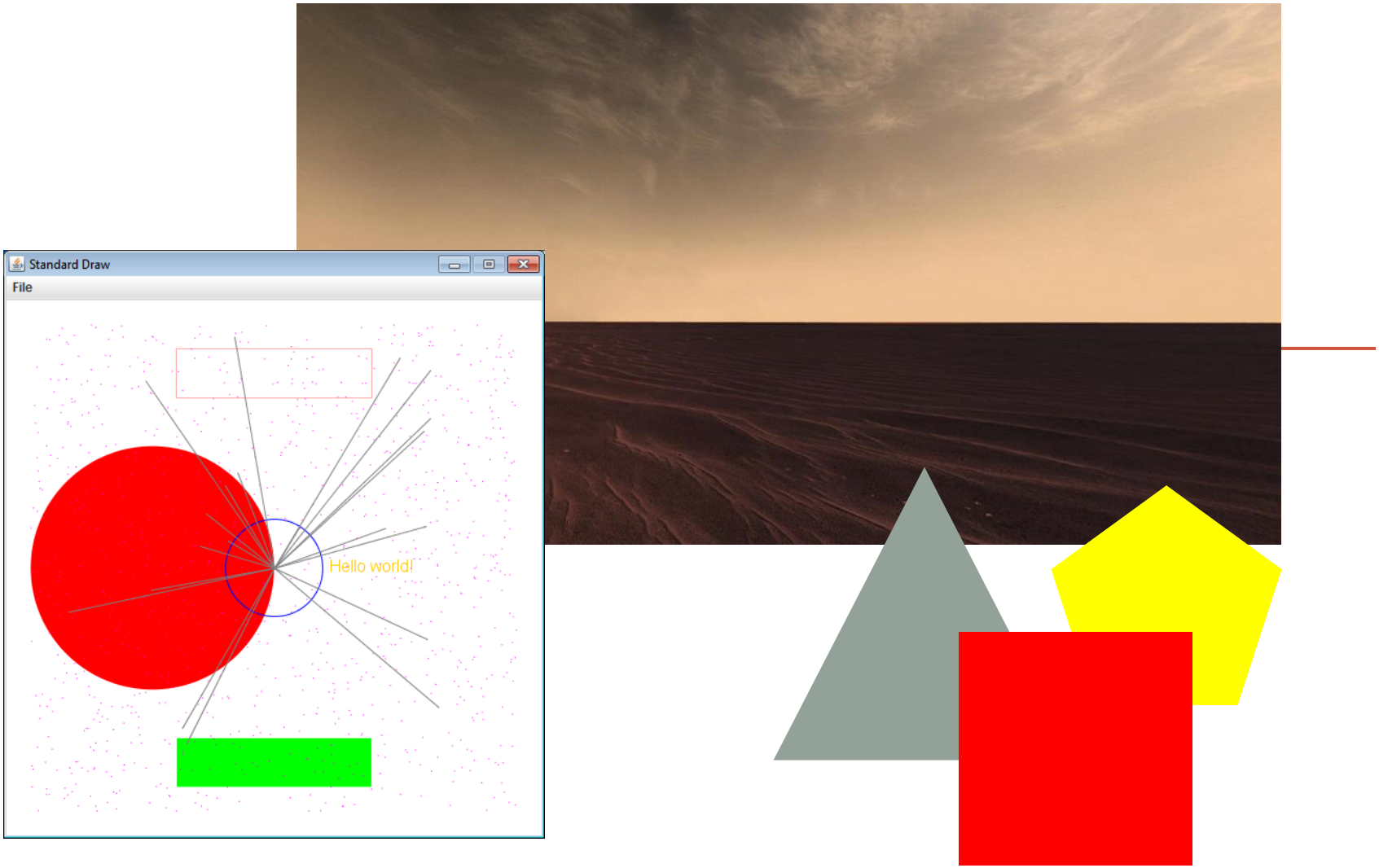
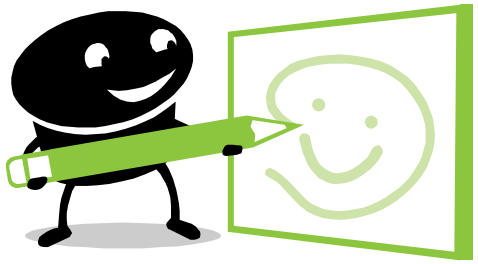
INPUT / OUTPUT PRACTICE



