

Multimedia-Programmierung

Übung 2

Ludwig-Maximilians-Universität München
Sommersemester 2015

Today





- 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

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

Where is the Pygame API?

- <http://pygame.org/docs/ref/index.html>



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:

```
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()
```

← import locals, mainly constants (e.g. QUIT)

← initializes ALL python modules (loads drivers etc.)

← initialize a display (Surface object)

← only necessary if alpha channel exists

← used to update the display

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:

```
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:

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

Output:

```
<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

```
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

```
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

```
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:

```
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:

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

A photograph of a man with glasses and a beard, wearing a black t-shirt with the word "Google" on it. He is holding a glass of beer. A speech bubble next to him contains the text "cooooool".

“cooooool”

Surfaces

Converts

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

Example:

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

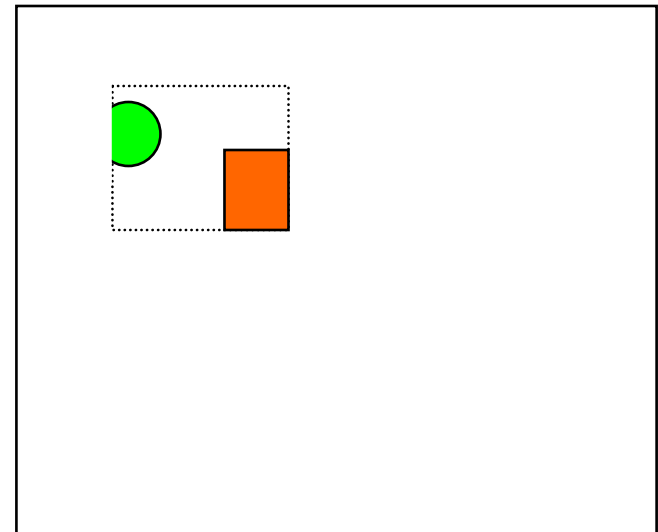
Surfaces

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)
```



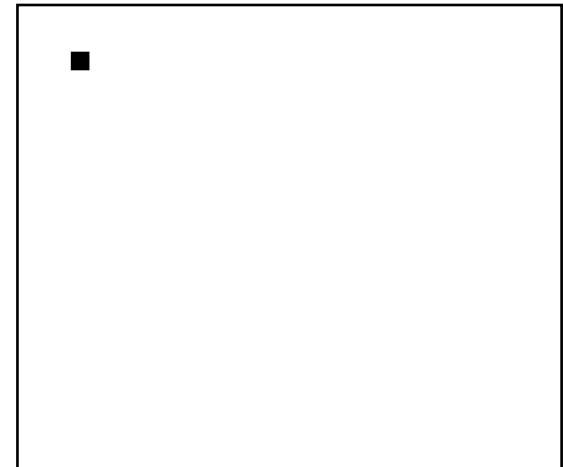
Surfaces

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:

```
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:

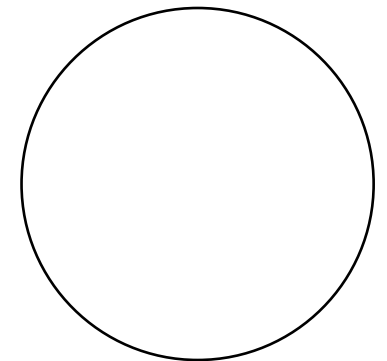
```
mouse_cursor = pygame.image.load("cursor.gif").convert_alpha()  
screen.blit(mouse_cursor, (0, 0))
```

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:

```
pygame.draw.circle(screen,(0,0,0),(100,100),100,1)
```





Do it yourself!

Useful Links

- Pygame API !!!!

<http://pygame.org/docs/>

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