Galaktisk Galskab i PyGame Zero – Workshop 2

I første workshop tegnede vi en stjernehimmel, der bevæger sig ved `blit` af en grafik-fil. Derefter placerede vi et rumskib på skærmen, som kan styres med keyboardet.

Alternativ stjernehimmel

Vi kan også tegne stjernerne enkeltvis som sprites med `Actor`. Nedenfor tilføjer vi 50 stjerner til en *liste* med tilfældig placering på skærmen. Filen `stjerne.png` kan du finde i `images` på https://github.com/stee0640/pgzero2025/. Gem den i dit programs `images` mappe.

Indsæt dette øverst i dit program:

```
import random

stjerne_liste = []
for _ in range(50):
    stjerne = Actor('stjerne.png')
    stjerne.x = random.randrange(WIDTH)
    stjerne.y = random.randrange(HEIGHT)
    stjerne_liste.append(stjerne)
```

For at tegne stjernerne på skærmen skal vi indsætte denne kode i `draw` callback:

```
    screen.clear() # Kun hvis kode til blit af baggrund er fjernet
    for stjerne in stjerne_liste:
        stjerne.draw()
```

Vi flytter stjernerne frem, indtil de når uden for skærmen. Derefter indsætter vi dem igen lige over toppen af skærmen. Det kan klares ved at tilføje denne kode i `update` callback:

```
    for stjerne in stjerne_liste:
        if stjerne.y < HEIGHT + 8:
            stjerne.y += 1
        else:
            stjerne.x = random.randrange(WIDTH)
            stjerne.y = -8
```

Hold rumskibet på banen

For at holde rumskibet på banen kan man udnytte at `Actor`-objekter har egenskaber som `top`, `bottom`, `left` og `right`, så vi ikke behøver beregne koordinater ud fra størrelsen på billedet:

```
def rumskib_bliv_paa_banen():
    if rumskib.top < 0:
        rumskib.top = 0
    if rumskib.bottom > HEIGHT:
        rumskib.bottom = HEIGHT
    if rumskib.left < 0:
        rumskib.left = 0
    if rumskib.right > WIDTH:
        rumskib.right = WIDTH
```

- Indsæt kaldet til funktionen `rumskib_bliv_paa_banen()` sidst i dit programs `update` callback og tjek at rumskibet ikke kan bevæge sig uden for skærmen.

Boost og ændring af grafik på Actor

Når vi trykker på "op"-tasten, vil vi have rumskibet til at lave et *boost*. Vi tilføjer samtidig en tyngdekraft, som trækker rumskibet nedad hele tiden. Under *boost bliver* billedet af rumskibet skiftet til et billede med raketmotoren tændt. Tilføj dette til `update`:

```
    if keyboard.up:
        rumskib.y -= 5
        rumskib.image = "rumskib2.png"
    else:
        rumskib.image = "rumskib1.png"
    rumskib.y += 2
```

Skyd!

Funktionen `on_mouse_down` bliver kaldt som callback, når vi trykker på musen. Nedenfor laver vi en ny `Actor` med et nyt skud, hver gang vi trykker på museknappen. Skuddet starter i punktet `rumskib.midtop`. Hvert nyt skud bliver tilføjet til en liste, så vi holder styr på dem:

```
skud_liste = []

def on_mouse_down():
    skud = Actor("skud1", pos=rumskib.midtop)
    skud_liste.append(skud)
```

Ved hver opdatering flytter vi alle skud på listen opad og sletter dem fra listen, når de kommer uden for skærmen. Tilføj følgende til `update` callback:

```
    for skud in skud_liste:
        skud.y -= 6
        if skud.bottom < 0:
            skud_liste.remove(skud)
```

- Cooldown: Kan du begrænse antallet af skud, man må have ude at flyve ad gangen?

Ramt?

Vi skal have noget at skyde på, så lav en ufo med `Actor` i toppen af dit program:

```
ufo = Actor('ufo1', pos=(300,100))
```

Husk at kalde `ufo.draw()` fra callback-funktionen `draw` for at få tegnet ufoen.

Vi kan tjekke om et skud rammer ufoen med funktionen `colliderect`, som vist nedenfor. Koden skal sættes ind i callback-funktionen `update` i `for`-loopet, der gemmemløber skuddene:

```
        if skud.colliderect(ufo):
            skud_liste.remove(skud)
            animate(ufo, x = random.randrange(0, WIDTH))
```

Når ufoen bliver ramt flytter den sig bare indtil videre. I næste workshop vil vi tilføje eksplosioner og flere ufoer, som skal kunne bevæge sig og skyde tilbage. Har du tid til over kan du lave fancy ting med `animate`-funktionen. Prøv at google "pgzero animate".