Multimedia-Programmierung
Übung 2

Ludwig-Maximilians-Universität München
Sommersemester 2016
Today

... oldie but goldie
• Sam Lantinga, 1998: Simple DirectMedia Layer (SDL) framework, to simplify porting games among platforms
  – Common and simple way to create displays and process input abstracting from platform particularities
  – Originally written in C

• Pygame is a language binding for SDL to the Python language

Download: http://pygame.org/docs/ref/index.html

Literature: W. McGugan, Beginning Game Development with Python and Pygame, Apress 2007
Pygame Modules

**pygame.cdrom** Accesses and controls CD drives  
**pygame.cursors** Loads cursor images  
**pygame.display** Accesses the display  
**pygame.draw** Draws shapes, lines, and points  
**pygame.event** Manages external events  
**pygame.font** Uses system fonts  
**pygame.image** Loads and saves an image  
**pygame.joystick** Uses joysticks and similar devices  
**pygame.key** Reads key presses from the keyboard  
**pygame.mixer** Loads and plays sounds  
**pygame.mouse** Manages the mouse  
**pygame.movie** Plays movie files  

**pygame.music** Works with music and streaming audio  
**pygame.overlay** Accesses advanced video overlays  
**pygame** Contains high-level Pygame functions  
**pygame.rect** Manages rectangular 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
Pygame Modules
Testing if Modules are available on a Platform

Test:

```python
if pygame.font is None:
    print "no font module"
```

Some modules might not be available on a platform depending on the hardware settings. In this case Pygame sets them to `None`. 
import pygame
from pygame.locals import *
from sys import exit

player_image = 'cursor.gif'

pygame.init()
screen = pygame.display.set_mode((640, 480), 0, 32)
pygame.display.set_caption("Hello, Pygame!")

mouse_cursor = pygame.image.load(player_image).convert_alpha()

while True:
    for event in pygame.event.get():
        if event.type == QUIT:
            exit()

    screen.fill((255,255,255))
    x, y = pygame.mouse.get_pos()
    x-= mouse_cursor.get_width() / 2
    y-= mouse_cursor.get_height() / 2
    screen.blit(mouse_cursor, (x, y))
    pygame.display.update()
Events

- Module `pygame.event`
- Generated all the time by different entities
- Stored in an event queue
- `pygame.event.wait()` waits until the list is not empty
- `pygame.event.get()` returns a list of the last events
- `pygame.event.poll()` returns the next event of the queue
- The type of the event is specified by `event.type`

Print all events in the list:

```python
for event in pygame.event.get():
    print event.type
```
Events

Parameters

• Events can have parameters

• Examples:
  – QUIT: no parameters
  – MOUSEBUTTONDOWN: pos, button
  – VIDEORESIZE: size, w, h
  – Etc.

Left click with the mouse:

```python
if event.type == MOUSEBUTTONDOWN:
    if event.button == 1:
        print event
```

Output:

```python
<Event(5-MouseButtonDown {'button': 1, 'pos': (231, 207)})>
```
Events

Mouse Events

- MOUSEMOTION: pos, rel, buttons
  - Example print event:

```
<Event(4-MouseMotion {'buttons': (1, 0, 0), 'pos': (660, 313), 'rel': (-4, -4)})>
```

- MOUSEBUTTONDOWN: pos, button
- MOUSEBUTTONUP: pos, button

- Example: check whether the left mouse button is pushed during mouse movement

```python
if event.type == MOUSEMOTION:
    if event.buttons[0] == 1:
        pass  # or do something
```
Events

Keyboard Events

- **KEYDOWN**: unicode, key, mod
- **KEYUP**: key, mod
  - `key` is the number of the key that has been pressed
  - `mod` represents combination keys like alt, ctrl and shift
  - `unicode` is the unicode value of the pressed key

- Example: check whether the left key has been pressed

```python
if event.type == KEYDOWN:
    if event.key == K_LEFT:
        pass  # or do something
```
Events
(un)blocking events

• `pygame.event.set_blocked(events)` blocks events from the event queue

• `pygame.event.set_allowed(events)` unblocks the events

• Example: block all keyboard events

```python
pygame.event.set_blocked([KEYDOWN,KEYUP])
```
Events

custom events

- `pygame.event.post(event)` posts a user event
- The value for events created by the user must have the value of `USEREVENT` or higher

- Example:

```
MMPROCKS = USEREVENT+1
new_event = pygame.event.Event(MMPROCKS, message="MMP Rocks")
pygame.event.post(new_event)
```
Fonts

- `pygame.font.SysFont(font, size)` loads a system font
- `pygame.font.Font(font, size)` loads a font from a file
- `Font.render(text, aliasing, color, bg_color)` creates a surface of a text

- Example:

```python
    test_font = pygame.font.SysFont("arial", 16)
    test_surface = test_font.render("test", True, (0,0,0))
    screen.blit(test_surface, (0,0))
```
Images

- Pygame can load different image types:
  - JPG
  - PNG
  - GIF (non animated)
  - BMP
  - PCX
  - TGA (uncompressed)
  - TIF
  - LBM (and PBM)
  - PBM (and PGM, PPM)
  - XPM

