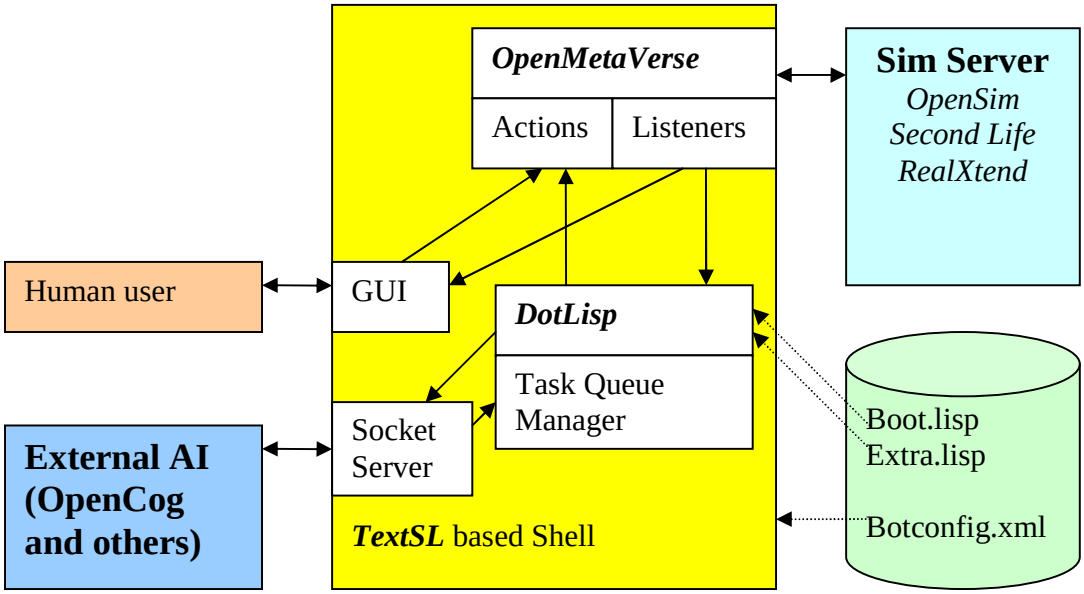
Cogbot: an Opensimulator avatar interface for external AI systems

By Kino Coursey

OVERVIEW

OpenSim4OpenCog/Cogbot is a server written in C# that connects to a avatar acting as "bot" in OpenSim/Second Life/RealXtend. The basic shell is based on the TextSL client for visually impared users. The system can accept in XML (or a pointer to a file with XML commands). The system is updated to use the latest Openmetaverse library to interface with the sims. It also has an internal LISP interperter (based on DotLisp) and task management queue. The task queue allows operations to be persistent, by automatically requing the evaluation of the Lisp code fragment.



Cogbot Basic Design

COMPILING 4 WINDOWS

Cogbot was developed using Windows XP and Visual Studio 2008.

You can use Visual Studio C# Express which is free from Microsoft.
http://microsoft.com/express

To compile, Open Visual Studio and open the cogbot.sln file
located in the project's root directory.
This should open Cogbot in the IDE and show the several sub projects.

In the Solution Explorer, make sure that the "cogbot" project
is bold (the startup project) and if it is not make it so by right
click the project and select "Set as Startup Project"

To build, go to Build -> Build Solution (or just press F6)

On the status bar at the bottom of the Visual Studio window - you should see "Build succeeded", once the system is finished compiling.

Time to Configure the system.

COMPILING 4 LINUX

BIG TODO:
Awaiting definition of how to compile on a linux machine.

We do know the follwing will be needed:

* nANT
* Mono 1.2.4 or better (with .NET 2.0 package)
* gmcs (known as the mono-gmcs package on Debian)

Once installed, cd to the main directory of your copy of cogbot
and run "nant" in the command line.

After 10-50 seconds of compiling you should see "Build Succeeded".
(Or something like it).

