RedUniverse - a simple toolkit

Mark d'Inverno & Fredrik Olofsson

This is basically a set of tools for sonification and visualisation of dynamic systems. It lets us build and experiment with systems as they are running. With the help of these tools we can quickly try out ideas around simple audiovisual mappings, as well as code very complex agents with strange behaviours.

The toolkit consists of three basic things... Objects, Worlds and a Universe. Supporting these are additional classes for things like particle systems, genetic algorithms, plotting, audio analysis etc. but preferably many of these functions you will want to code your self as a user.

We have chosen to work in the programming language SuperCollider (www.audiosynth.com) as it provides tight integration between realtime sound synthesis and graphics. It also allows for minimal classes that are easy to customise and extend. SuperCollider is also open for communication with other programs and it run cross-platform.

So to take full advantage of our toolkit, good knowledge of this programming language is required. We do provide helpfiles and examples as templates for exploration, but the more interesting features, like the ability to live-code agents, are hard to fully utilise without knowing this language.

Detailed overview

In SuperCollider we have the three base classes: RedObject, RedWorld and RedUniverse.

RedObject - things like particles, boids, agents, rocks, food etc.
RedWorld - provides an environment for objects.
RedUniverse - a global collection of all available worlds.

Objects all live in a world of some sort. There they obey a simplified set of physical laws. They have a location, velocity, acceleration, size and a mass. They know a little about forces and can collide nicely with other objects.

Pendulums are objects that oscillates. They have an internal oscillation or resonance of some sort.

Particles are objects that ages with time. They keep track of how long they have existed.

Boids are slightly more advanced particles. They have a desire and they can wander around independently seeking it.

Agents are boids that can sense and act. They also carries a state 'dictionary' where basically anything can be stored (sensory data, urges, genome, phenome, likes, dislikes, etc). Both the sense and act functions as well as the state dictionary, can be manipulated on the fly. Either by the system itself or by the user in runtime.

Worlds provide an environment for the objects. They have properties like size, dimensions, gravity etc and they also keep a list of all objects currently in that world.
For now there are three world classes:
RedWorld - endless in the sense that objects wrap around its borders.
RedWorld2 - a world with soft walls. Objects can go through but at a cost. How soft these walls are and how great the cost is depends on gravity and world damping.
RedWorld3 - a world with hard walls. Objects bounce off the borders - how hard depends on gravity and world damping.

The Universe is there to keep track of worlds. It can interpolate between different worlds. It can sequence worlds, swap and replace, and also migrate objects between worlds. All this while the system is running.
The RedUniverse class also does complete system store/recall to disk of all objects and worlds.

So the above are the basic tools. They should be flexible enough to work with e.g. objects can live in worlds of any number of dimensions. But as noted, one can easily extend functionality of these classes by subclassing.

Conclusion

How the objects and worlds behave, sound and look like are open for experimentation. That is, this is left for the user to code. So while there is great potential for customisation, it also requires more work form its users. The RedUniverse as a whole tries not to enforce a particular type of system. E.g. one can use it purely without any visual output or vice-verse.
We see it both as a playground for agent experiments as well as a serious tool for music composition and performance. We hope it is simple and straightforward and while there is nothing particularly novel about it, we have certainly had fun with it so far. Foremost it makes it easy to come up with interesting mappings between sound and graphics. In a way we just joyride these simple dynamic systems to create interesting sounds.

The software and examples will be available online on the LAM site. Of course as open source.