Lecture 8

Chapter 3 Getting Started with Graphics

3.3 Rotating, Scaling, and Flipping

3.4 Cloning

3.3 Rotating, Scaling, and Flipping

By design, all of the Drawable objects can be rotated, scaled, or flipped.

For these operations, the object's reference point is very important (it serves as a point of the shape that stays fixed during the transformation).

Recall that initially
for `Square`, `Circle`, `Rectangle`, `Image` or `Text` :
reference point is the **center** of an object,

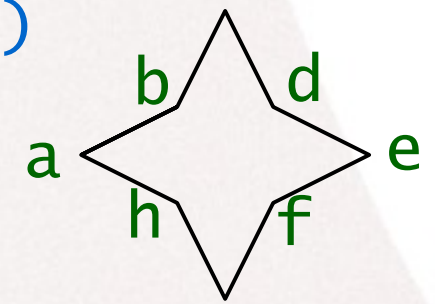
for `Polygon` and `Path`:
reference point is the location of the **first point**.

Rotating

`rotate(angle)` is a method of `Drawable` class

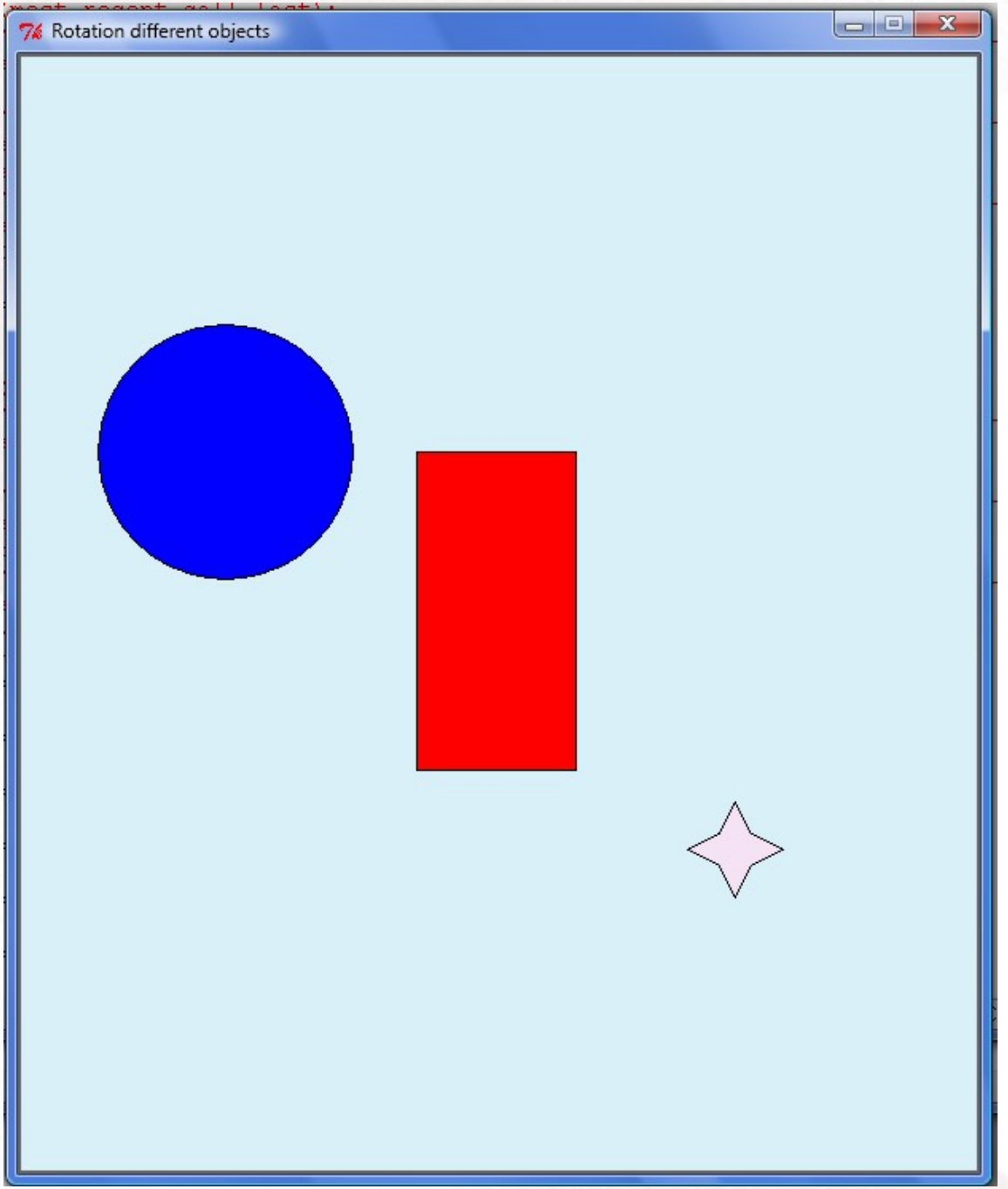
Example:

```
r=Rectangle(100,200,Point(300,350))
r.setFillColor('Red')

c=Circle(80,Point(130,250))
c.setFillColor('Blue')

a,b,c_=Point(10,25),Point(20,20), Point(25,10)
d,e,f = Point(30,20), Point(40,25), Point(30,30)
g,h =Point(25,40), Point(20,30)
p=Polygon(a,b,c_,d,e,f,g,h)
p.setFillColor((245,226,245))
p.move(400,400)

paper.add(r), paper.add(c), paper.add(p)
```



