Multimedia-Programmierung
Übung 5

Ludwig-Maximilians-Universität München
Sommersemester 2009
Today

• Sprite animations in Pygame
• Advanced collision detection
Keyframe Animations

• Keyframes are defined
• Intermediate steps are interpolated
• Basic interpolators/tweens/... built into many programming environments (e.g. Flash, JavaFX)
• Examples: motion, color, shape
Keyframe Animations

Keyframe Animations in Pygame

- Pygame has no built-in interpolators
- Logic has to be added by the programmer
- Question: How can we calculate the intermediate points?
Horizontal Animation (old slides)

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

player_image = 'head.jpg'
pygame.init()

screen = pygame.display.set_mode((640, 280), 0, 32)
pygame.display.set_caption("Animate X!")
mouse_cursor = pygame.image.load(player_image).convert_alpha()

x = 0 - mouse_cursor.get_width()
y = 10

while True:
    for event in pygame.event.get():
        if event.type == QUIT:
            exit()
        screen.fill((255,255,255))
        if x > screen.get_width():
            x = 0 - mouse_cursor.get_width()
        screen.blit(mouse_cursor, (x, y))
        x+=10
        pygame.display.update()
```

Result:
Sprite Animations

• Animations consist of frames that are displayed one after another
First Sprite Animation 1

```python
class Box(pygame.sprite.Sprite):
    def __init__(self, color, initial_position, fps):
        pygame.sprite.Sprite.__init__(self)
        self.act_frame = 0
        self.create_images(color)
        self.rect = self.image.get_rect()
        self.rect.topleft = initial_position
        self.fps = fps
        self.change_time = 1.0/self.fps
        self.time = 0

...  # remember the current frame
# create the images for the animation
create_images(color)
rect = image.get_rect()
rect.topleft = initial_position
fps = fps
change_time = 1.0/fps
time = 0
```

- Remember the current frame.
- Create the frames (defined later).

Based on the frames per second (fps) calculate the time needed for animation changes.
First Sprite Animation 2

```python
... def create_images(self, color):
    self.frames = []
    image = pygame.Surface((20,20), pygame.SRCALPHA, 32)
    pygame.draw.rect(image, color, (0,0,10,10))
    self.frames.append(image)
    image = pygame.Surface((20,20), pygame.SRCALPHA, 32)
    pygame.draw.rect(image, color, (10,0,10,10))
    self.frames.append(image)
    image = pygame.Surface((20,20), pygame.SRCALPHA, 32)
    pygame.draw.rect(image, color, (0,10,10,10))
    self.frames.append(image)
    image = pygame.Surface((20,20), pygame.SRCALPHA, 32)
    pygame.draw.rect(image, color, (10,10,10,10))
    self.frames.append(image)
    self.image = self.frames[self.act_frame]

def update(self, time_passed):
    self.time += time_passed
    if self.time >= self.change_time:
        self.act_frame = (self.act_frame + 1) % len(self.frames)
        self.image = self.frames[self.act_frame]
        self.time = 0
```

... pygame.init()

screen = pygame.display.set_mode((640, 480), 0, 32)
box1 = Box((255, 0, 0), (0, 0), 4)  # animated box in red

clock = pygame.time.Clock()

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

    screen.fill((100, 200, 0))
    time_passed = clock.tick() / 1000.0
    box1.update(time_passed)
    screen.blit(box1.image, box1.rect)
    pygame.display.update()
Multiple Parallel Animations

... 
```
pygame.init()

screen = pygame.display.set_mode((640, 480), 0, 32)
box1 = Box((255,0,0),(0,0),4)
box2 = Box((0,100,255),(40,40),2)
clock = pygame.time.Clock()

while True:
    for event in pygame.event.get():
        if event.type == QUIT:
            exit()
    screen.fill((100, 200, 0))
    time_passed = clock.tick() / 1000.0
    box1.update(time_passed)
    screen.blit(box1.image,box1.rect)
    box2.update(time_passed)
    screen.blit(box2.image,box2.rect)
    pygame.display.update()
```
Collision Detection

Rect

- Rect provides several methods to test collisions

- `Rect.collidepoint(point)` tests whether a point is within the Rect’s area

  ![Diagram](https://via.placeholder.com/150)

- `Rect.colliderect(rect)` tests whether two Rects intersect

  ![Diagram](https://via.placeholder.com/150)
Collision Detection

Rect II

• `Rect.collidelist(list)` tests whether the Rect collides with at least one Rect in the given list
• `Rect.collidelistall(list)` tests whether the Rect collides with all Rects in the list
• `Rect.collidedict(dict)` tests whether the Rect collides with at least one Rect in the given dictionary
• `Rect.collidedictall(dict)` tests whether the Rect collides with all Rects in the dictionary
Collision Detection

Sprites

• The module sprite provides several method to test collision
  \( \text{http://www.pygame.org/docs/ref/sprite.html} \)

• \texttt{sprite.spritecollide(\ldots)} returns a list of sprites within a group
  that intersect with a given sprite

• \texttt{sprite.collide_rect(a,b)} checks whether two sprites intersect
  (must have rects)

• \texttt{sprite.collide_circle(a,b)} checks whether the radius of two
  sprites intersect. Radius attribute should be defined in the
  sprite.

\begin{center}
\begin{tabular}{c c}
\text{False} & \text{True} \\
\end{tabular}
\end{center}
Collision Detection

Sprites 2

• `sprite.groupcollide(a,b)` returns a list of sprites of two groups that intersect
• `sprite.collide_mask(a,b)` checks whether two Sprite collide on a bitmap level (non-transparent pixels overlap)

```python
if pygame.sprite.collide_mask(head1, head2):
    print "collide"
```

False

True
Collision Detection

Masks

• Masks are 1bit per pixel representations of areas that can collide

• Module mask contains functions and classes to create and use masks
  http://www.pygame.org/docs/ref/mask.html

• mask.from_surface(surface,threshold=127) creates a mask of a surface. Threshold defines the alpha value that counts as collideable

• Class Mask contains methods to work with classes

Original

Mask

collision area
Collision Detection

Conclusion

• Pygame offers various ways to check for collisions
• **Choose your collision detection algorithm wisely depending on the task**
• Pixel based collision detection is precise but slow
• Rect or radius based collision detection is fast but imprecise
Useful Links

• Pygame Sprite Tutorial
  http://kai.vm.bytemark.co.uk/~piman/writing/sprite-tutorial.shtml

• Pygame API !!!!
  http://www.pygame.org/ctypes/pygame-api/