Graphical Structures

Eric Roberts
CS 106A
January 29, 2010

The `GPolygon` Class

- The `GPolygon` class is used to represent graphical objects bound by line segments. In mathematics, such figures are called ***polygons*** and consist of a set of ***vertices*** connected by ***edges***. The following figures are examples of polygons:

- Unlike the other shape classes, that location of a polygon is not fixed at the upper left corner. What you do instead is pick a ***reference point*** that is convenient for that particular shape and then position the vertices relative to that reference point.
- The most convenient reference point is often the geometric center of the object.

Constructing a `GPolygon` Object

- The `GPolygon` constructor creates an empty polygon. Once you have the empty polygon, you then add each vertex to the polygon, one at a time, until the entire polygon is complete.
- The most straightforward way to create a `GPolygon` is to use the method `addVertex(x, y)`, which adds a new vertex to the polygon. The *x* and *y* values are measured relative to the reference point for the polygon rather than the origin.
- When you start to build up the polygon, it always makes sense to use `addVertex(x, y)` to add the first vertex. Once you have added the first vertex, you can call any of the following methods to add the remaining ones:
  - `addVertex(x, y)` adds a new vertex relative to the reference point
  - `addEdge(dx, dy)` adds a new vertex relative to the preceding one
  - `addPolarEdge(r, theta)` adds a new vertex using polar coordinates

  Each of these strategies is illustrated in a subsequent slide.

Using `addVertex` and `addEdge`

- The `addVertex` and `addEdge` methods each add one new vertex to a `GPolygon` object. The only difference is in how you specify the coordinates. The `addVertex` method uses coordinates relative to the reference point, while the `addEdge` method indicates displacements from the previous vertex.
- Your decision about which of these methods to use is based on what information you have readily at hand. If you can easily calculate the coordinates of the vertices, `addVertex` is probably the right choice. If, however, it is much easier to describe each edge, `addEdge` is probably a better strategy.
- No matter which of these methods you use, the `GPolygon` class closes the polygon before displaying it by adding an edge from the last vertex back to the first one, if necessary.
- The next two slides show how to construct a diamond-shaped polygon using the `addVertex` and the `addEdge` strategies.

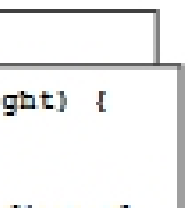
Drawing a Diamond (`addVertex`)

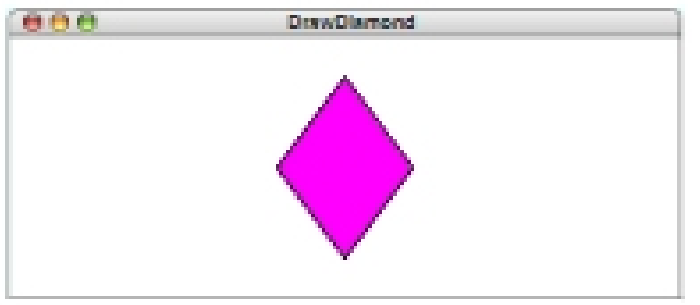
The following program draws a diamond using `addVertex`:

```
public void run() {
```

```
private GPolygon createDiamond(double width, double height) {
   GPolygon diamond = new GPolygon();
   diamond.addVertex(-width / 2, 0);
   diamond.addVertex(0, -height / 2);
   diamond.addVertex(width / 2, 0);
   diamond.addVertex(0, height / 2);
   return diamond;
}
```

diamond





Drawing a Diamond (`addEdge`)

This program draws the same diamond using `addEdge`:

```
public void run() {
```

```
private GPolygon createDiamond(double width, double height) {
   GPolygon diamond = new GPolygon();
   diamond.addVertex(-width / 2, 0);
   diamond.addEdge(width / 2, -height / 2);
   diamond.addEdge(width / 2, height / 2);
   diamond.addEdge(-width / 2, height / 2);
   diamond.addEdge(-width / 2, -height / 2);
   return diamond;
}
```

diamond