Bugs and annoyances, Version 5.3

Input/output

- When you save the current state of an animation using `File->Export` on the graph panel some of the information might not be saved. In particular, weights and labels are saved, but highlighting is not (since it is not equivalent to an actual color change of a node or edge).

Text editing (of programs or graphs)[1]

- 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.
- A text panel for a graph is often marked as having been changed when this is not really the case. This will happen if an algorithm runs on the graph and sometimes immediately when the graph is read from a file.
- If you attempt to do anything in the text window (File menu or tabs) while an algorithm is running, Galant hangs; it appears that you can exit (quit) from the file menu of the graph window, however.
- Syntax highlighting of functions and macros is oblivious of comments. Thus, if the name of a Galant function such as `edges` appears in a comment, it will be highlighted.
- Some preferences, such as syntax highlight colors, require the user to exit Galant before they take effect.
- 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)

- 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. This has been partially mitigated. If you use the Ctrl-i keyboard shortcut, you are asked for a "degree repelling boost" – nodes with high degree will repel each other more if this number is large.
- The force directed layout places nodes too close to the bottom edge so that they are obscured when an algorithm is running.
- Semantics of force directed layout when combined with adding edges is not intuitive (force directed layout takes over if the button is pressed, so state dependent).
- 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.
- 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.
- If a node is selected for editing (other than immediately on creation) its weight is set to 0 when the spinner shows up. Initially a node has no weight at all; this should continue to be the case unless the user specifies a weight.
- If user attempts to change weight/label of a node/edge and clicks on the graph panel in the middle of the operation, the change is lost. There is no "ok" prompt as there is with color.
- When user creates a new edge via keyboard shortcut, there is no obvious way to enter the weight and label.

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

Compilation and execution

- 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.
- A query, if the user ignores it and advances the algorithm, will treat the answer as if it were whatever the answer to the previous such query, if any, was (if there was no previous answer, it's null).
- 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.