

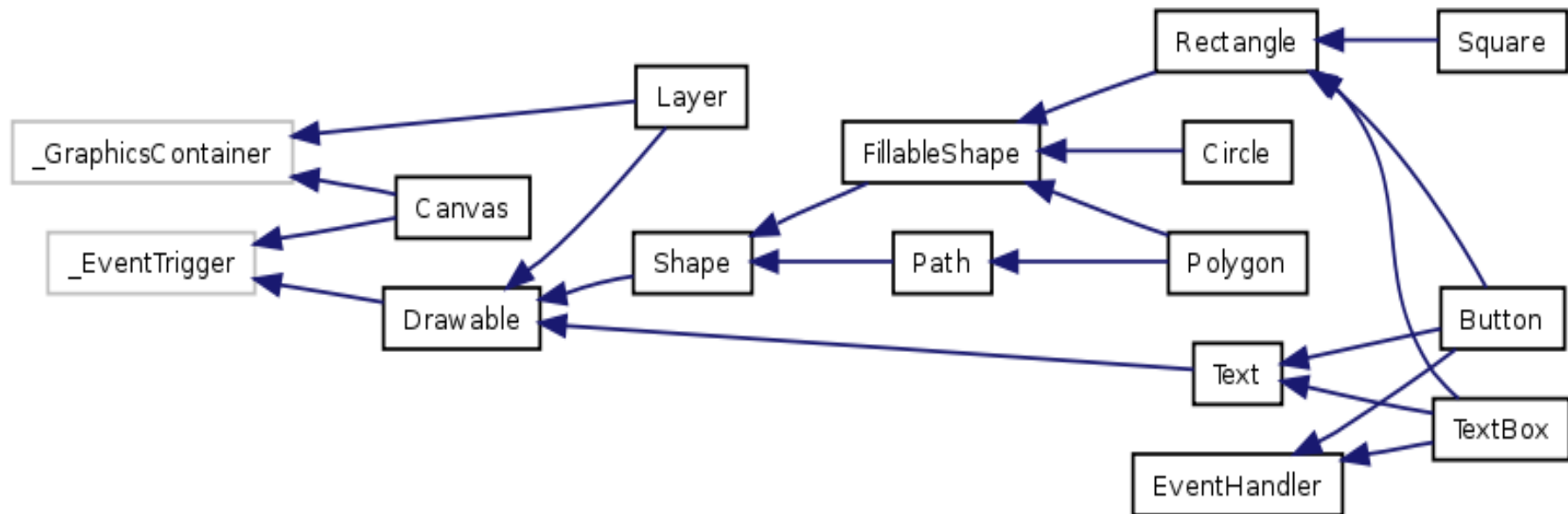
Lecture 17

Chapter 9 Inheritance

9.4 Class Hierarchies and cs1graphics

9.4 Class Hierarchies and cs1graphics

Let's explore the rich hierarchy in cs1graphics module.



This graph can be found on Documentation page of www.cs1graphics.org

9.4 Class Hierarchies and cs1graphics

We can recall that in some of the examples given in Chapter 3, I used a 'star' that was defined as a polygon.

Why don't we try to design a **Star** class (new drawable class)? If we do this, there will be no need for future users to deal with geometry of stars, i.e. all the user will need is to say is how many rays he/she wants the star to have.

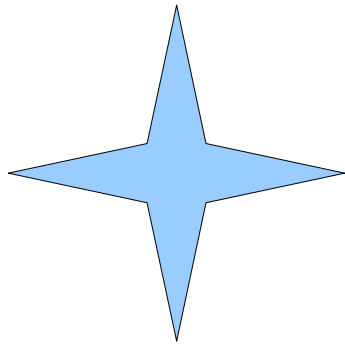
Since a star is a special case of a **Polygon**, let the Polygon be a parent class of the Star class.

```
class Star(Polygon)
```

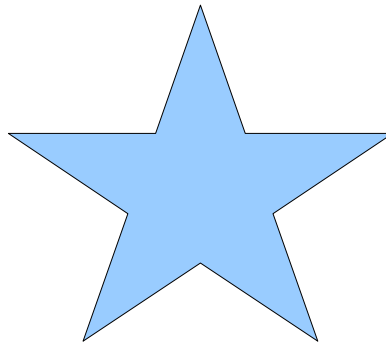
By doing this, we will inherit many useful behaviors (like drawing a polygon, adjusting the border, fill color, moving, scaling, and so on).