Rotating

Result:



Rotating

`rotate(angle)` is a method of `Drawable` class (clockwise rotation by `angle` degrees)

Example: next we add the following loop:

```
for i in range(100):
        r.rotate(15)
        c.rotate(15)
        p.rotate(15)
        time.sleep(0.25)
```

What can we expect to happen?

See program rotation-example1.py

We can slightly change program – move the reference point of the circle, see program rotation-example2.py

! text rotation was not implemented before – this needs to be checked

Scaling

- reference point remains fixed, all other points in an object are scaled relative to the reference point

`scale(factor)` is a method of `Drawable` class

Example:

```
 r=Rectangle(100,200,Point(300,350))
r.setFillColor('Red')

c=Circle(80,Point(130,250))
c.setFillColor('Blue')
c.adjustReference(50,10)

paper.add(r), paper.add(c)
time.sleep(1)

r.scale(0.5), c.scale(0.5)
time.sleep(1)

r.scale(0.5), c.scale(0.5)
```

What will this code do? See program scaling-example.py

Flipping

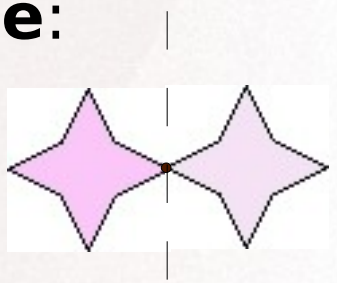
- taking a mirror image of an object

`flip(angle)` is a method of `Drawable` class

- if no argument is given to `flip` method, the flip takes place across a *vertical axis of symmetry*, passing through the reference point.

- parameter `angle` specifies the clockwise rotation of *axis of symmetry* away from vertical.

Example:



`star.flip()`

`star.flip(20)`

Flipping

Example:

```
r=Rectangle(100,200,Point(300,350))
r.setFillColor('Red')
 r.adjustReference(-30,-50)

star=Polygon(a,b,c,d,e,f,g,h)
star.move(400,450)
star.adjustReference(60,0)

r.flip(), star.flip()
time.sleep(2)

r.flip(), star.flip()
time.sleep(3)

r.flip(20), star.flip(20)
```

What can we expect to happen?

See program flipping-example.py

3.4 Cloning

Drawable types support a convenient `clone` method that returns a **brand new copy**.

The clone has precisely the same settings as the original element, but **is not automatically added to any canvases**.

