1 Introduction

The CreateGraphics package allows the developer to:

- Generate basic graphics
- Add event listeners
- Sound functionality for built-in mp3 files
- Basic "tween" animations

Functionality is performed using the createjs javascript library available at:
`http://www.createjs.com/`.

2 Graphics

In order to work with the CreateGraphics package, you need to include the following line at the top of your Grace file:

```
import createGraphics as cg
```

The graphics object needs to be created with the following command:

```
def graphics = createGraphics(width,height)
```

where width and height correspond to the desired graphics window width and height. For example:

```
def width = 300
def height = 300
def graphics = cg.createGraphics(width, height)
```

3 Shapes

The following shape objects are available in CreateGraphics:Circle, Rectangle, Rounded Rectangle, PolyStar, Ellipse, and Text. To draw one of these objects to the screen, pass a message to the graphics object:

- addCircle
- addRect
- addPolyStar
- addRoundRect
- addEllipse

For instance, to add a circle to the window, you might do the following:

```
import createGraphics as cg
def graphics = cg.createGraphics(300, 300)
graphics.addCircle
graphics.draw
```

Each shape has different parameters that are used to create it. These parameters have default values, so you don't need to set each one every time you create an shape.

3.1 Common Parameters

There are a few parameters that are common to each type of shape.

- **location** (Point): The x,y coordinates where the shape will be placed in the graphics window. Coordinates are expressed in Grace "Point" notation: x@y. Keep in mind that the origin is in the upper left corner of the window, so 10@10 will be 10 down and 10 right from the corner of the window.
- **color** (String): The color of the shape. Most basic colors can be set as "red", "blue", etc. However, you can also use 6-digit hex numbers such as "#CC3300" that corresponds to an HTML 5 hex colors. See `http://www.w3schools.com/tags/ref_colorpicker.asp` for more details.
- **fill** (Boolean): Whether or not you want to fill in the shape when it is drawn on the window.

3.2 Circle

Create: graphics.addCircle

Parameters:

- **radius** (Number): The length of the circle radius

3.3 Rectangle

Create: graphics.addRect

Parameters:

- **width** (Number): Width of the rectangle
- **height**(Number): Height of the rectangle

3.4 Rounded Rectangle

Create: graphics.addRoundRect

Parameters:

- **width** (Number): Width of the rectangle
- **height**(Number): Height of the rectangle
- **radius** (Number): Radius of the rounded corners

3.5 PolyStar

Create: graphics.addPolyStar

Parameters:

- **size** (Number): Length of each side of the star

- **sides** (Number): Number of sides
- **pointSize** (Number): Size of the points
- **angle** (Number): Angle between the points

3.6 Ellipse

Create: graphics.addEllipse

Parameters:

- **width** (Number): Width of the ellipse
- **height**(Number): Height of the ellipse

3.7 Text

Create: graphics.addText

Parameters:

- **content** (String): The content of the string

4 Drawing a Shape

To draw a shape on the graphics window, first create it, then configure it, and then draw it. The following code creates the output down in Figure 1.

```
import createGraphics as cg
def graphics = cg.createGraphics(200, 200)
def circle = graphics.addCircle
circle.color := "red"
circle.radius := 20
circle.position := 30@30
circle.fill := true
graphics.draw
```

Figure 1: Creating a red circle

5 Adding a Click Handler

Adding a click handler to a shape defines a block of code that will be executed when the shape is clicked. For instance, let's say that we want the red circle to turn blue when it is clicked, and we want to add a message to the user. Then you would add something like this:

```
import createGraphics as cg
def graphics = cg.createGraphics(200, 200)
def circle = graphics.addCircle
circle.color := "red"
circle.click := {
  print("clicked circle")
  circle.color := "blue"
  graphics.draw
}
graphics.draw
```

Notice that "graphics.draw" is located twice in this code. The second instance of graphics.draw (located in the last line of the code block) initially draws the circle onto the window. The first instance of graphics.draw (located at the end of the circle.click block), updates the circle on the window after it has been clicked. This instance of graphics.draw will occur at some point in the future when the circle is clicked.

6 Adding sound

CreateGraphics supports basic sounds. All sounds are preloaded in the browser and cannot be customized at this time. To play a sound, just use the "play" method of the graphics object. For example:

```
import createGraphics as cg
def graphics = cg.createGraphics(200, 200)
def circle = graphics.addCircle
circle.color := "red"
circle.click := {
  print("clicked circle")
  graphics.play("bicycle_bell")
  graphics.draw
}
graphics.draw
```

The following sounds are available: note1, note2, note3, note4, note5, note6, note7, note8, bicycle_bell, snap, whoosh, shutter.