CONFIGURATION

To configure the system you will need to provide or modify the botconfig.xml file. In particular you will want to have an avatar already defined on the sim you will be using. The "tcp" paramaters define the IP address of the Cogbot service. The other parameters define the login for the virtual viewer used by the sytem to connect to the avatar in the sim.
You can find the botconfig.xml file in
.\cogbot\bin\Debug\botconfig.xml
.\cogbot\bin\Release\botconfig.xml
.\cogbot\bin\botconfig.xml
Depending on the build or run options you use.

The key parameters to change are

```
  <tcpPort>5555</tcpPort>
  <tcpIPAddress>127.0.0.1</tcpIPAddress>

  <firstName>My</firstName>
  <lastName>Bot</lastName>
  <password>MyBotPassword</password>
  <simURL>http://localhost:8002/</simURL>

  <startupLisp>(thisClient.ExecuteCommand "login")</startupLisp>
```

Once started you can fill in the login parameters using the login form. Any changes there will be saved to the config file (which will be created if it does not already exist).

New: <startupLisp> allows an initial Lisp expression to be evaluated after the task manager starts executing and has loaded its boot files.

MANUAL OPERATION

Since Cogbot is based on TextSL, you can drive things manually.
You can start the system by executing:
\cogbot\bin\cogbot.exe
which uses *\cogbot\bin\botconfig.xml*

The menu system is fairly simple for now
File -> Exit
Client->Login
Client->Logout

Under "Client->Login" is a form for modifying the username and password. Information on the status of commands and messages are provided in the central text box. User input is in the white text entry box and executed by pressing the submit button.

An example of the report of logging in :

```
-------------------------------------------
About to initialize port.
Listening for a connection... port=5555
LoginForm Start Attempt 0
TextForm Network_OnLogin : [ConnectingToSim] Connecting to simulator...
LoginForm NOT logged in for reason:0 Timed out
TextForm Network_OnLogin : [ConnectingToSim] Connecting to simulator...
You see 0 people.
TextForm client_OnLogMessage: Info <Kotoko Irata>: Connecting to (127.0.0.1:9011)
TextForm Objects_OnNewAvatar:
TextForm Objects_OnNewAvatar:
TextForm client_OnLogMessage: Info <Kotoko Irata>: Received a region handshake for
Citadel (127.0.0.1:9011)
TextForm Network_OnSimConnected: Citadel (127.0.0.1:9011)
TextForm Network_OnLogin : [Success] Welcome to OGS
TextForm Objects_OnNewAvatar:
TextForm Avatars_OnLookAt: 7dbd61c6-90cf-49df-bf77-94f5a7223c19 to 7dbd61c6-90cf-49df-
bf77-94f5a7223c19 at 7dbd61c6-90cf-49df-bf77-94f5a7223c19 with type FreeLook duration 2
You see the objects 1: DIR, 2: Deck, 3: Desk, 4: StucoBeachHouse, 5: WallSectionSolid, 6:
FW_Steps, 7: Landscape, 8: HellBox, 9: Blue, 10: marble, 11: six, 12: one, 13: DaxSymbol,
14: 2_Walls, 15: ML866, and 16: Window02_Tall,4-Pane.
TextForm Avatars_OnLookAt: 7dbd61c6-90cf-49df-bf77-94f5a7223c19 to 7dbd61c6-90cf-49df-
bf77-94f5a7223c19 at 7dbd61c6-90cf-49df-bf77-94f5a7223c19 with type FreeLook duration 2
Logged in successfully.
-------------------------------------------
```

The results of typing help:

```
-------------------------------------------
login: Login to Secondlife
logout: Logout from Secondlife
stop: Cancels a particular action
teleport: Teleport to a location.
describe: Describe location, people, objects, or buildings.
say: Say a message for everyone to hear.
whisper: Whisper a message to a user.
help: Print this help message.
sit: Sit on the ground or on an object.
stand: Stand up.
jump: Jump.
crouch: Crouch.
mute: Toggle Mute or unmute a user
move: Move to a person or object, or in a direction.
use: Use an item from inventory.
fly: You start flying.
stop-flying: You stop flying.
where: Finds out in which direction an object or a building or a person is.
locate: Gives the coordinates of where you are.
follow: Start or stop following a user.
stop following: Start or stop following a user.
stop-following: Start or stop following a user.
tutorial1: Teaches you how to navigate using basic commands move, sit, stand
-------------------------------------------------
```

