$Id: asg4-oop-inheritance.mm,v 1.7 2017-07-24 15:56:02-07 - - $
PWD: /afs/cats.ucsc.edu/courses/cmps109-wm/Assignments/asg4-oop-opengl
URL: http://www2.ucsc.edu/courses/cmps109-wm/:/Assignments/asg4-oop-opengl/

1. Overview

In this assignment you will implement a simple graphics package using the OpenGL graphics library. Using inheritance, geometric objects will be displayed in a window. The lab workstations all run X11 so there is no problem there. When you log into the server remotely, be sure to specify the **-X** option, as in:

ssh -X *username*@unix.ucsc.edu

Before you log into the server make sure you have an X11 client running on your own workstation.

2. Program Specification

The program is presented in the form of a Unix **man**(1) page.

NAME

gdraw — drawing program displaying objects in a window

SYNOPSIS

gdraw [*options*] [*filename*]

OPTIONS

All options are recognized by **getopt**(3).

-@ *flags*

The flags specified are passed to the debugging macro. Debug output is printed to the standard error.

-w *width*

The initial width of the window. Default: 640.

-h *height*

The initial height of the window. Default: 480.

OPERANDS

Commands are read from the file whose name is given as an operand. If no filename is given, commands are read from the standard input.

COMMANDS

When the program begins, a window is created for displaying information, which consists of text and geometric objects of various kinds. The project proceeds in two parts:

(a) Shapes are defined in terms of their sizes (length, width, or vertices).

(b) Objects are drawn in the window by drawing commands, after which they may be altered by keyboard commands.

Sizes, lengths, widths, and coördinates are all floating point numbers, and measure pixels.

Colors may be specified by name using one of the names in the file **rgb.txt**, or by their hexadecimal values in the form **0xRRGGBB**, where **RR**, **GG**, and **BB** are the red, green, and blue components each specified as two hexadecimal digits in the range **00** to **FF**.

The **`define`** command makes a record of the definition of an shape and makes it available later for a draw command. The **`draw`** command creates an object in the screen's object list, to be displayed when the program starts.

The following commands are recognized. Each command is on a line by itself and terminates with the newline character. If the last character on a line is a backslash (\), the command is continued onto the next line.

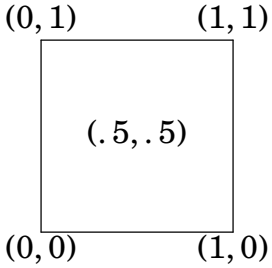
`#` ...
: A comment line is ignored, as are empty lines and lines consisting only of white space.

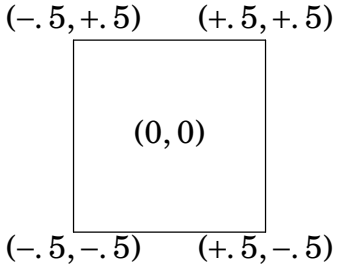
`define` *name* **`text`** *font words*...
: A text object is created by concatenating all of the words together, each separated from the next by spaces. The font is any of the seven GLUT bitmap fonts:
  - **`Fixed-8x13`**
  - **`Fixed-9x15`**
  - **`Helvetica-10`**
  - **`Helvetica-12`**
  - **`Helvetica-18`**
  - **`Times-Roman-10`**
  - **`Times-Roman-24`**

`define` *name* **`ellipse`** *width height*
: An ellipse is specified with the major and minor diameters specified. When drawn, the drawing coördinate is the center.

`define` *name* **`circle`** *diameter*
: A circle is just an ellipse whose length and width are equal.

`define` *name* **`polygon`** *x0 y0 x1 y1 x2 y2 ...*
: A polygon is specified with the number of vertices being equal to the number of (x, y) coördinates given. The polygon must be convex (each interior angle less than 180 degrees). The convexity is not verified by the program, so the appearance of a non-convex polygon is undefined. The center of the polygon is at $(\bar{x}, \bar{y})$, i.e., at the average of all the x coördinates and the average of the y coördinates. After computing the average, it is subtracted from each of the vertices to normalize them. For example, if the vertices are specified as shown in the left side of the diagram, they are normalized to that shown on the right side, so that $(\bar{x}, \bar{y})$ changes from $(.5, .5)$ to $(0, 0)$.





