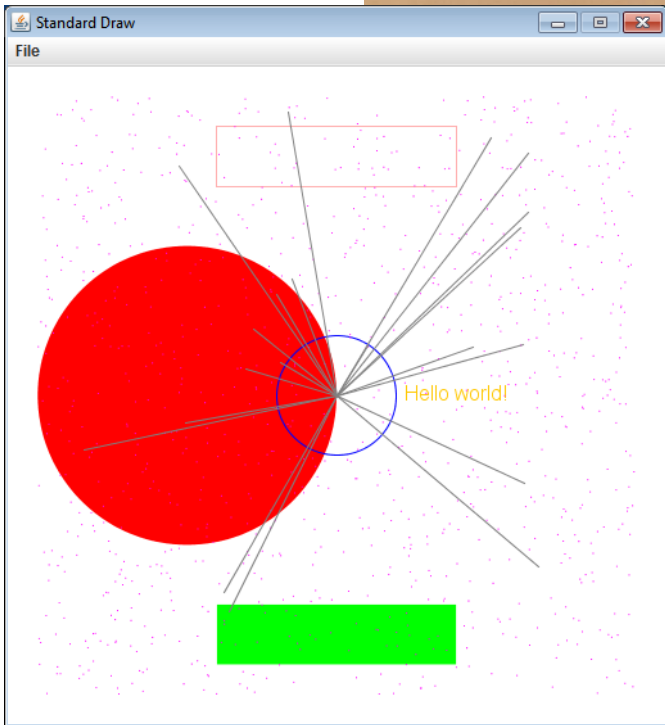
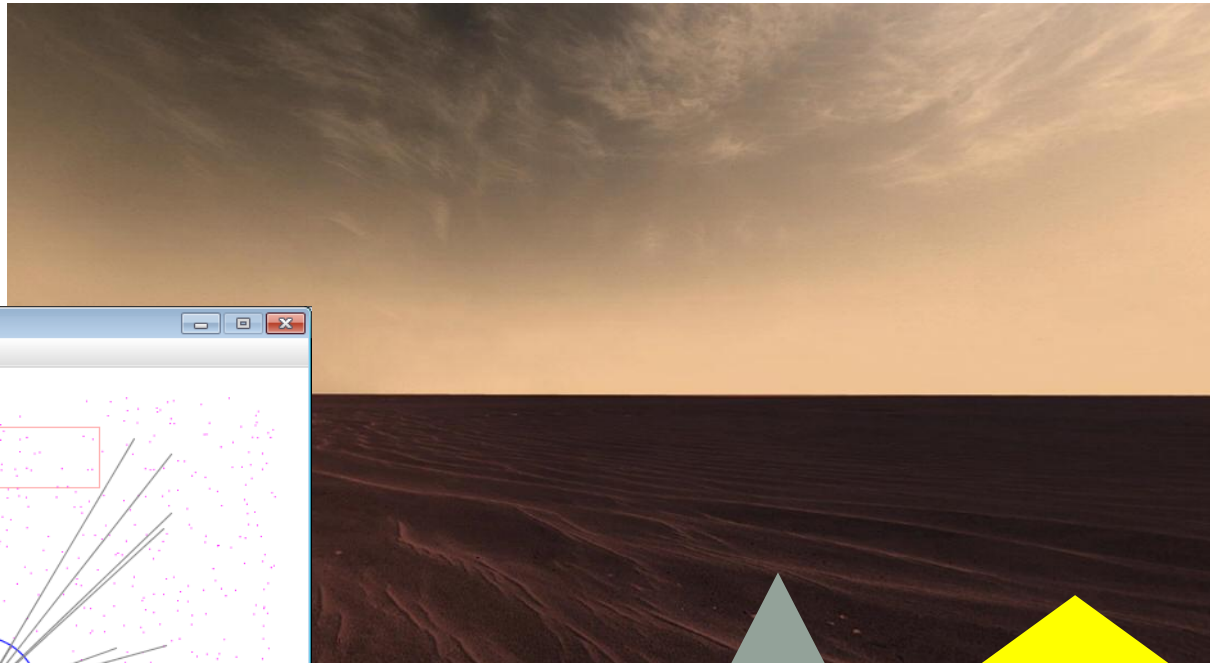
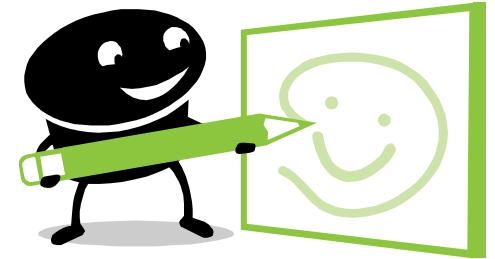