- Images are loaded by `pygame.image.load(image)` (returns a `Surface` object)
Images

• Saving is limited to:
  – BMP
  – JPEG
  – PNG
  – TGA

• Images are saved by `pygame.image.save(surface, file)`
Surfaces

Creating a Surface

- Surface objects are containers for images
- Used as canvases
- Even the Pygame screen is represented as a Surface

- Several functions return a Surface object (e.g. `pygame.image.load(image)`) 
- Blank surfaces can be created by calling the constructor `pygame.Surface((100,100))`
Surface 2 Image

- Any surface can directly be stored as an image
- `pygame.image.save(surface, name)`

Example:

```python
pygame.image.save(screen, "name.png")
```

“cooooool”
Surfaces

Converts

- Converts are used to convert surfaces to an efficient format
- Use `convert()` or `convert_alpha()` if the image contains transparency

Example:

```python
mouse_cursor = pygame.image.load(player_image).convert_alpha()
```
Surfaces
Clipping

• If clipping is set, only pixels in that area will be displayed
• `set_clip(Rect)`
• `set_clip()` resets the clipping area

Example:
```
screen.set_clip(100,100,200,200)
```
Surfaces
Filling and Setting and Getting Pixels

- `fill(color)` fills the surface with the defined color
- `set_at(pos,color)` can be used to manipulate single pixels
- `get_at(pos)` returns the pixel color of a surface

Set pixel 10,10 to black:

```python
screen.set_at((10,10),(0,0,0))
```
Surfaces

Blitting

- `blit(source, pos, sourcerect=None)` copies pixel data from one surface to another

Copy test_surface to 0,0:

```python
mouse_cursor = pygame.image.load("cursor.gif").convert_alpha()
screen.blit(mouse_cursor, (0, 0))
```
Drawing

- `pygame.draw.rect(surface, color, rect, width=0)` draws a rectangle to a surface
- `pygame.draw.polygon(surface, color, pointlist, width=0)` draws a polygon to a surface
- `pygame.draw.circle(surface, color, pos, radius, width=0)` draws a circle to a surface
- `pygame.draw.arc`, `pygame.draw.ellipse`, `pygame.draw.line` etc.

Draw an empty circle:

```python
pygame.draw.circle(screen, (0,0,0), (100,100), 100, 1)
```
Super Quick Wallpong Game
import pygame
from pygame.locals import *
from sys import exit
import random

pygame.init()
screen=pygame.display.set_mode((640,480),0,32)
pygame.display.set_caption("Pong Pong!")

#Creating 2 bars, a ball and background.
back = pygame.Surface((640,480))
background = back.convert()
background.fill((0,0,0))
bar = pygame.Surface((10,50))
bar1 = bar.convert()
bar1.fill((0,0,255))
wall = pygame.Surface((11,471))
wall = wall.convert()
wall.fill((0,0,255))
circ_sur = pygame.Surface((15,15))
circ = pygame.draw.circle(circ_sur,(0,255,0),(7,7),7)
circle = circ_sur.convert()
circle.set_colorkey((0,0,0))

# some dimensions and start positions
bar1_x, bar1_y = 10., 215.
Super Quick n Dirty Wallpong Game (2)

```python
circle_x, circle_y = 307.5, 232.5
bar1_move = 0.
speed_x, speed_y, speed_circ = 250., 250., 250.
clock = pygame.time.Clock()

while True:
    for event in pygame.event.get():
        if event.type == QUIT:
            exit()
        if event.type == KEYDOWN:
            if event.key == K_UP:
                bar1_move = -ai_speed
            elif event.key == K_DOWN:
                bar1_move = ai_speed
        elif event.type == KEYUP:
            if event.key == K_UP:
                bar1_move = 0.
            elif event.key == K_DOWN:
                bar1_move = 0.

    screen.blit(background,(0,0))
    frame = pygame.draw.rect(screen,(255,255,255),Rect((5,5),(630,470)),2)
    screen.blit(bar1,(bar1_x,bar1_y))
    screen.blit(wall,(625,5))
    screen.blit(circle,(circle_x,circle_y))
    bar1_y += bar1_move
....

bar1_y = 420
```

Game Loop
Make it Move!
# movement of ball
    time_passed = clock.tick(30)
    time_sec = time_passed / 1000.0
    circle_x += speed_x * time_sec
    circle_y += speed_y * time_sec
    ai_speed = speed_circ * time_sec

# since we don't know anything about collision, ball hitting bars goes like this.
    if circle_x <= bar1_x + 10.:
        if circle_y >= bar1_y - 7.5 and circle_y <= bar1_y + 42.5:
            circle_x = 20.
            speed_x = -speed_x
    if circle_x < 5.:
        circle_x, circle_y = 320., 232.5
        speed_x = -speed_x
        bar1_y = 215.
    elif circle_x > 608.:
        speed_x = -speed_x
        circle_x = 608.
    if circle_y <= 10.:
        speed_y = -speed_y
        circle_y = 10.
    elif circle_y >= 457.5:
        speed_y = -speed_y
        circle_y = 457.5

pygame.display.update()
Do it yourself!

http://inventwithpython.com/blog/2010/09/01/the-top-10-pygame-tutorials/
Useful Links

• Pygame API !!!!

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