



Asteroidespillet, Del 2



Introduksjon

I denne oppgaven skal vi jobbe videre med romskipsspillet vårt fra [del 1](#), og nå skal vi lære å styre romskipet!

Vi starter med den samme koden som vi hadde sist, men uten romskipet! Denne gangen skal vi nemlig tegne romskipet på en litt annen måte.

Det er viktig at du har de samme bildene som sist liggende sammen med koden.

1. Lag spillskjelettet

Endre koden fra del 1 slik at den ser ut som dette:

```
import pygame

# Variable for vindusstørrelsen, slik at vi slipper å endre mange steder hvis vi vil bytte størrelse
display_width = 600
display_height = 800

# Start opp pygame slik at det kan brukes. Viktig!
pygame.init()

# Start pygame-klokka. Denne holder rede på tiden i spillet vårt
clock = pygame.time.Clock()

# Sett opp spillvinduet
gameDisplay = pygame.display.set_mode((display_width, display_height))
pygame.display.set_caption('Asteroid Run')

# Les inn bakgrunnsbildet
bgImg = pygame.image.load("background.png")

# Tegn bakgrunnsbildet
def drawBackground():
    gameDisplay.blit(bgImg, (0, 0))

# Start "hovedløkka" i spillet
finished = False
while not finished:
    # Sørg for at denne løkken går 60 ganger i sekundet
    clock.tick(60)

    # Sjekk hendelser
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            finished = True

    drawBackground()

    # Oppdater vinduet med all grafikken som skal tegnes
    pygame.display.update()

# Når vi hopper ut av løkka, avslutter vi spillet.
pygame.quit()
```

2. Lag en romskipsklasse

Klasser og Objekter

Hvis du ikke har tatt oppgaven [Enkle objekter](#) anbefaler vi at du gjør denne først, siden vi skal lage oss noen klasser.

I spillet vårt vil vi ha et romskip som vi kan styre på skjermen. Dette romskipet er et *objekt*. Et objekt er en ting vi kan gjøre noe med (flytte det rundt på skjermen), og som har noen egenskaper (f.eks et bilde som kan tegnes). Vi programmerer dette som en *klasse* i programmet vårt, slik at vi kan lage flere *objekter* av samme type.

Sprites

For å enkelt kunne tegne og flytte romskipet har vi lyst til å lage det som en "sprite". Dette er engelsk og betyr noe sånt som en flygende magisk alv, men i spillprogrammering er det et bilde som du kan flytte rundt på skjermen.

Pygame har allerede en klasse for slike sprites, så vi skal utvide denne klassen med det vi trenger for romskipet vårt. Vi *arver* dermed egenskapene og metodene til *foreldreklassen*.

Vi kaller vår klasse for "*Spaceship*" og foreldreklassen er `pygame.sprite.Sprite`. Skriv inn følgende kode, et sted etter `pygame.init()`, men før hovedløkka.

```
class Spaceship(pygame.sprite.Sprite):
    def __init__(self):
        pygame.sprite.Sprite.__init__(self)

        self.image = pygame.image.load("spaceship.png")

        self.rect = self.image.get_rect()

        self.rect.centerx = 300
        self.rect.centery = 700
```

Den første linjen sier at vi vil ha en *klasse* med navn "Spaceship" og den skal arve fra `pygame.sprite.Sprite`.

Deretter lager vi en funksjon som heter `__init__` som tar ett enkelt argument, nemlig `self`. Init-funksjonen finnes i alle klasser, og er en funksjon som blir kjørt automatisk når du lager et objekt med denne klassen.

Variablen `self` er en magisk variabel som alltid inneholder det objektet du er i, altså det Spaceship-objektet som funksjonen din gjør noe med.

Alle Sprite-objekter har en egenskap som heter `image`. Dette er det bildet som blir tegnet, og vi leser derfor inn romskipsbildet vårt:

```
self.image = pygame.image.load("spaceship.png")
```

En annen egenskap alle sprites må ha er `rect`. Dette er et rektangel, altså en firkant som sier hvor på skjermen vi vil ha bildet vårt, og hvor stort det skal være. Vi kopierer rektanglet fra bildet vårt og setter midten av rektanglet til å være der vi vil at romskipet skal være.

```
self.rect = self.image.get_rect()

self.rect.centerx = 300
self.rect.centery = 700
```

3. Lag et romskip og tegn det

Nå har vi laget klassen vår for romskip, og det neste vi skal gjøre er å lage et *objekt* av denne klassen, slik at vi får et romskip i spillet vårt.

