Outline

1. Development Platforms for Multimedia Programming
   - 1.1. Classification of Development Platforms
   - 1.2. A Quick Tour of Various Development Platforms
2. Multimedia Programming with Python and Pygame
   - 2.1. Introduction to Python
   - 2.2. Pygame: A Multimedia/Game Framework for Python
3. Multimedia Programming with Java FX
4. Multimedia Programming with JavaScript and CreateJS
5. History of Multimedia Programming
6. Programming with Images
7. Programming with Vector Graphics and Animations
8. Programming with Sound
9. Programming with Video
10. Software Engineering Techniques for Multimedia Programs
    - 10.1. Design Patterns
    - 10.2. Multimedia Modeling Languages

2 Multimedia Programming with Python and SDL

2.1 Introduction to Python

2.2 SDL/Pygame:
Multimedia/Game Frameworks for Python

Literature:

www.pygame.org
www.libsdl.org

History of SDL & Pygame

- Sam Lantinga, 1998:
  Simple DirectMedia Layer (SDL) framework
  - To simplify porting games among platforms
  - Common and simple way to create displays and to process input, abstracting away from platform particularities; written in C
  - Basis for hundreds of games, among them *Angry Birds*, *Unreal Tournament*
  - *Current version: 2.0.3 (March 2014)*
- ***Pygame*** is a *language binding* for SDL *Version 1.2* to Python
  - Use the SDL library from Python code
- Pygame and SDL are open source projects
  - Version 1.9.1 (August 2009) is latest version of Pygame, using SDL 1.2.13
- Documentation :
  - www.pygame.org/docs
- Possible successor project, under development:
  - **PySDL2** (Python binding for SDL 2), see http://pysdl2.readthedocs.org

Modules in the Pygame Package

`pygame.cdrom`	Controls CD drives
`pygame.cursors`	Loads cursor images
`pygame.display`	Accesses display
`pygame.draw`	2D vector graphics
`pygame.event`	External events
`pygame.font`	Uses System fonts
`pygame.image`	Loads and saves an image
`pygame.joystick`	Special input
`pygame.key`	Keyboard input
`pygame.mixer`	Loads and plays sounds
`pygame.mouse`	Manages mouse
`pygame.movie`	Plays movie files

`pygame.music`	Works with music and streaming audio
`pygame.overlay`	Advanced video overlays
`pygame`	High level functions
`pygame.rect`	Manages areas
`pygame.sndarray`	Manipulates sound data
`pygame.sprite`	Manages moving images
`pygame.surface`	Manages images and the screen
`pygame.surfarray`	Manipulates image pixel data
`pygame.time`	Manages timing and frame rate
`pygame.transform`	Resizes and moves images

Slide Show Example (1)

```
import pygame
from pygame.locals import *
from sys import exit

background = pygame.Color(255, 228, 95, 0)
sc_w = 356
sc_h = 356

pygame.init()

# Create program display area
screen = pygame.display.set_mode((sc_w, sc_h), 0, 32)
pygame.display.set_caption("Simple Slide Show")

# Set background color
screen.fill(background)

# Load slide and show it on the screen
slide = pygame.image.load('pics/tiger.jpg').convert()
screen.blit(slide, (50, 50))
pygame.display.update()
...
```

Flags

Copy image to screen
(bit block image transfer)

Display Setup

`pygame.display.set_mode(`*`rect, flags, depth`*`)`

- **`Rect:`** Size of the display window (pixels)
- **`Flags:`** Properties of the display which can be switched on/off
  - **`FULLSCREEN`**
  - **`DOUBLEBUF`** Double buffering
  - **`HWSURFACE`** Hardware-accelerated display (must be full screen)
  - **`OPENGL`** OpenGL rendering
  - **`RESIZABLE`**
  - **`NOFRAME`**
- **`Depth:`** Bit depth of display
  - 8: 256 colors
  - 15: 32,768 colors
  - 16: 65,536 colors
  - 24: 16,7 million colors
  - 32 (eight spare bits): 16,7 million colors