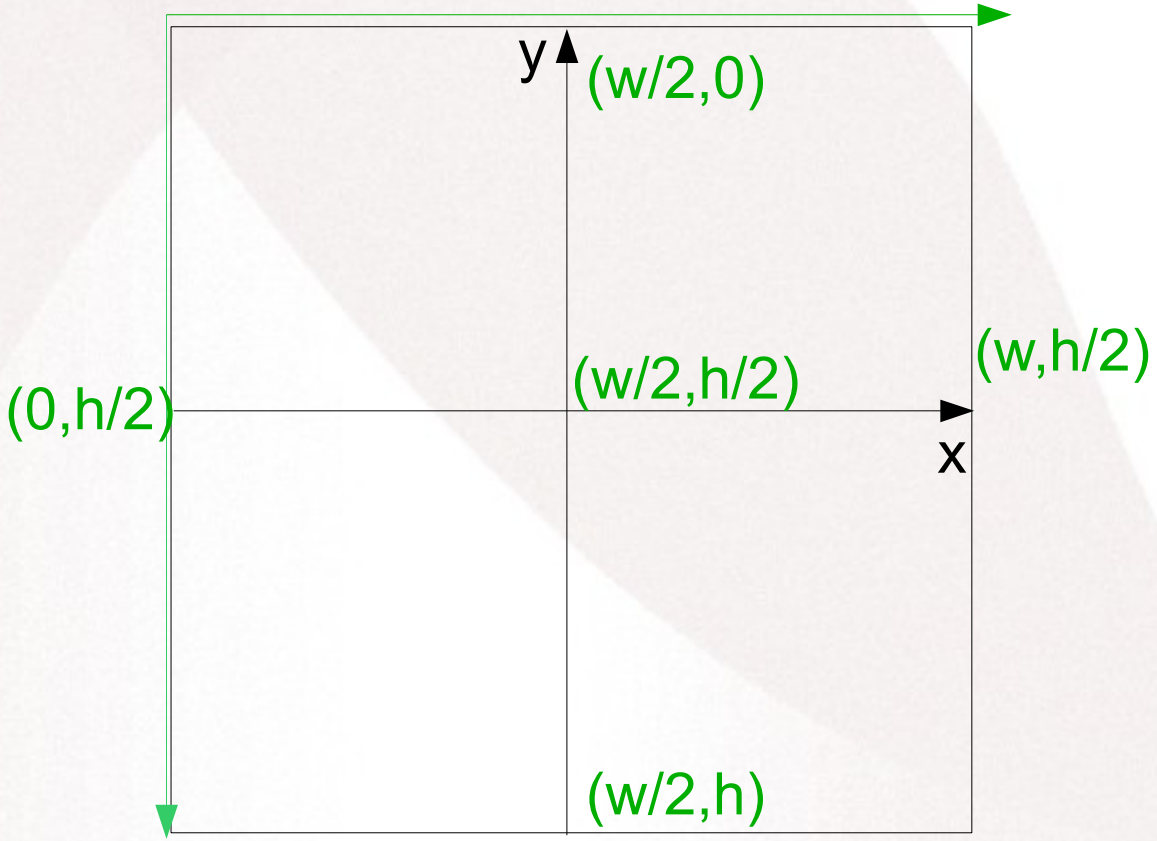
Example:

```
star=Polygon(a,b,c,d,e,f,g,h)
 star.move(400,450)

star2=star.clone()
star2.move(-100,-50)
star2.scale(2)
paper.add(star2)
```

See program cloning-example.py
And a more interesting one: cloning-fun.py

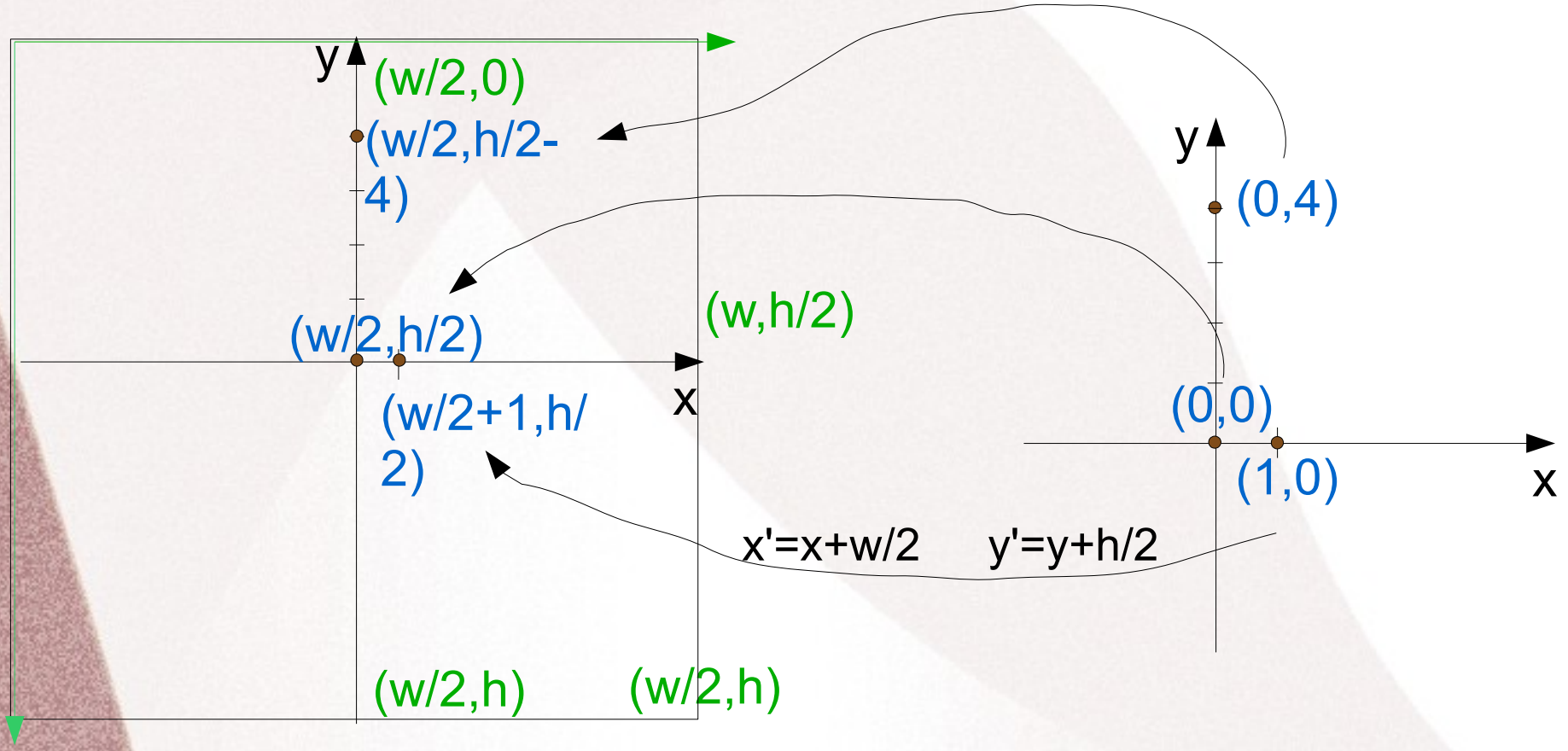
Rectangular coordinate system vs cs1graphics coordinate system

