Real-Time Visualization and Interaction with Computational Artefacts

Matthew Egbert\textsuperscript{1,2}

\textsuperscript{1} School of Computer Science, University of Auckland, Auckland, NZ
\textsuperscript{2} Te Ao Mārama — Centre for Fundamental Inquiry, University of Auckland, Auckland, NZ
m.egbert@auckland.ac.nz

Abstract
Artificial Life research often involves the development and analysis of computational artefacts such as simulations and models. This generally involves an iterative process that alternates between (i) modification of model and research goal and (ii) and investigation of the model—a process that repeats until the researcher is satisfied that they have produced a publishable result.

The first part of this paper argues that real-time visualization and interaction can improve this methodological process by facilitating the development of an intuitive understanding of the computational artefact, leading to more diverse and productive research questions and more interesting results.

Existing tools for developing real-time visualization and interaction involve substantial coding and are often designed with software engineers rather than scientists in mind. The second part of this paper thus introduces Realtime Visualization and Interaction Toolkit (RVIT), a cross-platform Python, Kivy and OpenGL based extendable framework, that has been designed to facilitate the augmentation of Python-based computational artefacts with real-time visualization and interaction. RVIT is freely available and published in alpha on github. With a critical mass of users, we hope it will become a commonly used tool among ALIFE and other scientific researchers and teachers.

Developing Simulations is an Interactive Process
Investigation of computational models is generally an iterative process involving

1. The proposal of a research target (a “question”).
2. The development of a computational model that will allow investigation of the question.
3. Informal investigation of the model (often before it is complete).
4. Repetition of steps 1–3 including modification of (a) the model; (b) the techniques used to study the model (what data is plotted, what parameters are varied experimentally, etc.) and (c) the research target itself—the ‘research question.’
5. A more formal investigation of the model. In some cases the results of this more formal investigation can also cause the researcher to return to steps 1-4.

Otherwise, when the researcher is satisfied, the results of this investigation are published.

This methodology has been described as a symmetric ‘dance of agency’ in the sense that the direction that the research takes is not something that is arbitrarily chosen by the researcher, but is instead influenced alternately by the researcher and the target of study itself (Pickering, 1995). The agency of the investigator is hopefully obvious, and the agency of the target of study (in ALIFE, often a model) can be made explicit with an example: when the model does not do what the investigator wanted or expected, the investigator changes the model, research goal, or the result of the research. In this way, the model’s dynamics orient the investigation, just as the investigator does.

When we recognize that this research process is interactive—that our early, perhaps less formal investigations, play an essential role in developing the ultimate research result—it becomes apparent that the form of interaction between investigator and artefact can also radically impact ultimate research outcomes.

In this vein, we suggest that the typical interface between investigator and computer model is rather unsatisfactory. Simulations are often run and then analyzed. To study the influence of a parameter change, a simulation has to be stopped so that the code can be edited and the simulation be restarted. While it is true that some models and some toolkits such as NetLogo (Wilensky, 1999) expose greater degrees of interactivity, allowing on-the-fly modification of parameters and real-time visualization, this is not the norm.

One reason for the status quo is historical. Computers used to be slower, graphics non-existent, and so real-time visualization and interaction (RVi) tools were essentially out of the question. These constraint on the early computational models developed into a methodological culture, where visualization was post hoc and interaction essentially non-existent. A second reason is the overly simplified received perspective of science and “the scientific method,” where in a desire to make results objective, the interaction between investigator and the object of investigation is swept under the carpet. A third reason for the status quo is the large degree of effort required to develop a rich real-time interface to a computational model. Established graphical user interface (GUI) libraries such GTK2 are designed for software...
extends the Kivy UI library (Virbel, 2011) with scientific visualization elements allowing researchers to rapidly generate and modify visualization and interaction elements. For example, a time-series style plot that tracks a scalar variable simply by adding code that specifies a few details of the visualization, which plots the var field of the model object is achieved in 5 lines:

```python
GraphRenderer:
    pos_hint: ('x':0.0, 'y':0.5)  # position on the screen
    size_hint: (0.75,0.25)      # size
    target_object: model        # the object
    target_varname: 'var'        # the field of obj to plot
```

A slider-controller for varying dynamically a scalar variable in the simulation is also easily implemented using RVIT by adding only the following lines.

```python
SkivySlider:
    pos_hint: ('x':0.8,'y':0.5)
    size_hint: (0.05,0.5)
    slider_min: 0.0
    slider_max: 5.0
    target_object: model
    target_varname: 'parameter'
```

At this stage we have only implements a few scientific visualization elements including GraphRenderer, PointRenderer, ArrayRenderer, BarChartRenderer. As time progresses we hope to have a wide variety of end-user contributed visualization elements. Currently the interaction elements are standard UI elements provided by Kivy. A second design priority was to provide high-speed visualization elements. Extending Kivy, we use OpenGL shader technology to produce efficient visualizations that require minimal programming.

With an active community we hope that RVIT can become a widespread tool that changes the way that people think about, develop and use computational models in scientific inquiry.

**References**