Alle sprites i Pygame legges i en liste, og så tegner vi hele denne listen av sprites på en gang. Dette er lurt, for da bruker datamaskinen mindre tid på hver tegning, og vi kan tegne flere ting uten at spillet hakker og går tregt.

Siden romskipsklassen vår er en type sprite, kan vi putte det i denne sprite-listen, og så blir det tegnet sammen med alle de andre spriteene.

Legg inn følgende kode, rett før hovedløkka:

```
ship = Spaceship()

# Lag en liste av "sprites" og legg til romskipet vårt i listen.
sprites_list = pygame.sprite.Group()
```

```
sprites_list.add(ship)
```

Først lager vi altså en variabel som vi kaller "ship", som skal være av type "Spaceship", altså den nye klassen vår.

Så lager vi sprite-listen som vi kaller for "sprites_list", og legger inn romskipet i den.

Til slutt kommer det viktigste, nemlig å tegne sprite-listen vår. Dette må vi gjøre hver gang hovedløkka kjører, og vi må gjøre det *etter* at vi har tegnet bakgrunnen (kan du tenke deg hvorfor?). Legg inn denne linjen rett etter der du kjører `drawBackground()`

```
sprites_list.draw(gameDisplay)
```

- ☐ Kjør programmet og se hva som skjer. Du skal nå få opp det samme vinduet som du hadde i steg 1, med stjernehimmel og et romskip.

4. Styr romskipet med piltastene

Men dette er jo ikke noe spennende sier du sikkert, for det er jo akkurat det samme vi hadde før, bare med mer vanskelig kode! Det er nok sant, men nå kan vi gjøre mange ting med romskipet vårt som vi ikke kunne før. Det neste vi skal gjøre er å styre romskipet, og senere skal vi se at du kan enkelt sjekke om romskipet kræsjer med andre sprites, noe vi får god bruk for!

For å kunne styre skipet trenger vi først en funksjon i klassen vår for å sette posisjonen. Legg inn følgende kode, rett under `__init__`-funksjonen i Spaceship-klassen:

```
def updatePosition(self, x_speed):
    self.rect.centerx += x_speed
```

Denne funksjonen flytter på rektanglet ved å si at "centerx", altså midten av firkanten i x-retning skal være det den var før pluss variablen "x_speed".

Tips

I stedet for å skrive `variabel = variabel + 1` kan vi skrive `variabel += 1`. += betyr altså "det du hadde fra før, pluss noe mer"

Det neste vi skal gjøre er å lage en variabel "x_speed" som styrer hvor mye vi flytter oss til høyre eller venstre. Skriv inn dette rett *før* hovedløkka:

```
x_change = 0
```

Flere hendelser


I del 1 snakket vi om "hendelser" (events på engelsk), og vi sjekker allerede etter `pygame.QUIT`-hendelsen i hovedløkka vår. Nå skal vi sjekke om piltastene er trykket ned, og i så fall skal vi flytte på romskipet vårt. Skriv om for-løkka som sjekker hendelser så den ser slik ut istedet:

```
for event in pygame.event.get():
    if event.type == pygame.QUIT:
        finished = True
    if event.type == pygame.KEYDOWN:
        if event.key == pygame.K_LEFT:
            x_change = -5
        if event.key == pygame.K_RIGHT:
            x_change = 5
    if event.type == pygame.KEYUP:
        if event.key == pygame.K_LEFT or event.key == pygame.K_RIGHT:
            x_change = 0
```

Her er det viktig å holde innrykkene riktig, det er nemlig to if-tester inni hverandre her; Vi sjekker først *om* en tast ble trykket ned, og hvis det er sant, så sjekker *hvilken* tast som ble trykket ned.

Det siste skrittet er å ta `x_change`-verdien vår og putte den inn i `updatePosition`-funksjonen vi laget. Skriv inn følgende, etter at vi har sjekket hendelsene, og før vi tegner romskipet:

```
ship.updatePosition(x_change)
```

-  Kjør programmet og se hva som skjer. Du skal nå kunne styre romskipet med høyre og venstre piltast

Utfordringer

- Klarer du å hindre at romskipet flyr utenfor skjermen? Hint
- Klarer du å styre romskipet opp og ned på skjermen? Hint

Lisens: [CC BY-SA 4.0](#) **Forfatter** Thomas Sevaldrud