember the goal of SearchMapCells: To find a position on the map that meets certain conditions. Using some functions, we can obtain information from the mapelement me and do our analysis.

Let's look at a few of these mapelement functions:

```
is_map_elem_all_land(mapelement)
is_map_elem_all_sea(mapelement)
is_map_elem_coast(mapelement)
```

These three functions require a mapelement and return TRUE of FALSE, so 0 or 1 as an integer.
Let's go back to the previous example and rewrite the function(me) part.

```
local c3d = MAP_XZ_2_WORLD_XYZ(214,220)
SearchMapCells(CIRCULAR, 0, 0, 6, world_coord3d_to_map_idx(c3d), function(me)
if (is_map_elem_coast(me) > 0) then
local shaman = getShaman(TRIBE_ORANGE)
createThing(T_SPELL, M_SPELL_LAND_BRIDGE, shaman.Owner, c3d, false, false)
return false
end
return true
end)
```

Our land bridge spell is back! In this piece of code, it is cast if "is_map_elem_coast(me) > 0" is true. So we get the land bridge spell only if we find a position on the map is that is at the intersection of land and sea. Then the function stops and we get the land bridge at the search position (214,220).
That's pretty useless still. But we wouldn't go through all of this searching trouble for nothing. Our next step is to read the position that met the conditions and cast the land bridge spell there, instead of on our set position (214,220). That sounds a lot better, does it not?

Here is the code that we need:

```
local c3dLookPosition = MAP_XZ_2_WORLD_XYZ(214,220)
SearchMapCells(CIRCULAR, 0, 0, 6, world_coord3d_to_map_idx(c3dLookPosition), function(me)
if (is_map_elem_coast(me) > 0) then
        local c2d = Coord2D.new()
       map_ptr_to_world_coord2d(me, c2d)
       local c3d = Coord3D.new()
       coord2D_to_coord3D(c2d, c3d)

local shaman = getShaman(TRIBE_ORANGE)
createThing(T_SPELL, M_SPELL_LAND_BRIDGE, shaman.Owner, c3d, false, false)
return false
end
return true
end)
```

See the changes? We added 4 lines to the if-statement that help us read the coordinates from the mapelement me. We created a c2d coordinate and c3d coordinate, using the syntax:

```
local c2d = Coord2D.new()
local c3d = Coord3D.new()
```

These coordinates are just empty, but they are required for the functions that read the position from the mapelement. Remember c2d is a 2D-coordinate, with XPos and Zpos. c3d is a 3D-coordinate, with XPos, Zpos, and Ypos, where YPos is height.

Now that we have these two coordinates, we can convert our mapelement to a 3D-coordinate using two functions.

```
map_ptr_to_world_coord2d(me, c2d)
coord2D_to_coord3D(c2d, c3d)
```

Then we have our 3D-coordinate c3d, which is the coordinate found by SearchMapCells(): The coast coordinate. Why? Because our whole conversion process is in the if-statement. It is only executed when SearchMapCells() finds a coast position.

Don't let all those conversions get you confused, it's just how the game works.

If you look at the code we have now, you'll notice there is one position that is still pre-defined and constant: the position where SearchMapCells() starts its search.
We are searching for a coast position to cast a land bridge spell, but what if we find one and our shaman is at the other side of the map? Casting over half the world would be totally ridiculous. Or cool. Well, anyway, let's set our function to analyze the map around the shaman, so that we don't get bridges where we don't want them.

Implementing this last step will give us an AI script for one of multiplayer's popular shaman moves: saving the shaman from drowning with a land bridge spell.

Remember this line?

```
local shaman = getShaman(TRIBE_ORANGE)
```

It allowed us to set the orange shaman to be the owner of a cast spell, but the game can tell us more about the shaman, as long as we ask for it. We can check if the shaman is actually alive or dead:

```
if (shaman ~= nil) then
Code to execute if Shaman is alive
end
```

We can also check what the shaman is currently doing using shaman.State. To check if she is drowning, we can use:

```
if (shaman.State == S_PERSON_DROWNING) then
        Code to execute if Shaman is drowning
end
```

We can read the shaman's 2D-position and 3D-position:

```
Shaman2DPos = shaman.Pos.D2
Shaman3DPos = shaman.Pos.D3
```

And of course there are a lot more states we can check out, such as:

```
S_PERSON_DYING
S_PERSON_ELECTROCUTED
S_PERSON_IN_WHIRLWIND
S_PERSON_ON_FIRE
```

Maybe you can find some creative use for those by yourself. Let's get back to our code.
If we put our SearchMapCells() function inside the drowning if-statement, we obtain:

```
local shaman = getShaman(TRIBE_ORANGE)
if (shaman ~= nil) then
if (shaman.State == S_PERSON_DROWNING) then
SearchMapCells(CIRCULAR, 0, 0, 6, world_coord2d_to_map_idx(shaman.Pos.D2), function(me)
if (is_map_elem_coast(me) > 0) then
local c2d = Coord2D.new()
map_ptr_to_world_coord2d(me, c2d)
local c3d = Coord3D.new()
coord2D_to_coord3D(c2d, c3d)
createThing(T_SPELL, M_SPELL_LAND_BRIDGE, shaman.Owner, c3d, false, false)
return false
```

```
end
return true
end)
end
end
```

And that is it. Now it's time to try it out. Start your level and blast the orange shaman into the water. You'll discover a minor flaw: each time the shaman drowns, she casts land bridge multiple times. And here we were trying to prevent her from wasting spells. Well, let's fix that, shall we?
We are going to introduce a spell delay to our script, to prevent the spam. Let's define a variable spell_delay at the beginning of our script:

```
-- Set the variables we are going to use
spell_delay = 0
```

What we are going to do, is change the value of spell_delay after creating the land bridge spell. And, we change our if-statement: we only want the land bridge spell if the value spell_delay equals 0.

```
local shaman = getShaman(TRIBE_ORANGE)
if (shaman ~= nil) then
if (shaman.State == S_PERSON_DROWNING and spell_delay == 0) then
SearchMapCells(CIRCULAR, 0, 0, 6, world_coord2d_to_map_idx(shaman.Pos.D2), function(me)
if (is_map_elem_coast(me) > 0) then
local c2d = Coord2D.new()
map_ptr_to_world_coord2d(me, c2d)
local c3d = Coord3D.new()
coord2D_to_coord3D(c2d, c3d)
createThing(T_SPELL, M_SPELL_LAND_BRIDGE, shaman.Owner, c3d, false, false)
spell_delay = 36
return false
end
return true
end)
end
end
```

Ever wondered how the spell delay in this example is going to get back to 0 after we set it to 36?
Yes, we need to do something about that ourselves. Let's make the value of spell_delay decrease with 1 every gameturn. Include the if-statement to prevent your spell delay from becoming negative.

