Scripting Languages

Python pygame Library

- overview
- program structure
- using documentation
- colors
- drawing figures
- exercise

Pygame library

- www.pygame.org
  - http://www.pygame.org/download.shtml
- License: Lesser GPL (LGPL)
  - permits use of the library in free software and proprietary programs
  - closed source and commercial games are OK
- Dependence
  - SDL (Simple DirectMedia Layer: http://libsdl.org/)
    - cross-platform development library to provide low level access to audio, keyboard, mouse, joystick, and graphics hardware via OpenGL and Direct3D
    - officially supports Windows, Mac OS X, Linux, iOS, and Android
    - zlib license: allows you to use SDL freely in any software for any purpose

Pygame Modules Overview

cdrom	manage cdrom devices and audio playback
cursors	load cursor images, includes standard cursors
display	control the display window or screen
draw	draw simple shapes onto a Surface
event	manage events and the event queue
font	create and render Truetype fonts
image	save and load images
joystick	manage joystick devices
key	manage the keyboard
mouse	manage the mouse
movie	playback of mpeg movies
sndarray	manipulate sounds with numpy
surfarray	manipulate images with numpy
time	control timing
transform	scale, rotate, and flip images

Pygame examples

http://www.pygame.org/docs/ref/examples.html

- Run the pygame examples:
  - from python3 interpreter:

```
>>> import pygame.examples.aliens
>>> pygame.examples.aliens.main()
```

  - from terminal session:

```
$ python3 -m pygame.examples.aliens
```

- Locate the example source-file: stars.py

```
>>> import pygame.examples.stars
>>> pygame.examples.stars.__file__
'/usr/local/lib/python3.5/dist-packages/pygame/examples/stars.py'
```

CLI vs GUI

- CLI (Command Line Interface) programs
  - explicit sequence of instructions
  - communication with print/input (blocking)
  - linear-oriented program structure
- GUI (Graphical User Interface) programs
  - background/real-time instructions execution
  - communication with events (non-blocking)
  - loop-oriented program structure

Pygame program structure

- Pre-game code
  - Initialize game state (constants, data structures, etc.)
- Initialize pygame GUI window
- Enter main game loop (infinite):
  - handle events
    - pygame.QUIT → close the GUI, exit the loop (or program)
  - update game state & redraw graphics
    - recalculate objects positions and appearance
  - update graphics
    - optionally → keep track of FPS
- Post-game code (optional):

Program structure

```
import pygame
import sys

size = (500, 300)

pygame.init()

pygame.display.set_mode(size)

while True:
    for event in pygame.event.get():
        if event.type in (pygame.QUIT,
                          pygame.KEYDOWN):
            pygame.quit()
            sys.exit()

    your code

    pygame.display.flip()
```

load pygame package

initialize pygame

create pygame window

enter main program loop

read all events

handle event: 'close window'

refresh window content (empty)

Using documentation

http://www.pygame.org/docs/ → References

```
import pygame
import sys


size = (500, 300)


pygame.init()


pygame.display.set_mode(size)


while True:
    for event in pygame.event.get():
        if event.type in (pygame.QUIT,
                          pygame.KEYDOWN):
            pygame.quit()
            sys.exit()

    your code

    pygame.display.flip()
```

http://www.pygame.org/docs/ref/pygame.html

http://www.pygame.org/docs/ref/display.html

http://www.pygame.org/docs/ref/event.html

Main window

```
import pygame
import os

#os.environ['SDL_VIDEO_WINDOW_POS']="100,100"
os.environ['SDL_VIDEO_WINDOW_POS']="center"

size = (500, 300)
pygame.init()

pygame.display.set_mode(size)
pygame.display.set_caption('Hello World!')

#pygame.display.set_mode(size,pygame.FULLSCREEN)
#pygame.display.set_mode(size,pygame.NOFRAME)

while True:
    for event in pygame.event.get():
        if event.type in (pygame.QUIT,
                          pygame.KEYDOWN,
                          pygame.MOUSEBUTTONDOWN):
            pygame.quit()
            sys.exit()
    pygame.display.flip()
```