Outline

- File Input
  - Practice
    - Reading configuration files
- Graphics
  - Practice
    - Background Images
    - Animation



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)
```

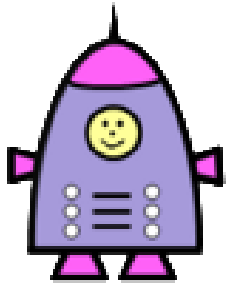
Configuration File: hitchhiker.txt

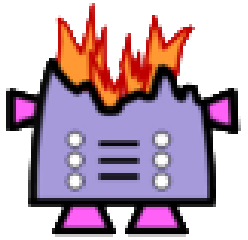
```
stars.jpg
dont_panic_40.png 0.5 0.5 0.035 100
6
asteroid_small.png 0.1 0.1 0.018 -0.002 -0.003
asteroid_medium.png 0.2 0.2 0.030 0.002 -0.003
asteroid_large.png 0.3 0.3 0.065 -0.002 0.003
asteroid_small.png 0.4 0.4 0.018 -0.001 -0.004
asteroid_medium.png 0.6 0.6 0.030 0.002 -0.003
asteroid_large.png 0.7 0.7 0.065 -0.0035 0.0025

# Hitchhikers Guide to the Galaxy: Avoid a bunch of asteroids
#  <background image>
#  <player image> <player x-position> <player y-position> <player radius>
<player speed factor>
#  <number enemies>
#  <enemy0 image> <enemy0 x-position> <enemy0 y-position> <enemy0 x-
velocity> <enemy0 y-velocity>
