1 Administrative Topics

- We look at projects

2 Graphics Objects

<table>
<thead>
<tr>
<th>Object Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Window</td>
<td>an object that allows you to put graphics into a window</td>
</tr>
<tr>
<td>Point</td>
<td>an object that allows you to specify the x-y coordinates of a graphics object</td>
</tr>
<tr>
<td>Circle</td>
<td>a circular graphics object – lets you draw a circle</td>
</tr>
<tr>
<td>Rectangle</td>
<td>a rectangular graphics object – lets you draw a rectangle</td>
</tr>
<tr>
<td>Oval</td>
<td>an elliptic graphics object – lets you draw an oval</td>
</tr>
<tr>
<td>Line</td>
<td>a line graphics object</td>
</tr>
<tr>
<td>Polygon</td>
<td>a graphics object that is constructed by providing all the vertices</td>
</tr>
</tbody>
</table>

The Graphwin methods we are interested in are:

- win.getMouse() – waits for a mouse click
- win.close() – closes the window
• win.checkMouse() – True if a mouse has recently clicked on the window

Zelle has organized the methods for the shape functions well. There are some methods (and functions) specific to each shape. Most notably, the functions used to create each object are unique:

• Circle( centerpt, radius ). centerpt is a Point specifying the location of the center of the circle. radius is a number.

• Rectangle( pt1, pt2 ). each pt is a Point specifying opposite corners of the rectangle (e.g. upper left and lower right)

• Oval( pt1, pt2 ). Each pt is a Point specify oppoosite corners of the rectangle that bounds the oval you want to draw

• Line( pt1, pt2 ). Each pt is an endpoint of the line segment.

• Polygon( pt1, pt2, pt3, ...) or Polygon( [pt1, pt2, pt3, ...] ). Each pt is a vertex of a polygon.

There are several methods that are general - -they can be used on any of the shape objects. Suppose that obj is a Circle, Rectangle, Oval, Line, or Polygon. Then, the follow methods are defined:

• obj.draw(win) – places obj in the window (win) (which must be a GraphWin)

• obj.move(dx, dy) – moves the obj by a given amount

• obj.setOutline(clr) – sets the outline color

• obj.setWidth(width) – sets the width of the outline (in pixels)

• obj.setFill(clr) – sets the fill color

For the color-setting methods, clr must be either a string (e.g. 'green', 'blue', etc.) or a color_rgb (e.g. graphics.color_rgb( 255, 150, 20 ) ).

To draw a simple circle, you use code like this:
# Make a window big enough to hold the scene
w = 300
h = 300
win = graphics.GraphWin("Crazy Toy", w, h)

# Make a circle centered at 150,150 with radius 20
c = graphics.Circle(graphics.Point(150, 150), 20)

# Draw the circle (i.e. put it in the window)
c.draw(win)

# Keep the window open for awhile
win.getMouse() # Pause to view result
win.close()    # Close window when done
The color/width-setting methods that allow us to do make more interesting scenes.

Below is a picture of a very simplified view of a Fisher Price Person.

```python
# Make a window big enough to hold the scene
w = 300
h = 300
win = graphics.GraphWin("Crazy Toy", w, h)

# Make a person
body = graphics.Rectangle(
    graphics.Point(w/2-20, 150),
    graphics.Point(w/2+20, 150+60))
body.setFill('blue')
body.setOutline('blue')
head = graphics.Circle(
    graphics.Point(w/2, 150-20), 20)
head.setOutline(graphics.color_rgb(0,0,0))
head.setFill(graphics.color_rgb(255,0,255))

# Draw the objects.
body.draw(win)
head.draw(win)

# Keep the window open for awhile
win.getMouse() # Pause to view result
win.close() # Close window when done
```

Let’s add more features to the person. How can we manage the code better? What about putting the objects into a list? That will make the drawing code take 2 lines. And, in the future, if we want to move the person, we can also do that with two lines.