GRAPHICS AND SOUND



Outline

- File Input
- Graphics
 - StdDraw.py
 - Draw primitive **shapes**
 - Draw **images** from a file
 - Create **animation** loops
 - Get **keyboard input** from users
- Audio
 - StdAudio.py
 - **Play audio** files



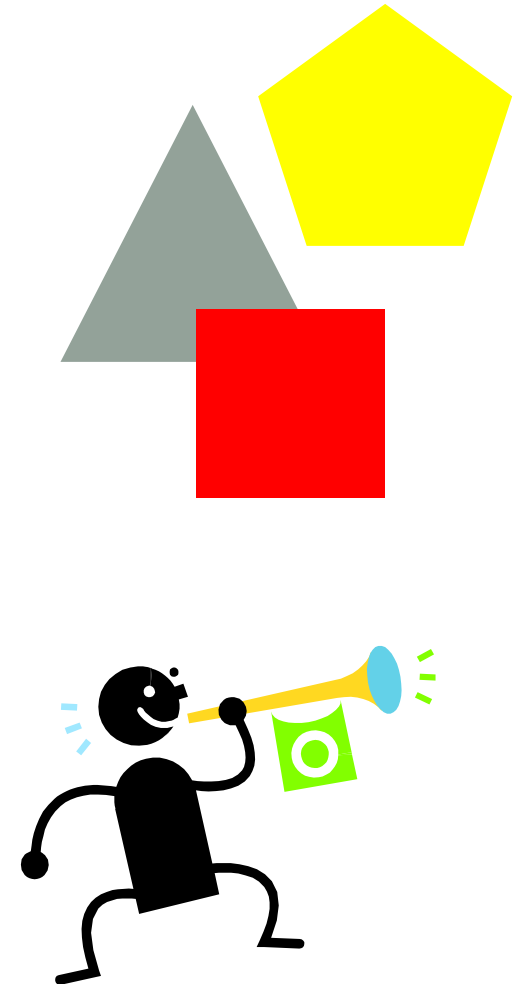
Input and Output Thus Far

- **Input**
 - Parsing **command line** arguments
 - **Reading interactively** from user
- **Output**
 - **Display text to console**
 - ... and some audio output ...

```
Level: 0
. . ! . . . . .
. . . . . . . .
. . . . . . . .
. . . . . . . .
. . . . * . . . .
. . . . . . . .
. . . . . . . .
. . . . . . . .
. . . . . . . .
. . . . . . . #
Direction? s
You walked south
Zombie went east
```

New Input/Output Capabilities

- **File Input**
- **StdDraw**
 - Draw shapes and images
 - Make animated programs
 - Get real-time keyboard input
- **StdAudio**
 - Playback of record sounds
 - Generate your own sounds



Input from Files

- What if..
 - There are too many values for a user to type interactively?
 - These values are stored in a text file?
- Can our program read these values from a file?
 - Yep! 😊

Python File Input

- We need to open the file:
 - **with open(fname, 'r') as f:**
 - fname is a string for the file name
 - f is just any variable that you want to use
 - 'r' means we want to read the file (as opposed to writing it)
- Once we are done with the file, we need to close it:
 - **f.close()**

File Input

```
import sys

sum = 0.0
count = 0

# Check if we need to print out command line help
if len(sys.argv) < 2:
    print("AvgNumsFile <filename>")
else:
    # Open up the text file for reading
    fname = sys.argv[1]
    with open(fname, 'r') as f:
        # Keep going as long as there is more text in the file
        for line in f:
            # Translate that line to a float
            sum += float(line)
            count += 1
    f.close()

# Print out the final average
print(sum / count)
```

