OpenPAR README

1 Introduction

PAR provides high-level semantics of actions and objects. It can be used as a part of an AI system for animation and other applications. Detailed information about PAR can be found here.

PAR actions and objects are stored in a MySQL database called the Actionary. We are supplying a number of uninstantiated PARs (uPARs) actions and objects, but the provided tutorials outline how to add others. MySQL's C++ connection interface to the Actionary.

Python is essentially used as the logic system for PAR. There are six files associated with each action to test the conditionals, as well as express the execution steps and assertions.

More information about the workings of PAR can be found on the website and in the provided tutorials.

2 Installation

These setup instructions have been written for a 64bit compilation of PAR using visual studio 2013, mysql version 5.6 64bit and python version 2.7.10 64bit. The subsequent tutorials are also based off of this configuration. We believe the system could be run using other configurations (for example, a 32-bit operating system), but we have not tested them. It is extremely important that the architecture for mysql and python are the same as PAR's compilation. Failure to insure this will result in errors.

1. **Notes:**

2. IF you only wish to use PAR in visual studio's release mode, then you can use the provided quickstart guide, which provides a faster and easier compilation of constituent components of the software.

2.1 MySQL

1. Install MySQL. We are currently using MySQL 5.6

   (a) Do a complete install.
   (b) Configure the server once installation has finished.
      i. Proceed with a *Detailed Configuration*, selecting the following:
         - Developer machine
         - Multifunctional Database
         - Decision Support (DSS)/OLAP
         - Include Bin Directory in Windows PATH
         - Leave all other options on default
      ii. Set the *root password* to *root* when prompted.

Optional Install MySQL GUI tools.
These are nice GUIs for examining and administrating the database.

2. Install the database.
   (a) In the command prompt, `cd` into the *OpenPAR* directory.
   (b) Type: `mysql -u root -p`
   (c) Enter the *root password* you created earlier.
   (d) Now in MySQL type:

```
create database openpardb;
quit;
```

(e) In the command prompt again, type:

```
mysql -u root -p openpardb < actionary.sql
```

or

```
mysql -u user_name --password=your_password database_name < actionary.sql
```

Once the MySQL database has been created and properly configured, create an environmental variable MYSQL_ROOT, which should point to the root folder of your MySQL server.

2.2 Boost

The Boost Library is used for building MySQL's C++ connector. PAR does not inherently use Boost, but it is necessary for PAR to be run in Visual Studio's debug mode.

1. Download the Boost Libraries here.
2. Follow this tutorial to install Boost onto the system.
3. Create an environmental variable *Boost_Root* to point to the start of your Boost directory.

2.3 MySQL C++ connector

PAR uses MySQL default connections to c++, which can be downloaded here. We provide a debug library file and dll for a 64bit installation of version 1.1.6, and the installation of the connector creates a release version. Please make sure that the libraries built (mysqlcppconn.lib) are in folders name *lib debug* and *lib opt*, if your compiling does not do so already. Failure to do so will require manually changing the paths in several of the PAR solution files.

After installing msyql connector, create an environmental variable called *CONNECTOR_ROOT* that points to the root folder of your MySQL installation. This folder should contain in it both the connector folder and the folder of your installation of MySQL.

To build MySQL C++ Connector from source (essential to using the libraries in Visual Studio's Debug mode), please follow the tutorial here.

1. **Notes**
2. We found an issue in building MySQL C++ Connector for Visual Studio 2013 64bit. Building the connector for Visual Studio 2012 does not seem to have an impact on the use of the created .lib and .dll files.

2.4 Python

We are currently using Python 2.7.10 64bit.

1. Install Python to `C:\Python27` (default location). After installation, create an environmental variable PYTHON_ROOT which points to the directory that contains python.exe (as well as an include folder for Python.h).

2. If you wish to use PAR in Visual Studio's Debug mode, you will also need to download and compile the same version of Python that you installed. This should not be the version of Python that PAR links to (using the environmental variable), and is only used to compile the debug .lib (called Python27_d). A tutorial on how to compile python for windows can be found here.

Notes: (a) We have included both a debug and release version of Python2.7.10 lib and dll. The debug lib and dll are necessary to run PAR in debug mode in Visual Studio.

2.5 PAR Code

The PAR directory contains the PAR code.

2.6 PAR Code

The PAR directory contains the PAR code.
Compile all of the code in either Debug, Release or both.
The compilation order is: lwnets, database, agentProc, then your main project. Compile all of the code in either Debug, Release or both.
The compilation order is: lwnets, database, agentProc, then your main project.

actions Contains all of the Python files required for the PAR actions.

agentProc Contains the PAR Agent Process code. It includes an action queue and code for processing the PAR actions to ensure that they are executed in the right context. The AgentProc class can be sub-classed to create individual agents with different AIs.

database Contains the Actionary interface code and PAR data structures. Much of this code is MySQL calls. It also contains the code that interfaces with Python, and defines the action and object structures used in PAR.
Note: This project at the beginning of actionary.cpp you will want to change the database connection code to reflect your login info to the database that you set up.

libs The directory where the various libraries are compiled to. In any project that you create, these should all be linked together[1].

lwnets Contains the PaTNets used by the action queue and process manager. PaTNets are Parallel Transition Networks or parallel finite state machines.

If your installed component libraries have a different folder structure then the ones that we have described, you may have to change the paths in the system settings. For example, the MySQL connector include path may be "*CONNECTOR_ROOT*\include" or within database's solution. Please go through and check the include paths (for All Configurations) and the Linker paths (separately for Debug and Release).

3 Potential Issues

1. A common execution problem is not finding MySQL and Python dlls. The simplest solution is to include mysqlcppconn.dll (or mysqlcppconn_d.dll) and python27.dll (python27_d.dll) in the same directory as your executable.

[1] See tutorials