How to draw a scale that will cut the graphics window into four equal quadrants?
Let **w** stand for width, and **h** stand for height of the canvas. Then



Rectangular coordinate system vs cs1graphics coordinate system

How do we jump from out rectangular coordinate system to graphics window (canvas) coordinate system)?



Rectangular coordinate system vs cs1graphics coordinate system

We also need to keep in minds that in the graphics window everything is done using pixels, which have smaller scale than when we draw a graph on paper.

2 options:

1) do a straightforward mapping
   $y = f(x) = x^2-10$, assume that function f: $\mathbf{Z} \rightarrow \mathbf{Z}$
   integers $\rightarrow$ pixels

2) do scaling:
   for example, let 1 be 10 pixels, 2 be 20 pixels, -3 be -30 pixels (multiply the number by 10)

Rectangular coordinate system vs cs1graphics coordinate system

In case of 1) : a straightforward mapping

$y = f(x) = x^2\text{-}10$, assume that function f: $\mathbf{Z} \rightarrow \mathbf{Z}$

integers $\rightarrow$ pixels

The formulas for conversion from our rectangular coordinate system to the Canvas coordinate system will be

$$x_{Canvas} = x + w/2 \qquad y_{Canvas} = -y + h/2$$

Rectangular coordinate system vs cs1graphics coordinate system

In case of 2): a scaling
for example, let 1 be 10 pixels, 2 be 20 pixels, -3 be -30 pixels (multiply the number by 10)

let
$scale_x$ be the scale (number of pixels for one unit) for x,
and
$scale_y$ be the scale (number of pixels for one unit) for y
then

$x_{Canvas} = x \cdot scale_x + w/2$ $\quad$ $y_{Canvas} = -y \cdot scale_y + h/2$

Rectangular coordinate system vs cs1graphics coordinate system

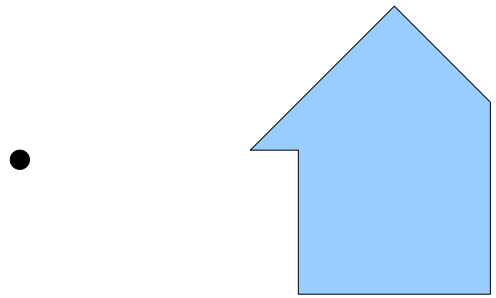
program rec_coord_system.py

draws the rectangular coordinate system in a window 800x800 pixels, and

a graph of the function $y = 3x+1$ (which is a straight line)

In-class assignment

Create a simple object, then display it in the graphics window (paper, and then rotate it around some point in the graphics window (on the paper)



Homework Assignment (not for grade)

- Use the figure you did in previous assignment. Perform the following actions:
  - shrink it,
  - stretch it,
  - flip it,
  - rotate it about some arbitrary point in the graphics window, which is not on/in the figure,
  - make few copies of it, with different colors and dimensions (shrink, stretch, rotate...).
- Draw a rectangular coordinate system, and using line segments (path) draw a “curve” of $y=x^2-10$ for $x = [-5,5]$