StdDraw Overview

- StdDraw

- Like random and sys, we'll use another library: StdDraw

- Put stdDraw.py in directory with your program

- You will also need:

- stdarray.py
- picture.py
- color.py

- Draw simple things:

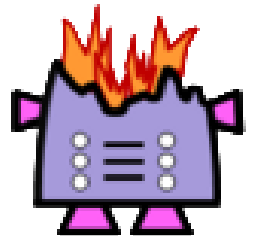
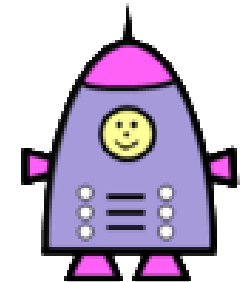
- Rectangles, circles, lines, polygons, text
- Make them different colors

- Draw images loaded from a file:

- e.g. spaceship, Mars background, etc.

- Animate things:

- e.g. bouncing ball, video games



Hello Drawing!

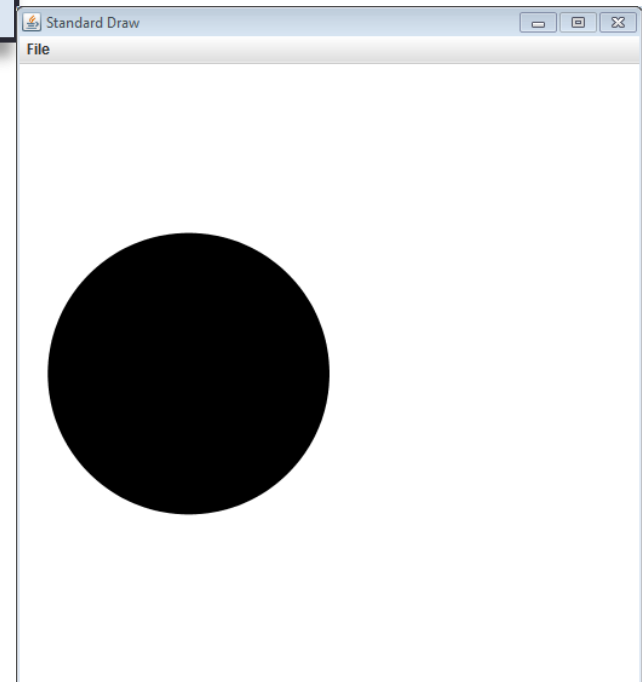
```
import StdDraw

StdDraw.filledCircle(0.25, 0.5, 0.25)
StdDraw.show(0.0)
```

