The Objectdraw Graphics Library

Contact Information: Kim B. Bruce (`kim@cs.williams.edu`), Andrea P. Danyluk (`andrea@cs.williams.edu`), and Thomas P. Murtagh (`tom@cs.williams.edu`).

Problem Statement: Computer graphics can be used in introductory CS courses as good early examples of the use of objects. As many have discovered over the years, the use of graphics also has many pedagogical benefits, including (1) making the actions of programs more concrete and visible, (2) in providing visible feedback when errors occur, and (3) making possible more interesting and motivating programs. Unfortunately, standard Java graphics suffer from several problems, most of which are listed in section A1 of the Task Force's Taxonomy of Problems, which make them harder to use effectively in introductory courses.

Solution Overview: The objectdraw library contains classes representing drawing canvases (one each for the use with the AWT and Swing libraries) and classes representing graphics objects that can be placed on the canvas.

Graphics classes in the library include `FramedRect`, `FilledRect`, `FramedOval`, `FilledOval`, `FramedArc`, `FilledArc`, `VisibleImage`, `Line`, and `Text`. (A visible image is used to create a graphic object from a jpeg or gif file.) A hierarchy of interfaces is also provided so that different kinds of graphics objects can be used in the same context.

The graphics objects associated with a canvas have the following characteristics:

1. Objects appear immediately on a canvas when created.
2. When the characteristics of graphics objects (e.g., location, size, color, etc.) are changed, the changes appear immediately.
3. The canvas automatically takes care of all repaints by keeping internally a list of all objects on the screen.
4. Coordinates on the canvas can be provided by a pair of doubles or by a `Location` object, where `Location` is an objectdraw class representing a pair of x and y coordinates. All coordinates are represented as doubles to make it possible for students to ignore problems of truncation when creating animations.

A drawing canvas is a GUI component that represents a drawing area on the screen. When a graphical object is created, the programmer must specify the canvas on which it should appear. Each canvas is implemented as a list of graphical objects rather than as an off-screen bit map. When the system requests a repaint of the screen or the user's code modifies any object, our library's version of `paint` erases the entire screen and redraws all the objects in each canvas. With this simple technique, screen updates appear to occur instantaneously. This removes a large burden from novice programmers. As a convenience for novices, the objectdraw library also supports a `WindowController` class, an extension of `JApplet` that has a canvas installed in the center.

Our goal with this library was not to provide a full-featured graphics library. Instead we wished only to provide simple primitives that could be used with as little overhead as possible. Because of the pedagogical goals, we made a conscious decision to leave out certain features. For example, we do not provide a graphics class that can be used to construct composite objects. Instead, we show

```
public class DragBall extends WindowController {
  private FilledOval ball; // ball
  private Location lastPos;  // last mouse posn

  // make the ball
  public void begin() {
    ball = new FilledOval(95,50,30,30, canvas);
    ball.setColor(Color.red);
    new Text("Drag the ball!", 75,20, canvas);
  }

  // Save starting point and if point in box
  public void onMousePress(Location point) {
    lastPos = point;
  }

  // if mouse is in ball, then drag the ball
  public void onMouseDrag(Location point) {
    if ( ball.contains(lastPos) ) {
      ball.move( point.getX()-lastPos.getX(),
                 point.getY()-lastPos.getY());
      lastPos = point;
    }
  }

  // if dragging when release, go back to start
  public void onMouseRelease(Location point) {
    if (ball.contains(lastPos))
      ball.moveTo(95,50);
  }
}
```

Figure 1: Class illustrating the objectdraw graphics library

our students how to build these on their own so that they gain experience in designing classes and writing methods.

Experience with the solution: We have been using the objectdraw library for 5 years. Faculty at more than a dozen colleges, universities, and high schools have used the library in conjunction with our on-line text and lecture notes. We have recently ported the library from AWT to Swing. Those using the library have reported that it is reliable and highly motivating for students.

We have found the library especially useful when combined with the support for event-driven programming that is also contained in the library and is described in a separate document. The combination of truly object-oriented graphics and event-driven programming leads to the possibility of writing very interesting and motivating programs from the first week of classes.

Figure 1 illustrates a very simple program that allows the user to drag around a filled oval on the screen. It illustrates the use of constructors as well as the `setColor`, `contains`, `moveTo`, and `move` methods. It also illustrates some of the features of the language that support event-driven programming.

API Documentation: Documentation for the classes in the library may be found at:

`http://cortland.cs.williams.edu/~cs134/eof/library/javadocs/`

Supplemental Materials: The web page

`http://cortland.cs.williams.edu/~cs134/eof/`

includes links to an early AWT version of the objectdraw library, textbook examples, papers on our approach, instructions on using the library with different IDE's, and a collection of resources for instructors. The resources for instructors includes lecture notes, other sample programs, and laboratory assignments. This page also has a link to another page that contains the Swing version of the objectdraw library as well as a more recent version of the text that supports the Swing version of the library.