ComS 229 Project 2 (part 2)

The Game of Life — Interfaces

April 2, 2013

0 Introduction

This is a "change" document. In other words, rather than writing another entire spec, this document focuses on what should be changed relative to part 1 of the project.

1 The programs

Write the following programs. As usual, this describes the *minimum* required functionality; you are free to implement extra if you like.

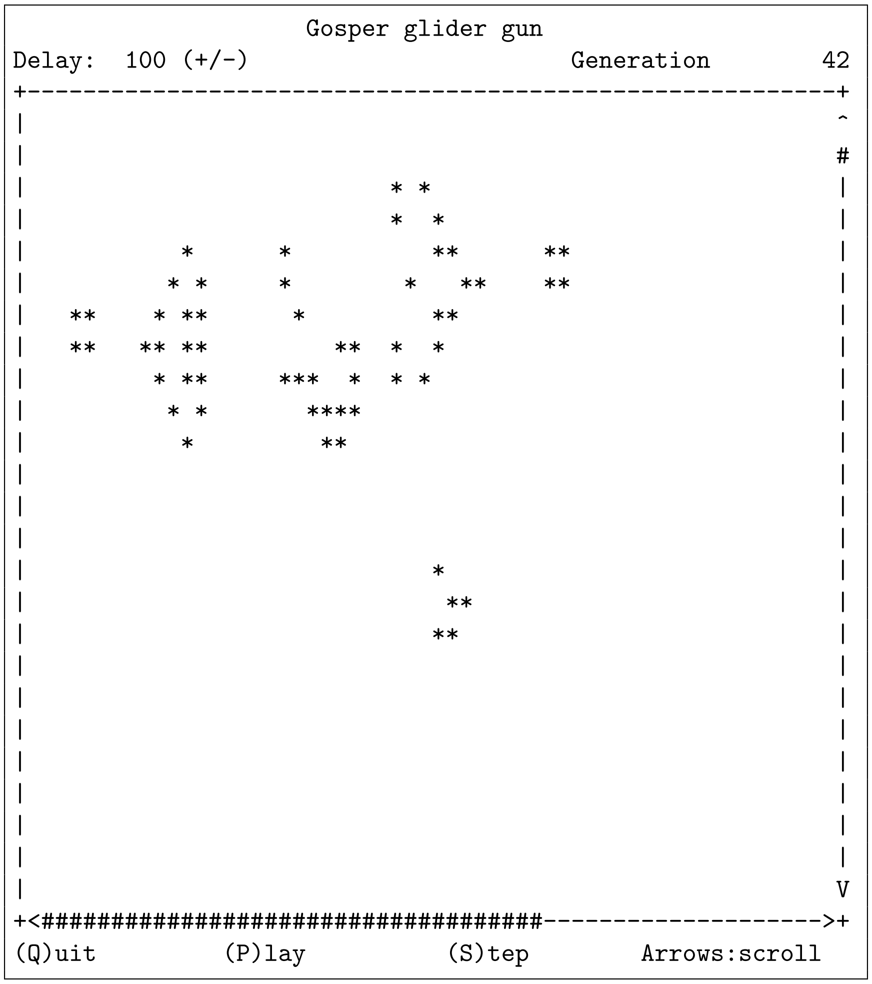
1.0 `showgen`

The output characters in visual mode may be different from '`~`' and '`1`'; see Appendix A. Also, when `showgen` produces an `.aut` file, be sure that all statements of the original file are present in the output (the `Initial` statement will most likely be different). Otherwise, there is no change.

1.1 `sim-tui`

Write a simple Textual User Interface (TUI) that updates the ASCII visualization of each generation, in a terminal window. Standard C++ cannot do this by itself, so you will use the `ncurses` library (a new, open–source replacement for the old "cursor optimization" library, `curses`). See `http://tldp.org/HOWTO/NCURSES-Programming-HOWTO/` for a nice tutorial, and consult your `man` pages to learn about the functions in more detail. Some useful functions include:

- `initscr`, `endwin`
- `cbreak`, `nocbreak`, `echo`, `noecho`, `keypad`, `timeout`
- `clear`
- `getch`
- `getmaxyx`
- `mvprintw`
- `refresh`
- `waddch`
- `wmove`



```
                     Gosper glider gun
Delay:  100 (+/-)                       Generation        42
+----------------------------------------------------------+
|                                                          ^
|                                                          #
|                          * *                             |
|                          *  *                            |
|           *      *          **      **                   |
|          * *     *        *   **    **                   |
|   **    * **      *         **                           |
|   **   ** **         **  *  *                            |
|         * **     ***  *  * *                             |
|          * *       ****                                  |
|           *         **                                   |
|                                                          |
|                                                          |
|                                                          |
|                             *                            |
|                              **                          |
|                             **                           |
|                                                          |
|                                                          |
|                                                          |
|                                                          |
|                                                          |
|                                                          |
|                                                          |
|                                                          V
+<####################################-------------------->+
(Q)uit         (P)lay          (S)tep        Arrows:scroll
```

