

pygame

Af: Michael Hansen, Coding Pirates Furesø, 2022-24, version 1.60

Dokumentet ligger her: <https://github.com/mhfalken/pygame/>




Her er en beskrivelse af hvordan man installerer pygame og hvordan man udvikler et lille Space invaders spil. Der er ikke noget specifik Python undervisning, men der er vist eksempler på kode konstruktioner der hvor man har brug for det.

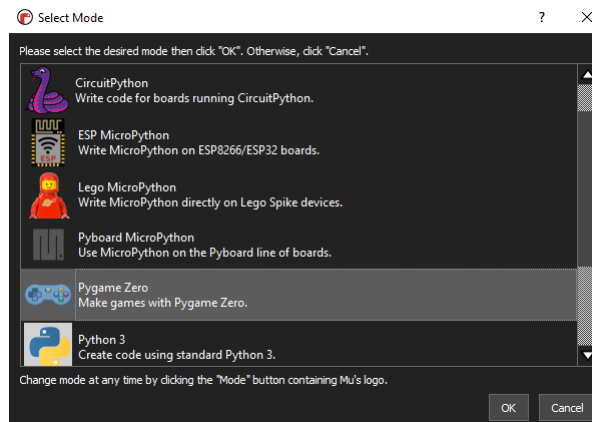
Det er skrevet til brug i et Coding Pirates forløb og beregnet til børn fra 12 år. Man kan let få noget ud af det selvom man kun bruger 2-4 gange på det. De fleste vil i løbet af første gang både få installeret det hele og få noget til at bevæge sig på skærmen.

| | | |
|------|----------------------------------|----|
| 1 | Installation af pygame | 3 |
| 2 | Space invaders | 3 |
| 2.1 | Intro..... | 3 |
| 2.2 | Flyt rumskibet..... | 5 |
| 2.3 | Grafik..... | 6 |
| 2.4 | Skud..... | 6 |
| 2.5 | Fjender..... | 7 |
| 2.6 | Kollisioner | 9 |
| 2.7 | Lyd..... | 9 |
| 2.8 | Point tæller | 9 |
| 2.9 | Alien's skyder igen | 10 |
| 2.10 | Lav flere rækker af aliens..... | 10 |
| 2.11 | Tempo | 10 |
| 2.12 | Game over..... | 11 |
| 2.13 | Tilføj et baggrundsbillede | 11 |
| 2.14 | Flere baner (levels) | 11 |
| 3 | Avanceret features..... | 12 |
| 3.1 | Flere rumskibe..... | 12 |
| 3.2 | Eksplosions animationer..... | 13 |
| 3.3 | UFO'er | 14 |
| 4 | Frivillig info | 14 |
| 5 | Links..... | 14 |

1 Installation af pygame

Det letteste er at installere MU editoren: <https://codewith.mu/en/download>.

Når den er installeret, så start MU og tryk så på **Mode** ikonet  oppe i venstre hjørne og vælg **Pygame Zero**.



2 Space invaders

Her er en overordnet beskrivelse af, hvordan man laver et simpelt Space invaders spil.

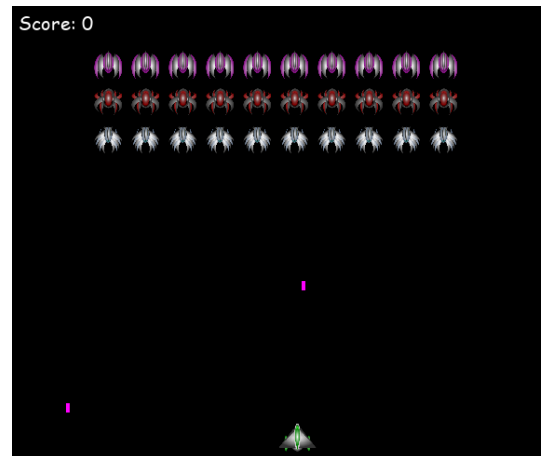
Spillet har et rumskib i bunden af skærmen, som kan skyde. Øverst er en række af fjender (alien's) som flytter sig og skyder på rumskibet. Det går ud på at udrydde alle fjenderne inden de dræber en først.

Man skal bruge en 'start' fil, som har den basale pygame-loop lavet.

1. Hent filen og gem den i en folder.

<https://github.com/mhfalken/pygame/blob/main/pygame-start.py>

Tryk på "Download raw file" og gem filen et fornuftigt sted.



Figur 1

2.1 Intro

2. Start med at kopiere filen **pygame-start.py** over i en ny fil som hedder **space_inv.py**.
3. Åben filen **space_inv.py** i MU. (Tryk på **Load** ikonet)
4. Start programmet (det kommer man til at gøre mange gange ...): Tryk på **Play** ikonet.

Nu skulle der gerne åbne et nyt vindue, hvor der er en lille gul flad firkant. Den kan flyttes til højre med højre piletast.

5. Stop spillet igen (Tryk på **Stop** ikonet). Det er vigtigt at stoppe spillet efter brug, da man ellers ikke kan starte spillet igen næste gang.