Put the circle at x
coordinate 0.25

Put the circle at y
coordinate 0.5

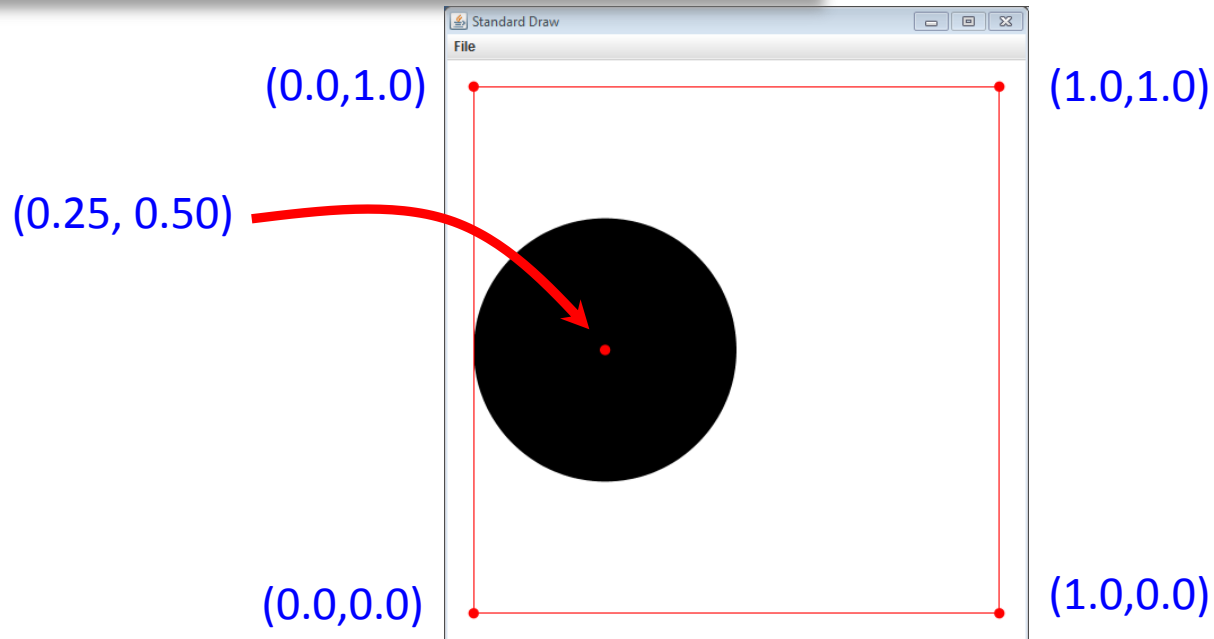
Make the circle be
of radius 0.25



Default Coordinate System

```
import StdDraw

StdDraw.filledCircle(0.25, 0.5, 0.25)
StdDraw.show(0.0)
```



Other Shapes and Text

```
import StdDraw
import random

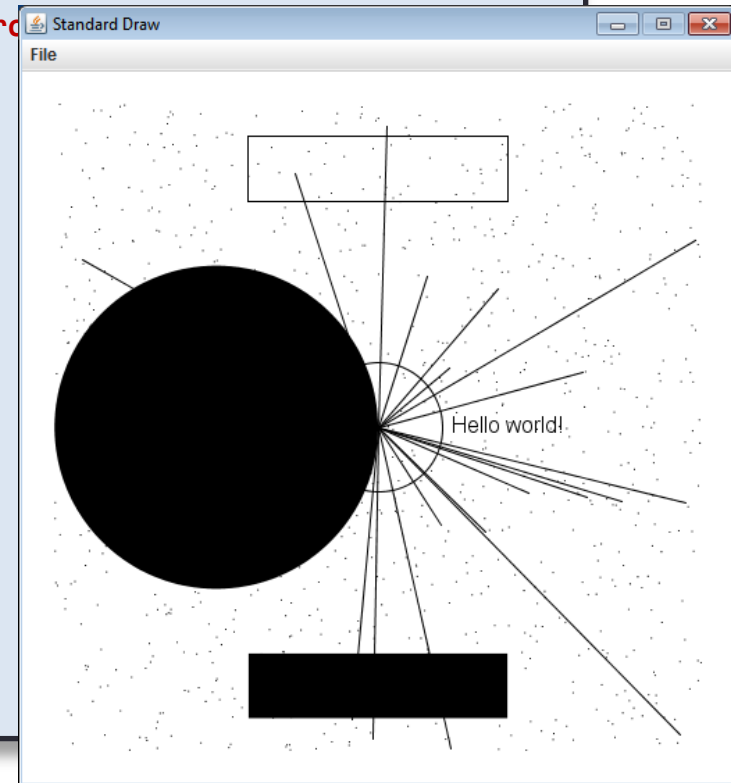
# Drawing circles involves sending in an (x,y) center plus a radius
StdDraw.filledCircle(0.25, 0.5, 0.25)
StdDraw.circle(0.5, 0.5, 0.1)
StdDraw.show(0.0)

# Rectangles are drawn at an (x,y) center with the passed in half
# width and half height argument (the distance to the edge from
# (x,y) center).
StdDraw.filledRectangle(0.5, 0.1, 0.2, 0.05)
StdDraw.rectangle(0.5, 0.9, 0.2, 0.05)
StdDraw.show(0.0)

# Text is drawn centered at the given (x,y)
StdDraw.text(0.7, 0.5, "Hello world!")
StdDraw.show(0.0)

# Scatter a 1000 random points around the screen
for i in range(0, 1000):
    StdDraw.point(random.random(), random.random())
StdDraw.show(0.0)

# 20 random lines radiating from the center
for i in range(0,20):
    StdDraw.line(0.5, 0.5, random.random(), random.random())
StdDraw.show(0.0)
```