Colors

- Tuple (R,G,B) or (R,G,B, A)
  - black = pygame.color.Color(0, 0, 0)
  - custom = pygame.color.Color(0x34F05B)
- HTML-named
  - white = pygame.color.Color("white")
    - http://www.w3schools.com/colors/colors_names.asp

```
>>> pygame.color.Color("black")
(0, 0, 0, 255)
>>> pygame.color.Color("gray")
(190, 190, 190, 255)
>>> pygame.color.Color(0x01020304)
(1, 2, 3, 4)
```

https://www.pygame.org/docs/ref/color.html

Drawing: pygame.draw

- Create surface object (prior to the main loop)

  `s = pygame.display.set_mode(`*size*`)`

- Clean (fill) surface before drawing
  - `s.fill(`*background_color*`)`
- Draw figures on surface:

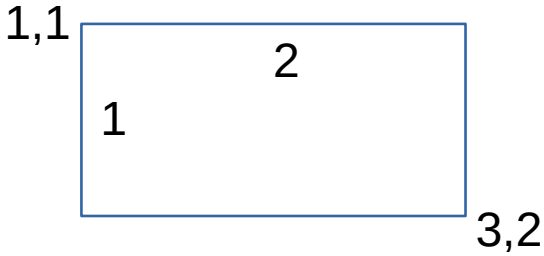
  `pygame.draw.figure(s, ...)`

- Update display
  - `pygame.display.flip()`

pygame.draw.rect
pygame.draw.polygon
pygame.draw.circle
pygame.draw.ellipse
pygame.draw.arc
pygame.draw.line
pygame.draw.lines
pygame.draw.aaline
pygame.draw.aalines

Rectangles

- `pygame.draw.rect(s, color, rectangle)`
  - http://www.pygame.org/docs/ref/draw.html
- rectangle is defined by tuple:
  - (left, top, width, height)
  - ((left, top), (width, height))
  - ((left, top), size)
  - (position, (width, height))
  - (position, size)

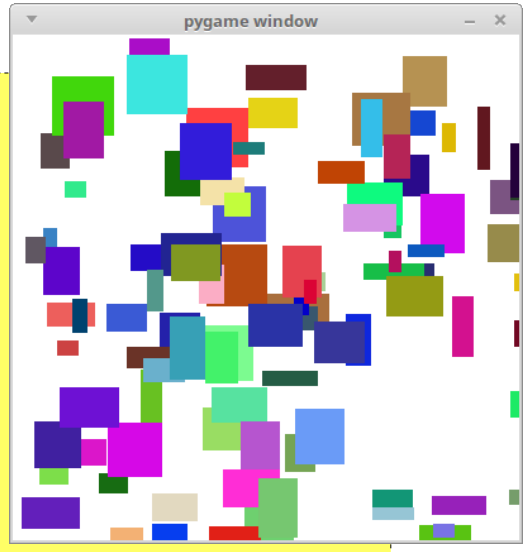



```
(1, 1, 3 ,2)
((1, 1), (2, 1))
```

```
orig = (1, 1)
dims = (2, 1)
```

```
((1, 1), dims)
(orig, (2, 1)
(orig, dims)
```

