Boris.NET User Guide

Boris.NET is programmed for .NET framework version 3.5 and Visual Studio 2008 and compiled to a DLL file. It has also been tested on Mono platform [http://www.mono-project.com/] which allows using it in other operating systems such as Linux and Mac.

In this user guide we will look at different functionalities of Boris.NET.

Adding Boris.NET to a project

To add Boris.NET to the project we used “Add reference” from the “Project menu” in Visual Studio 2008. We used the “Browse” tab to find and add “Boris.dll” to the project.

We also needed to add Boris.NET namespace to the project. We did this by adding the following code to the beginning of our Main Program file. This file is called “Program.cs” in a C# console application.

```
using Boris;
```

Creating a Portal

Before creating a new agent we need to have a portal as all agents need to be connected to a portal.

To create a portal we need to create an instance of portal class. The following codes will be placed in the Main() method of our application.

```
Portal myPortal = new Portal("London");
```

This will create a myPortal object. Every agent, including portals, need to have a name. This portal is called London.

Creating a new Agent

An Agent as is an object instantiated from MetaAgent class. To create new Agent the following command is used.

```
MetaAgent agentA = new MetaAgent("Ali");
```

Now we have agent agentA. This agent’s name is Ali.

Connecting a Portal

Portals can connect to routers or other portals, but not to both of them at the same time.

Connecting to a Router

To connect to a router we need to know the IP and the port of the machine where the router is located. For this example we assume that the router's IP is 80.81.82.83 and it is advertising on port 1234.

```
myPortal.Connect("80.81.82.83", 1234);
```

If the router is located on the localhost, there is no need to mention the IP.

```
myPortal.Connect(1234);
```

If the router is not available on in the IP and the port mentioned, the portal will not be able to connect to the router. This will result in the application termination and the log file will register the reason. The error message logged will look as follows:

```
Log Entry : 09:04:10 PM, 25 March 2009

Router on the chosen port is not available or not advertising

-----------------------------------

Log Entry : 09:04:10 PM, 25 March 2009

Application aborted

-----------------------------------
```

It is always necessary to make sure that the router is advertising and the correct IP and port are entered in the application.

Connecting to another Portal

To connect to another portal we need to use the name of the portal object. In this example we create a new portal and connect myPortal to it.

```
Portal superPortal = new Portal("UK");

myPortal.Connect(superPortal);
```

Now myPortal is connected to superPortal.

Registering Agents

Agents can be registered before or after a portal connects to a router or another portal. The following code is use to register an agent

```
myPortal.AddAgent(agentA);
```

This will add agentA to myPortal.

Sending message

To send a message, an agent needs to know the other agent's name. The following code will show how an agent (agentA), whose name is Ali, sends the message "Hi, I am Ali!" to another agent called Sam.

```
agentA.SendMessage("Sam", "Hi, I am Ali!");
```

Receiving Message

In Boris the behaviour of an agent is a function associated to that agent. That function will be called when they receive a message. To call the function when an agent receives a message the following code is used:

```
agentA.MessageReceived +=

new MetaAgent.MessageReceivedHandler(agentA_MessageReceived);


static void agentA_MessageReceived(Communication message)

{

       Console.WriteLine("{0} said {1}",

              message.Sender, message.Body);

}
```

In the above example agentA_MessageReceived is the function that describes the behavior of agentA. In here we only publish the name of the sender, message.Sender, and the message it sends to us, message.Body, to the command line. agentA_MessageReceived needs to be programmed in a way to describe the behaviour of agentA.