2.1.1 Gennemgang af koden

Her er en kort gennemgang af koden (se billede næste side). Den består af nogle standarddefinitioner (grå) som man ikke skal røre ved, på nær `WIN_WIDTH` og `WIN_HEIGHT`. De blå områder er dem som laver og flytter den gule firkant og det er meningen af man skal ændre og tilføje kode her. De gule områder er her, hvor meget af den nye kode skal sættes ind.

Game loop'en kører rundt 60 gang i sekundet (60 FPS), så alt man laver i den, bliver gjort 60 gange i sekundet!

```

1  # Standard control loop in pygame
2  # Made by Michael Hansen 2021-23
3  import pygame
4  import os
5  import random
6
7  pygame.font.init()
8  pygame.mixer.init()
9
10 # Colors (R, G, B)
11 WHITE = (255, 255, 255)
12 BLACK = (0, 0, 0)
13 RED = (255, 0, 0)
14 YELLOW = (255, 255, 0)
15
16 # Game window
17 FPS = 60
18 WIN_WIDTH, WIN_HEIGHT = 600, 500
19 WIN = pygame.display.set_mode((WIN_WIDTH, WIN_HEIGHT))
20
21 SHIP_WIDTH = 20
22 SHIP_HEIGHT = 20
23
24 def main():
25     ship = pygame.Rect(200, 300, SHIP_WIDTH, SHIP_HEIGHT)
26     clock = pygame.time.Clock()
27     run = True
28     while run:
29         clock.tick(FPS)
30         for event in pygame.event.get():
31             if event.type == pygame.QUIT:
32                 run = False # Spillet stopper
33
34             if event.type == pygame.KEYDOWN:
35                 if event.key == pygame.K_SPACE:
36                     pass # Her er trykket paa SPACE tasten
37             keys_pressed = pygame.key.get_pressed()
38             if keys_pressed[pygame.K_RIGHT]:
39                 ship.x += 1 # Her er trykket paa RIGHT tasten
40
41         # Update display (WIN)
42         WIN.fill(BLACK)
43         pygame.draw.rect(WIN, YELLOW, ship) # Tegner den gule firkant
44         pygame.display.update()
45
46 # -----
47
48 main()
49 pygame.quit()

```

Standard definitioner

Game vindue setup

Globale definitioner skal stå her

Rumskib

Keyboard håndtering

Game logik skal stå her

Grafik

loop

Figur 2

Linje 23 definerer rumskibet.

```
ship = pygame.Rect(x, y, width, height)
```

hvor (x, y) er placeringen af rumskibet og (width, height) er bredden og højden. Ved at ændre på x og y kan man flytte rundt på rumskibet. Man tilgår felterne på følgende måde:

```
ship.x  
ship.y  
ship.width  
ship.height
```

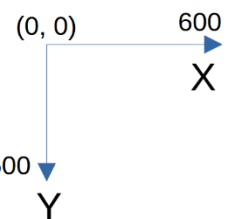
Følgende linje flytter rumskibet en til højre

```
ship.x = ship.x + 1
```

OBS: Nede i bunden af filen er en lang kommentar (ikke vist på billedet) som indeholder en masse små kode stumper som man med stor fordel kan bruge til at kopier fra, når man skal udvide spillet.

2.2 Flyt rumskibet

Skærmen er lavet som et stort koordinatsystem, men med den ændring at (0, 0) er i **øverste venstre** hjørne. Dvs. at x går fra 0->max som 'normalt', men y går fra top (0) til bund (max).



Lige nu er skærmens størrelse sat til 600x500, men det kan ændres ved at ændre værdierne i toppen af koden: WIN_WIDTH og WIN_HEIGHT.

1. Find de to linjer kode som flytter rumskibet til højre når man trykket på højre piletast. (kik efter: `if keys_pressed[pygame.K_RIGHT]:`)
2. Tilføj nu to linjer som får rumskibet til at flytte sig til venstre når man trykker på venstre piletast. (Det er en fordel at kopiere de to linjer som flytter rumskibet til højre og så rette lidt i dem.)
3. **Det er vigtigt at gemme koden inden den testes, da det ellers ikke virker! (Brug CTRL-S)**

Hvis man hellere vil bruge tasterne A og D så hedder de: `K_a` og `K_d`.

4. Rumskibet bevæger sig lidt langsomt. Prøv at få det til at bevæge sig lidt hurtigere.

Hvis man bliver ved med at trykke på piletasterne, så bevæger rumskibet sig ud af skærmen – det skal vi have fikset. Der skal nu laves noget kode som forhindrer at det sker.

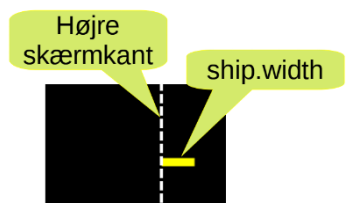
Det kan være en fordel at udvide den `if` sætning som checker piletasten til også at checke om man er inden for kanten af skærmen.

Det kan se sådan ud for den venstre kant af skærmen:

```
if keys_pressed[pygame.K_LEFT] and ship.x > 5:
```

5. Prøv at lave rettelsen og se om den virker.
6. Lav nu en tilsvarende rettelse for den højre kant. Her skal man bruge den konstant der hedder `WIN_WIDTH` som angiver hvor mange prikker skærmen er bred.