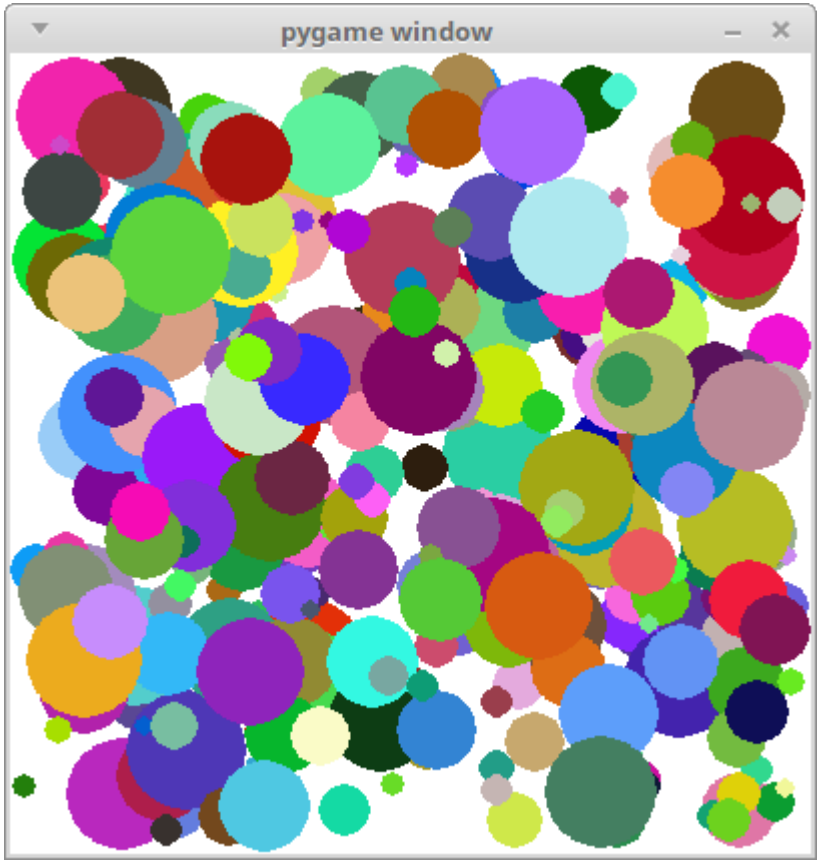
Example → random rectangles




```
import pygame, random, sys

win = (400, 400)
white = pygame.Color("white")
pygame.init()
s = pygame.display.set_mode(win)
s.fill(white)

for i in range(100):
    rgb = random.randint(0,0xFFFFFFFF)
    color = pygame.color.Color(rgb)

    size = (random.randint(10,50), random.randint(10,50))
    orig = (random.randint(0,win[0]), random.randint(0,win[1]))

    rect = (orig, size)
    pygame.draw.rect(s, color, rect)
    pygame.display.flip()

while True:
    for event in pygame.event.get():
        if event.type in (pygame.QUIT, pygame.MOUSEBUTTONDOWN):
            pygame.quit()
            sys.exit()
```

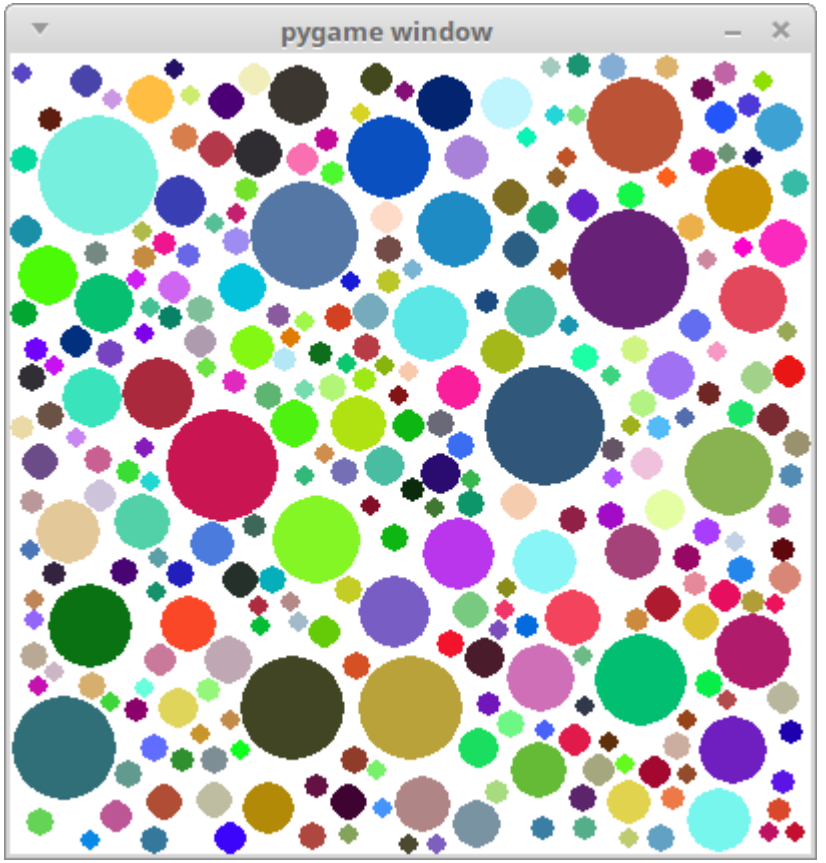
Exercise → random circles

- All circles within window



Exercise → random circles

- All circles within window
- No overlapping



Exercise → checkerboard n×m

```
import pygame
import sys

#define your variables
...

#draw your board in a function
def draw_board():
    ...

pygame.init()
s = pygame.display.set_mode(board)
draw_board()
pygame.display.flip()

while True: # main game loop
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()
```