-- Changing the spell delay every turn

```
if (spell_delay > 0) then
spell_delay = spell_delay - 1
end
```

If you paid close attention so far, your code will now look something like this:

SPOILER: HIDE

```
-- LUA Test script for orange tribe in SP.

-- Import modules and include files
import(Module_System)
import(Module_Players)
import(Module_Defines)
import(Module_PopScript)
import(Module_Game)
import(Module_Objects)
import(Module_Map)
import(Module_Math)
include("UtilPThings.lua")
include("UtilRefs.lua")

-- Variables we're going to use
spell_delay = 0

-- Initiate the orange tribe
computer_init_player(_gsi.Players[TRIBE_ORANGE])

-- Set spell attributes for orange
PThing.SpellSet(TRIBE_ORANGE, M_SPELL_CONVERT_WILD, TRUE, FALSE)
PThing.SpellSet(TRIBE_ORANGE, M_SPELL_BLAST, TRUE, FALSE)
PThing.SpellSet(TRIBE_ORANGE, M_SPELL_LAND_BRIDGE, TRUE, FALSE)
PThing.SpellSet(TRIBE_ORANGE, M_SPELL_INSECT_PLAGUE, TRUE, FALSE)

-- Set building attributes for orange
```

```
PThing.BldgSet(TRIBE_ORANGE, M_BUILDING_TEPEE, TRUE) -- Hut
PThing.BldgSet(TRIBE_ORANGE, M_BUILDING_DRUM_TOWER, TRUE)
PThing.BldgSet(TRIBE_ORANGE, M_BUILDING_WARRIOR_TRAIN, TRUE)
PThing.BldgSet(TRIBE_ORANGE, M_BUILDING_TEMPLE, TRUE)
PThing.BldgSet(TRIBE_ORANGE, M_BUILDING_SUPER_TRAIN, TRUE)

-- Set internal attributes for orange
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_EXPANSION, 30)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_HOUSE_PERCENTAGE, 50)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_MAX_BUILDINGS_ON_GO, 10)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_PREF_WARRIOR_TRAINS, 1)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_PREF_WARRIOR_PEOPLE, 20)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_PREF_SUPER_WARRIOR_TRAINS,1)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_PREF_SUPER_WARRIOR_PEOPLE, 40)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_MAX_TRAIN_AT_ONCE, 5)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_AWAY_BRAVE, 0)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_AWAY_WARRIOR, 50)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_AWAY_SUPER_WARRIOR, 50)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_AWAY_RELIGIOUS, 0)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_AWAY_SPY, 0)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_AWAY_MEDICINE_MAN, 100)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_MAX_ATTACKS, 999)

-- Set states for orange
STATE_SET(TRIBE_ORANGE, TRUE, CP_AT_TYPE_FETCH_WOOD)
STATE_SET(TRIBE_ORANGE, TRUE, CP_AT_TYPE_CONSTRUCT_BUILDING)
STATE_SET(TRIBE_ORANGE, TRUE, CP_AT_TYPE_TRAIN_PEOPLE)
STATE_SET(TRIBE_ORANGE, TRUE, CP_AT_TYPE_AUTO_ATTACK)
STATE_SET(TRIBE_ORANGE, TRUE, CP_AT_TYPE_POPULATE_DRUM_TOWER)
STATE_SET(TRIBE_ORANGE,TRUE, CP_AT_TYPE_MED_MAN_GET_WILD_PEEPS)

-- Don’t build the drum tower
DELAY_MAIN_DRUM_TOWER(TRUE,TRIBE_ORANGE)


-- Spell entries
SET_SPELL_ENTRY(TRIBE_ORANGE, 0, M_SPELL_INSECT_PLAGUE, 25000, 64, 3, 0)
SET_SPELL_ENTRY(TRIBE_ORANGE, 1, M_SPELL_INSECT_PLAGUE, 25000, 64, 3, 1)

-- Marker entries
for i = 0,7 do
SET_MARKER_ENTRY(TRIBE_ORANGE, i, i, i, 0, 0, 2, 0)
end

function OnTurn()

-- Activating the marker entries
if (_gsi.Counts.GameTurn == 512) then
for i = 3,7 do
MARKER_ENTRIES(TRIBE_ORANGE, i, -1, -1, -1)
end
end

-- Attacking the blue tribe
if (every2Pow(10) and _gsi.Players[TRIBE_ORANGE].NumPeople > 30 and PLAYERS_PEOPLE_OF_TYPE(TRIBE_ORANGE, M_PERSON_WARRIOR) > 5 and PLAYERS_PEOPLE_OF_TYPE(TRIBE_ORANGE, M_PERSON_SUPER_WARRIOR) > 5) then
ATTACK(TRIBE_ORANGE, TRIBE_BLUE, 10, ATTACK_BUILDING, -1, 999, M_SPELL_WHIRLWIND, M_SPELL_LIGHTNING_BOLT, M_SPELL_LIGHTNING_BOLT, ATTACK_NORMAL, 0, NO_MARKER, NO_MARKER, NO_MARKER)
end

-- Placing tower plans
if (_gsi.Counts.GameTurn == 512) then
BUILD_DRUM_TOWER(TRIBE_ORANGE, 0, 10)
BUILD_DRUM_TOWER(TRIBE_ORANGE, 10, 0)
BUILD_DRUM_TOWER(TRIBE_ORANGE, 240, 240)
BUILD_DRUM_TOWER(TRIBE_ORANGE, 230, 0)
end

-- Saving the shaman from drowning with a land bridge spell
    local shaman = getShaman(TRIBE_ORANGE)
    if (shaman ~= nil) then
        if (shaman.State == S_PERSON_DROWNING and spell_delay == 0) then
            SearchMapCells(CIRCULAR, 0, 0, 6, world_coord2d_to_map_idx(shaman.Pos.D2), function(me)
            if (is_map_elem_coast(me) > 0) then
                local c2d = Coord2D.new()
                map_ptr_to_world_coord2d(me, c2d)
                local c3d = Coord3D.new()
                coord2D_to_coord3D(c2d, c3d)
                createThing(T_SPELL, M_SPELL_LAND_BRIDGE, shaman.Owner, c3d, false, false)
                spell_delay = 36
                return false
```

```
            end
            return true
            end)
        end
    end

-- Changing the spell delay every turn
    if (spell_delay > 0) then
    spell_delay = spell_delay - 1
    end

end
```

Congratulations, you have just mastered one of the most difficult parts of this tutorial. But there is still more! Take a cup of coffee, relax a bit, I'll wait.

