◆ PGZ - Sprettball

Skrevet av: Ole Kristian Pedersen, Kodeklubben Trondheim

Oversatt av: Stein Olav Romslo

Kurs: Python
Tema: Tekstbasert, Spill
Fag: Matematikk, Programmering
Klassetrinn: 5.-7. klasse, 8.-10. klasse

Introduksjon

I denne oppgåva skal du lage ein ballanimasjon. Du skal bruke det du har lært i oppgåva om enkle objekt (../enkle_objekter/enkle_objekter_nn.html), og Pygame og Pygame Zero. Viss du ikkje hugsar noko om objekt kan du gå tilbake til oppgåva og repetere.

Steg 0: Installere Pygame Zero

For å gjere denne oppgåva må du installere Pygame Zero (https://pygame-zero.readthedocs.io/en/latest/installation.html). Start med å sjekke at du har installert Python 3, altså at Python-versjonen din er nummerert på forma 3.X.X.

Åpne kommandolinja (engelsk: command prompt) på datamaskina di. Brukar du Windows kan du åpne start-menyen og skrive cmd (eventuelt *Ledetekst*, som er det norske namnet på programmet som skal køyre). På Mac og Linux åpnar du terminalvindauget. Skriv inn følgjande:

Windows og Mac:

```
pip install pgzero
```

Linux:

```
sudo pip install pgzero
```

Nokre Linux-system kallar den `pip3`, i så fall må du skrive det i staden for `pip` i koden over. Viss pip ikkje er installert kan du prøve å skrive

```
sudo python3 -m ensurepip
```

før du prøver `sudo pip install pgzero` att.

Steg 1: Høgde og breidde

- [ ] Lag eit nytt Python-program med følgjande kode:

```
import pgzrun

HEIGHT = 400
WIDTH = 600

pgzrun.go()
```

- [ ] Køyr programmet og sjå kva som skjer. Du skal sjå eit svart vindauge som er 400 pikslar høgt, og 600 pikslar breitt.

Ein **piksel** er eit lyspunkt på skjermen og nøyaktig kor stort dette lyspunktet er avhenger av kva skjerm du har - så det kan vere at vindauget får ulik storleik på andre datamaskiner enn din.

Steg 2: Lag ein ball!

- [ ] Me skal lage ein ball som me kan vise på skjermen. Me startar med å lage ei `Ball` -klasse, som har variablane `radius` og `color` , og ein posisjon som består av `x` og `y` .

```
COLORS = {
    'red': (255, 0, 0),
    'green': (0, 255, 0),
    'blue': (0, 0, 255),
    'white': (255, 255, 255),
    'black': (0, 0, 0)
}

class Ball:
    radius =  20
    color = COLORS['red']
    x = WIDTH // 2
    y = HEIGHT // 2
```

Her har me valt å ha ein raud ball, men du kan velje ei anna farge frå `COLORS`-ordboka viss du vil det. Hugs at `//` tyder 'heiltalsdivisjon', det vil seie at svaret blir runda av nedover, slik at me får eit heiltal som svar.

- [ ] I tillegg må me ha ein funksjon som kan teikne ballen vår. Denne skal me kalle `draw()`. Hugs at funksjonane som skal vere ein del av klassa må ha eit innrykk. Me må endre på klassa så den ser slik ut:

```
class Ball:
    radius = 20
    color = COLORS['red']
    x = WIDTH // 2
    y = HEIGHT // 2

    def draw(self):
        screen.draw.filled_circle((self.x, self.y), self.radius, self.c
olor)
```

No er du nesten ferdig. Me må lage eit `Ball`-objekt, `ball1` og ein global `draw`-funksjon. Dette vil sjå slik ut:

```
ball1 = Ball()

def draw():
    screen.clear()
    ball1.draw()
```

Prosedyra `screen.clear()` syt for at me teiknar på ein blank skjerm, og må alltid kome fyrst i den globale funksjonen `draw()`.

 Test programmet ditt

No kan du teste programmet ditt. Du kan få opp ein einsfarga sirkel midt i vindauget.

Steg 3: Rørsle

- [ ] Me vil at ballen vår skal bevege seg. Korleis skal me få til dette? Me lagar funksjonen `update()`.