9.4 Class Hierarchies and cs1graphics

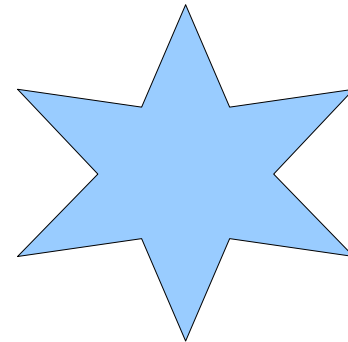
Our biggest challenge will be implementing a constructor that establishes the initial geometry of a star.



4-point star



5-point star



6-point star

A star with n rays has $2n$ points.

Also we have two radii (outer and inner):

outer is measured from the center of the star to the tips of the rays,
inner is measured from the center of the star to the place where two neighboring rays meet.

9.4 Class Hierarchies and cs1graphics

A decision:

Let's allow the user to specify the outer radius and the ration between the inner and outer radii.

In addition, we can allow the user to specify the reference point too.

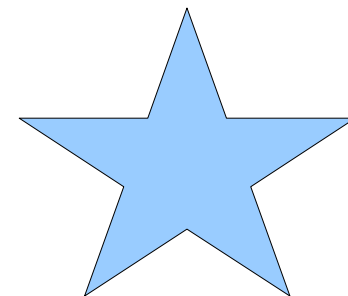
So, the constructor will be:

```
__init__(self, numRays=5, outerRadius=30, innerRatio=0.5,  
         center=Point(0,0))
```

$$InnerRatio = \frac{(inner\ radius)}{(outer\ radius)} < 1$$

The smaller the ratio (closer to 0), the “thinner” the star,
The larger the ratio (closer to 1), the “puffier” the star.

9.4 Class Hierarchies and cs1graphics

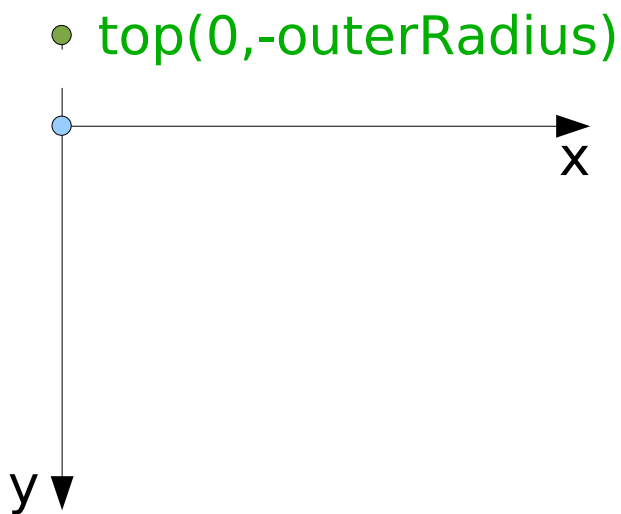


Let's begin to build the class Star:

```
class Star(Polygon):
    def __init__(self, numRays=5, outerRadius=30,
                 innerRatio=0.5, center=Point(30,30)):
        Polygon.__init__(self) # call the parent constructor
        # a polygon with 0 points is created

        top = Point(0, -outerRadius) # top point of the star
        print "Top point:", top

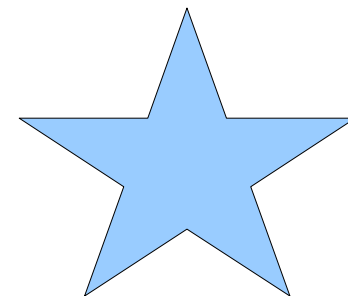
        angle = 360.0/(2*numRays) # angle between points
        print "Angle:", angle
```



We put the top point of the outerRadius units above the origin.