Her rammer man et typisk problem, da rumskibet (den gule firkant) har nulpunkt i øverst venstre hjørne og dermed når ud af skærmen inden den stopper (se billede). Det løses ved at lave lidt matematik og bruge bredden af rumskibet (`ship.width`) i `if` sætningen.



7. Lav nu den rettelse der gør at rumskibet bliver inden for den højre kant. (Hint: Matematikken kan se sådan ud: `ship.x+ship.width+5 < WIN_WIDTH`).

Nu skulle rumskibet gerne kunne bevæge sig til begge sider og blive på skærmen.

2.3 Grafik

Ind til videre så har vi 'bare' flyttet rundt på en gul firkant. Nu skal vi have lidt grafik på, så det ligner et rumskib.

1. Brug internettet til at finde et fint billede af et rumskib som du vil bruge i dit spil. Det skal være en `.png` fil.
2. Kald filen `ship.png` og gem den ved siden af din `space_inv.py` fil.

For at bruge billedet i spillet skal man gøre to ting.

3. Hente billedet ind i spillet.

```
SHIP_IMAGE = pygame.image.load('ship.png')
```

hvor `ship.png` er navnet på filen. Den linje skal stå i 'Globale definitioner' området – se "Gennemgang af koden".

4. Udskifte den linje som tegner den gule firkant med en som tegner billedet (Grafik området).

```
#pygame.draw.rect(WIN, YELLOW, ship) # Tegner den gule firkant
WIN.blit(SHIP_IMAGE, ship)
```

Som det ses, så er der sat en kommentar foran den gamle linje (`#`), så den ikke er aktiv længere. (Man kan også bare slette linjen.)

Nu skulle der gerne komme et billede frem på skærmen...

Billedet er sandsynligvis lidt stort og vender måske også forkert. Det løses ved at indsætte nogle flere linjer lige under `SHIP_IMAGE = ...` linjen.

```
SHIP_IMAGE = pygame.transform.scale(SHIP_IMAGE, (SHIP_WIDTH, SHIP_HEIGHT))
SHIP_IMAGE = pygame.transform.rotate(SHIP_IMAGE, 90)
```

Den første linje ændrer størrelsen (`scale`) til `SHIP_WIDTH` og `SHIP_HEIGHT`. Hvis man ønsker at rette størrelsen af sit rumskib, så skal man rette `SHIP_WIDTH` og `SHIP_HEIGHT` lidt højere oppe i koden. Den næste linje drejer (`rotate`) billedet et vist antal grader – her 90 grader. Den skal kun bruges hvis billedet skal drejes.

5. Lav nu de ændringer så rumskibet får en god størrelse og vender den rigtige vej.

Til sidst skal vi have flyttet vores rumskib ned i bunden af skærmen.

6. Ret `y` i rumskibet der hvor det er defineret, så skibet kommer ned i bunden af skærmen – (se "Gennemgang af koden").

Nu skulle man gerne have et billede af et rumskib i bunden af skærmen, som kan flyttes til siderne og blive inden for skærmen.

2.4 Skud

Nu skal vi have vore rumskib til at kunne skyde nogle skud op ad skærmen. Da den skal kunne skyde mere end et skud ad gangen, skal vi bruge en liste til at holde styr på skuddene. En liste defineres på følgende måde:

```
shipBullets = []
```

1. Sæt linjen lige under der, hvor `ship` er defineret.

Nu har vi en tom list af skud.

2. Man skyder ved at trykke på **space tasten**, så find det sted i koden og sæt de følgende linjer ind der. Man laver et skud (`bullet`) på følgende måde:

```
bullet = pygame.Rect(x, y, 4, 10)
```

hvor `x`, `y` skal have nogle "fornuftige" værdier – det kommer vi til senere – start med 300, 400.

3. Skuddet skal ind i listen og det gør man på følgende måde:

```
shipBullets.append(bullet)
```

Nu har man en liste med skud.

For at få skuddet til at komme frem på skærmen, skal vi vise det. Det gøres ved at 'løbe' hele listen igennem og for hvert skud vise det på skærmen.

4. Indsæt følgende linjer (koden skal stå i "Grafik" området under `WIN.blit(...)`.)

```
for b in shipBullets:
    pygame.draw.rect(WIN, RED, b)
```

Koden her løber skudlisten igennem og viser dem et af gangen.

5. Kør spillet og se hvad der sker.

Der er flere ting galt: Skuddene bevæger sig ikke, men står stille i bunden af skærmen og kommer heller ikke ud hvor rumskibet er.

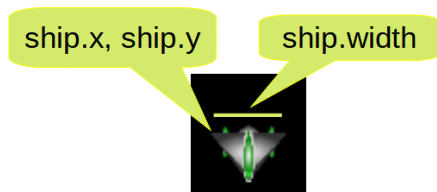
6. Man får skuddene til at bevæge sig ved at løbe skudlisten igennem og så flytte dem en af gangen.

```
for b in shipBullets:
    b.y = b.y - 5
```

koden skal stå i "Game logik" området.

Nu skulle skuddene gerne bevæge sig op ad skærmen.

