Skolan för Datavetenskap och kommunikation

DD1311 Programmeringsteknik med PBL

Föreläsning 17

Nyheter

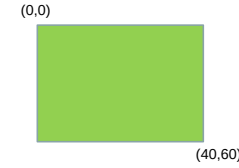
- Torsdagens grupptimme flyttad (prata med assen om ny tid).
- Sista föreläsningen på torsdag:
  - Redovisning av P-uppgiften
  - Fortsättningskurser
  - Kursenkät
  - Animationer, musik mm i Pygame/Livewires

Dagens föreläsning

- Mer om GUI med Tkinter
  - Rita former
  - Visa bilder
  - GUI med objektorientering - exempel
- Spelprogrammering med LiveWires
  - Klasser
  - Styra en Sprite med musen

Rita i Tkinter

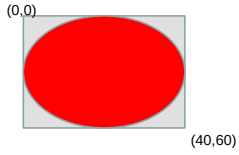
- Man kan rita geometriska former, t ex ovaler (och cirklar), rektanglar (och fyrkanter), mm
- Koordinaterna för figuren börjar i övre vänstra hörnet:



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**

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.

```
class Application(Frame):

    def __init__(self, master):
        Frame.__init__(self, master)
        self.grid()
        self.create_widgets()

    def create_widgets(self):
        """Här skapas alla
        komponenter som attribut i
        Application-klassen"""
        self.person_ent=Entry(self)
        …
```

ur mad_lib.py (Dawson kap 10)

fsguix3.pyw

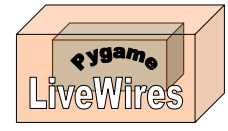
- Programmet letar efter ett danspass på Friskis & Svettis
- Komponenterna är attribut i klassen FS
- Vi vill ge alla komponenter liknande utseende
- I exemplet nedan ändras font och bakgrundsfärg av metoden `pynta`

```
class FS(Frame):

    def __init__(self, master):
        Frame.__init__(self, master)
        self.master.title("Hitta pass")
        self.pack()
        self.favoritlokaler = ["City"]
        self.egenfont = ("Courier", 10)
        self.egenfarg = "#d9d9ff"
        self.skapaLokalval()
        self.skapaPassletarknapp()
        self.skapaResultatruta()

    def pynta(self,komponent):
        komponent["font"] = self.egenfont
        komponent["bg"] = self.egenfarg
```

ur fsguix3.pyw

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