Ready? All-right, let's continue!

Counting objects during a search
Our last example was just one example of the many things we can do with SearchMapCells(). There is more information in a mapelement than just its location. We can scan for contents: buildings, followers, scenery, etc. Let's say, for an example, that we want to count the number of enemy buildings within a radius 8 of the shaman's position. We would start in the same way as we did in our previous example:

```
local shaman = getShaman(TRIBE_ORANGE)
if (shaman ~= nil) then
SearchMapCells(CIRCULAR,0,0,8,world_coord2d_to_map_idx(shaman.Pos.D2),
function(me)
Code
return true
end)
end
```

Now comes the new part. We are going to build in a processList, which is a list of all the objects inside the mapelement me. It's syntax is similar to that of the function(me) part that we've seen:

```
me.MapWhoList:processList(function(t)
Code
return true
end)
```

The processList is a simple loop through all the objects t that are in the mapelement me. We return true at the end of the function to keep the loop running. Inside the function statement we are going to write conditions that our objects should meet.
Two properties that we want to read from the object are its type and its owner. We can obtain those using t.type and t.owner. t.type can be a T_PERSON, T_BUILDING, or T_SCENERY. t.owner describes the tribe that owns it. We can build in if-statements to filter all the objects t for enemy buildings:

We can check for the type and owner using these two if-statements:

```
if (t.Type == T_BUILDING) then
if (t.Owner ~= TRIBE_ORANGE) then
        Some code that describes an action
end
end
```

We just asked two questions: 1. Is our object a building? And 2: Is orange not the building's owner?
If those two questions return true as an answer, our object is an enemy building. Agree?

For the last step, we need to add a way to count the number of enemy buildings. To do this, we add a variable count and increase its value by one for every enemy building we find:

```
local count = 0;
local shaman = getShaman(TRIBE_ORANGE)
if (shaman ~= nil) then
SearchMapCells(CIRCULAR,0,0,8,world_coord2d_to_map_idx(shaman.Pos.D2),
function(me)
me.MapWhoList:processList(function(t)
if (t.Type == T_BUILDING) then
if (t.Owner ~= TRIBE_ORANGE) then
                        count = count+1
end
end
return true
end)
return true
end)
end
```

Using functions
We just made a nice piece of code to count the number of enemy buildings near the shaman. But what if we want to know more? Say that we want to know the enemy buildings near the mid, or near our base, or near a stone head that's on the map? Yes, we would basically need the same code, all we need to do is give in another search location to SearchMapCells().

To do such a thing in a nice way, we use functions. With a function, we don't need to copy paste code and adjust a few parameters each time. We give the parameters to the function as input, and adjust our code to use the input. We need to create something like this:

```
function function_name(parameter1,parameter2,parameter3)
Code that uses parameter1, parameter2, parameter3
return answer
end
```

In our main code, we can then use:

```
answer1 = function_name(parameter1,parameter2,parameter3)
answer2 = function_name(parameter4,parameter5,parameter6)
```

Get the idea? That will save us a lot of space and time.
Let's now adjust our latest example and build a function from it. Let's call it count_enemy_buildings. Let's give the X and Z coordinates as input, and the search radius. Our code structure:

```
function count_enemy_buildings(x,z,radius)
Code that uses x, z, radius
return count
end
```

We replace the numbers we used previously as x, z and radius by variables, like this:

```
function count_enemy_buildings(x,z,radius)
local count = 0;
local c3d = MAP_XZ_2_WORLD_XYZ(x,z)
SearchMapCells(CIRCULAR,0,0,radius,world_coord3d_to_map_idx(c3d),function(me)
me.MapWhoList:processList(function(t)
if (t.Type == T_BUILDING) then
if (t.Owner ~= TRIBE_ORANGE) then
                        count = count+1
end
end
return true
end)
return true
end)
return count
end
```

In our main code, we can then use, for example:

```
num_enemy_buildings_pos1 = count_enemy_buildings(10,10,8)
num_enemy_buildings_pos2 = count_enemy_buildings(200,64,10)
num_enemy_buildings_pos3 = count_enemy_buildings(18,105,6)
```

With the count_enemy_buildings() function we just wrote, this code:

```
num_enemy_buildings_pos1 = count_enemy_buildings(10,10,8)
```

does the same thing as:

```
local count = 0;
local c3d = MAP_XZ_2_WORLD_XYZ(10,10)
SearchMapCells(CIRCULAR,0,0,8,world_coord3d_to_map_idx(c3d),function(me)
me.MapWhoList:processList(function(t)
if (t.Type == T_BUILDING) then
if (t.Owner ~= TRIBE_ORANGE) then
      count = count+1
end
end
return true
end)
return true
end)
num_enemy_buildings_pos1 = count
```

Functions make your code flexible and overall, shorter. If you constantly use code sections that are very similar, see if you can combine some of those sections into a function.

Counting all objects to the map
Similar to the processList we've seen in the SearchMapCells function, there is a function GlobalTypeList() that we can use to count the total number of objects on the map.
The syntax is similar:

```
ProcessGlobalTypeList(TYPE, function(t)
Code
return true
end)
```

where TYPE = T_BUILDING, T_PERSON, T_SCENERY.