#  <enemy1 image> <enemy1 x-position> <enemy1 y-position> <enemy1 x-
velocity> <enemy1 y-velocity>
#  ...
```

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





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)`

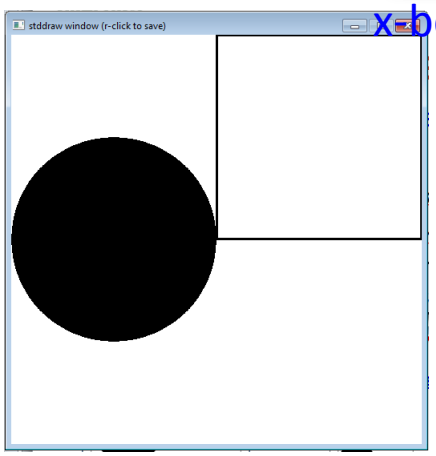
```
StdDraw.filledCircle(0.25, 0.5, 0.25)
StdDraw.rectangle(0.5, 0.5, 0.5, 0.5)
```

x-bottom of rectangle

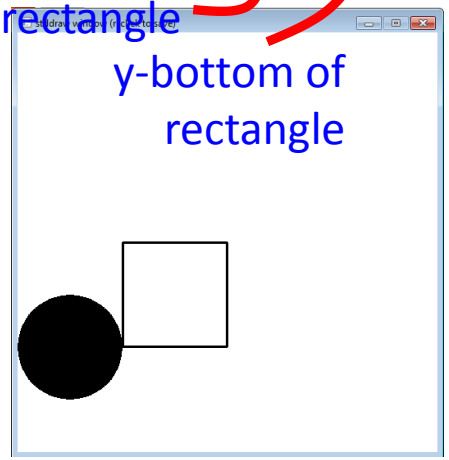
y-bottom of rectangle

width

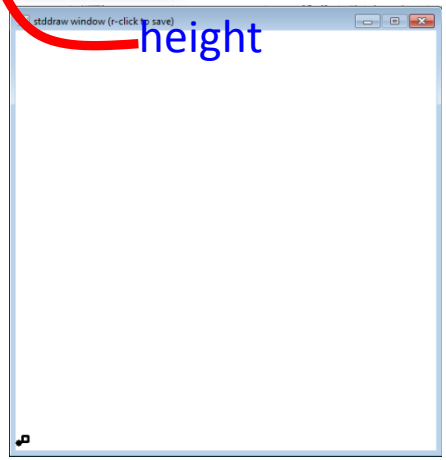
height



`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

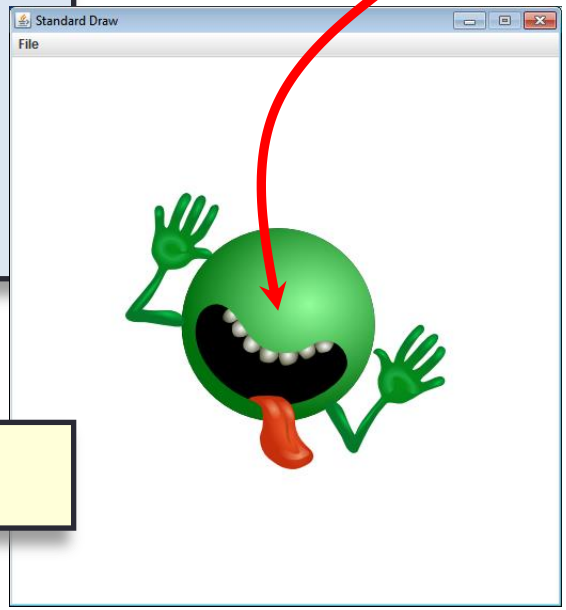
- Loading image from file
  - Supports various formats such as JPG and PNG
  - Put image files in same directory with program
  - `StdDraw.picture(picture, x, y)`

y-center

x-center

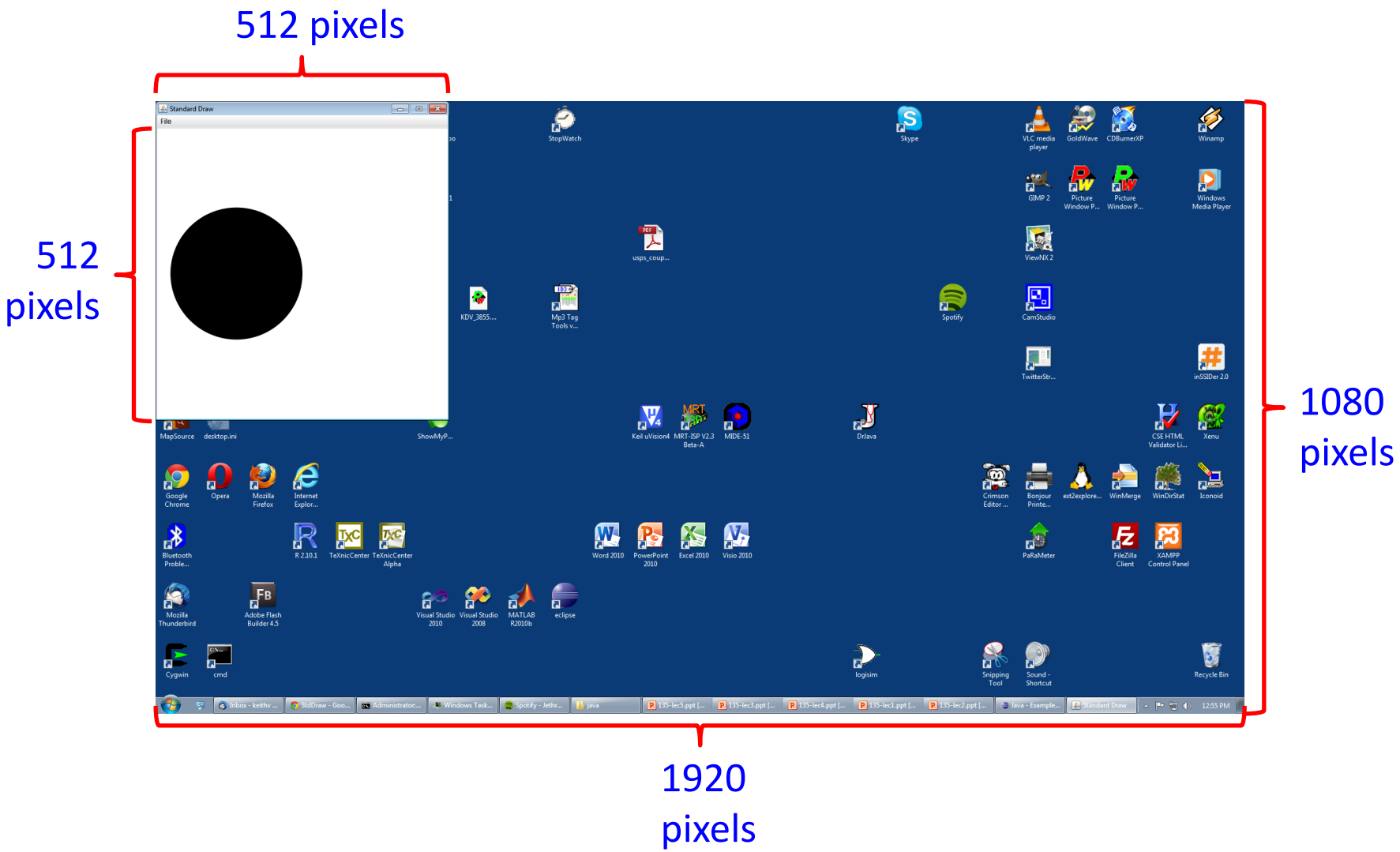
```
import StdDraw
import sys
import picture as p

