



## 2008 Special Issue

The Emergent neural modeling system<sup>☆,☆☆</sup>Brad Aisa<sup>\*</sup>, Brian Mingus, Randy O'Reilly*Computational Cognitive Neuroscience Lab, Department of Psychology, University of Colorado at Boulder, United States*

## ARTICLE INFO

## Article history:

Received 1 November 2007

Received in revised form

10 June 2008

Accepted 17 June 2008

## Keywords:

Neural networks

Robotics

Simulator

## ABSTRACT

Emergent (<http://grey.colorado.edu/emergent>) is a powerful tool for the simulation of biologically plausible, complex neural systems that was released in August 2007. Inheriting decades of research and experience in network algorithms and modeling principles from its predecessors, PDP++ and PDP, Emergent has been redesigned as an efficient workspace for academic research and an engaging, easy-to-navigate environment for students. The system provides a modern and intuitive interface for programming and visualization centered around hierarchical, tree-based navigation and drag-and-drop reorganization. Emergent contains familiar, high-level simulation constructs such as Layers and Projections, a wide variety of algorithms, general-purpose data handling and analysis facilities and an integrated virtual environment for developing closed-loop cognitive agents. For students, the traditional role of a textbook has been enhanced by wikis embedded in every project that serve to explain, document, and help newcomers engage the interface and step through models using familiar hyperlinks. For advanced users, the software is easily extensible in all respects via runtime plugins, has a powerful shell with an integrated debugger, and a scripting language that is fully symmetric with the interface. Emergent strikes a balance between detailed, computationally expensive spiking neuron models and abstract, Bayesian or symbolic systems. This middle level of detail allows for the rapid development and successful execution of complex cognitive models while maintaining biological plausibility.

© 2008 Elsevier Ltd. All rights reserved.

## 1. Introduction

Emergent (<http://grey.colorado.edu/emergent>) is a powerful tool for the simulation of biologically plausible, complex neural systems that was released in August 2007. The immediate predecessor to Emergent is PDP++ v3.2, a tool used by a variety of researchers for neural modeling and teaching. PDP++ was itself an extension of the PDP software released by McClelland and Rumelhart in 1986 with their groundbreaking book, *Parallel Distributed Processing* (McClelland & Rumelhart, 1986). Emergent represents a near complete rewrite of PDP++, replacing an aging and largely unsupported graphical user interface (GUI) framework called Interviews with a well supported, more modern one called Qt (<http://trolltech.com/products/qt>). A major benefit of Qt is its seamless integration into all major platforms, allowing Emergent

to not only be easily installed on them, but also to adopt their native look and feel. With this in mind, we completely redesigned the user interface, employing a now-familiar tree-based browser approach (with tabbed edit/view panels) for project exploration and interaction (Fig. 1). We also radically redesigned or even replaced several core constructs from the previous product, such as *Environments* and *Processes*, replacing them with the more general-purpose *DataTable* and *Program* constructs that will be discussed later.

More important than technical or interface changes, we also extended the intended scope of the tool. Whereas the previous versions were primarily intended for relatively small research and teaching models, typically aimed at demonstrating some isolated or delimited piece of functionality, the new version is intended to support very large-scale simulations of entire integrated brain-like systems. And whereas the previous versions were primarily designed for closed simulations using simple fixed data patterns as input and output, Emergent has been designed to accommodate external “closed-loop” sensory and motor connections both by plugins and with a built-in simulation environment that includes a rigid-body physics simulation for creating virtual robot-like agents.

This article will give a general overview of Emergent's features and capabilities, ending with a comparison with other neural network simulators and a discussion of the features we plan to implement in the near future.

<sup>☆</sup> Supported by grants: NIH R01 MH069597, ONR N00014-07-1-0651, DARPA/ONR N00014-05-1-0880, ONR N00014-03-1-0428 (O'Reilly); NIH IBSC 1 P50 MH 64445 (McClelland).