Inside the statement, we can again provide the following code to count enemy buildings:

```
if (t.Type == T_BUILDING) then
if (t.Owner ~= TRIBE_ORANGE) then
        count = count+1
end
end
```

However the line if (t.Type == T_BUILDING) is not needed as ProcessGlobalTypeList() has TYPE as the input. So the code to count the total number of enemy buildings would be:

```
local count = 0
ProcessGlobalTypeList(TYPE_BUILDING, function(t)
if (t.Owner ~= TRIBE_ORANGE) then
count = count+1
end
return true
end)
```

Now say the we want to count the number of troops? Say, warriors.
We can use the model property, t.Model, which describes the type person.
This can be M_PERSON_BRAVE, M_PERSON_WARRIOR, M_PERSON RELIGIOUS, M_PERSON_SUPER_WARRIOR, M_PERSON_SPY.

We then need to include an additional if statement which checks the model:

```
if (t.Model == M_PERSON_WARRIOR) then
end
```

So:

```
local count = 0
ProcessGlobalTypeList(TYPE_PERSON, function(t)
if (t.Owner ~= TRIBE_ORANGE) then
if (t.Model == M_PERSON_WARRIOR) then
count = count+1
end
end
return true
end)
```

Intelligent conversion of wildmen
During our Populous careers, we all have spent some time wondering why AI converting wildmen works the way it does. The built-in functions for converting wildmen are slow and always require the shaman to walk towards the wildmen even if they are in range.
With the createThing() function, we can change this. We use a ProcessGlobalTypeList to find the wildmen on the map and then create spell effect.
We start with this:

```
        local shaman = getShaman(TRIBE_ORANGE)
        if (shaman ~= nil) then
            ProcessGlobalTypeList(T_PERSON, function(t)
            if (t.Model == M_PERSON_WILD) then
                createThing(T_SPELL, M_SPELL_CONVERT_WILD, shaman.Owner, t.Pos.D3, false, false)
                return false
            end
            return true
            end)
        end
```

That is a perfect way to let the orange shaman instantly convert all the wildmen on the map, all at once.

Do you also find that a little too much? Let's do something about that by adding some conditions.

We can introduce a spell delay, but we can also just say:

```
if (every2Pow(6)) then
        local shaman = getShaman(TRIBE_ORANGE)
        if (shaman ~= nil) then
            ProcessGlobalTypeList(T_PERSON, function(t)
            if (t.Model == M_PERSON_WILD) and (get_world_dist_xyz(shaman.Pos.D3, t.Pos.D3) < 8192 + shaman.Pos.D3.Ypos*3) then
                createThing(T_SPELL, M_SPELL_CONVERT_WILD, shaman.Owner, t.Pos.D3, false, false)
                return false
            end
            return true
            end)
        end
    end
```

Now, every 64 gameturns (5,3 seconds), the orange shaman will cast a convert spell on a wildmen on the map.
To prevent her from taking our wildmen, we want to add a range. We can obtain the distance between two coordinates:

```
distance = get_world_dist_xyz(c3d_1, c3d_2)
```

To obtain the distance between the shaman and a wildman (object t) that came up in the ProcessGlobalTypeList search, we can type:

```
get_world_dist_xyz(shaman.Pos.D3, t.Pos.D3)
```

The spell range for convert so just happens to be described by:

```
convert_range = 8192 + shaman.Pos.D3.Ypos*3
```

You can see the shaman.Pos.D3.YPos syntax here, that's the current height of the shaman. That has an effect on the spell range, agree?

So in total, our code for intelligent wildman conversion would be:

```
if (every2Pow(6)) then
        local shaman = getShaman(TRIBE_ORANGE)
        if (shaman ~= nil) then
            ProcessGlobalTypeList(T_PERSON, function(t)
            if (t.Model == M_PERSON_WILD) and (get_world_dist_xyz(shaman.Pos.D3, t.Pos.D3) < 8192 + shaman.Pos.D3.Ypos*3) then
                createThing(T_SPELL, M_SPELL_CONVERT_WILD, shaman.Owner, t.Pos.D3, false, false)
                return false
            end
            return true
            end)
        end
    end
```

Of course you can change the code to change the range, or conversion pace.

Enemy shaman battles
We can use the same trick with object range to write a code that makes the AI shaman aggressive towards the player shaman. See if you can understand the code below.

```
local shaman = getShaman(TRIBE_ORANGE)
if (shaman ~= nil) then
ProcessGlobalTypeList(T_PERSON, function(t)
       if (t.Owner == TRIBE_BLUE) then
if (get_world_dist_xyz(shaman.Pos.D3, t.Pos.D3) < 3092 + shaman.Pos.D3.Ypos*3 and t.Model == M_PERSON_MEDICINE_MAN) then
if (spell_delay == 0 and is_thing_on_ground(shaman) == 1) then
createThing(T_SPELL, M_SPELL_BLAST, shaman.Owner, t.Pos.D3, false, false)
                        spell_delay = 24
                        return false
                        end
elseif (get_world_dist_xyz(shaman.Pos.D3, t.Pos.D3) < 6000 + shaman.Pos.D3.Ypos*3 and t.Model == M_PERSON_MEDICINE_MAN) then
if (spell_delay == 0 and is_thing_on_ground(shaman) == 1) then
createThing(T_SPELL, M_SPELL_LIGHTNING_BOLT, shaman.Owner, t.Pos.D3, false, false)
                        spell_delay = 60
                        return false
                        end
                end
       end
       return true
        end)
end
```

You're right. It is a big chunk of code containing many if statements. We need to do a lot of checks to make sure we are not creating blast effects when the shaman would not be able to cast them.

It makes perfect sense:

- We first make sure that we're talking about the orange shaman and that she is actually alive.
- Then we check if there are people present using a ProcessGlobalTypeList().
- Then we check that the people we find belong to the blue tribe, even better, that it is the blue shaman.
- We check if a blast spell can be cast within range.
- We check that the orange shaman is one the ground. Not flying or drowning, then she can not cast.
- We check if the spell delay equals 0.
- And then we use createThing() to cast the blast spell at the blue shaman's position. We return false to end the search. Say that the shaman would be out of range for blast, we check if we would be able to cast a lightning spell. As we all know, the range for the lightning spell is much larger.

