

# RePast 2.0: Major Changes and New Features

Nicholson Collier and Thomas R. Howe  
Social Science Research Computing  
University of Chicago

## Abstract

The University of Chicago's Social Science Research Computing's RePast is a software framework for creating agent-based simulations using the Java language. It provides a library of objects for creating, running, displaying, and collecting data from an agent-based simulation. In addition, RePast includes several varieties of charts for visualizing data (e.g. histograms and sequence graphs) and can take snapshots of running simulations and create QuickTime movies of such. This paper describes some of the major changes and new features provided by the recently released RePast 2.0.

## The Scheduler

Prior to 2.0, the semantics of the RePast scheduler were more of a discrete time simulator where every tick was checked to see if any actions were scheduled for that tick. As result the tick count proceeded in a regular sequence (e.g. 1, 2, 3, 4 ...), causing confusion about the nature of ticks as merely indexes for the ordered execution of action. Version 2.0 rectifies this shortcoming by re-implementing the scheduler as a true discrete event simulator. Consequently, the tick count now accurately reflects the nature of a tick. For example, if we have three actions scheduled at tick 1, 5 and 10 respectively, the tick count will start at 1, jump to 5 and then to 10 without iterating through the non-existent intervening ticks. This change aligned the observable behavior of a simulation with the semantics of scheduling making it easier to write more temporally complex models.

Perhaps more importantly, the re-write of the scheduler also allowed for the scheduling of actions to occur at fractional ticks (e.g. 2.4) as well as introducing the notion of actions with duration.<sup>1</sup> Fractional ticks provide the ability to easily model processes that take place at different time scales. For example, if we want to model a process where some action occurs 5 times for every single occurrence of another process, then can schedule the first action with an interval of .2 and the second at an interval of 1. So, if we start this first process at .2 it will occur at .2, .4, .6, .8, 1.0, 1.2 and so. If we start the second action at 1.1, it will occur at 1.1, 2.1, 3.1 and so on.

The introduction of actions with duration introduces flexibility in the scheduling of an event that is naturally modeled as a background action, that is, as an action that can be started and run in the background for some period of time but then blocks the execution of other actions when that time period has elapsed (assuming that the action itself has not yet completed.) For example, if some process to be modeled contains some long-running and complicated calculation that can be started at time  $t$ , but whose results are needed at time  $t + 5$ , and that there are actions that can be run concurrently over time  $t - t + 4$ , then we can model this calculation as an action with duration. In terms of implementation, this action runs in its own thread and is amenable to being run on a separate processor or even on another machine.

## GIS

The other major feature added to RePast in version 2.0 is the introduction of interoperability with Geographical Information Systems. Agents can now operate on landscapes imported from GIS and features

---

<sup>1</sup> A fractional tick is not a fraction of a tick but rather a tick whose numerical index is a floating point number. As mentioned above, *tick* is just shorthand for an index for the ordering of actions.

in such landscapes can now behave as agents themselves. More specifically, GIS vector data can be imported from shapefiles, gml, postgis. (Soon oracle and sde will also be supported.) Once in RePast, the features are translated into network components. Polygons and points are translated into nodes and lines are translated into edges between a series of nodes. The inclusion of a delaunay triangulation “neighborhood” allows the user to create edges between the node components based on the triangulation, establishing the “neighbors” of those components. Raster file support is also provided, giving access to ascii raster data with a few geographic operators. In the near future, additional geographic operators will be provided for both vector and raster gis. In addition, tools will be added to make it easier to have agents living on a continuous spaces, instead of being tied to a network or grid structure.

## **SimBuilder**

SimBuilder, the RePast Rapid Application Development environment, has been released at 1.0. This release provides support for simple Network models and Grid models. Conceived of as a pedagogical tool, SimBuilder provides an easy-to-use front end to developing RePast models. The system is a drag and drop interfaces consisting of components from the RePast library. Actions are then coded using NqPY (Not quite Python), a subset of the Python language. This is a simple to use scripting language that allows modelers to write actions without needing to learn java.

Future plans for SimBuilder include:

- The integration of other space/agent types
- Including the facility to add new components as beans to SimBuilder
- Exporting models in java code
- Expanding the NqPY support

## **Miscellaneous Improvements**

In addition to the major features mentioned above there have been the following miscellaneous improvements.

- Improved loading of models via the RePast toolbar
- Improved network drawing
- Automation of network link creation
- Easier creation of graph sequences and histograms
- Dynamic re-sizing of display surfaces
- Window size and location persist across model instances
- Refactored core classes allow experienced programmers to more easily extend RePast

## **Near-Term Future**

We have several projects in the works for upcoming releases. A few of them follow:

- Support for distributed batch models using ProActive
- Support for distributed models for extended actions.
- Additional support for batch control (new language, parameter steering).
- Rewriting the “space library” to allow for more easily swappable topologies
- Continued refactoring of the core libraries for easier extensibility