JiST Graphical User Interface Event Viewer

Mark Fong
mjf21@cornell.edu

Table of Contents

Introduction

Java in Simulation Time, or JiST, is a simulation framework written in pure Java that transparently and efficiently enables programmers to simulate many different types of computer scenarios.

One of the key constructs in JiST is an event. An event is an occurrence that causes other events, which causes more events, and so on and so forth. While a simulation is running, a programmer would like to view the results of the events that took place. JiST provides the option of writing the simulation results to log files. These log files can be extremely verbose, forcing the reader to sort through many lines of text.

For a typical event X, the programmer would like to see two types of relationships that X possesses: the event that spawned or called X, and the events that X calls. We will designate these events the Parent and Children, respectively. Children events can have children as well, giving the event-causality picture a one-to-many tree structure, with each node having one parent and zero or more children.

The goal of this project is to create a Java-based graphical user interface (GUI) that allows the programmer to easily view which events cause which events. The GUI is an intuitive method whereby the user can easily traverse from event to event while skipping undesired content.

What it does

The application transforms the existing simulation logs into a hierarchy of events. While using a table to display the events, the application enables the user to traverse events (via mouse clicks) in causal order while listing the events in temporal order.

Design

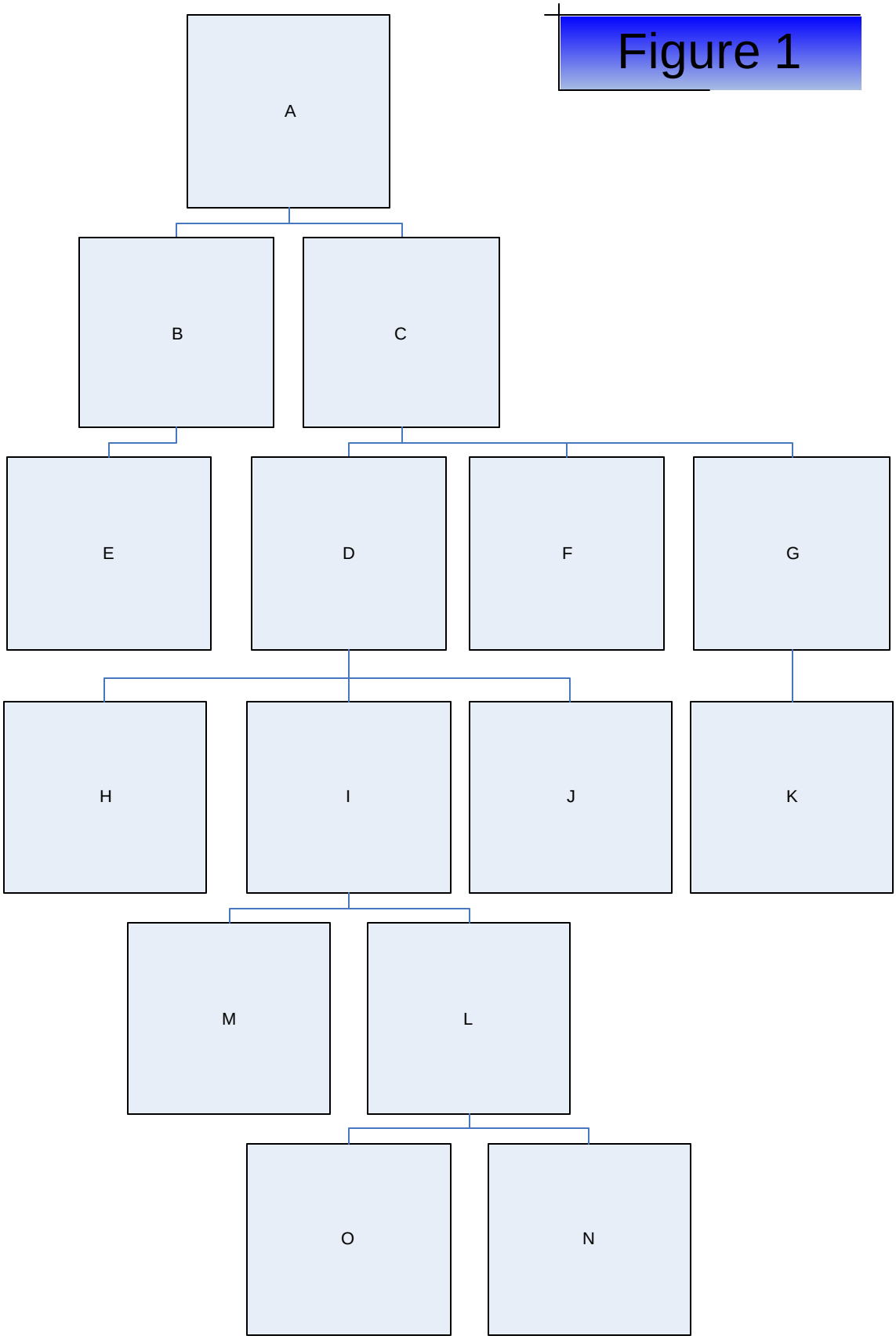
Following Sun's Model-View-Controller (MVC) architecture for creating GUIs, designing this project involved three main steps:

1) Model - creating a model that logically represented the data, i.e. events.
2) View - creating the GUI components and the view of the data.
3) Controller - managing user interaction with the GUI.

Model

The model that describes the data to be displayed in the GUI involves events and their relationships. Events already exist within JiST, but the concept of parent and children events had to be included. An EventNode encapsulates the construct formed by an event and the relationships it has with its parent and children events. An EventNode consists of the event itself, the event's parent, and the event's children. EventNodes strung together form a one-to-many tree structure, as shown in Figure 1 below.

Figure 1



The data model describes EventNodes in the context of a table. When JiST creates an Event, it notifies the data model of the new Event and its parent. The model creates a list of EventNodes that will be displayed in the table. The model specifies functions for finding the Parent and Children events with respect to the list.

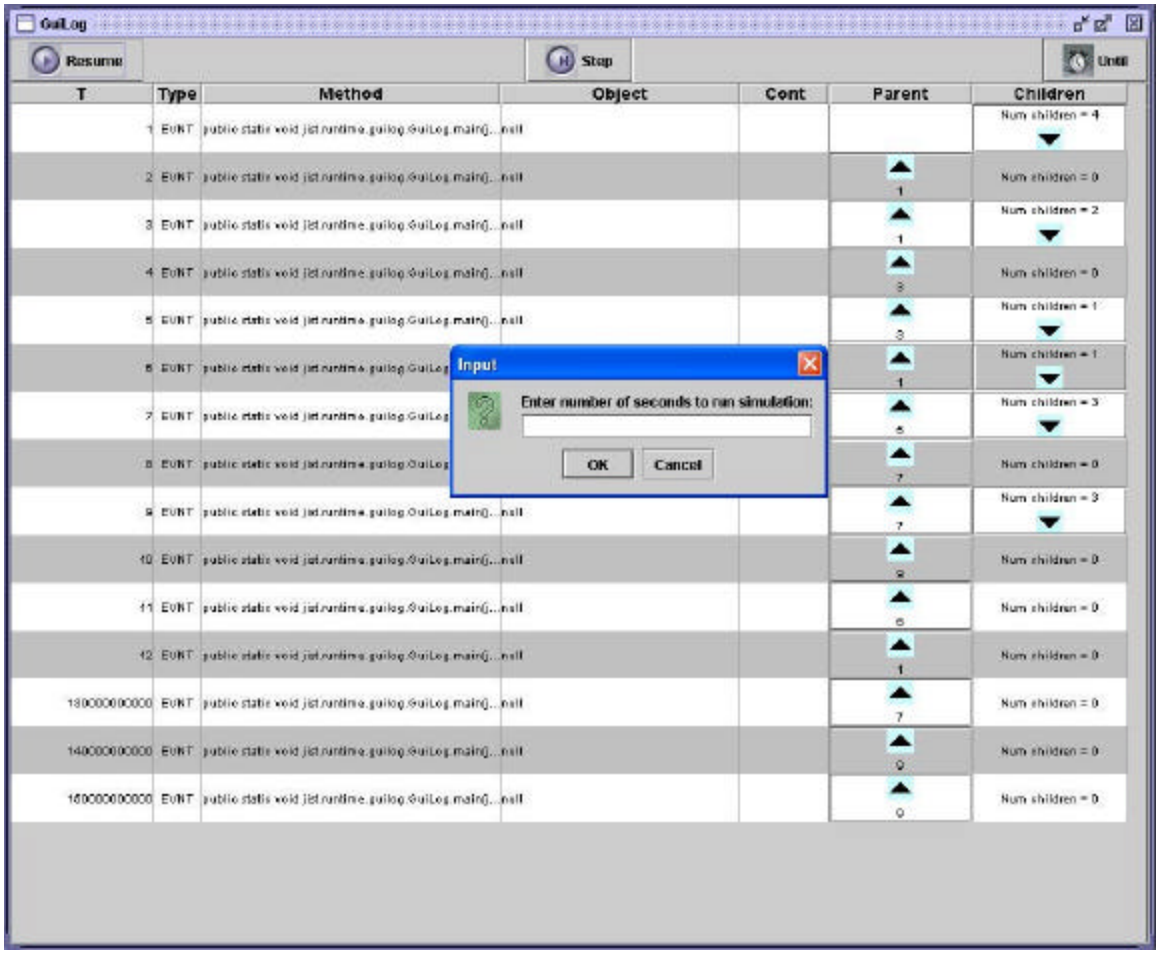
View

The View portion of the MVC architecture involves creating the GUI components and determining how the data will be presented to users. There are two components, a button and a table. As each event is created during a simulation, the table displays the JiST event's properties. As can be seen from the screenshot below, the table has a total of seven columns. The first five display the events' properties of time, type, method, object, and continuation. The last two columns display information regarding the parent and children events.

The parent and children columns display the time property of the parent and children events. Since events can have more than one child, the children column contains a drop-down list that allows users to select which child is of interest to them. Users can make selections on these individual children entries (as well as the cell-contents of the parent column), but the details of how this works is deferred to the Controller section.

There are three buttons, all of which are separate from the table. The leftmost one allows users to pause the simulation so that they can leisurely view the table. The middle one allows one Event to be added to the GUI. The rightmost button allows users to specify an amount of time to allow Events to be added before pausing the GUI.

Below is a screenshot of the paused GUI after the Until button is initially pressed.


GuiLog
Resume
Stop
Until
T
Type
Method
Object
Cont
Parent
Children
Input
Enter number of seconds to run simulation:
OK
Cancel

Controller

As mentioned before, the Controller portion of the MVC architecture describes how the program manages interaction with the user. The user is able to interact with five portions of the GUI:

1) The parent column in the table.
2) The children column in the table.
3) The Pause/Resume button.
4) The Step button.
5) The Until button.

Parent and Children Columns

Since the parent and children column only display the time property of these events, the user will want to see the expanded details. Since the parent and children are events themselves, they have their own complete entries in the table. Thus, when users click on the parent column, the Controller redirects the user to the row of the table that contains the parent event. Similarly, when users select an entry (i.e. child) in the drop-down list in the children column, the Controller redirects the user to the row of the table containing the selected child event. For short, this action of redirection within the table is called Parent and Children jumping.

Pause/Resume Button

The pause/resume button toggles between pausing and resuming the Controller. Each time the button is pressed, it changes to the opposite state. The pause button's function is to suspend changes to the table so that the user can view the state of the table at an arbitrary time. When the pause button is pressed, the Controller blocks and prevents events from being added to the data model. Pressing the resume button removes the block and recommences addition of events.

Step Button

Similar to stepping through a code debugger, this button allows the user to step through a simulation one event at a time. The Step button is available while the GUI is paused. Each time the Step button is pressed, the Controller un-pauses/resumes the GUI by turning event-blocking off, and allows one event to be added to the data model. After the event is added, the Controller pauses the GUI again and blocks events from being added.

Until Button

This button allows the user to specify an amount of time to run a simulation. The Until button is available when the GUI is paused. When the button is pressed, the user is prompted to enter an amount of time. The Controller then un-pauses/resumes the GUI, waits the specified amount of time, and then pauses the GUI again.

Code Review

Files

There are two source files, GuiLog.java and LogInterface.java which are located in jist\runtime\guilog. GuiLog.java contains the entire program

and LogInterface.java contains the methods that the main JiST engine calls.

Data Structures

Data structure: EventNode
In order to capture the relationship among events, a data structure was used called an EventNode that wraps around each Event and accomplishes the one-to-many tree structure. An EventNode contains three objects:

1) The Event itself.
2) An EventNode representing the Event's parent.
3) A list of EventNodes, each of which represents the Event's children.