And will be building a star as if it is centered around the origin

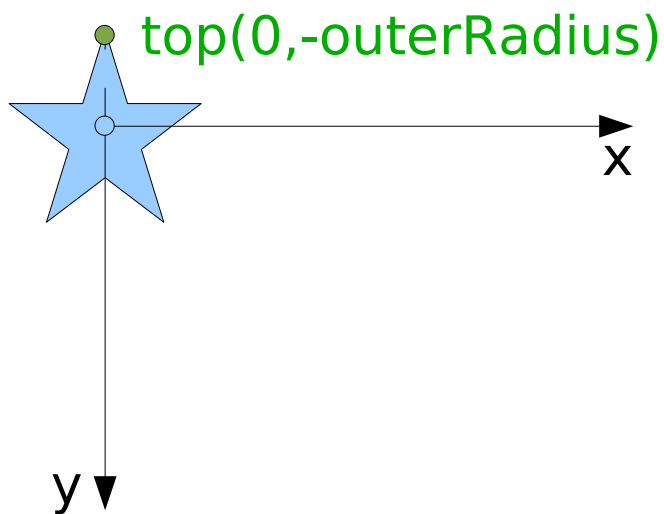
9.4 Class Hierarchies and cs1graphics



Let's begin to build the class Star:

```
class Star(Polygon):
    def __init__(self, numRays=5, outerRadius=30,
                 innerRatio=0.5, center=Point(30,30)):
        Polygon.__init__(self) # call the parent constructor
        # a polygon with 0 points is created

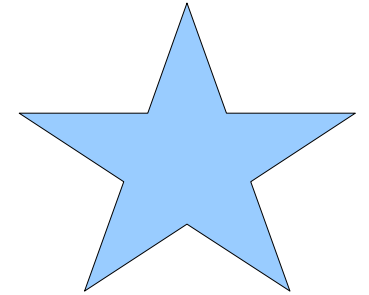
        top = Point(0, -outerRadius) # top point of the star
        print "Top point:", top
```



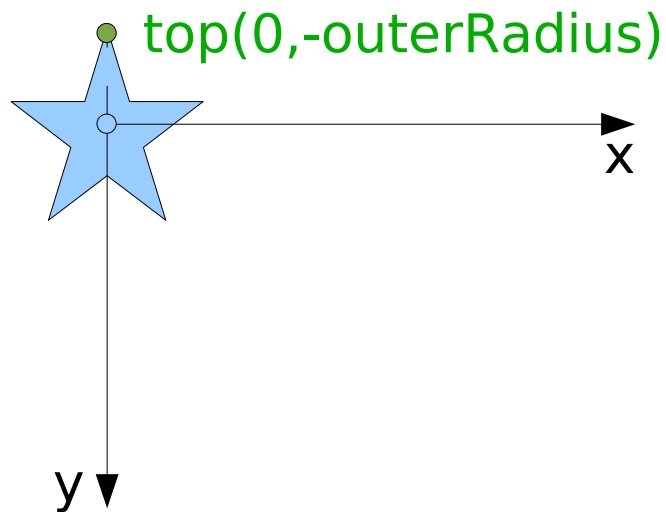
We put the top point of the outerRadius units above the origin.

And will be building a star as if it is centered around the origin

9.4 Class Hierarchies and cs1graphics



There is a reason for centering the star around the origin: we will see it later.



We put the top point of the `outerRadius` units above the origin.

And will be building a star as if it is centered around the origin

9.4 Class Hierarchies and cs1graphics

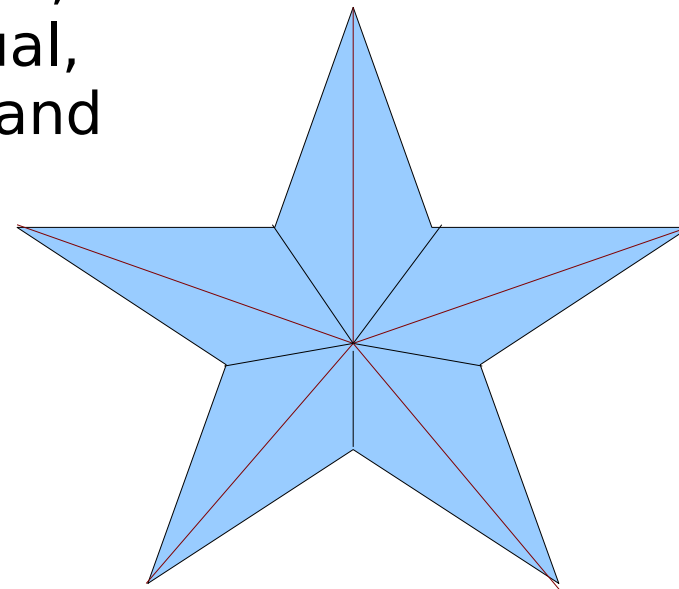
Let's proceed:

```
angle = 360.0/(2*numRays) # angle between points  
print "Angle:",angle
```

The angles between outer points are equal,
the angles between inner points are equal,
and the angles between adjacent outer and
inner points are equal too.