Slide Show Example (2)

```
...
pygame.time.wait(4000)

# Load slide and show it on the screen
slide = pygame.image.load('pics/butterfly.jpg').convert()
screen.blit(slide, (50, 50))
pygame.display.update()
pygame.time.wait(4000)
...
# Event loop
while True:
    for event in pygame.event.get():
        if event.type == QUIT:
            exit()
```

Event Loop

- Why ask for events?
  - Program termination (clocking close icon of window) is QUIT event
- Typical standard structure: Infinite loop:
  - Asking for new events
  - Breaking the loop if termination event found
  - “Active waiting” is not optimal, better wait for events in background
- Next version:
  - Uses generic code to deal with all slides
  - Switches interactively between slides (arrow keys)

QUIZ

- How can we achieve that the slideshow application can be terminated by the user before running to its end?

Interactive Slide Show – Keyboard Control

```
slides = []
slides.append(pygame.image.load('pics/tiger.jpg').convert())
...
slides.append(pygame.image.load('pics/butterfly.jpg').convert())

slideindex = 0

while True:
    for event in pygame.event.get():
        if event.type == QUIT:
            exit()
        if event.type == pygame.KEYDOWN:
            if event.key in [K_SPACE, K_RIGHT]:
                if slideindex+1 < len(slides):
                    slideindex += 1
            if event.key == K_LEFT:
                if slideindex > 0:
                    slideindex -= 1
            if event.key == K_q:
                exit()
    screen.blit(slides[slideindex], (50, 50))
    pygame.display.update()
```

List

Key pressed

Individual keys

Pygame vs. PySDL2: What's in a Name?

Pygame

```
background =
pygame.Color(255, 228, 95, 0)

pygame.image.load(name)
.convert()

pygame.event.get()

event.key == K_LEFT


screen.blit(image,(50, 50))



pygame.display.update()
```

PySDL2

```
background = sdl2.ext.Color
(0, 255, 228, 95)

sdl2.ext.load_image(name)


sdl2.ext.get_events()

event.key.keysym.sym ==
SDLK_LEFT

sdl2.surface.SDL_BlitSurface
(image, None, windowsurface,
SDL_Rect(50, 50))

window.refresh()
```

Slide Show in PySDL2 (1)

```
import sdl2.ext
from sdl2 import *

RESOURCES = sdl2.ext.Resources(__file__, "pics")
sc_w = 356
sc_h = 356
background = sdl2.ext.Color(0, 255, 228, 95)

sdl2.ext.init()

# Create program display area
window = sdl2.ext.Window("Slide Show - Use Arrow Keys",
size=(sc_w, sc_h))
window.show()
windowsurface = window.get_surface()

# Set background color
sdl2.surface.SDL_FillRect(windowsurface, None, background)
```

...

Slide Show in PySDL2 (2)

...

```
# Preload slide files
slides = []
slides.append(sdl2.ext.load_image(RESOURCES.get_path("frog.jpg")))
slides.append(sdl2.ext.load_image(RESOURCES.get_path("cows.jpg")))
slides.append(sdl2.ext.load_image(RESOURCES.get_path("elephant.jpg"
slides.append(sdl2.ext.load_image(RESOURCES.get_path("tiger.jpg")))

# Event loop
slideindex = 0
running = True
while running:
    for event in sdl2.ext.get_events():
        if event.type == sdl2.SDL_QUIT:
            running = False
            break
```

...

Slide Show in PySDL2 (3)

```
...
        if event.type == sdl2.SDL_KEYDOWN:
            if event.key.keysym.sym in [SDLK_SPACE, SDLK_RIGHT]:
                if slideindex+1 < len(slides):
                    slideindex += 1
            if event.key.keysym.sym == SDLK_LEFT:
                if slideindex > 0:
                    slideindex -= 1
            if event.key.keysym.sym == SDLK_q:
                running = False
                break
    sdl2.surface.SDL_BlitSurface(slides[slideindex], None,
       windowsurface, SDL_Rect(50, 50))
    window.refresh()

sdl2.ext.quit()
```

Lesson Learnt

It is not important
to learn the details of a certain platform!

It is important
to learn how to get familiar with any new platform quickly!