Adding Color

```
import StdDraw
import random

# Drawing circles involves sending in an (x,y) center plus a radius
StdDraw.setPenColor(StdDraw.RED)
StdDraw.filledCircle(0.25, 0.5, 0.25)
StdDraw.setPenColor(StdDraw.BLUE)
StdDraw.circle(0.5, 0.5, 0.1)
StdDraw.show(0.0)

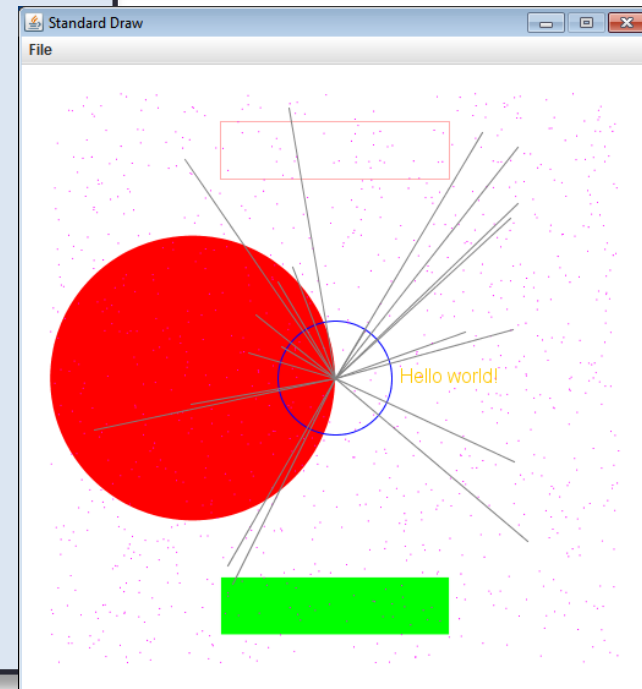
# Rectangles are drawn at an (x,y) center with the passed in half
# width and half height argument (the distance to the edge from the
# (x,y) center).
StdDraw.setPenColor(StdDraw.GREEN)
StdDraw.filledRectangle(0.5, 0.1, 0.2, 0.05)
StdDraw.setPenColor(StdDraw.PINK)
StdDraw.rectangle(0.5, 0.9, 0.2, 0.05)
StdDraw.show(0.0)

# Text is drawn centered at the given (x,y)
StdDraw.setPenColor(StdDraw.ORANGE)
StdDraw.text(0.7, 0.5, "Hello world!")
StdDraw.show(0.0)

# Scatter a 1000 random points around the screen
StdDraw.setPenColor(StdDraw.MAGENTA)
for i in range(0, 1000):
    StdDraw.point(random.random(), random.random())
StdDraw.show(0.0)

# 20 random lines radiating from the center
StdDraw.setPenColor(StdDraw.GRAY)
for i in range(0, 20):
    StdDraw.line(0.5, 0.5, random.random(), random.random())
StdDraw.show(0.0)
```

```
StdDraw.BLACK
StdDraw.BLUE
StdDraw.CYAN
StdDraw.DARK_GRAY
StdDraw.GRAY
StdDraw.GREEN
StdDraw.LIGHT_GRAY
StdDraw.MEGENTA
StdDraw.ORANGE
StdDraw.PINK
StdDraw.RED
StdDraw.WHITE
StdDraw.YELLOW
```



Changing Coordinate Size

- Often convenient to use different coordinates
 - 0.0 to 1.0 is default x-size and y-size
 - Change x-size `StdDraw.setXscale(min, max)`
 - Change y-size `StdDraw.setYscale(min, max)`

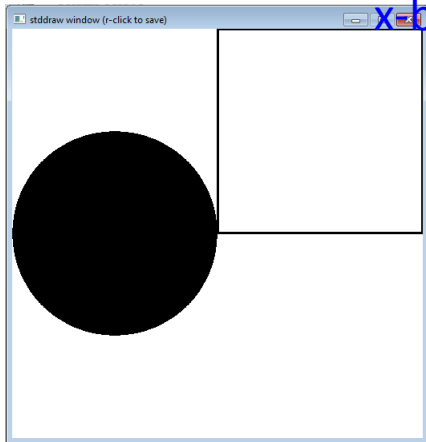
```
StdDraw.filledCircle(0.25, 0.5, 0.25)
StdDraw.rectangle(0.5, 0.5, 0.5, 0.5)
```