Let's see what our code looks like now we have written all this:

SPOILER: HIDE

```
-- LUA Test script for orange tribe in SP.

-- Import modules and include files
import(Module_System)
import(Module_Players)
import(Module_Defines)
import(Module_PopScript)
import(Module_Game)
import(Module_Objects)
import(Module_Map)
import(Module_Math)
include("UtilPThings.lua")
include("UtilRefs.lua")

-- Variables we're going to use
spell_delay = 0

-- Initiate the orange tribe
computer_init_player(_gsi.Players[TRIBE_ORANGE])

-- Set spell attributes for orange
PThing.SpellSet(TRIBE_ORANGE, M_SPELL_CONVERT_WILD, TRUE, FALSE)
PThing.SpellSet(TRIBE_ORANGE, M_SPELL_BLAST, TRUE, FALSE)
PThing.SpellSet(TRIBE_ORANGE, M_SPELL_LAND_BRIDGE, TRUE, FALSE)
PThing.SpellSet(TRIBE_ORANGE, M_SPELL_INSECT_PLAGUE, TRUE, FALSE)

-- Set building attributes for orange
PThing.BldgSet(TRIBE_ORANGE, M_BUILDING_TEPEE, TRUE) -- Hut
PThing.BldgSet(TRIBE_ORANGE, M_BUILDING_DRUM_TOWER, TRUE)
PThing.BldgSet(TRIBE_ORANGE, M_BUILDING_WARRIOR_TRAIN, TRUE)
PThing.BldgSet(TRIBE_ORANGE, M_BUILDING_TEMPLE, TRUE)
PThing.BldgSet(TRIBE_ORANGE, M_BUILDING_SUPER_TRAIN, TRUE)

-- Set internal attributes for orange
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_EXPANSION, 30)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_HOUSE_PERCENTAGE, 50)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_MAX_BUILDINGS_ON_GO, 10)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_PREF_WARRIOR_TRAINS, 1)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_PREF_WARRIOR_PEOPLE, 20)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_PREF_SUPER_WARRIOR_TRAINS,1)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_PREF_SUPER_WARRIOR_PEOPLE, 40)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_MAX_TRAIN_AT_ONCE, 5)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_AWAY_BRAVE, 0)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_AWAY_WARRIOR, 50)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_AWAY_SUPER_WARRIOR, 50)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_AWAY_RELIGIOUS, 0)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_AWAY_SPY, 0)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_AWAY_MEDICINE_MAN, 100)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_MAX_ATTACKS, 999)

-- Set states for orange
STATE_SET(TRIBE_ORANGE, TRUE, CP_AT_TYPE_FETCH_WOOD)
STATE_SET(TRIBE_ORANGE, TRUE, CP_AT_TYPE_CONSTRUCT_BUILDING)
STATE_SET(TRIBE_ORANGE, TRUE, CP_AT_TYPE_TRAIN_PEOPLE)
STATE_SET(TRIBE_ORANGE, TRUE, CP_AT_TYPE_AUTO_ATTACK)
STATE_SET(TRIBE_ORANGE, TRUE, CP_AT_TYPE_POPULATE_DRUM_TOWER)
STATE_SET(TRIBE_ORANGE,TRUE, CP_AT_TYPE_MED_MAN_GET_WILD_PEEPS)

-- Don't build the drum tower
DELAY_MAIN_DRUM_TOWER(TRUE,TRIBE_ORANGE)
```

```
-- Spell entries
SET_SPELL_ENTRY(TRIBE_ORANGE, 0, M_SPELL_INSECT_PLAGUE, 25000, 64, 3, 0)
SET_SPELL_ENTRY(TRIBE_ORANGE, 1, M_SPELL_INSECT_PLAGUE, 25000, 64, 3, 1)

-- Marker entries
for i = 0,7 do
SET_MARKER_ENTRY(TRIBE_ORANGE, i, i, i, 0, 0, 2, 0)
end

function OnTurn()

-- Activating the marker entries
if (_gsi.Counts.GameTurn == 512) then
for i = 3,7 do
MARKER_ENTRIES(TRIBE_ORANGE, i, -1, -1, -1)
end
end

-- Attacking the blue tribe
if (every2Pow(10) and _gsi.Players[TRIBE_ORANGE].NumPeople > 30 and PLAYERS_PEOPLE_OF_TYPE(TRIBE_ORANGE, M_PERSON_WARRIOR) > 5 and PLAYERS_PEOPLE_OF_TYPE(TRIBE_ORANGE, M_PERSON_SUPER_WARRIOR) > 5) then
ATTACK(TRIBE_ORANGE, TRIBE_BLUE, 10, ATTACK_BUILDING, -1, 999, M_SPELL_WHIRLWIND, M_SPELL_LIGHTNING_BOLT, M_SPELL_LIGHTNING_BOLT, ATTACK_NORMAL, 0, NO_MARKER, NO_MARKER, NO_MARKER)
end

-- Placing tower plans
if (_gsi.Counts.GameTurn == 512) then
BUILD_DRUM_TOWER(TRIBE_ORANGE, 0, 10)
BUILD_DRUM_TOWER(TRIBE_ORANGE, 10, 0)
BUILD_DRUM_TOWER(TRIBE_ORANGE, 240, 240)
BUILD_DRUM_TOWER(TRIBE_ORANGE, 230, 0)
end

-- Saving the shaman from drowning with a land bridge spell
    local shaman = getShaman(TRIBE_ORANGE)
    if (shaman ~= nil) then
        if (shaman.State == S_PERSON_DROWNING and spell_delay == 0) then
            SearchMapCells(CIRCULAR, 0, 0, 6, world_coord2d_to_map_idx(shaman.Pos.D2), function(me)
            if (is_map_elem_coast(me) > 0) then
                local c2d = Coord2D.new()
                map_ptr_to_world_coord2d(me, c2d)
                local c3d = Coord3D.new()
                coord2D_to_coord3D(c2d, c3d)
                createThing(T_SPELL, M_SPELL_LAND_BRIDGE, shaman.Owner, c3d, false, false)
                spell_delay = 36
                return false
            end
            return true
            end)
        end
    end

-- Attacking the blue shaman with blast or lightning spells
    local shaman = getShaman(TRIBE_ORANGE)
    if (shaman ~= nil) then
        ProcessGlobalTypeList(T_PERSON, function(t)
        if (t.Owner == TRIBE_BLUE) then
            if (get_world_dist_xyz(shaman.Pos.D3, t.Pos.D3) < 3092 + shaman.Pos.D3.Ypos*3) then
                 if (spell_delay == 0 and is_thing_on_ground(shaman) == 1) then
                    createThing(T_SPELL, M_SPELL_BLAST, shaman.Owner, t.Pos.D3, false, false)
                    spell_delay = 24
                    return false
                end
            elseif (get_world_dist_xyz(shaman.Pos.D3, t.Pos.D3) < 6000 + shaman.Pos.D3.Ypos*3 and t.Model == M_PERSON_MEDICINE_MAN) then
                if (spell_delay == 0 and is_thing_on_ground(shaman) == 1) then
                    createThing(T_SPELL, M_SPELL_LIGHTNING_BOLT, shaman.Owner, t.Pos.D3, false, false)
                    spell_delay = 60
                    return false
                end
            end
        end
        return true
        end)
    end

-- Intelligent conversion of wildmen
    if (every2Pow(6)) then
        local shaman = getShaman(TRIBE_ORANGE)
        if (shaman ~= nil) then
            ProcessGlobalTypeList(T_PERSON, function(t)
            if (t.Model == M_PERSON_WILD) and (get_world_dist_xyz(shaman.Pos.D3, t.Pos.D3) < 8192 + shaman.Pos.D3.Ypos*3) then
```

```
                createThing(T_SPELL, M_SPELL_CONVERT_WILD, shaman.Owner, t.Pos.D3, false, false)
                return false
            end
            return true
            end)
        end
    end

-- Changing the spell delay every turn
    if (spell_delay > 0) then
    spell_delay = spell_delay - 1
    end

-- Ending the OnTurn() function
end
```