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



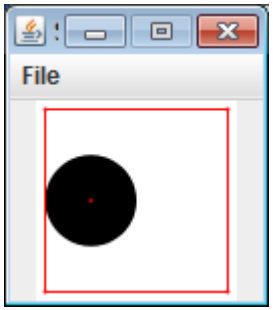

```
% python DrawImage.py dont_panic.png
```

Window Size

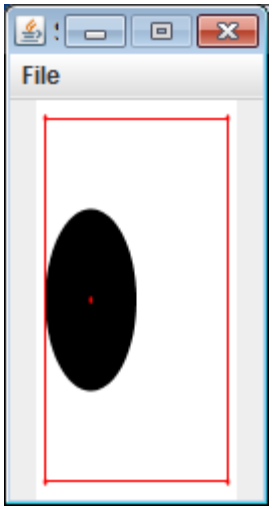

512 pixels
512 pixels
1080 pixels
1920 pixels
Standard Draw
File

Changing Window Size

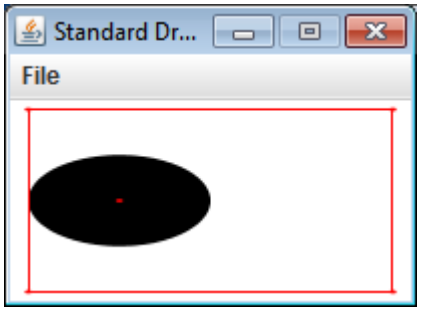
- Window size
  - Default size: 512 x 512 pixels
  - Set different size:
    - `StdDraw.setCanvasSize(width, height)`
  - Call just once at start of program



100 x 100



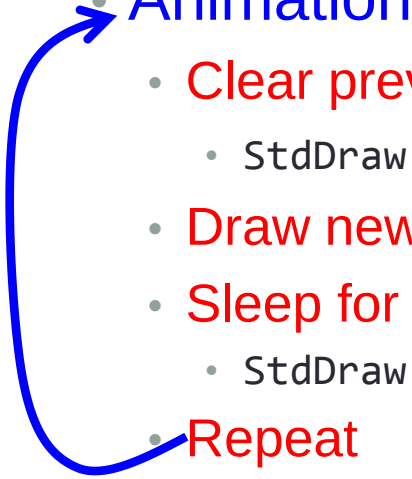
100 x 200



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)
```

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])
```

Summary

- File Input
  - Practice
    - Reading configuration files
- Graphics
  - Practice
    - Background Images
    - Animation