CISC 2000—Computer Science II
Fall, 2013

Programming Project # 1: Some Computer Graphics Examples

Date Due: Wednesday 18 September 2013

Write two graphics programs that use the author's GUI.

- The first (`proj1a.cc`) should prompt a user for her name. It should draw a `Simple_window` that contains said name, using a nice-looking font at a reasonably nice size.[1] This can be done with the tools in Chapter 12. Please note that a user might input her full name, which means that simply using `cin >> name` isn't good enough.
- The second (`proj1b.cc`) should draw a standard 8 × 8 checkerboard, containing alternating black and red squares. In principle this can be done using the tools in Chapter 12, but it will be extremely tedious; the tools in Chapter 13 make this much easier (see Section 13.10).

A few considerations:

1. I would judge the second program to be about twice as hard as the first. So I will give Project 1a and Project 1b relative weights of 35% and 65%, respectively.
2. I have made executable versions of these programs for you to try out. The executables are named `proj1a` and `proj1b`. These programs are available in the project's share directory

   ```
   ~agw/class/cs2/share/proj1
   ```

   on the Departmental Linux machines. This means that once you log in and `cd` into said directory, you execute them by simply typing in their names. *Please try them out before you start working on the assignment!*
3. You should do this project in a working directory named "`~/private/cs2/proj1`".
   (a) You'll need to create this directory, by issuing the shell command[2]

   ```
   mkdir␣-p␣~/private/cs2/proj1
   ```

   You only need to do this once.

   (b) To guarantee that any files you create for this project, you will need to issue the shell command

   ```
   cd␣~/private/cs2/proj1
   ```

   before you start doing any work on the project (e.g., before firing up `emacs`). You'll do this each time you log in and start work on the project.

   Please note that I'll need to actually run your programs, to see whether they work. (After all, this *is* graphics.) So please make sure that you do all your work within `~/private/cs2/proj1` as indicated, or I'll have to mount a massive search expedition to find your programs.

[1] I used 30-point Times italic.

[2] The `-p` flag means to create any necessary parent directories. Since this is the first project for CS2, your `private` directory probably doesn't have a `cs2` subdirectory. This explains the `-p`.

4. The directory `~agw/class/cs2/share/proj1` also contains a `Makefile`, which you should copy to your working directory (`~/private/cs2/proj1`). Once you've done this, you can use the `make` command (from within this working directory) as follows:

   (a) `make proj1a` will compile and link the source file `proj1a.cc`, producing an executable program named `proj1a`. *Do not execute the command `make proj1a` until there's a file named `proj1a.cc` in the working directory!*

   (b) Similarly, `make proj1b` will compile and link the source file `proj1b.cc`. *Do not execute the command `make proj1b` until there's a file named `proj1b.cc` in the working directory!*

   (c) `make` (by itself) will build both `proj1a` and `proj1b`.

   (d) `make clean` will clean out the directory. It gets rid of the executable files `proj1a` and `proj1b`, compiled object files `*.o`, as well as other stuff you probably don't care about (core dumps, `emacs` backup files, and the like).

   The reason that you should use `make`, rather than directly using `g++`, is that the `Makefile` contains the extra information needed to access the author's GUI.

5. The `photo` program is pretty useless for capturing the output of a graphics program. So it will suffice for you to send me a listing of your two programs. The `a2ps` program can be used to produce a "pretty-printed" listing. So when you're ready to turn in your listing for this project, issue the shell command[3]

   ```
   a2ps␣-o␣-␣proj1a.cc␣proj1b.cc␣|␣ps2pdf␣-␣proj1.pdf
   ```

   This will create a PDF file `proj1.pdf`, containing a nicely formatted listing of your two programs. Now take a moment to see what the listing looks like, by issuing the shell command

   ```
   xpdf␣proj1.pdf␣&
   ```

   If you're happy with what you see, quit `xdvi` (this is in a menu provided by the File button), and you can then mail me `proj1.pdf` by issuing the shell command

   ```
   mail␣-s␣"Project␣1"␣-r␣harry@bovik.com␣agw␣<␣proj1.pdf
   ```

   Here, `harry@bovik.com` is to be replaced by the email address to which my confirmation message should be sent, i.e., the email address you most commonly use.

   Please do not actually use the email address `harry@bovik.com`!!! It's only a sample return address!!![4]

Have fun! After all, this *is* graphics, and graphics should be fun.

[3] `a2ps` produces a pretty-printed listing of the two programs, written in POSTSCRIPT. (POSTSCRIPT is an ancestor of PDF.) `ps2pdf` translates `PostSctipt` to PDF. The two commands are linked by a pipe, with "`a2ps␣-o␣-`" telling `a2ps` to put its output onto standard output (rather than a file) and "`ps2pdf␣-`" telling `ps2pdf` to read its input from standard input (instead of a file). This is an example of how to use the Unix shell, along with filters (programs that read from standard input and write to standard output), to build "on-the-fly" commands.

[4] Believe it or not, every semester some student manages to send me email with this return address.