describe

```
------------------
You are in Citadel.
You see 2 people.
You see the objects 1: CEMA, 2: Sit, 3: Clever, 4: 5_Flooring, 5: Boardman Bedroom, 6:
Wood, 7: Keyboard, 8: Medical, 9: House03_PostHickory, 10: Low, 11: CLEAR, 12: marble
end, 13: Banana, 14: Banana Plant, 15: Clay, and 16: Imperial.
You see 2 buildings.
You see 2 people.
------------------
```

```
describe people
------------------
You see one person: 1: Daxxon Kinoc.
------------------
describe Daxxon
------------------
Daxxon Kinoc is standing in Citadel.
Daxxon Kinoc is 2.112267 distant.
------------------
describe Clever
------------------
Clever Zebra Small Office (left): http://www.cleverzebra.com/
This object is for sale for L10
------------------
------------------------------------------------------------
and so on.

To get the inventory you can issue "describe inventory".
The system also accepts
  "use <inventory-item-name> to wear"
  "use <inventory-item-name> to animation-start"
  "use <inventory-item-name> to animation-stop"

The system recives events from the sim like what is being looked at.

TextForm Avatars_OnLookAt: 7dbd61c6-90cf-49df-bf77-94f5a7223c19
              to da717612-e98f-469b-b6c3-f9145ca84e64
              at da717612-e98f-469b-b6c3-f9145ca84e64
              with type Focus duration 1.701412E+38
  (TARGET IS SELF)

 meaning that the user is looking at the bot.
```

External Access

The whole point is to be something an AI running as an external process can use to get into a virtual world. To do this the system accepts commands via the socket it is listening to.

For testing we setup a file on a server with the commands to the lisp interperter XML encoded.

The construct *(thisClient.ExecuteCommand "<command-string>")* will execute any command-string you could execute manually from the client.

Another useful fragment to know is:
(set thisTask.requeue (to-bool false))
which means not to requeue this code fragment for later execution.

If you did want this fragment to be constantly requeued you would use
(set thisTask.requeue (to-bool true))

So the dotlisp equivelent of "Hello World" would be:

```
(block
  (thisClient.ExecuteCommand "say Hello World with a L I S P 2 …")
  (thisClient.msgClient "(knows world (exists me))" )
  (set thisTask.requeue (to-bool-false))
 )
```

Which we then translate into XML and stick in a URL accessible file.

```
--------------------------------------------------
testlisp2.xlsp
--------------------------------------------------
<?xml version="1.0" encoding="utf-8" ?>
<op name="block">
   <op name="thisClient.ExecuteCommand">
```

```
   "say Hello With a L I S P 2..."
 </op>
   <op name="thisClient.msgClient">
    "(knows world (exists me))"
  </op>
 <op name="set">
  <arg name="1">thisTask.requeue</arg>
  <arg name="2">(to-bool false)</arg>
  </op>
</op>
```

Using Putty we connect via a raw socket to the Cogbot server
(in our case localhost:5555).

And assuming we have a command stored as an XML file on a server
we can simply type in the URL
http://pandor6/temp/testlisp2.xlsp

The system will return
```
'(enqueued)(knows world (exists me))
```
execute the command and the bot will say "Hello With a L I S P 2..."
in-world.

```
SockClient:http://pandor6/temp/testlisp2.xlsp
EvaluateXmlCommand :http://pandor6/temp/testlisp2.xlsp
XML2Lisp =>'(block(thisClient.ExecuteCommand
   "say Hello With a L I S P 2..."
  )(thisClient.msgClient
    "(knows world (exists me))"
   ) )'
 taskTick Results>nil
 taskTick continueTask=False
Kotoko Irata says, "Hello With a L I S P 2...".
```

To find out more about the lisp system see
 \cogbot\dotlisp\dotlisp.html

The system should automatically load
 \cogbot\bin\boot.lisp
 \cogbot\bin\extra.lisp

TODO: 2008-08-24

At this time all the subprojects "aligned" to the point to allow Cogbot to function. OpenSim and OpenMetaverse are currently moving and OpenCog is being written. So there are many things that are needed besides just keeping up with new releases.