<sup>☆☆</sup> Thanks go to Dave Jilk for being the intrepid early adopter; Jay McClelland of Carnegie Mellon University and Jonathan Cohen of Princeton University for their financial assistance during Emergent's development; and all members of the CCN Lab at CU Boulder for their valuable input and patient testing of the software.

<sup>\*</sup> Corresponding author. Tel.: +1 720 233 0225; fax: +1 303 492 2967.

E-mail address: [Brad.Aisa@colorado.edu](mailto:Brad.Aisa@colorado.edu) (B. Aisa).

**Fig. 1.** The Emergent Project Browser. The main workspace in Emergent showing, from left: (a) the Toolbox with widgets for Programming and similar tasks; (b) the main Browser, a hierarchical tree of all objects in the project; (c) the Panel area, for editing and viewing the details of objects, in this case displaying a Doc; and (d) the 3D viewer, for viewing simulation objects in true 3D.

## 2. Emergent

### 2.1. Supported algorithms

Out of the box, Emergent supports classic back-propagation (BP) (Rumelhart, Hinton, & Williams, 1986), and recurrent back-propagation in several variants (Almeida, 1987; Pineda, 1987; Williams & Zipser, 1989); Constraint-satisfaction (CS) including the Boltzmann Machine (Ackley et al., 1985), Interactive Activation and Competition, and other related algorithms; Self-organized learning (SO) including Hebbian Competitive learning and variants (Rumelhart & Zipser, 1986) and Kohonen's Self-Organizing Maps and variants (Kohonen, 1984); and Leabra (an acronym for "local error-driven and biologically realistic algorithm") which includes key features from each of the above algorithms in one coherent framework (O'Reilly & Munakata, 2000).

The previous version of the software (PDP++ v3.2) also served as the basis for some other neural algorithms or extensions, including the Real-time Neural Simulator RNS++ (<http://ccsrv1.psych.indiana.edu/rns++/>) (Josh Brown); Long Short Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997); and the oscillating inhibition learning mechanism (Norman et al., 2006). The enhanced user-friendliness of the software and our new plugin technology make these kinds of extensions very easy to implement, hopefully encouraging more researchers to consider Emergent as the architectural base of their research algorithms. Unlike the tools such as MATLAB, Emergent is completely free and open-source; in addition, its network algorithms run at compiled C++ speed, rather than in an interpreter.

### 2.2. General features

Emergent opens to present a familiar tree-based browser (on the left) plus detail panel (on the right.) The user can select any

object in the left-hand tree to see its detailed properties on the right, and open container nodes to reveal the sub-contents. Many objects have several detail sub-panels that present the object and its content in different views, depending on the purpose of the user. For example, the table object provides a panel with the properties of the table itself, one that lists the columns, and one that enables the user to browse or edit data. The user can open up any number of new browsers rooted at any point in an existing browser.

Clipboard and drag-and-drop manipulation of objects are supported wherever it makes sense. Many "action-like" operations, such as assigning an object to a program variable element, can be done via drag-and-drop.

When the user opens or creates a project, an additional viewer pane appears in the browser; this viewer supports one or more frames which display a true 3D rendering of one or more objects in the system, such as networks, graphs and virtual environments.

### 2.3. Networks

The basic unit of modeling in Emergent is the *Unit*, which is a neuron-like object that represents a small population of like-coding spiking neurons, such as might be observed in a cortical column. Its output is typically a time continuous value ranging from 0 to 1, which represent the extremes of "no firing in the population" to "maximal firing" in the population. Maximal firing is a product of the number of individual neurons times the rate of firing per each neuron. For more detailed neural models it is also possible to run the units in discrete spiking mode. These Units perform separate integrations of excitatory, inhibitory, and leak inputs to accommodate shunting inhibition effects, replicating the classic equivalent circuit dynamics of real neurons. The output is typically based on a thresholded, parameterized, bounded and









