Skolan för Datavetenskap och kommunikation

DD1311 Programmeringsteknik med PBL

Föreläsning 17 & 18

Nyheter

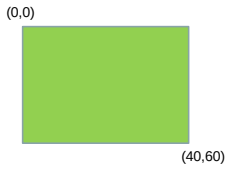
- Fler redovisningstider kommer upp på webben (i första hand på tisdagarna).
- Labbar och gruppmöten som vanligt denna vecka, men föreläsningen i eftermiddag är i sal F2.
- VM i programmering på KTHB tisdag 21 april kl 9-15. Följ tävlingen från Kåren eller på Axess-TV som direktsänder alltihop!

Dagens föreläsningar

- Mer om GUI med Tkinter
  - Rita former
  - Visa bilder
  - Lambda-uttryck
  - Funktionsanrop med parameter
  - GUI med objektorientering - exempel
- Spelprogrammering med LiveWires
  - Styra en Sprite med musen
  - Animationer
  - Ljud
  - Musik

Rita i Tkinter

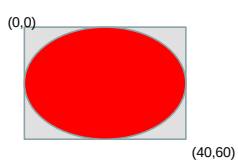
- Man kan rita geometriska former, t ex ovaler (och cirklar), rektanglar (och fyrkanter), mm
- Koordinaterna för figuren börjar i övre vänstra hörnet:



Exempel

- Så här kan man rita en oval:

```
create_oval(0,0,40,60,fill="red")
```



I programmet **ritaOval.pyw** finns ett fullständigt exempel!

Bilder i Tkinter

- I Tkinter finns klassen PhotoImage som representerar en bild.
- Bilden blir ett objekt som kan skickas med som parameter till komponenter:

```
foto = PhotoImage(file="herrK.gif")
label = Label(roten, image = foto)
```

- Hela programmet finns i filen **foto.pyw**

Spara en referens till bilden!

- Om referensen till bilden försvinner (om den t ex är en lokal variabel) så blir bilden osynlig! Därför bör man alltid spara en explicit referens till bilden:

```
def bildknapp(roten, filnamn):
  bild = PhotoImage(file = filnamn):
  knapp = Button(roten, image=bild)
  knapp.image = bild
  return knapp
```

Hela programmet finns i filen **bokknappar.pyw**

Koppla ihop funktion med knapp

- Alla komponenter har attributet `command`.
- Där anger man vilken metod/funktion som ska anropas när komponenten används.
- Om vi skriver en funktion byttext() som ska anropas när nån trycker på knapp så kan vi koppla ihop funktion med knapp så här:

```
knapp["command"] = byttext
```

Anropa funktion med knapptryck!

```
from Tkinter import *

def byttext():
    knapp["text"] = "Aj!"

roten = Tk()
knapp = Button(roten,
               text = "Tryck inte",
               command = byttext)
knapp.pack()
roten.mainloop()
```

Parametrar då?

- I bägge varianterna ovan skickar man bara med *namnet* på funktionen.
- Hur ska man göra om funktionen har parametrar?
- Lösning: Använd *lambda-uttryck*!

Lambda-uttryck…

- Ett lambda-uttryck är ett alternativt sätt att skapa en funktion.
- Istället för att definiera funktionen på vanligt sätt

```
def dubbla(x):
   return 2*x
```

- Skriver man

```
dubbla = lambda(x) : 2*x
```

…som parameter till Button

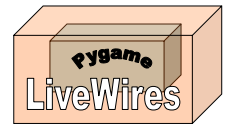
```
from Tkinter import *

def byttext(t):
    knapp["text"] = t

roten = Tk()
knapp = Button(roten,
    text="Tryck inte",
    command=(lambda: byttext("Aj!")))
knapp.pack()
roten.mainloop()
```

GUI med objektorientering

- När man skriver större program gör man klasser av sitt GUI, så blir det enklare att strukturera och återanvända koden
- De grafiska komponenterna får vara attribut.
- Hantering av komponenter kan delas upp i flera metoder.
- Två exempel: **mad_lib.py** (ur Dawson kap 10) och **PyShell.py** (en av filerna i IDLE).

Spelprogrammering

- Pygame är en samling moduler för spelprogrammering .
- Se exempel i Dawsons bok: moving_pan (kap 11), astrocrash (kap 12), och på webben: PyDance (icculus.org/pyddr/), FretsOnFire (fretsonfire.sourceforge.net/)



- LiveWires är ett extralager som gör det enklare att använda Pygame.

Vad finns i LiveWires?

- Modulerna `games` och `color` (färgkonstanter)
- `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)

moving_pan.py

Ett exempel från Dawsons kap 11, där

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

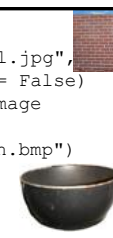
```
# Moving Pan
# Demonstrates mouse input

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

Animationer

- En Animation är en serie bilder som kan spelas upp som en film.
- Explosionen i explosion.py i Dawson 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!

Fortsättningskurser

- Nästa obligatoriska kurs från oss är DD1214 Numeriska metoder i höst
- Valfria datakurser att fortsätta med är DD1320 Tillämpad datalogi, följt av t ex
  - DD2385 Programutvecklingsteknik
  - DD2334 Databasteknik
- Välkomna!