This data structure allows one to traverse the event tree from one event to another.

Of particular interest are the parent and children events. The relationship among an event, its parent, and its children form an n-way tree. That is, each node of the tree has exactly one node above it, and can have any number of nodes below it.

Private Members

The following are instance members of class GuiLog

- frame (JFrame) – The JFrame for the GUI.
- panel (JPanel) – The JPanel for the GUI.
- table (JTable) - The JTable that displays the Event information.
- model (EventTableModel) – The JTable's EventTableModel.
- list (LinkedList) – A list used to track the events internally so that they can easily be deleted.
- numEventsThreshHold(int) - The maximum number of Events that can be displayed.
- pauseButton(JButton) – The Pause/Resume button for the GUI.
- stepButton(JButton) – The Step button for the GUI.
- untilButton(JButton) – The Until button for the GUI.

Static Variables

The following are static variables used for pausing the Controller.

- pauselock (Object) – Object that acquires the lock.
- paused (Boolean) – Pause-status of the Controller.
- isStep (Boolean) - Whether to Step through one event or not.

Classes and Core Methods

All of the classes are contained within GuiLog.java.

GuiLog
The constructor of the GuiLog class creates and displays the table (JTable) and button (JButton). It creates a JFrame and adds a JPanel to it. It then configures various objects

such as the MouseAdapterHandler, ButtonHandler, and customized versions of the TableModel, TableCellRenderer, and TableCellEditor, which are all associated with the JTable and described in greater detail below.

GuiLog also contains methods for pausing the Controller. Toggling the status of the Pause/Resume button component calls these methods.

Methods

- add(Event id, Event parent) – Adds Event id to the EventTableModel and specifies parent as the parent event. This also adds id to the LinkedList list. Before adding id, this method also checks if numEventsThreshHold has been reached.
  Before this occurs, the method first calls checkLock() to determine if it can proceed execution.
- del (Event id) – Deletes Event id from the EventTableModel and removes it from the LinkedList list.
- checkLock() – This method attempts to acquire a lock on pauseLock. If paused is false, the lock is acquired and the method returns. If paused is true, the lock is not acquired and the method waits to be notified by resume(). If isStep is true, this method returns only once.
- pause() – This method sets paused to true. This method is called by the ButtonHandler.
- resume() – This method sets paused to false and releases the lock on the pauselock object by notifying the checkLock() method. This method is called by the ButtonHandler.
- stepEvent() – This method sets paused to false, isStep to true, and releases the lock on the pauselock object by notifying the checkLock() method. Since isStep is true, checkLock() will only return once, and thus the net effect of this method is that one Event is allowed to be added. This method is called by the ButtonHandler.
- untilTimer() – This method, called by the ButtonHandler, prompts the user for a number of seconds. It then creates two pairs of Timers and UntilTasks: one pair for calling GuiLog.Resume(), and the other pair, which is scheduled to run after the user-specified number of seconds, for calling GuiLog.Pause().

EventTableModel

As described in the MVC architecture, the table obtains its data from a data model. The EventTableModel class represents the data model for the JTable. The EventTableModel implements the TableModel interface. It uses an array of EventNodes to store and access the Events. It has methods for managing the EventNodes, e.g. finding EventNodes in the array, finding Parent and Children indices in the array.

Methods

- add(Event ev, Event parent) – Creates a new EventNode with ev as the Event, and parent as the parent Event. This new EventNode is then added to the array. Finally, this method notifies all listeners that the rows in the table have changed.

- del (Event id) – Deletes Event id and its children from the array. Finally, this method notifies all listeners that the rows in the table have changed.

EventCellRenderer

The EventCellRenderer is a customized TableCellRenderer. This object describes what information to display and in what format.

Methods

- getTableCellRendererComponent (JTable table, Object value, boolean isSelected, boolean hasFocus, int row, int column) – This method returns a component that is used for drawing the cell in the table. Based on the column number, the method extracts the appropriate Event information from the value object and returns either a JLabel for basic Event information, or a JButton for the Parent Event. The JButton displays the time that the Parent Event occurred, and an arrow indicating where the Parent Event is located on the table.

ChildrenCellEditor

The ChildrenCellEditor is a customized TableCellEditor. This object describes what information to display in the Children Event column and in what format. It also describes how to handle mouse events.