x-bottom of rectangle

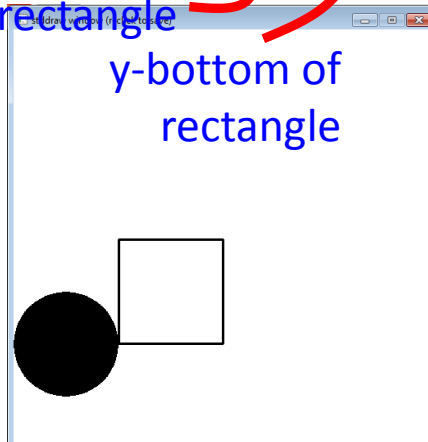
y-bottom of rectangle

half width

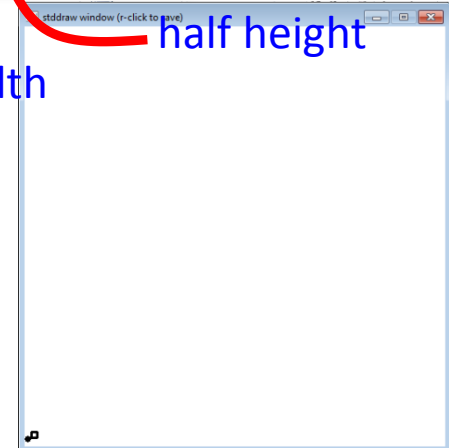
half height



```
StdDraw.setXScale(0.0, 1.0)
StdDraw.setYScale(0.0, 1.0)
```



```
StdDraw.setXScale(0.0, 2.0)
StdDraw.setYScale(0.0, 2.0)
```

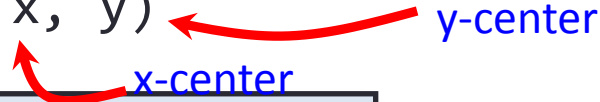


```
StdDraw.setXScale(0.0, 30.0)
StdDraw.setYScale(0.0, 30.0)
```

Drawing Images

- Loading image from file

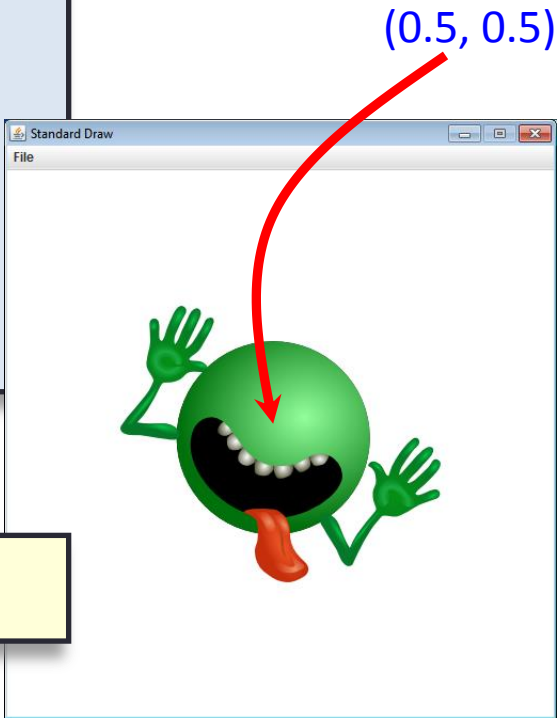
- Supports various formats such as JPG and PNG
- Put image files in same directory with program
- `StdDraw.picture(pic, x, y)`



```
import StdDraw
import sys
import picture as p

pic = p.Picture(sys.argv[1])
StdDraw.picture(pic, 0.5, 0.5)
while True:
    StdDraw.show(0.0)
```