Koden skal nu rettes så skuddene kommer ud af rumskibet der hvor det er og ikke bare midt på skærmen. Dette gøres ved at rette i den linje hvor man laver skuddene (`x`, `y`) (der hvor vi bare skrev 300, 400) og så lave lidt matematik ved at bruge `ship.x`, `ship.y` og `ship.width`.



7. Ret nu den linje som laver skuddene så de kommer ud af rumskibet, i stedet for midt på skærmen.

Skuddene forsætter ud af skærmen, men bliver ved med at forsætte uden for skærmen også. Vi skal derfor have lavet et check for om skuddene er uden for skærmen og så fjerne dem fra listen som er udenfor (ellers vil systemet blive overbelastet på sigt).

Selve checket ligner det for rumskibet kantcheck. Man fjerner et skud fra listen på følgende måde:

```
shipBullets.remove(b)
```

8. Ret nu koden til der hvor skuddene bliver flyttet, til også at slette dem, hvis de forlader skærmen.

Man kan skyde lige så mange skud man vil på en gang. Det kan godt blive lidt vildt så der skal laves en begrænsning på, hvor mange skud man kan skyde på en gang.

Antallet af skud svarer til antallet af elementer i skudlisten og findes på følgende måde:

```
len(shipBullets)
```

9. Tilføj nu et check så man kun kan have max 3 skud i luften ad gangen ved at tilføje til `if` sætningen inden man tilføjer et skud til listen. Det er denne linje der skal rettes i:

```
if event.key == pygame.K SPACE and xxx:
```

hvor `xxx` skal erstattes med checket for længden af listen.

Nu skulle man gerne kunne skyde op til 3 skud ad gangen og de starter det rigtige sted (ud fra rumskibet).

10. Test at det hele virker og at man kan skyde igen, når skuddene er ude af skærmen.

2.5 Fjender

Spillet er ikke sjovt uden fjender, så nu skal vi have tilføjet nogle af dem.

1. Start med at lave en tom liste af `aliens` lige under `shipBullets = []`.
2. Lav to konstanter og kald dem `ALIEN_WIDTH` og `ALIEN_HEIGHT` (som for rumskibet) og giv dem samme værdier til at starte med.
3. Man tilføjer en `alien` til listen på følgende måde (skal stå lige under hvor `aliens` er defineret):

```
alien = pygame.Rect(x, y, ALIEN_WIDTH, ALIEN_HEIGHT)
aliens.append(alien)
```

Husk at (`x`, `y`) angiver hvor `alien`'en er placeret. Lige nu har vi kun én alien i listen, men vi tilføjer flere senere.

4. Find et alien billedet på nettet, kald det `alien.png` og gør ligesom med rumskibsbilledet, men kald det `ALIEN_IMAGE` i stedet.

Hint: Det skal se sådan ud:

```

ALIEN_WIDTH = 40
ALIEN_HEIGHT = 30
ALIEN_IMAGE = pygame.image.load('alien.png')
ALIEN_IMAGE = pygame.transform.scale(ALIEN_IMAGE, (ALIEN_WIDTH, ALIEN_HEIGHT))

```

For at vise billedet, skal man løbe alien listen igennem (som for shipBullets) og vise dem (som for SHIP_IMAGE).

5. Tilføj de to linjer i "Grafik" området, som løber listen igennem og viser billedet.
6. Kør programmet og se at der kommer et alien billede frem.
7. Tilpas ALIEN_WIDTH og ALIEN_HEIGHT så dimensionerne og størrelsen passer.
8. Lav en loop som tilføjer 10 alien's til listen og sørg for at de ikke overlapper hinanden. Man laver en loop på følgende måde:

```

for i in range(10):
    ...

```

Der skal lidt matematik til at fordele alien'sne jævnt over skærmen som på billedet i Figur 1. (Beregn x . Det er vigtigt at de bliver sat ind i listen med stigende x værdier).

x kan med fordel beregnes noget ala det her: $x = 50 + i*50$.

Hint: Det skal ligne noget ala det her:

```

for i in range(10):
    alien = pygame.Rect(50+i*50, 100, ALIEN_WIDTH, ALIEN_HEIGHT)
    aliens.append(alien)

```

Nu skal vi have dem til at bevæge sig. De skal bevæge sig fra side til side og for hver gang de når en kant, skal de flytte sig lidt nedad. På den måde kommer de tættere og tættere på rumskibet som tiden går.

Man starter med at lave en variabel som holder styr på hvilken vej de bevæger sig lige nu (venstre eller højre). (Skal stå i området hvor ship er defineret).

9. Opret variable

```

alienMoveX = -1

```

10. Lav en løkke som løber alle alien's igennem og flytter dem alle sammen med alienMoveX. Skal stå i "Game logik" området).

```

for a in aliens:
    a.x = a.x + alienMoveX

```

Derefter skal man lave nogle if sætninger som checker om alien'en i enden af listen er ude over skærmkanten. Det er nok at checke den alien som er i den ende som er tættest på kanten. De kan findes på følgende måde – første og sidste alien:



- `aliens[0]` Første alien (venstre side)
- `aliens[-1]` Sidste alien (højre side)

11. For venstre side af skærmen kan det se således ud:

```

if alienMoveX < 0 and aliens[0].x < 10:
    alienMoveX = -alienMoveX
    for a in aliens:
        a.y = a.y + 20

```

12. Test koden og lav den tilsvarende kode for højre side af skærmen. (Hust at bruge WIN_WIDTH og `aliens[-1]`)

Når en alien når bunden af skærmen uden af være blevet slået ihjel inden, så skal den forsvinde.