Case: Improved shaman-only attacks
If we combine all the previous knowledge on LUA scripting, we can script a smart shaman-only attack on a player base, just like we know them from multiplayer games.
We will do this in several steps:

- Script a shaman-only attack using the ATTACK command or by MOVE_SHAMAN_TO_MARKER()
- Count the number of enemy buildings around the shaman using SearchMapCells()
- Connect this to a spell action at an intelligent position using createThing()

We don't try to destroy single buildings using an earthquake spell.
Earthquake spells should be cast in between buildings.

Tips:

- Try your spell action with tornado first, where the shaman casts tornado on any nearby building. Create the spell at the building's position.
- To find an intelligent map position for the earthquake, use the count_enemy_buildings function we've written. Use a small radius and connect your building count with an if statement to find a small area with high enemy building density. Create the eq spell there.

Furthermore, include some real AI shaman power into your script:

- Kill the player's shaman with lightning
- Blast enemy followers that are sent to defend

Finally, build in some logs to help you keep track of where scripting goes wrong.
To include a log, use:
log("Message")

Then access your logs by using Shift+F1 --> Debug output.

Good luck!

Case solution

SPOILER: HIDE

```
-- Populous the Beginning: LUA Scripting Case: Advanced Scripting

-- Import modules and include files
import(Module_System)
import(Module_Players)
import(Module_Defines)
import(Module_PopScript)
import(Module_Game)
import(Module_Objects)
import(Module_Map)
import(Module_Math)
include("UtilPThings.lua")
include("UtilRefs.lua")

-- Variables we're going to use
spell_delay = 0
count = 0
building_count = 0
```

```
-- Initiate the orange tribe
computer_init_player(_gsi.Players[TRIBE_ORANGE])

-- Set spell attributes for orange
PThing.SpellSet(TRIBE_ORANGE, M_SPELL_CONVERT_WILD, TRUE, FALSE)
PThing.SpellSet(TRIBE_ORANGE, M_SPELL_BLAST, TRUE, FALSE)
PThing.SpellSet(TRIBE_ORANGE, M_SPELL_LAND_BRIDGE, TRUE, FALSE)
PThing.SpellSet(TRIBE_ORANGE, M_SPELL_INSECT_PLAGUE, TRUE, FALSE)

-- Set building attributes for orange
PThing.BldgSet(TRIBE_ORANGE, M_BUILDING_TEPEE, TRUE) -- Hut
PThing.BldgSet(TRIBE_ORANGE, M_BUILDING_DRUM_TOWER, TRUE)
PThing.BldgSet(TRIBE_ORANGE, M_BUILDING_WARRIOR_TRAIN, TRUE)
PThing.BldgSet(TRIBE_ORANGE, M_BUILDING_TEMPLE, TRUE)
PThing.BldgSet(TRIBE_ORANGE, M_BUILDING_SUPER_TRAIN, TRUE)

-- Set internal attributes for orange
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_EXPANSION, 30)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_HOUSE_PERCENTAGE, 50)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_MAX_BUILDINGS_ON_GO, 10)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_PREF_WARRIOR_TRAINS, 1)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_PREF_WARRIOR_PEOPLE, 20)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_PREF_SUPER_WARRIOR_TRAINS,1)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_PREF_SUPER_WARRIOR_PEOPLE, 40)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_MAX_TRAIN_AT_ONCE, 5)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_AWAY_BRAVE, 0)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_AWAY_WARRIOR, 50)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_AWAY_SUPER_WARRIOR, 50)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_AWAY_RELIGIOUS, 0)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_AWAY_SPY, 0)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_AWAY_MEDICINE_MAN, 100)
WRITE_CP_ATTRIB(TRIBE_ORANGE, ATTR_MAX_ATTACKS, 999)

-- Set states for orange
STATE_SET(TRIBE_ORANGE, TRUE, CP_AT_TYPE_FETCH_WOOD)
STATE_SET(TRIBE_ORANGE, TRUE, CP_AT_TYPE_CONSTRUCT_BUILDING)
STATE_SET(TRIBE_ORANGE, TRUE, CP_AT_TYPE_TRAIN_PEOPLE)
STATE_SET(TRIBE_ORANGE, TRUE, CP_AT_TYPE_AUTO_ATTACK)
STATE_SET(TRIBE_ORANGE, TRUE, CP_AT_TYPE_POPULATE_DRUM_TOWER)
STATE_SET(TRIBE_ORANGE,TRUE, CP_AT_TYPE_MED_MAN_GET_WILD_PEEPS)


-- Don’t build the drum tower
DELAY_MAIN_DRUM_TOWER(TRUE,TRIBE_ORANGE)

-- Spell entries
SET_SPELL_ENTRY(TRIBE_ORANGE, 0, M_SPELL_INSECT_PLAGUE, 25000, 64, 3, 0)
SET_SPELL_ENTRY(TRIBE_ORANGE, 1, M_SPELL_INSECT_PLAGUE, 25000, 64, 3, 1)

-- Marker entries
for i = 0,7 do
SET_MARKER_ENTRY(TRIBE_ORANGE, i, i, i, 0, 0, 2, 0)
end

function OnTurn()

    -- Activating the marker entries
    if (_gsi.Counts.GameTurn == 512) then
        for i = 3,7 do
        MARKER_ENTRIES(TRIBE_ORANGE, i, -1, -1, -1)
        end
        log("Just activated the marker entries!")
    end

    -- Attacking the blue tribe
    if (every2Pow(10) and _gsi.Players[TRIBE_ORANGE].NumPeople > 30 and PLAYERS_PEOPLE_OF_TYPE(TRIBE_ORANGE, M_PERSON_WARRIOR) > 5 and PLAYERS_PEOPLE_OF_TYPE(TRIBE_ORANGE, M_PERSON_SUPER_WARRIOR) > 5) then
        log("I will attack!")