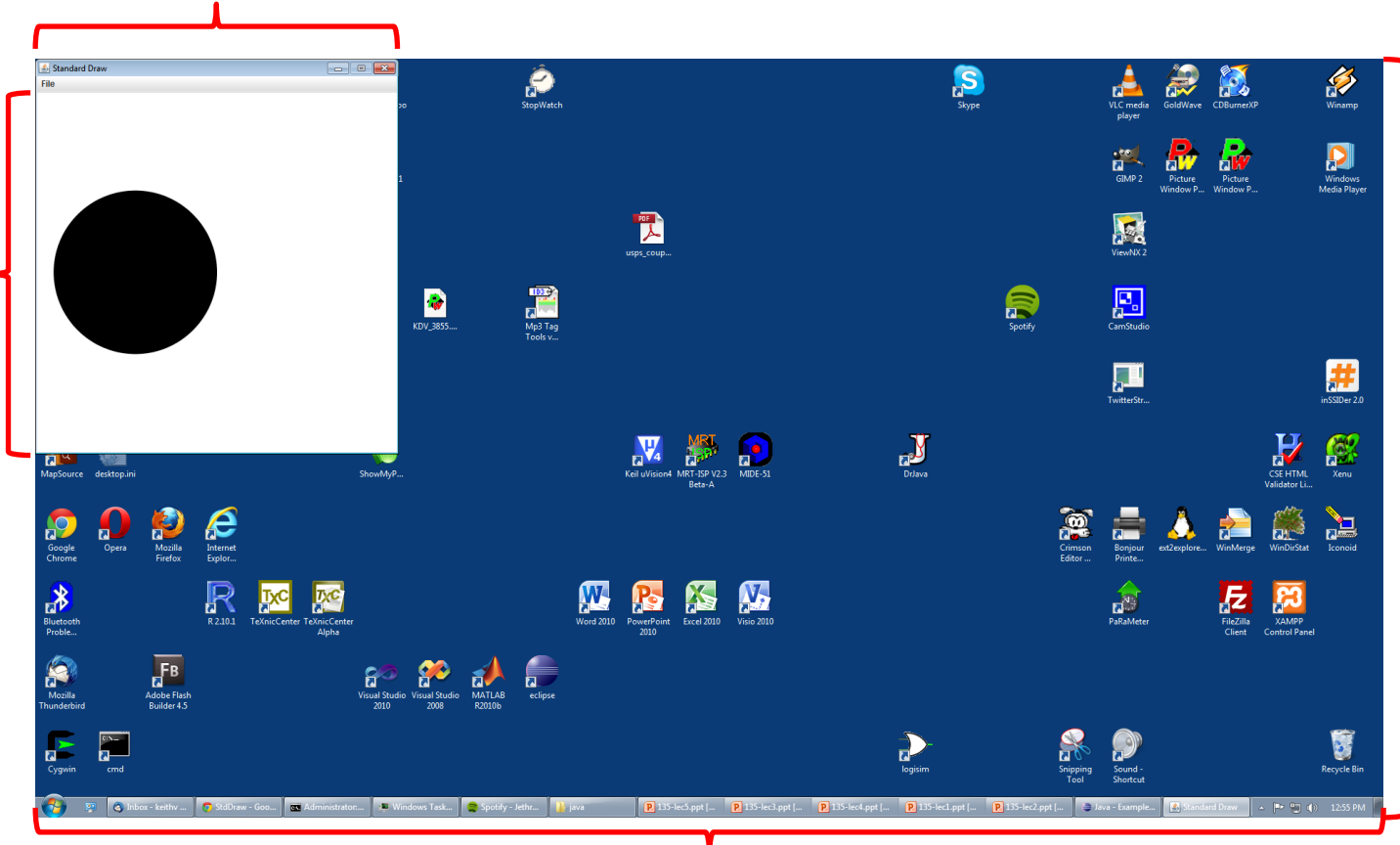
```
% python DrawImage.py dont_panic.png
```