13. Lav et check som ser om en alien har nået bunden af skærmen og fjern den i givet fald (Brug WIN_HEIGHT.)

2.6 Kollisioner

Nu skal vi have rumskibets skud til at virke, dvs. slå nogle alien's ihjel.

Pygame har en speciel funktion, som kan beregne om to figurer overlapper (rører) hinanden, som fx om et skud rører en alien. Følgende funktion checker om et skud rører en alien:

```
bullet.collidect(alien)
```

Lav en løkke som løber igennem `shipBullets` listen og for hver skud checker om de rører en alien i `aliens` listen. Det kræver at man laver en løkke inden i en anden løkke! Det kan se således ud:

```
for b in shipBullets:
    for a in aliens:
        if b.collidect(a):
            <her er alien ramt>
```

1. Tilføj koden og ret den sidste linje så den sletter både skuddet og alien'en. (Kræver 2 linjer). Man sletter alien'en på følgende måde: `aliens.remove(a)`. (Koden skal stå efter at bullets er blevet 'flyttet')
2. Tilføj selv linjen for at slette skudet.
3. Test at både alien og skud forsvinder når de bliver ramt.

Man kan se om en liste ikke er tom med følgende linje (undersøger om længden af listen er større end 0):

```
if len(aliens) > 0:
    <...>
```

4. Når alle alien's er væk, så kommer der sandsynligvis en fejl i koden. Prøv at forstå hvorfor og indsæt en ekstra `if` sætning for at undgå det opstår. (Hint: man må ikke tilgå et element i en liste som er tom.)

Hvis en alien kommer hent ned i bunden af skærmen, så rammer den vores rumskib. Hvis det sker så skal spillet stoppe (man er død) og det gør man med følgende linje:

```
run = False
```

5. Tilføj noget kode som undersøger om en alien rammer vores rumskib og så stopper spillet. Man kan med fordel sætte koden ind der hvor man flytter aliens frem og tilbage. Man checker om en alien rammer vores rumskib på følgende måde:

```
if a.collidect(ship):
```

hvor `a` er en alien og `ship` er vores rumskib.

2.7 Lyd

Nu mangler vi bare at sætte lyd på spillet. Det kan godt være lidt svært at finde lydfiler på nettet, så jeg har nogle man kan få, hvis man ikke selv kan finde nogen. I bunden af dokumentet er det nogle links til gode steder, hvor man kan finde lydfiler.

Ligesom med grafik, så skal man i "globale definitioner" af filen inkludere de lydfiler som man vil bruge:

```
SHIP_FIRE_SOUND = pygame.mixer.Sound('laser1.ogg')
```

Når man så nede i sin kode skal have lyd, så sætter man bare følgende linje ind:

```
SHIP_FIRE_SOUND.play()
```

1. Prøv at få lyd på når rumskibet skyder.
2. Sæt også lyd på når en alien bliver skudt.

Nu har man et fint lille spil, som demonstrerer hvad pygame kan.

2.8 Point tæller

Tilføj en point tæller oppe i toppen af skærmen. Man laver en variabel til at gemme scoren i og tæller den op hver gang man dræber en alien.

1. Opret en variabel og kald den `score` og sæt den lige under der, hvor `alienMoveX` er defineret.
2. Opdater `score` der hvor man rammer en alien.

Når man skal vise selve score teksten på skærmen gøres følgende:

3. Man skal først definere den 'font' (skrifttype) man vil skrive med i "Globale definitioner" området.

```
SCORE_FONT = pygame.font.SysFont('comicsans', 20)
```

20 angiver størrelsen af fonten.

4. For at vise scoren gøres følgende: ("Grafik" området)

```
text = SCORE_FONT.render("Score: %i" %(score), 1, WHITE)
WIN.blit(text, (20, 5))
```

Dette viser `score` i øverste venstre hjørne af skærmen. Hvis man hellere vil have scoren til at stå et andet sted på skærmen, så rettes tallene 20 og 5 i ovenstående kode.

2.9 Alien's skyder igen

Vi skal nu have alien'sne til at skyde igen. Man laver en list til skuddene (`enemyBullets`) ligesom til rumskib skuddene. Der hvor man løber alien listen igennem for at flytte dem, kan man for en tilfældig alien en gang imellem få den til at sætte et skud ind i `enemyBullets` listen, ligesom man gør med rumskibet.

1. Opret `enemyBullets` lige under `shipBullets`.

Når man skal vælge en tilfældig alien, skal man bruge en funktion som genererer et tilfældig tal.

```
random.randint(0, 10)
```

Denne funktion laver et tilfældigt tal mellem 0 og 10.

