Föreläsning 10

Spelprogrammering med bilder och ljud

Grafikpaket

- Pygame är ett grafikpaket (en samling moduler).
- LiveWires är ett extralager som gör det enklare att använda Pygame.

- Appendix A i boken beskriver LiveWires.

Vad finns i LiveWires?

- Modulerna `games` och `color`
- `games` innehåller klasserna
  - `Screen` (grafikfönstret)
  - `Sprite` (figurer som kan flytta sig i grafikfönstret)
  - `Text` (text i grafikfönstret)
  - `Message` (text som bara syns en stund)
  - `Animation` (en samling bilder som bildar en film)
  - `Mouse` (tar emot inmatning från musen)
  - `Keyboard` (tar emot tangenttryckningar)
  - `Music` (laddar och spelar musikfiler)
- `color` innehåller färgkonstanter

moving_pan.py

Ett exempel från kap 11, med en Sprite som styrs av användaren (med musrörelse):

1. Grafikfönstret öppnas
2. En bakgrundsbild laddas in
3. Figuren "Pan" (en Sprite) skapas
4. Inmatning från musen tas emot för att styra Pan

```
# Moving Pan
# Demonstrates mouse input

from livewires import games

games.init(screen_width=640,
           screen_height=480,
           fps=50)

class Pan(games.Sprite):
    """ A pan. Controlled by the mouse. """

    def update(self):
        """ Move to mouse coordinates. """
        self.x = games.mouse.x
        self.y = games.mouse.y
```

uppdateringshastighet

ärver attribut och metoder

ger muspekarens koordinater

```
def main():
    wall_image=games.load_image("wall.jpg",
                                transparent = False)
    games.screen.background = wall_image

    pan_image = games.load_image("pan.bmp")

    the_pan = Pan(image = pan_image,
                  x = games.mouse.x,
                  y = games.mouse.y)

    games.screen.add(the_pan)
    games.mouse.is_visible  = False
    games.screen.event_grab = True
    games.screen.mainloop()

main()
```

muspekaren håller sig innanför grafikfönstret

Klassen Sprite

Klassen Sprite har inbyggda attribut, t ex:

- image (bilden)
- x och y (positionen)
- dx och dy (hastighet i x- och y-led)
- right, left, top, bottom (bildkanternas position)

och inbyggda metoder, t ex:

- update (anropas automatiskt i varje steg)
- destroy (tar bort alla referenser till objektet)
- overlapping_objects (lista med överlappare)

bouncing_pizza.py

Annat exempel från kap 11, med en Sprite (pizzan) som styrs av programmet:

- dx och dy sätts till 1 när pizzan skapas
- vi definierar om pizzans update-metod så att den byter riktning om den kommer till någon av fönsterkanterna

```
class Pizza(games.Sprite):
    """ A bouncing pizza. """

    def update(self):
        """ Reverse a velocity component
            if edge of screen reached. """
        if self.right > games.screen.width
           or self.left < 0:
            self.dx = -self.dx
        if self.bottom > games.screen.height
           or self.top < 0:
            self.dy = -self.dy
```

Animationer

- En Animation är en serie bilder som kan spelas upp som en film.
- Explosionen i explosion.py i kap 12 är ett exempel.
- Först skapar man en lista med bildfiler.
- Sen ett Animation-objekt, med bildlistan, postition och visningsparametrar.

Bildlistan

```
explosion_files = ["explosion1.bmp",
                   "explosion2.bmp",
                   "explosion3.bmp",
                   "explosion4.bmp",
                   "explosion5.bmp",
                   "explosion6.bmp",
                   "explosion7.bmp",
                   "explosion8.bmp",
                   "explosion9.bmp"]
```

Animation-objekt

```
explosion = games.Animation(
    images = explosion_files,
    x = games.screen.width/2,
    y = games.screen.height/2,
    n_repeats = 0,
    repeat_interval = 5)
```

bildlistan

position

antal filmvisningar

tid mellan successiva bilder

Ljud

- Ljudfiler måste vara på formatet *.wav
- Ljudet laddas med metoden `load_sound`
- och styrs med metoderna `play` och `stop`
- Exempel:

```
missile_sound  =
 games.load_sound("missile.wav")
missile_sound.play(3)
```

- Man kan ha upp till åtta ljud igång samtidigt!

Musik

- Musik fungerar ungefär som ljud, men
- Det finns bara en musikkanal (motsvaras av objektet games.music)
- Man kan ladda musikfiler på flera olika format, t ex MP3 och MIDI
- Exempel:

```
games.music.load("theme.midi")
games.music.play(-1) #-1=evigt
```

- sound_and_music.py demonstrerar!

Interaktion

- I spel vill man att objekten ska reagera när dom kommer i kontakt med andra objekt.
- Sprite har ett attribut overlapping_sprites - en lista med de objekt som överlappar.
- I astrocrash-programmet definieras en klass Collider som ska hantera krockar.

```
class Collider(games.Sprite):
        def update(self):
        """ Check for overlapping sprites. """
        super(Collider, self).update()

        if self.overlapping_sprites:
          for sprite in self.overlapping_sprites:
                sprite.die()
          self.die()

    def die(self):
        """ Destroy self and explode. """
        new_explosion =
            Explosion(x = self.x, y = self.y)
        games.screen.add(new_explosion)
        self.destroy()
```

Programutveckling

- Alla större program i boken har utvecklats på följande sätt:
- Skriv först en algoritm för programmet.
- Skriv en superenkel förstaversion av programmet och provkör den. När den fungerar…
- Bygg på med lite till och provkör igen.
- Upprepa tills programmet är klart (astrocrash finns i åtta versioner i bokens kapitel 12).
- Detta kallas *prototypprogrammering*!

Fortsättningskurser

- Nästa obligatoriska kurs från oss är DN1214 Numeriska metoder i höst
- Valfria datakurser att fortsätta med är DD1320 Tillämpad datalogi, följt av t ex
  - DD1385 Programutvecklingsteknik
  - DD1334 Databasteknik
- Välkomna!

Kursenkät

- Kursenkäten finns på kursens webbsida, under rubriken "Kursanalys"
- Berätta för oss hur vi skulle kunna göra kursen bättre till nästa år!