* (DONE) Provide feedback to the client. This is as simple as adding a "msgClient" method to the "thisClient" object.

* (INWORK) Patch the events hooks through using the "msgClient" function. Each event would simply post a lisp code fragment to the task queue.

Working on the "heard" listener first by adding to Chat.cs:
```
    parent.enqueueLispTask("(thisClient.msgClient \"(heard (" + fromName + ") '" + message + "' )\" )");
```

when I say 'hi there' to the bot in-world cogbot returns to the tcp client:
```
(heard (Daxxon Kinoc) 'hi there' )
```

This could of course be changed into calls like "(on_chat (fromName) message)" where "on_chat" is a lisp function, and could be redefined by the AI using Cogbot. Also such a function could translate into something like XML or use system provided methods to do so.

In general define all the events that occur, map them to function calls, then have a file for each type of message each type of client AI expects. So OpenCog would have one, OpenCyc its own, Soar its own, etc...

* (SEMI-DONE) Set the system up for auto-login. The command line does have login and logout. However setting the other parameters would be nice. Also the sim may report that the system is already logged on, and may require a second attempt.

*(DONE) Lisp initialization string in the config file, to be executed on start up.

Adding
```
<startupLisp>(thisClient.ExecuteCommand "login")</startupLisp>
```
to the startup file causes the system to automatically login as it's first action after booting. This addresses the previous TODO but requires more lisp.

Something like:

```
<startupLisp>
(block
  (set thisClient.config.simURL "http://myothersim.org:8002/")
  (set thisClient.config.firstName "EvilTwin")
  (thisClient.ExecuteCommand "login")
  )
</startupLisp>
```

or load additional config or operational files.

* Having multiple bots. Currently the system provides single bot access, with each socket serving one bot. Being able to run multiple bots would be nice.

* Time based requeuing . The system can requeue a task but it is not timed on an individual task basis, using a simple round robin scheduler.

*Document the objects the lisp system has access to. These are basically the same as the client object in Cogbot since "lisp thisClinent" == "cogbot client" in the code. However the method set is still evolving.

* More graceful socket shutdown.

* RealXtend functions. The system uses OpenMetaverse and thus should work with the systems it works with. However some RealXtend function may go beyond the

```
set supported by Secondlife/Opensim. This is more of a wish for OpenMeteverse
extension.

* Intelligently support lisp over the socket. Ideally, lisp, XML-encoded lisp,
and the current pointer to XML files.

* Port dotLisp to Opensim. I already ported Yield Prolog, and dotLisp is
"safely dead", meaning it works but is not being rapidly extended (like
everything else). So it would provide an AI-ish scripting language which
can access sim methods, being both simple, dynamic and complete. Might
provide a method to let AI's script objects...

* Connecting up OpenCog. In general create the set of "mapping files" in lisp
described above for any set of external AI programs.

* Provide features to specify login location coordinates.
```

NOTES:

- Simply affordable: With the general ability to evaluate lisp functions in strings or use the strings as URL's to functions, one can implement a form of affordance processing similar to The Sims. In The Sims new objects transmitted to passing agents code fragments which included conditions checked against the agents internal state blackboard. A message from a refrigerator might in effect say "if your hunger is high and you like fruit then open me by grasping handle 3234" or some such. While most learning AI systems might consider it "cheating" it might be used to simulate extra perceptions like smell.

Links:

- Initial CogBot at Google Code
  - http://code.google.com/p/opensim4opencog/
- TextSL at Google Code
  - http://code.google.com/p/textsl/
- DotLisp
  - http://dotlisp.sourceforge.net/dotlisp.htm
- OpenMetaverse
  - http://openmv.org/wiki/Main_Page
  - http://www.libsecondlife.org/wiki/Main_Page
- OpenSimulator
  - http://www.opensimulator.org
  - http://osgrid.org/forums/
- Second Life
  - http://secondlife.com/
- RealXtend
  - http://www.realxtend.org/
  - http://www.rexdeveloper.org/forum/