Man kan bruge funktionen på mange måder, og her er et eksempel som kan bruges til at få en alien til at skyde på et tilfældigt tidspunkt. (Koden skal stå inde i den loop hvor man flytter alien'sne).

```
if random.randint(0, 10) < 5:
    <tilføj bombe>
```

Man styrer hvor ofte en alien skyder ved at ændre på 10 og 5 tallene i linjen ovenover.

2. Tilføj en bombe til `enemyBullets` inde i `if` sætningen ovenover, ligesom man laver et skud fra vores rumskib (`append`).

Når man har fået tilføjet nogle skud til `enemyBullets`, skal de også flyttes ligesom med `shipBullets`.

3. Kopier den kode fra `shipBullets` som flytter skuddene og ret den til så den flytter `enemy` skuddene i stedet. (De skal bevæge sig i den modsatte retning!)
4. Kopier den kode som fjerner `shipBullets` når de ryger ud i toppen af skærmen og ret den til så `enemy` skuddene forsvinder når de ryger ud i bunden af skærmen (brug `WIN_HEIGHT`).

Hvis `enemyBullets` rammer vores rumskib så skal vi dø.

5. Tilføj den kode som undersøger om `enemyBullets` rammer vores skib og sæt `run = false`, hvis det sker. Kik på koden for hvordan `shipBullets` rammer en alien.

2.10 Lav flere rækker af aliens

Vi skal nu lave flere rækker af alien's. Tilføj en loop mere inde i den loop som opretter alien'sne, og ændre den `y` værdi som alien'sne bliver oprettet med.

Det kan se sådan ud (den nye linje er i **fed** – husk at rykke linjerne neden under ind):

```
for i in range(10):
    for j in range(3):
```

Man skal også ændre lidt i hvor de 'nye' aliens bliver placeret (`y` værdi), da de ellers bliver placeret lige oven i de andre og dermed ikke kan ses.

```
alien = pygame.Rect(50+i*40, 50+j*40, ALIEN_WIDTH, ALIEN_HEIGHT)
```

2.11 Tempo

Tempo handler om hvordan tingene bevæger sig og man kan med fordel ændre tempoet over tid, så tingene kører hurtigere jo længere man er kommet.

Få alien'sne til at bevæge sig hurtigere jo længere de kommer ned over skærmen.

1. Ændrer `alienMoveX` over tid. Her er det bedst at lægge lidt til `alienMoveX` hver gang alien'sne flytter en linje ned ad. Det kan se sådan ud:

```
alienMoveX = -alienMoveX+0.3
```

Hver opmærksom på at man det ene sted skal lægge lidt til og det anden sted skal man trække lidt fra! Pga. forskellig afrunding, så vil de ikke bevæge sig lige hurtig i begge retninger.

2.12 Game over

Når spillet er slut så skriv på skærmen om man har vundet eller tabt. Tilføj en variabel som angiver om man er død og så check den når spillet er slut og skriv en tekst ud.

1. Lav en variabel i bunden af "Globale definitioner", som skal bruges til at huske om vi er døde i spillet:

```
dead = False
```

2. I toppen af `def main():` funktionen, tilføj følgende linje:

```
global dead
```

3. Find det sted i koden, hvor man dør og sæt `dead = True`.

4. Helt nede i bunden af sin kode, lige før `pygame.quit()`, indsæt følgende linjer:

```
pygame.draw.rect(WIN, (100, 100, 100), (200, 200, 200, 100))
text = SCORE_FONT.render("Game over", 1, YELLOW)
WIN.blit(text, (WIN_WIDTH/2-text.get_width()/2, WIN_HEIGHT/2 - 40))
pygame.display.update()
pygame.time.delay(4000)
```

5. Kør spillet og se at den skriver *Game over* når spillet er slut.

Problemet er at spillet kun stopper hvis man dør, men spillet skal jo også stoppe, når der ikke er flere alien's tilbage.

6. Tilføj nu følgende linjer i bunden af "Game logik" området:

```
if len.aliens) == 0:
    run = False
```

7. Test spillet og se at det også stopper, hvis man vinder.

Brug nu `dead` variablen vi lige har lavet til at afgøre om man har vundet eller tabt.

8. Tilføj følgende linjer lige efter `WIN.blit(...)` i koden overover.

```
if (dead):
    text = SCORE_FONT.render("You lost", 1, RED)
    WIN.blit(text, (WIN_WIDTH/2-text.get_width()/2, WIN_HEIGHT/2))
```

Nu skriver den ud hvis man har tabt.

9. Tilføj selv en `else` sætning, som skriver en tekst når man har vundet.

2.13 Tilføj et baggrundsbillede

For at få spillet til at se lidt mere fedt ud, kan man tilføje et baggrundsbillede til sit spil.

Find et godt billede og gem det ved siden af alle de andre billeder i spillet.

1. Tilføj følgende linje i "Globale definitioner" området:

```
BACKGROUND_IMAGE = pygame.image.load('background.png')
```

Hvor 'background.png' skal rettes til navnet på det billede man har valgt.

2. I toppen af "Grafik" området (lige efter `WIN.fill(BLACK)`), tilføj følgende linje:

```
WIN.blit(BACKGROUND_IMAGE, (0, 0))
```

2.14 Flere baner (levels)

Vi skal nu lave flere baner efter hinanden, hvor sværhedsgraden stiger for hver bane. Vi genbruger den samme kode, men laver små ændringer, så banerne kommer til at se forskellige ud.