Therefore each small angle is $\frac{360}{(2*n)}$,
where n is the number of
star rays.

Why do we need this?



9.4 Class Hierarchies and cs1graphics

Why do we need this?

We will start with the top point, and will be calculating all the other points by dully rotating and scaling vector, represented by the origin and the top point.

For this will will use operations such as `*` and `^`:

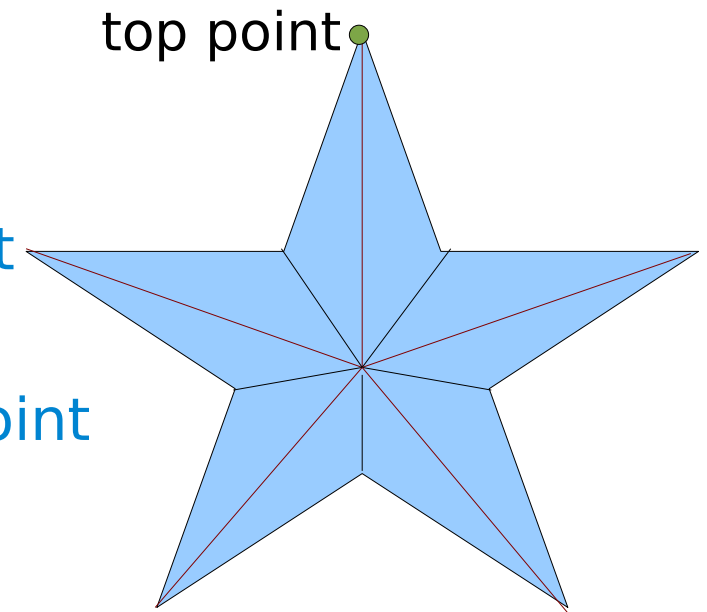
`Point*n` - multiplies vector with the `Point` as the end-point by the scalar `n`

`Point^angle` - rotates the vector with `Point` as the end-point of the vector to `angle` degrees clockwise about the origin.

See methods of Point class:

`__mul__` (`self,operand`) and
`__xor__` (`self,angle`)

in the documentation of cs1graphics



9.4 Class Hierarchies and cs1graphics

Let's proceed:

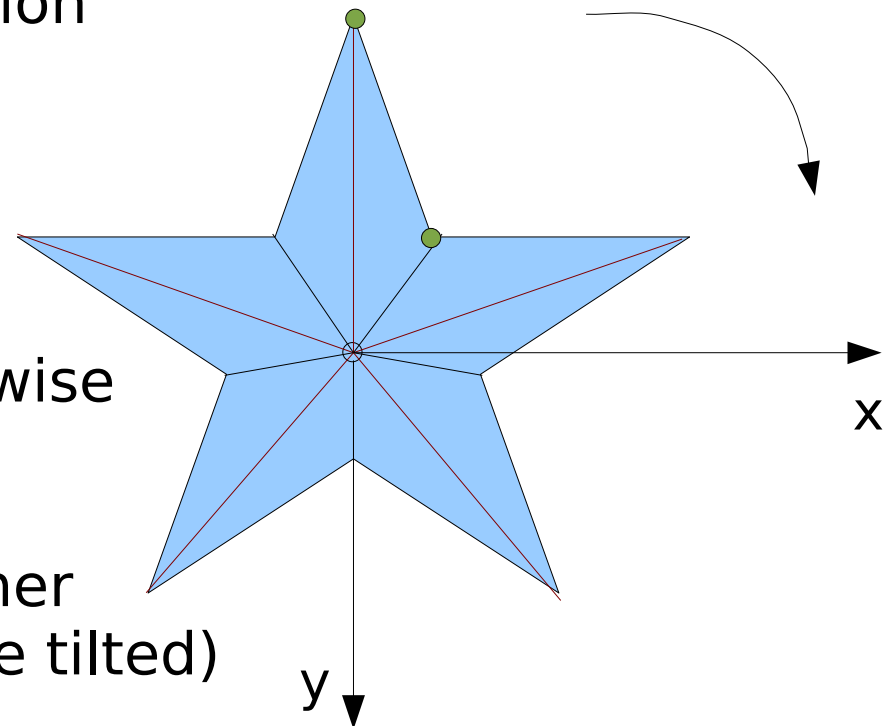
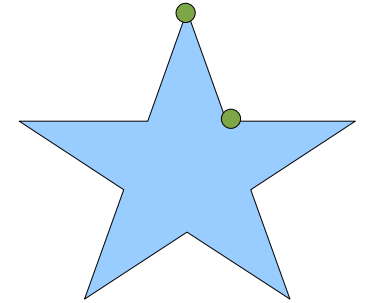
```
for i in range(numRays):
    # adding ray point
    self.addPoint(top^(angle*2*i))

    # adding inner point
    self.addPoint(innerRatio*top^(angle*(2*i+1)))
```

