CS450/550 — Program 1 — Due January 23, 1996

In this assignment, you will use the C++ package from the textbook to implement Bresenham's circle drawing algorithm. Your program should open an X window and then enter a loop. In each iteration, it should ask the user to enter the center and radius of a circle (using 2 mouse clicks). Then it should display the circle (via Bresenham's algorithm) and then wait until the user presses the mouse button again. Then it should clear the window and repeat the process. The loop should terminate after 20 iterations.

To make the assignment a bit more challenging, your code should cause the circle to "wrap-around" if it is too big to fit into the window. For example, a point at $(-5, 10)$ should be mapped to $(W - 5, 10)$, and a point at $(800, 10)$ should be mapped to $(800 - W, 10)$, where W is the width of the window. Your code should be general so that it could properly map points such as $(-948474, 467364)$ back into the visible window.

The code for this project is available as a `tar` file on the College of Engineering machines as `~tgd/cs450/code/p1.tar`. When untarred, this file will create the following directories and files:

```
opsys.h
Shape2d:
        const.h
        application.h
        circle.h                circle.c
        convex.h                convex.c
        edge.h                  edge.c
        elist.h                 elist.c
        glist.h                 glist.c
        line.h                  line.c
        mouse.h                 mouse.c
        pattern.h               pattern.c
        point.h                 point.c
        polygon.h               polygon.c
        prim.h
        scene.h                 scene.c
        vertex.h                vertex.c
        win_view.h              win_view.c
        Makefile
Xglib:
        xglib.h                 xglib.c
        Makefile
p1:
        p1.c
        Makefile
```

These are only a subset of all of the files that come with the textbook. However, they are all that we will need for the first two weeks. Here is a description of some of these files—the ones that are relevant to this assignment.

`opsys.h` This file specifies which kind of window system you are using. I believe all of the color X terminals in the COE lab have 8-bit color systems, so I have configured this file to define the system `Xwin8`.

`const.h` This file defines several constants of which the two most important are `DWID` (768) and `DHEI` (576). These specify the width and height of the X window in pixels.

application.h This defines a simple class called `application_t`. (All class names in this package end with `_t`). Your programs should normally create an instance of this class as the first thing in their `main()` routine. This class takes care of creating and destroying the X window. It creates an X window of dimensions 768 by 576 pixels. Technically, the X window is called a **viewport**. The `application_h` also defines the relationship between world coordinates and X window coordinates. The X window coordinate system has (0,0) in the upper left hand corner, (767,0) in the upper right-hand corner, (0,575) in the lower left-hand corner, and (767,575) in the lower right-hand corner. The corresponding world coordinates are $(0.0, 575.0)$, $(767.0, 575.0)$, $(0.0, 0.0)$, and $(767.0, 0.0)$. Hence, these correspond to a standard X-Y cartesian coordinate system positioned at the lower left corner of the screen. This is described in the textbook (Section 4.1, pages 108–114).

The `application_t` has one member, `wv`, which is an instance of the `win_view_t` class.

circle.h, circle.c These define the `circle_t` class. A circle is defined by its center and its radius (in world coordinates). It has several different constructors including one that takes a `mouse_t` as an argument and prompts the user to enter the two points by clicking the mouse (at the center and at any point on the perimeter).

mouse.h, mouse.c These define the `mouse_t` class, which encapsulates the interface to the mouse. Unqualified function names (such as `::wait_button_down`) are calls to the X window interface routines is `xglib.c`. The most useful function defined here is `mouse_t::wait_button_click()`, which prompts the user to click the mouse, and waits for the click before continuing. We will usually call this routine after putting up a display to wait for the user to look at the display.

point.h, point.c These define a 2-D point as having two `double` members, `x` and `y`.

win_view.h, win_view.c These define three classes: `window_t`, `viewport_t`, and `win_view_t`. A `window_t` defines a rectangle in world coordinates in terms of four values (left, right, top, bottom). It will be responsible for clipping off parts of objects that extend beyond the window. The `viewport_t` defines the viewport in *normalized device coordinates* (such that the window has width 1.0 and height 1.0). The `win_view_t` is the key class. It has two methods, `win_dev` and `dev_win`, that destructively convert a `point_t` between world and device coordinates. To do this, it contains a `window_t` and a `viewport_t`.

Shape2d/Makefile The primary purpose of this Makefile is to create a binary library called `lib2d.a`. This is created by issuing the command `make lib2d.a` while connected to the `Shape2d` directory.

xglib.h, xglib.c, Makefile These define a set of routines for creating and manipulating X windows. You should read through `xglib.h` for nice documentation of each of these routines. The routines we will most often invoke are `col(int)`, which changes the current drawing color of the X window, `set_font_color(int)`, which changes the current font color, and `draw_string(int x, int y, char * s)`, which draws string `s` at device coordinates `(x,y)`. For this assignment, you will also want to call `setpix(int x, int y)`, which draws pixel `(x,y)` in device coordinates using the current color.

p1.c Makefile This is a "skeleton" file for the first program. You must write the code for the body of `bresenham` and any other functions you may need (e.g., `setpix_oct`). You will want to convert the center and radius of the circle into device coordinates and then apply Bresenham's circle-drawing algorithm. The `Makefile` is set up so that you can type `make p1` to compile and load the program.

Please note that all of this only works on the COE HP machines.

Please email your solution to the TA (`muenchp@cs.orst.edu`) in the form of a `shar` file that contains your `p1.c` program and written instructions on how to run it. The "Subject:" of your message should be "PROGRAM1". The program should be linkable using the current `lib2d.a` and `gxlib.o` files described above. Programs will be graded according to the following criteria: Correct Execution (50%), Clean Design and Structure (25%), Clear Documentation and Execution Instructions (25%).