Vi starter med at genbruge den samme bane 3 gange, for på den måde at lave 3 ens baner efter hinanden.

1. Lav en ny variabel lige efter `dead = False`:

```
level = 0
```

2. Find det sted i bunden af koden hvor `main()` bliver kaldt og tilføj følgende linjer (dem i **fed** er de nye linjer):

```
for level in range(3):
    main()
    if dead:
        break
```

3. Prøv spillet og se at man nu har 3 baner efter hinanden, med mindre man dør undervejs.

Som man kan se, så bliver ens point nulstillet efter hver bane. Det er jo ikke meningen, så det skal vi have fikset.

4. Flyt `score = 0` ud af `main()`, så den står lige under `level = 0`.
5. Lav følgende ændring i koden (toppen af `main()` funktionen):

```
global dead, level, score
```

Prøv spillet igen og se at ens score overlever fra bane til bane.

Nu skal vi ændre i koden så de 3 baner bliver forskellige for hvert level. Vi laver flere forskellige ændringer, så spillet bliver mere varieret.

Første skridt er at styre antallet af aliens pr. level, så level 0 (første level) har 1 række, level 1 har 2 rækker og level 2 har 3 rækker alien's.

1. Det gør vi ved at ændre lidt i den loop, som laver antallet af alien's rækker. Find den linje som laver antallet af rækker alien's og ret den fra:

```
for j in range(3):
```

til:

```
for j in range(level+1):
```

2. Prøv spillet og se at det virker.

For hvert level skal vi have et nyt alien billede, så det er tydeligt at vi er kommet et level højere op.

3. Find to billeder mere som skal være vores nye alien billeder og gør som for det første alien billede – se 2.5 Fjender, men kald det noget andet, fx `ALIEN_IMAGE2` og `ALIEN_IMAGE3`.
4. I "Grafik området", der hvor man viser alien billedet, indsæt en `if` sætning, som vælger et image afhængig af level. Det kan se sådan ud (nye linjer i **fed**):

```
if level == 0:
    WIN.blit(ALIEN_IMAGE, a)
elif level == 1:
    WIN.blit(ALIEN_IMAGE2, a)
```

5. Tilføj selv den sidste `else:` linje for `ALIEN_IMAGE3`.
6. Kør spillet og se at alien billedet ændres for hvert level.

For hvert level skal vores alien's skyde oftere, så de i level 0 skyder sjældent, mens de i level 2 skyder ofte. Det gør man ved at ændre lidt i den 'formel' som genererer skuddene.

7. Finde linjen med `random.randint` og ændre den til:

```
if random.randint(0, 10) < (2*level + 1):
```

Man styrer, hvor ofte de skyder ved at rette lidt i tallene i linjen ovenover.

Kør spillet og se at de skyder oftere jo længere man kommer i spillet. Trim eventuelt på tallene så det får den 'rigtige' sværhedsgrad.

8. Man kan også skifte baggrundsbilledet for hver level. Det gøres på samme måde som da vi skiftede alien billedet for hvert level, ved at bruge en `if` sætning der, hvor man viser baggrundsbilledet.

3 Avanceret features

Her beskrives nogle lidt mere avanceret features, hvor det er meningen at man selv skal lave lidt mere.

3.1 Flere rumskibe

Man kan få et ekstra rumskib fx når man når et vist antal point. Rumskibet sættes ved siden af det man allerede har og hvor de bevæger sig og skyder sammen.

Ideen her er at man alle de steder, hvor man bruger `ship` ændre det til en liste af `ship` i stedet for.

1. Start med at lave en liste til at holde `ship` i. Den skal indsættes lige over der hvor man lave sit rumskib (`ship`):

```
ships = []
```

2. Tilføj nu vores rumskib til listen.

```
ships.append(ship)
```

3. For at gøre det lettere at teste koden, så tilføj et ekstra rumskib til listen, som ligger lige til højre for det andet.

```
ship = pygame.Rect(WIN_WIDTH/2 + SHIP_WIDTH, WIN_HEIGHT - 50, SHIP_WIDTH, SHIP_HEIGHT)
ships.append(ship)
```

4. Gå nu koden igennem og ret alle de steder, hvor `ship` bliver brugt til at bruge listen i stedet. Det kan se sådan ud:

Før:

```
ship.x += 1
```

Efter:

```
for s in ships:
    s.x += 1
```

5. Når man har rettet det alle steder, så prøv og se at alt stadig virker. Der skal være to rumskibe, som kan bevæge sig sammen og som begge kan skyde. De skal kunne dø et ad gangen. Ret koden de steder som fejler eller mangler.
6. Slet nu det ekstra rumskib vi tilføjede i starten og tilføj det i stedet når man har fået et vist antal point (som bonus). Brug de slettede linjer som guide.
7. Afspil også en lyd når man får et ekstra rumskib.
8. Test alt virker.

Man kan godt have mere en 2 rumskibe, hvis man har lyst til det.

3.2 Eksplosions animationer

Her skal vi lave en eksplosionseffekt, når en alien bliver skudt.