Window Size

512 pixels

512 pixels

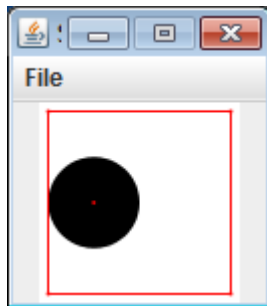


1080 pixels

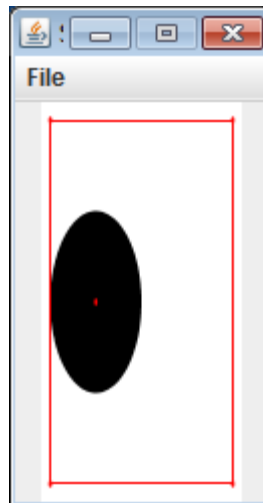
1920 pixels

Changing Window Size

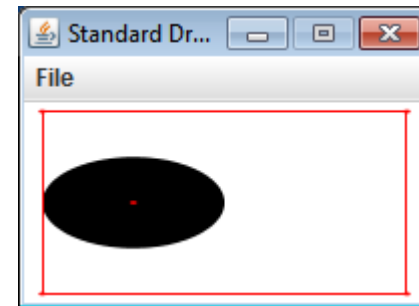
- **Window size**
 - Default size: **512 x 512 pixels**
 - **Set different size:**
 - `StdDraw.setCanvasSize(width, height)`
 - **Call just once** at start of program



100 x 100



100 x 200



200 x 100

Animating Things

• Animation loop

- Clear previous drawing
 - `StdDraw.clear()` (or draw a picture over the screen)
- Draw new stuff
- Sleep for awhile
 - `StdDraw.show(timeMs)`
- Repeat

```
import StdDraw
import sys
import picture as p

x, y = 0.5, 0.5
xOffset, yOffset = 0.01, 0.01
pic = p.Picture(sys.argv[1])
while True:
    StdDraw.clear()
    StdDraw.picture(pic, x, y)
    x += xOffset
    y += yOffset
    if x > 1.0 or x < 0.0:
        xOffset *= -1
    if y > 1.0 or y < 0.0:
        yOffset *= -1
    StdDraw.show(50)
```

Keyboard Input

- Responding to keyboard input
 - Problem: Interactive input waits for text then enter key
 - `StdDraw` gives us real-time keyboard input
 - Check if key was pressed: `StdDraw.hasNextKeyTyped()`
 - Find out the key: `StdDraw.nextKeyTyped()`
 - Note: **must click on drawing window first**
 - Example:
 - Make image spin clockwise on 'a'
 - Make image spin counterclockwise on 's'
 - Stop spinning on any other key

Interactive Bouncing Image

```
import StdDraw
import sys
import picture as p

x, y = 0.5, 0.5
xDirection, yDirection = 0.0, 0.0
xOffset, yOffset = 0.01, 0.01
pic = p.Picture(sys.argv[1])
while True:
    StdDraw.clear()
    StdDraw.picture(pic, x, y)
    if StdDraw.hasNextKeyTyped():
        ch = StdDraw.nextKeyTyped()
        if ch == 'a':
            xOffset += 0.05
        elif ch == 's':
            yOffset += 0.05
    x += xOffset
    y += yOffset
    if x > 1.0 or x < 0.0:
        xOffset *= -1
    if y > 1.0 or y < 0.0:
        yOffset *= -1
    StdDraw.show(50)
```

Adding Sound

- **StdAudio**
 - **Plays sound files** in .wav format
 - Plays one time
 - StdAudio.playFile(filename)
 - Also can play raw audio in double []
 - For creating your own sounds
 - Example, add audio to our bouncing image:

```
import StdAudio
...
else:
    xOffset = 0.01
    yOffset = 0.01
    StdAudio.playFile(sys.argv[2])
```

Additional information

- Many more functions in `StdDraw` and `StdAudio`
 - Can look at the individual programs to see the functions you can call:

```
line(x0, y0, x1, y1)
```

```
point(x, y)
```

```
circle(x, y, radius)
```

```
filledCircle(x, y, radius)
```

```
rectangle(left_x, bottom_y, width, height)
```

```
filledRectangle(left_x, bottom_y, width, height)
```

```
square(left_x, bottom_y, length)
```

```
filledSquare(left_x, bottom_y, length)
```

```
Polygon(listOfXs, listOfYs)
```

```
text(x, y, string)
```

```
setFontFamily(fontFamily)
```

```
setPenRadius(radius)
```

```
setPenColor(color)
```

```
...
```

Summary

- File Input
- Graphics
 - StdDraw.py
 - Draw primitive **shapes**
 - Draw **images** from a file
 - Create **animation** loops
 - Get **keyboard input** from users
- Audio
 - StdAudio.py
 - **Play audio** files