        ATTACK(TRIBE_ORANGE, TRIBE_BLUE, 10, ATTACK_BUILDING, -1, 999, M_SPELL_WHIRLWIND, M_SPELL_LIGHTNING_BOLT, M_SPELL_LIGHTNING_BOLT, ATTACK_NORMAL, 0, NO_MARKER, NO_MARKER, NO_MARKER)
    end

    -- Our shaman-only attack
    if (every2Pow(10)) then
        local shaman = getShaman(TRIBE_ORANGE)
        if (shaman ~= nil) then
            if IS_SHAMAN_AVAILABLE_FOR_ATTACK(TRIBE_ORANGE) then
                log("Moving shaman to marker...")
                MOVE_SHAMAN_TO_MARKER(TRIBE_ORANGE, 13)
                log("Here comes a shaman-only attack!")
            end
```

```
    end
end

-- Our shaman behaviour during the shaman-only attack
local shaman = getShaman(TRIBE_ORANGE)
if (shaman ~= nil and spell_delay == 0) then
    SearchMapCells(CIRCULAR,0,0,10,world_coord2d_to_map_idx(shaman.Pos.D2),function(me)
    me.MapWhoList:processList(function(t)
    if (t.Type == T_BUILDING) then
        if (t.Owner ~= TRIBE_ORANGE) then
            building_count = building_count + 1
            log("Building count: " .. building_count)
            local c2d = Coord2D.new()
            map_ptr_to_world_coord2d(me, c2d)
            local c3d = Coord3D.new()
            coord2D_to_coord3D(c2d, c3d)
            if (building_count > 1) then
                no_enemy_buildings = count_enemy_buildings(c3d,2)
                if (no_enemy_buildings > 1 and spell_delay == 0) then
                    createThing(T_SPELL, M_SPELL_EARTHQUAKE, shaman.Owner, c3d, false, false)
                    spell_delay = 120;
                    return false
                end
            end
        end
    end
    return true
    end)
    return true
    end)
    building_count = 0
end

-- Placing tower plans
if (_gsi.Counts.GameTurn == 512) then
    log("Trying to build towers!")
    BUILD_DRUM_TOWER(TRIBE_ORANGE, 0, 10)
    BUILD_DRUM_TOWER(TRIBE_ORANGE, 10, 0)
    BUILD_DRUM_TOWER(TRIBE_ORANGE, 240, 240)
    BUILD_DRUM_TOWER(TRIBE_ORANGE, 230, 0)
end

-- Saving the shaman from drowning with a land bridge spell
local shaman = getShaman(TRIBE_ORANGE)
if (shaman ~= nil) then
    if (shaman.State == S_PERSON_DROWNING and spell_delay == 0) then
        SearchMapCells(CIRCULAR, 0, 0, 6, world_coord2d_to_map_idx(shaman.Pos.D2), function(me)
        if (is_map_elem_coast(me) > 0) then
            local c2d = Coord2D.new()
            map_ptr_to_world_coord2d(me, c2d)
            local c3d = Coord3D.new()
            coord2D_to_coord3D(c2d, c3d)
            createThing(T_SPELL, M_SPELL_LAND_BRIDGE, shaman.Owner, c3d, false, false)
            spell_delay = 36
            return false
        end
        return true
        end)
    end
end

-- Attacking the blue shaman with blast or lightning spells
local shaman = getShaman(TRIBE_ORANGE)
if (shaman ~= nil) then
    ProcessGlobalTypeList(T_PERSON, function(t)
    if (t.Owner == TRIBE_BLUE) then
        if (get_world_dist_xyz(shaman.Pos.D3, t.Pos.D3) < 3092 + shaman.Pos.D3.Ypos*3) then
             if (spell_delay == 0 and is_thing_on_ground(shaman) == 1) then
                createThing(T_SPELL, M_SPELL_BLAST, shaman.Owner, t.Pos.D3, false, false)
                spell_delay = 24
                return false
            end
        elseif (get_world_dist_xyz(shaman.Pos.D3, t.Pos.D3) < 6000 + shaman.Pos.D3.Ypos*3 and t.Model == M_PERSON_MEDICINE_MAN) then
            if (spell_delay == 0 and is_thing_on_ground(shaman) == 1) then
                createThing(T_SPELL, M_SPELL_LIGHTNING_BOLT, shaman.Owner, t.Pos.D3, false, false)
                spell_delay = 60
                return false
            end
        end
```

```
        end
        return true
        end)
    end

    -- Intelligent conversion of wildmen
    if (every2Pow(6)) then
        local shaman = getShaman(TRIBE_ORANGE)
        if (shaman ~= nil) then
            ProcessGlobalTypeList(T_PERSON, function(t)
            if (t.Model == M_PERSON_WILD) and (get_world_dist_xyz(shaman.Pos.D3, t.Pos.D3) < 8192 + shaman.Pos.D3.Ypos*3) then
                createThing(T_SPELL, M_SPELL_CONVERT_WILD, shaman.Owner, t.Pos.D3, false, false)
                return false
            end
            return true
            end)
        end
    end

    -- Changing the spell delay every turn
    if (spell_delay > 0) then
    spell_delay = spell_delay - 1
    end

-- Ending the OnTurn() function
end

-- Function to count enemy buildings from c3d coordinate
function count_enemy_buildings(c3d,radius)
local count = 0;
SearchMapCells(CIRCULAR,0,0,radius,world_coord3d_to_map_idx(c3d),function(me)
me.MapWhoList:processList(function(t)
if (t.Type == T_BUILDING) then
if (t.Owner ~= TRIBE_ORANGE) then
                count = count+1
end
end
return true
end)
return true
end)
return count
end
```

And that's a wrap on this tutorial! Give yourself a big compliment, you deserve one. I could definitely use a drink. Cheers!