- [ ] Fyrst må me legjge til eit par variablar som bestemmer farta på ballen. Me skal ha ein variabel for farta i y-retning og ein variabel for farta i x-retning.

```
class Ball:
    radius =  20
    color = COLORS['red']
    x = WIDTH // 2
    y = HEIGHT // 2
    speed_x = 3
    speed_y = 3
```

- [ ] Så må me lage ein funksjon `update()` som er ein del av `Ball`. Denne syt for at ballen beveger seg `speed_x` pikslar i x-retninga, og `speed_y` i y-retninga.

```
class Ball:
    # ...

    def update(self):
        self.x += self.speed_x
        self.y += self.speed_y
```

- [ ] I tillegg må me ha ein global funksjon `update()` som kallar `ball1.update()`:

```
def update():
    ball1.update()
```

Test programmet ditt

No kan du teste programmet ditt att. Ballen skal bevege seg viss alt er gjort riktig.

Kva skjer når den kjem til kanten? I neste steg skal me syte for at ballen ikkje forsvinn ut av vindaugeet.

Steg 4: Veggkollisjonar

Me vil la ballen sprette tilbake når den treff ein vegg. Her er det eit par ting me må tenke på: korleis oppdagar me at ballen treff, og korleis kan me endre variablane slik at den sprett vekk frå veggen? Ballen sin posisjon er bestemt av `x` og `y`, men den har òg ein `radius` som me må ta omsyn til når me skal oppdage om ballen treff veggen. Når ballen treff den øvste eller nedste veggen vil me at farta blir reversert i y-retning, det samme gjeld for farta i x-retning når me treff høgre eller venstre vegg.

- [ ] Me må endre `update()` -funksjonen i `Ball` -klassa:

```
class Ball:
    # ...

    def update(self):
        self.x += self.speed_x
        self.y += self.speed_y

        # sjekkar for kollisjon i x-retning
        if self.x + self.radius >= WIDTH or self.x - self.radius <= 0:
            self.speed_x = -self.speed_x

        # sjekkar for kollisjon i y-retning
        if self.y + self.radius >= HEIGHT or self.y - self.radius <= 0:
            self.speed_y = -self.speed_y
```

🏁 Test programmet ditt

Køyr programmet ditt, og pass på at ballen sprett tilbake når den treff ein av veggane.

Steg 5: Styre farta til ballen

Me skal la brukaren styre farta til ballen ved hjelp av piltastane. Når brukaren trykkar på 'Pil opp' skal ballen gå raskare oppover (eventuelt mindre fort nedover), det motsette skal skje om brukaren trykkar 'Pil ned'. Det same skal skje viss brukaren trykkar på 'Pil høgre' eller 'Pil venstre', men då skal fartsendringa skje i x-retning.

- [ ] For å få til dette skal me lage ein `on_key_down()` -funksjon i `Ball` -klassa:

```
class Ball:
    # ...

    def on_key_down(self, key):
        if key == keys.LEFT:
            self.speed_x -= 1
        elif key == keys.RIGHT:
            self.speed_x += 1
        elif key == keys.UP:
            self.speed_y -= 1
        elif key == keys.DOWN:
            self.speed_y += 1
```

Legg merke til at funksjonen har ein parameter, `key`, som brukast til å avgjere kva tast brukaren trykka på.

- [ ] Me treng òg ein global `on_key_down()`-funksjon. Denne har òg ein `key`-parameter, som blir sendt videre til `ball1.on_key_down()`.

```
def on_key_down(key):
    ball1.on_key_down(key)
```

Test programmet ditt

No skal du ha ein ball som sprett mellom kantane i vindauget, og du skal kunne styre farta ved hjelp av piltastane.

Utfordring: Stopp ballen

Me ynskjer å bruke mellomromtasten for å stoppe ballen. Det vil seie at me skal setje `speed_x` og `speed_y` til `0`. Prøv å endre funksjonen `on_key_down(key)` i `Ball`-klassa for å sjekke om brukaren har trykka på mellomromtasten.

Hint: `key == keys.SPACE` vil vere sann viss brukaren trykkar på mellomromtasten.