Motivation: How to create these pattern?

How to achieve repetition?
How to store color values so as to repeat them in the same order?

Graphics with Loops

A simple flower

```
from cslgraphics import *
paper = Canvas(800, 700, 'skyBlue', 'Rotation Designs')
for i in range(12):
    petal = Ellipse(150, 30)
    petal.setFillColor('yellow')
    # 75 is one half 150; try -95 instead
    petal.adjustReference(-75, 0)
    petal.rotate(i*30)  # 30 is 360/12
    petal.moveTo(200, 200)
paper.add(petal)
```
Abstracting over our flower with makeFlower

Define a function `makeFlower` that takes as arguments (1) the number of petals (2) the color of each petal (3) the width of each petal and (4) the height of each petal and returns a `Layer` object with an appropriately constructed flower object.

```python
makeFlower(12, 'yellow', 30, 150)
makeFlower(10, 'brown', 90, 150)
makeFlower(30, 'magenta', 20, 150)
```

A simple nautilus shell

```python
for i in range(50):
    ring = Circle(100)
    ring.setFillColor('white')
    # adjust by radius size
    ring.adjustReference(-100, 0)
    # 10 is just a small amount
    ring.rotate(i*10)
    # 0.95 just makes it smaller by a tad
    ring.scale(0.95**i)
    ring.moveTo(500, 150)
paper.add(ring)
```

Parameterize it: makeNautilus

```python
shell = makeNautilus(50, 100, 10, .95, 'pink')
shell.moveTo(200, 200)
paper.add(shell)
```

```python
def makeNautilus(num, size, angle, shrink, color):
    nautilus = Layer()
    for i in range(num):
        ring = Circle(size)
        ring.setFillColor(color)
        ring.adjustReference(-size, 0)
        ring.rotate(i*angle)
        ring.scale(shrink**i)
        nautilus.add(ring)
    return nautilus
```

Make it fancy: makeColorfulNautilus

```python
colorfulShell = makeColorfulNautilus(50, 100, 10, .95, ['pink', 'blue', 'green', 'magenta'])
colorfulShell.moveTo((400, 400))
paper.add(colorfulShell)
```

```python
def makeColorfulNautilus(num, size, angle, shrink, colorList):
    nautilus = Layer()
    for i in range(num):
        ring = Circle(size)
        ring.setFillColor(colorList[i % len(colorList)])
        ring.adjustReference(-size, 0)
        ring.rotate(i*angle)
        ring.scale(shrink**i)
        nautilus.add(ring)
    return nautilus
```

The % operator makes sure that despite the value of num (and as a result, of i), the indices always are only between 0 and len of colorList.
### Rotated squares

The class `Color` is defined in es1graphics. The function `randomColor` will generate new colors, thus, when running this code you’ll see a differently colored graphics.

```python
for i in range(16):
    s = Square(200, Point(200, 500))
    s.rotate(6*i)
    s.scale(0.9**i)
    s.setFillColor(Color.randomColor())
    paper.add(s)
```

### Rose-colored squares

#### # Helper function
```python
def makeColor(redFraction, greenFraction, blueFraction):
    return Color((255.0*redFraction, 255.0*greenFraction, 255.0*blueFraction))
```

```python
for i in range(25):
    s = Square(10 + 10*i, Point(600,500))
    s.rotate(15*i)
    s.setDepth(i)
    s.setFillColor(makeColor(i/24.0, 0.5, 0.5))
    paper.add(s)
```

If we don’t use `setDepth`, the bigger square (the last one) will be displayed on top. Comment out that line to see for yourself.