Animationer er en liste af billeder, som tilsammen laver en lille film (animation). Find nogle billeder, som kan bruges til en eksplosionseffekt og kald dem `ex1.png`, `ex2.png` etc. Der skal være 10-20 billeder for at det bliver godt.

Først skal vi have alle vores billeder ind i spillet, og her udnytter vi at deres navn er sammensat af et navn og et tal.

1. Indsæt følgende linjer under der hvor vores aliens billeder bliver lavet (globale definitioner)

```
EXPLOSIONS = []
for i in range(16):
    EXPLOSIONS.append(pygame.transform.scale(pygame.image.load('ex%i.png' % (i+1)),
    (ALIEN_WIDTH, ALIEN_HEIGHT)))
```

Vi skal nu have lavet en liste til at holde vores aktive eksplosioner i, dvs. at når en alien dør, så sætter vi den ind i eksplosionslisten. Eksplosionslisten består af et par, dvs. to værdier:

- Alien som er død – for at få fat i dens koordinater (x, y) så vi ved, hvor eksplosionen skal være.
- Et tal, som angiver hvilket billede i animationen vi er nået til at vise. Her starter vi med 0.

2. Opret eksplosionslisten:

```
explosions = []
```

3. Der hvor vores alien dør, indsæt følgende linje:

```
explosions.append((a, 0))
```

Nede i grafikområdet skal vi vise eksplosionerne. Det smart her at, at vi bruger den samme kode til at vise alle eksplosionerne med uanset hvor alien'en døde og hvor langt den er i dens animation.

Planen her er at vi løber hele eksplosionslisten igennem og viser et billedet for hver død alien, men samtidig så fjerner vi elementet og sætter det ind igen, men hvor vi har talt vores tal op. Det bliver vi ved med indtil der ikke er flere billeder tilbage.

4. Det ser sådan ud:

```
expl = [] # Temp list
for (a, i) in explosions:
    WIN.blit(EXPLOSIONS[i], a)
    explosions.remove((a, i))
    if i < 15:
        expl.append((a, i+1))
expl = expl
```

5. Kør spillet, og se at der kommer eksplosioner frem, hver gang en alien dør. (Koden er lavet til en animation med 16 billeder, så hvis man har et andet antal billeder, skal man rette tallet 15 overover.)

Det smarte ved den måde vores eksplosionsliste er lavet på er, at vi kan genbruge al koden til andre eksplosioner, som når vores eget rumskib bliver ramt.

6. Find det sted i koden, hvor vores rumskib bliver ramt, og sæt det ind i eksplosionslisten, som vi gør når en alien dør. Eksplosionsanimationen for vores rumskib virker kun, når man har mere end et rumskib, da spillet ellers stopper.

3.3 UFO'er

Lav en UFO som flyver hen over skærmen (vandret) og som kan rammes og give ekstra point (bonus).

1. Find et billede til vores UFO.
2. Få billedet ind i spillet som for vores aliens. Kald det `UFO_IMAGE`.
3. Lav en liste til vores UFO'er. Kald den `ufos = []`.
4. Lav en UFO i venstre side af skærmen oppe i toppen og indsæt den i `ufos` listen.
5. Lav den kode som får UFO'en til at bevæge sig vandret hen over skærmen – kig på den kode som får vores alien's til at bevæge sig til siderne.
6. Når UFO'en når højre side af skærmen, skal den fjernes, lige som vores skud gør det når det ryger ud af skærmen.
7. I den kode som checker om vores skud har ramt en alien, tilføj her kode som også checker om vi har ramt UFO'en.
8. Når vi har ramt UFO'en, skal vi have nogle point (bonus) og der skal også afspilles en lyd.
9. Lav nu noget kode om en gang i mellem sætter en UFO ind i `ufos` listen. Kik på hvordan koden til aliens skud ser ud.

Nu skulle spillet gerne have nogle UFO'er som en gang i mellem flyver hen over skærmen og giver point hvis de rammes.

10. Når UFO'en rammes, så tilføj den til eksplosionslisten.
11. Få nu UFO'erne til at skyde en gang i mellem. Det gøres på samme måde som når aliens skal skyde.

4 Frivillig info

På github ligger der en fuld version af Space Invaders spillet som kan bruges til at gøre det lettere at hjælpe piraterne, uden at man skal kode hele spillet selv først. Versionen følger i store træk dette dokument, men et par steder er der brugt lidt mere avanceret kodning for at få en bedre kode.

Version hedder `space_inv_150.py` og ligger her:

https://github.com/mhfalken/pygame/tree/main/space_invaders

Hvis man vil prøve spillet, skal man også bruge nogle billede- og lydfiler som ligger i `/effects/` folderen. Der ligger også et mere komplet Space Invaders spil, som kan bruges til at vise lidt mere af hvad pygame kan. Det hedder `space_inv.py` og har små animationer, flere levels og high score.

5 Links

| | |
|---------------|---|
| Pygame | www.pygame.org |
| Grafik og lyd | https://opengameart.org/ |
| Grafik og lyd | https://kenney.nl/ |
| Lyd | https://www.fesliyanstudios.com/ |