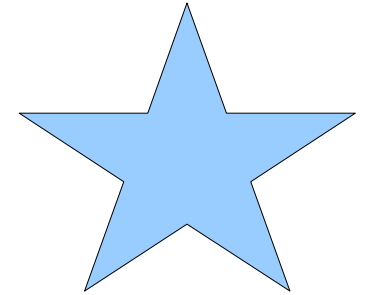
Two points are added at each iteration of this for loop – one outer point, one inner point.

And the reason for centering the star around the origin is that operator \wedge rotates the vector clockwise **about the origin**.

See what happens if you choose other top point (most likely the star will be tilted)



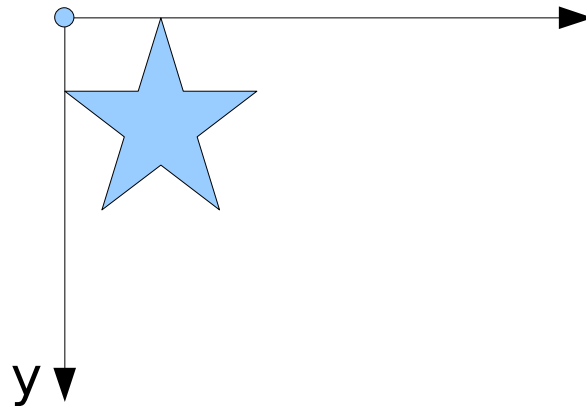
9.4 Class Hierarchies and cs1graphics



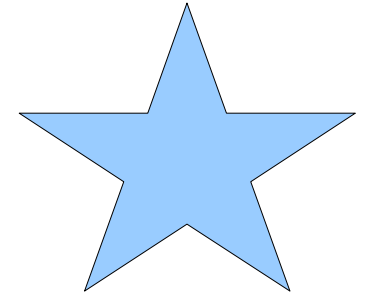
Let's proceed:

```
# moving Reference Point from the  
#top point to the center of the star  
self.adjustReference(0,outerRadius)
```

```
# moving the star to the location given by user,  
# or to the default location  
self.moveTo(center.getX(),center.getY())
```



9.4 Class Hierarchies and cs1graphics



One more addition:

We can allow user to change the innerRatio.

In this case we need to create one more attribute:

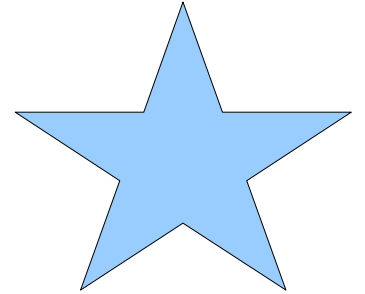
`_innerRatio`, and then write a method that will allow to update the innerRatio

```
# record the inner ratio as attribute  
self._innerRatio=innerRatio
```

```
def setInnerRatio(self,newRatio):  
    factor=newRatio/self._innerRatio  
    self._innerRatio=newRatio
```

```
#we will modify only inner points  
for i in range(1,self.getNumberOfPoints(),2):  
    self.setPoint(factor*self.getPoint(i),i)
```

9.4 Class Hierarchies and cs1graphics



See the program [star_class.py](#)