`define` *name* **`rectangle`** *width height*
: A rectangle with the given width and height is defined with strictly horizontal and vertical lines with the given thickness. When drawn, the drawing coördinates will be the center.

`define` *name* **`square`** *width*
: A square is a rectangle with equal width and height.

`define` *name* **`diamond`** *width height*
: A diamond's width is from the leftmost to rightmost point and its height is from topmost to bottommost height. The draw command specifies its center.

`define` *name* **`triangle`** *x0 y0 x1 y1 x2 y2*
: A triangle is just another name for a polygon.

`define` *name* **`equilateral`** *width*
: An equilateral triangle has a horizontal base (two points have the same y-value). The apex has a higher y-value. The constructor just uses the constructor for a triangle.

`border` *color thickness*
: Specifies the thickness in pixels of the border surrounding the selected object, and its color. Defaults: red, 4.

`moveby` *pixels*
: The move parameter indicates how many pixels an object moves when directed by one keystroke. Default: 4.

`draw` *color name xcenter ycenter*
: A geometrical object (an ellipse or a polygon) is drawn with the center at the coördinates (x, y), using the color specified. A text object is drawn using (x, y) as the coördinate of the left end of the string at the baseline.

KEYBOARD INPUT

Once the input file is finished, the keyboard is used to command the movements of the objects. The following keystrokes control the program:

`q`	Quit (exit the program).
`h`	Move the selected object left.
`j`	Move the selected object down.
`k`	Move the selected object up.
`l`	Move the selected object right.
`n`	Set the selection to the next object.
`p`	Set the selection to the previous object.
`0` ... `9`	Set the selection to objects 0 through 9, respectively.

An object is always displayed using the coördinates of the center. If an object is moved off the window, it appears at the other end of the window. That is, if the x or y coördinate exceeds the width (height) of the windor, it is set to 0. If it becomes negative, it is set to the width (height). The selected object is always surrounded by a border to make it visible. For objects 0 through 9, the object's number is displayed in the center of the object.

EXIT STATUS

0 No errors were detected.

1 Errors were detected and messages printed to the standard error.

3. Code

OpenGL and GLUT will be used to display the window. See the short example programs in the `misc/` subdirectory. Starter code is in the `code/` subdirectory. For documentation, Google using the terms "OpenGL" and/or "GLUT" and the name of the function, then look for results naming the web pages in

`www.opengl.org/documentation`

Following are some functions you might need:

```
glBegin
glClear
glClearColor
glColor3ub
glColor3ubv
glEnable
glEnd
glFlush
glHint
glLineWidth
glLoadIdentity
glMatrixMode
glOrtho
glPointSize
glPopMatrix
glPushMatrix
glRasterPos2f
glRasterPos2i
glRotatef
glTranslatef
glVertex2f
glViewport
gluOrtho2D
glutAddMenuEntry
glutAddSubMenu
glutAttachMenu
glutBitmapCharacter
glutBitmapHeight
glutBitmapLength
glutBitmapString
glutCloseFunc
glutCreateMenu
glutCreateWindow
glutDisplayFunc
glutEntryFunc
glutInit
```

```
glutInitDisplayMode
glutInitWindowPosition
glutInitWindowSize
glutKeyboardFunc
glutKeyboardUpFunc
glutMainLoop
glutMotionFunc
glutMouseFunc
glutPassiveMotionFunc
glutPostRedisplay
glutReshapeFunc
glutSetIconTitle
glutSetWindowTitle
glutSpecialFunc
glutSpecialUpFunc
glutSwapBuffers
glutTimerFunc
glutWireTeapot
```

4. What to Submit

Submit all C++ source files and the **`Makefile`**. If you are doing pair programming, also submit the **`PARTNER`** file.