- getTableCellEditorComponent (JTable table, Object value, boolean isSelected, int row, int column) – This method returns a JComboBox component that is used for editing cells in the Children Event column. The times that each Child Event occurs is extracted from the value object and entered into the JComboBox. This method also registers the JComboBox as an ActionListener.
- actionPerformed(ActionEvent e) – This method runs when a mouse selection occurs on the JComboBox. When a Child Event is selected from the JComboBox, this method identifies the table row that corresponds to the selected Child. The method then changes the view of the JTable to show the selected Child Event.

MouseAdapterHandler

This is a customized version of MouseAdapter. The purpose of this class is to handle mouse clicks on the Parent Event column.

- mouseClicked(MouseEvent e) – This method is similar to the ChildrenCellEditor's actionPerformed() method. When a cell in the Parent Event column is clicked on, the method identifies the table row that corresponds to the Event's Parent. The method then changes the view of the JTable to show the selected Parent Event.

UntilTask

This is a customized version of TimerTask. The constructor (which takes in a String) determines whether the UntilTask's run() method will call GuiLog's pause() or resume() method. This class is used by GuiLog's untilTimer() method.

ButtonHandler

This class defines what action to take when one of the buttons is pressed.

- actionPerformed(ActionEvent e) – If the Pause/Resume button is pressed, this method toggles the GuiLog's static variable, paused, between true and false by calling GuiLog's pause() and resume() methods.
  If the Step button is pressed, this method calls GuiLog's stepEvent() method.
  If the Until button is pressed, this method calls GuiLog's untilTimer() method.

Testing and Verification

Both correctness and performance were tested and evaluated by different test programs.

Correctness

The following functionality criteria were tested:

1. For an Event with zero, one, or two or more Children, the Event, Parent, and Children information must be displayed correctly, and the Parent and Children jumping must function correctly.
2. The table must continue to function properly when the number of viewable Events is varied.
3. The table must continue to function properly while the Pause button is enabled.
4. The table must continue to function properly when the Step button is pressed.
5. The table must continue to function properly when the Until button is pressed.

To test #1, Events were created and added to the GUI with zero, one, or two or more children. The Event, Parent, and Children information for each of these was inspected. The Parent and Children jumping was also tested.

For #2, values from a range of 15 to 50 were used, and for each value the criteria in #1 was tested. The testing for #3-5 was incorporated into these tests as well by clicking the Pause, Step, and Until buttons intermittently and verifying the criteria in #1.

Performance

Performance was evaluated on a qualitative basis. Test programs were used to evaluate the program's ability to handle different workloads.

In earlier stages of development, rendering of the GUI was quite sluggish. To improve upon this, static variables were used to replace components and properties that were constantly being instantiated or set, such as fonts, icons, and buttons. This improved the rendering performance dramatically. Throughout the correctness tests, performance was monitored to ensure that the rendering speed was maintained and did not decline during the testing process.

Using the Code

Running Simulations: Seeing it in Action

To see the GuiLog work, one can run the same simulations that were used for testing. The following are step-by-step instructions for windows. The home directory of the CVS JiST repository is /code. Before running each example, the user must perform the following:

I. Put all the files in the /code/libs directory into the CLASSPATH variable.
II. From a command prompt, negotiate to the source directory of the JiST repository: /code/src.
III. Compile the entire project by typing: make.bat

GuiLog.Main

1. Type the following from the source directory: java jist.runtime.guilog.GuiLog

Jist.minisim.hello

1. In /src/jist/runtime/Main.java, change the GUILOG_SIZE variable to a positive number.
2. Repeat step III.
3. Type the following from the source directory:
   java jist.runtime.Main -c debug-controller.properties jist.minisim.hello

Incorporating the Code into Existing Projects

To use the Event Viewer, one has to import jist.runtime.guilog.GuiLog. There are two methods available to the user: add() and del(), which allows the user to add or delete an event from the GUI, respectively. These methods were described earlier in the GuiLog Code Review section on page 9.

Suggestions for Improvement

Since the JiST GUI Event Viewer is a debugging tool, there are more ways to improve and augment its debugging capabilities. Some of these include:

1. Consolidating results of Parent and Children event information so that the user can view all the relatives of a specified event simultaneously.
2. Pausing JiST simulations and allowing the user to step through and into Events, much like a traditional debugger does with code.
3. Add a progress indicator to illustrate how much time has passed when the Until button is used.