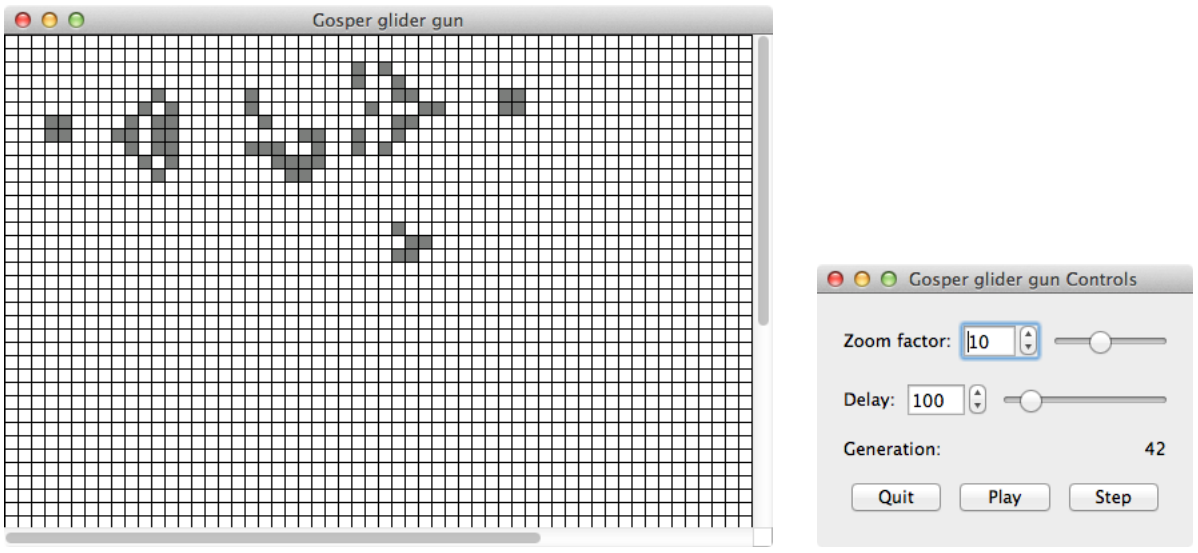
Figure 1: Sample screenshot for `sim-tui` (terminal is 30 rows, 60 cols).

This program reads a single `.aut` file passed as an argument. The input file specifies the size of the terrain and the initial configuration for generation 0. The program determines and displays the state of each cell as generations advance. The program accepts the following command–line switches.

`-h`	Display a help screen that describes the program.
`-tx` l, h	Set the x range of the terrain; overrides values specified in the `.aut` file.
`-ty` l, h	Set the y range of the terrain; overrides values specified in the `.aut` file.

The following information should be displayed in the terminal window:

- The name of the simulation (see Appendix A).
- The generation count (leave space for at least 5 digits).
- The delay between generations, when the simulation is running.
- The current state of each cell. This should take up most of the terminal window.
- Horizontal and vertical scrollers, if the terrain size is larger than the area that may be displayed in the terminal window. In this case, the arrow keys should cause the display to scroll in the direction of the arrow.



Zoom factor: 10

Delay: 100

Generation: 42

Figure 2: Sample screenshot for `sim-gui`.

- A reminder of what keys may be pressed to control the simulator. These should be:

  `'S'` to step to the next generation.

  `'P'` to pause or play the simulation. "Play" means to step through successive generations and display them, with a delay in between.

  `'Q'` to quit.

  These should *not* be case sensitive. Initially, the simulator should be "Paused".

An example screenshot is shown in Figure 1.

1.2 `sim-gui`

Write a simple Graphical User Interface (GUI) that updates a graphical view of each generation, in a separate window. Standard C++ cannot do this by itself, so you will use the Qt 4 library (an open–source, C++ library for GUIs). You are strongly encouraged to download "C++ GUI Programming with Qt 4" from Blackboard, and to work through all examples in Chapter 1, the first example in Chapter 2 (up to "Rapid Dialog Design"), the icon editor example in Chapter 5, and the scrolling example in Chapter 6.

You do not need to implement a "complete" GUI (e.g., with a "File" menu, drag–and–drop capability, etc.). Your executable should read a single `.aut` file passed as an argument, and should accept the following command–line switches.

`-h`	Display a help screen (to standard output) that describes the program.
`-tx` l, h	Set the x range of the terrain; overrides values specified in the `.aut` file.
`-ty` l, h	Set the y range of the terrain; overrides values specified in the `.aut` file.

The GUI should have the same functionality as the TUI: the simulation name should be displayed for each window, the grid display should include scrollers if the terrain size is larger than what is displayed, and there should be a window with the simulation controls. An example screenshot is shown in Figure 2. Note that the window style is dictated by your platform, and cannot be configured in Qt.

2 Limits

For `sim-tui`, you may require that the terminal window is at least 40 columns wide, and 20 rows tall.

3 Makefiles

As you work with Qt, you will discover that Qt will build its own makefile for any given project, and these makefiles are not guaranteed to be portable across systems. You are still required to have one "master" `Makefile` at the top level, so that typing `make` builds everything (including the Qt makefile). There are a few ways to handle this, but all of them are a bit tricky, so be sure to test this out early on `popeye`. From a "Unix–like" system you can test GUI applications remotely on `popeye` if you connect with `ssh -X`.

A `.aut` file format

You must handle the following new keywords.

`Name` Sets the name of the simulation, which should be displayed by the simulators. This statement must come before the `Initial` statement, and has the format

```
Name "name string";
```

where a name string is allowed to contain any character except '`"`'.

`Chars` Sets the characters to use for displaying the *dead* and *alive* states. This statement must come before the `Initial` statement. Format:

```
Chars dead, alive;
```

where the characters are specified as ASCII codes (in base 10). These values should be ignored by `sim-gui`.

`Colors` Sets the colors to use for displaying the *dead* and *alive* states. This statement must come before the `Initial` statement. Format:

```
Colors (rd, gd, bd), (ra, ga, ba);
```

where `(rd, gd, bd)` are the RGB values for the dead cells, and `(ra, ga, ba)` are the RGB values for the alive cells. Each of these values should be between 0 and 255, in base 10. These values should be ignored by `showgen` and `sim-tui`.