Bugs and annoyances, Version 6.0.2

The following are known bugs/problems with Galant. Some of them, marked * have been observed only on a Mac (on my Mac specifically).

Input/output

- If a graph has been saved via a File→Export in the graph window, runtime attributes such as hidden may cause problems when the graph is read and displayed. This is because there is no state information available at the point when the graph is read. It's not clear what the right solution is. Probably check if an algorithm is running, and, if not, use the stateless variant of the attribute retrieval.
- * Occasionally there is an exception involving the user interface with a stack entry that says something about GlyphView. This can happen at any time. Just hit Continue when the error window pops up (usually a good policy if you're not sure what the error is about and it appears not to be affecting your session).

Text editing (of programs or graphs)[1]

- There is no "undo" mechanism, either for the text editor or the editor in the graph window.
- * Tabs for graphs and algorithms are often hard to deal with: (a) if you reread a graph or algorithm, it appears twice; (b) you can only run an algorithm on a graph if the tabs for the two appear at the top of the window at the same time – this may be impossible if there are other intervening graphs and algorithms and the window is not wide enough.
- * Occasionally Galant simply hangs; it appears that you can exit (quit) from the file menu of the graph window or from the the Galant menu on the task bar. Oddly, when Galant hangs, you can bring it back to life by executing a command in a separate terminal window.
- Some preferences, such as syntax highlight colors, require the user to exit Galant before they take effect.
- * The Open/Save preference that specifies the default directory for files does not persist from one session to the next (unlike all other preferences).
- When saving a file, Galant complains if the extension is not correct (.graphml or .alg) but does not fill it in automatically.

Graph editing (in graph panel or via keyboard shortcuts)

- Editing is mode-driven: the effect of a mouse action is determined by which of the four modes (select, create node, create edge, delete) is selected on the toolbar. This has unpleasant consequences if, for example, the user forgets that she is in delete mode.
- Nodes have to be moved individually. In a large graph there is no way to select a collection of nodes and move them all at once.
- The force directed layout algorithm clusters nodes too close to each other when there are cliques or near cliques.
- Semantics of force directed layout when combined with adding edges is not intuitive (force directed layout takes over if the button is pressed, so adding nodes/edges is dependent on the state of the button). It's generally a bad idea to change the graph when the smart reposition button is pressed.
- It is not possible to change the thickness of an edge or node boundary directly from the editor or an algorithm nor is it possible to change the fill color of a node. The only way to change these properties is via highlighting, selecting, and marking nodes/edges during the animation.

[1] Galant's editor is primitive, but programs can easily be edited externally. Text representations of graphs can either be edited or generated externally.

- Once you choose a color for a node in the editor, you can't uncolor it in the editor. The best you can do is set it to black, but then it appears thicker. However, the algorithm strip_attributes.alg can be run to reset colors and other nonessential attributes. You can save the cleaned up graph using File→Export in the graph window.
- It is not possible to change a weight from nothing to 0. You need to set it to 1 first and then back to 0.
- When user creates a new node/edge via keyboard shortcut, there is no obvious way to enter the weight and label (except to click in the appropriate text field).

Compilation and execution

- * If you try to manipulate anything in the text window (e.g., click on File) when an algorithm is running, Galant hangs. You must quit Galant completely without saving anything.
- When there are compilation errors the user cannot scroll the text window (or make modifications in it) while the popup window showing the errors is displayed. There are two possible workarounds: (i) open the algorithm in an external editor, or (ii) view the error messages in the console.
- Every once in a while an exception occurs when an error-free algorithm is executed or when Galant initially fires up, but it is possible to step through the animation normally after hitting the Continue button.
- When an algorithm controls visibility of node/edge labels or weights and the user overrides in the middle of execution, Galant sometimes freezes when user terminates the algorithm. This appears to happen more frequently if user does a lot of fast forward/reverse between visibility changes. The problem does not seem to occur if user toggles visibility via keyboard shortcuts.
- Compiler error messages can be cryptic (but at least they refer to the correct line numbers). Because of the macro preprocessing, it may be necessary to look at the console to get an idea of what is causing a particular error. If the parentheses/brackets/braces inside a function definition or body of one of the for_... macros are unbalanced, the macro preprocessor will simply report the fact with no indication of the location of the error except for an excerpt from the beginning of the body.
- Line numbers do, however, get out of sync if the header of a function declaration takes up more than one line. For example, in

```
function foo(Node v,
             Node w,
             Edge e) {
}
```

  The first three lines are treated as one.
- If a macro is used incorrectly, the preprocessor does not report a line number.
- After hitting Enter or Return at the end of a query, user still needs to step forward to do the next step of the algorithm.
- There is no way to execute the animation in a continuous fashion with a controllable speed. The current workaround is the use of arrow keys as keyboard shortcuts for stepping forward or backward – these can be held down to generate multiple steps in rapid succession, but finer grained control is difficult.