Re: Scripting in Populous Beta: LUA tutorial

Posted: **Thu May 28, 2020 10:23 pm**
by **TheGabber**
Kudos!

Re: Scripting in Populous Beta: LUA tutorial

Posted: **Thu May 28, 2020 11:09 pm**
by **[D]ystopia**
nice thread
i finished first post, but didn't understand some stuff, if someone can help with comments on each line with (???)

-- Marker entries
for i = 0,7 do (???)
SET_MARKER_ENTRY(TRIBE_ORANGE, i, i, i, 0, 0, 2, 0) (???)
end

function OnTurn()

-- Activating the marker entries
if (_gsi.Counts.GameTurn == 512) then
for i = 3,7 do (???)

MARKER_ENTRIES(TRIBE_ORANGE, i, -1, -1, -1) (???)
end
end

Re: Scripting in Populous Beta: LUA tutorial

Posted: **Fri May 29, 2020 8:31 am**
by **410172_Chief**
Nice to see you trying out some LUA scripting! Of course I'll help.

What you are looking at is a for loop. The syntax for a for loop in LUA is:

```
for i = x1,x2 do
Your code here
end
```

Where x1 and x2 should both be integers and x2 should be larger than x1.

What does a for loop do? It loops through the code that is inside, and changes the value of i for every turn.

So:

```
for i = 0,2 do
code
end
```

does the same as:

```
i = 0
code
i = 1
code
i = 2
code
```

In that example with the marker entries, I use the for loop to set all the marker entries at once.

```
for i = 0,7 do
SET_MARKER_ENTRY(TRIBE_ORANGE, i, i, i, 0, 0, 2, 0)
end
```

is the same as:

SPOILER: HIDE

```
SET_MARKER_ENTRY(TRIBE_ORANGE, 0, 0, 0, 0, 0, 2, 0)
SET_MARKER_ENTRY(TRIBE_ORANGE, 1, 1, 1, 0, 0, 2, 0)
SET_MARKER_ENTRY(TRIBE_ORANGE, 2, 2, 2, 0, 0, 2, 0)
SET_MARKER_ENTRY(TRIBE_ORANGE, 3, 3, 3, 0, 0, 2, 0)
SET_MARKER_ENTRY(TRIBE_ORANGE, 4, 4, 4, 0, 0, 2, 0)
SET_MARKER_ENTRY(TRIBE_ORANGE, 5, 5, 5, 0, 0, 2, 0)
SET_MARKER_ENTRY(TRIBE_ORANGE, 6, 6, 6, 0, 0, 2, 0)
SET_MARKER_ENTRY(TRIBE_ORANGE, 7, 7, 7, 0, 0, 2, 0)
```

Let's look at that SET_MARKER_ENTRY syntax for the first line: SET_MARKER_ENTRY(TRIBE_ORANGE, 0, 0, 0, 0, 0, 2, 0)
First number: the entry of the marker entry (0, 1, 2 etc. for the amount of marker entries you want to set)
Second and third number: the two markers the troops will patrol between. These are set using the world editor and can be any number depending on how many markers you have placed. If I set the same number, like I did in my examples, troops will patrol at one marker, so hin and forth on the spot.
Fourth-Seventh number: the number of each troop that will patrol. The numbers are for braves, warriors, firewarriors, preachers. No spies, sorry.

So, looking at SET_MARKER_ENTRY(TRIBE_ORANGE, 0, 0, 0, 0, 0, 2, 0), that means:
This is marker entry number 0. Patrol 0 braves, 0 warriors, 2 firewarriors and 0 preachers between markers 0 and 0.
Or in other words: patrol two firewarriors at the spot on marker 0.

We set these marker entries at the start of the game. During the game, we need to activate them to make the AI do what's in those lines. There's where we have the line:

```
MARKER_ENTRIES(TRIBE_ORANGE, entry1, entry2, entry3, entry4)
```

If you want to activate less than 4 marker entries, you use -1. I can use either:

```
MARKER_ENTRIES(TRIBE_ORANGE, 0, 1, 2, 3)
```

or

```
MARKER_ENTRIES(TRIBE_ORANGE, 0, 1, -1, -1)
MARKER_ENTRIES(TRIBE_ORANGE, 2, 3, -1, -1)
```

or

```
MARKER_ENTRIES(TRIBE_ORANGE, 0, -1, -1, -1)
MARKER_ENTRIES(TRIBE_ORANGE, 1, -1, -1, -1)
MARKER_ENTRIES(TRIBE_ORANGE, 2, -1, -1, -1)
MARKER_ENTRIES(TRIBE_ORANGE, 3, -1, -1, -1)
```

to do exactly the same thing. In theory. But in practise, you may experience different results.

After some tests and a lot of frustration, I chose for the last way to activate my entries.
So if I need to activate markers 3 till 7, I write:

```
MARKER_ENTRIES(TRIBE_ORANGE, 3, -1, -1, -1)
MARKER_ENTRIES(TRIBE_ORANGE, 4, -1, -1, -1)
MARKER_ENTRIES(TRIBE_ORANGE, 5, -1, -1, -1)
MARKER_ENTRIES(TRIBE_ORANGE, 6, -1, -1, -1)
MARKER_ENTRIES(TRIBE_ORANGE, 7, -1, -1, -1)
```

Or in a for loop:

```
for i = 3,7 do
MARKER_ENTRIES(TRIBE_ORANGE, i, -1, -1, -1)
end
```

And that is what you see in the case solution.

Hope this helps!
