Tutorial 8
OGRE Integration

OGRE is an open source 3D graphics engine.
In this tutorial, we use OGRE as an example integration of Open PAR with a game or graphics engine. Most engines have similar structures, so our hope is that you can use this as a template for your own system requirements. This tutorial uses OGRE Intermediate Tutorial 1 as a basis.

Project Configuration

In addition to the project configuration settings for PAR[1], youll need to set up the project for use with OGRE. See the OGRE setup tutorial for details.

main.cpp

Here is where we initialize both OpenPAR and OGRE. The particular setup we have chosen to use for OGRE puts the resulting executable file in the bin directory of the OGRE installation. To compensate for this, we obtain the path to main.cpp (line 14) and set the path to the PAR action files relative to this location.

We then set up the PAR time indicators as we have in earlier tutorials (lines 19-21) and setup the Actionary, Python interrupter, and PAR actions (lines 24-27).

Finally, we create an OGRE application and start its execution (lines 30-31).

GameApplication.cpp

Loading characters is done in the loadCharacters method (line 86). A new instance of the Agent class is created and then stored on a list of agent in the GameApplication class. The Agent class is a subclass of the Open PAR AgentProc class. More on this in an upcoming section.

The findAgent method (line 94) allows Agents to be retrieved from the agent list by name.

The addTime method (line 106) is called by OGRE at every frame and has the amount of time since the last class passed as a parameter. In this method, we call LWNetList::advance(&error) to advance the PaTNets. We then call the update method of each agent on the agent list.

[1] See Tutorial 1

PARactions.h

This file links PAR actions to their implementations as shown in previous tutorials. The one difference is that now we include calls to start animations in the action implementations (lines 11-16). This code simply calls OGRE methods that start playing back an animation clip on the specified agent.

Agent.h

Much of this file is the same as from basic OGRE tutorials. The main difference is that weve made the Agent class a sub-class of the PAR AgentProc class (line 9).

Agent.cpp

Most of the implementations of the methods in the Agent class are straight from OGRE tutorials. We will highlight a few modifications. First, the constructor also calls the AgentProc constructor (line 3). Within the constructor, we also set the capabilities for this agent (line 17) and add an action to the agents action queue (lines 20-22). Instead of explicitly starting an action when the agent is created, you might put such calls inside a richer agent AI.

In the update method, we check to see if the agent is currently executing any PARs (line 34). If there are no actions currently active, then we tell OGRE to stop playing any animations.

The updateAnimations method contains code from informing PAR when an animation clip has finished (lines 132-133). There are multiple ways to terminate an action. A duration can be specified and the action will be terminated at the end of that duration or you might want to play an animation clip through once and then signal that the action is finished. That is where this code would come into play. You could also both specify the duration and use the code for indicating the animation is finished. In this case, the action would end after as soon as one of the conditions is met.