pygame

Af: Michael Hansen, Coding Pirates Furesø, 2022-23, version 1.31

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

Først skal man installere Python og der gøres ved at downloade Python og installerer det.
https://www.python.org/downloads/

Derefter skal pygame installeres og metoden afhænger lidt af om man kører Windows eller MAC.
Windows:
Åbent en Windows PowerShell (tryk på Windows tasten og skriv powershell). Skriv følgende i PowerShell vinduet:

```
py -m pip install -U pygame --user
```

MAC:
Åbent et Terminal vindue (tryk på søg i toppen af skærmen og skriv terminal). Skriv følgende i Terminal vinduet:

```
pip3 install pygame
```

Nu er pygame klart til brug. Lidt senere beskrives hvordan det startes.

2 Visual Studio Code editor

For at kunne skrive Python kode, skal man bruge en editor. Her er mange muligheder, men jeg vil til pygame anbefale Visual Studio Code (VSC).
Man skal installere følgende fil:
https://code.visualstudio.com/

Editoren har mange gode egenskaber, hvoraf nogle få vil blive omtalt her.
Oppe i toppen af vinduet er en Menu. For at åbne en fil vælges File->Open file.
Når man skriver pygame kode, så hjælper VSC med at komme med forslag til de funktioner og værdier som man er ved at indtaste. Så det er vigtigt at kikke på skærmen mens man taster og udnytter denne hjælp.
Ude til venstre er der linje numre. Disse bruges hvis man får en fejl, da der altid stå et linjenummer som hjælp til at lede efter fejlen.
Gode genvejstaster: (CTRL = CMD på MAC)

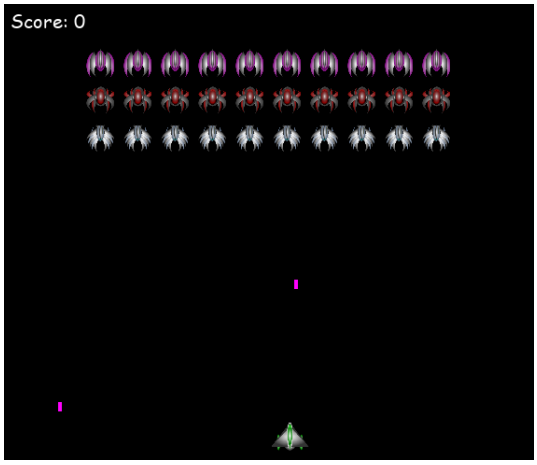
Tast	Funktion
CTRL-S	Gem filen
CTRL-Z	Undo
CTRL-C	Kopier
CTRL-V	Indsæt

Specielle MAC taster:

Tast	Funktion
ALT-8	[
ALT-9	]

3 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 (aliens) som flytter sig og skyder på rumskibet. Det går ud på at udrydde alle fjenderne inden de dræber en først.



Figur 1

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
Det kan godt være lidt svært at hente filen, så det letteste er at åbne en tom fil i VSC (**file->New file**) og så kopiere indholdet af pygame-start.py over i den. Gem nu filen (**file->Save As…**) og kald den *pygame-start.py*.

3.1 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 VSC. (file->open file)

4. Start programmet (det kommer man til at gøre mange gange …):

Windows:
Åben en Windows PowerShell (se under Installering) og skift bibliotek til der hvor filen ligger og skriv følgende kommando:

```
python space_inv.py
```

Man kan også skrive:

```
py space_inv.py
```

MAC:
Åben en Terminal (se under Installering) og skift bibliotek til der hvor filen ligger og skriv følgende kommando:

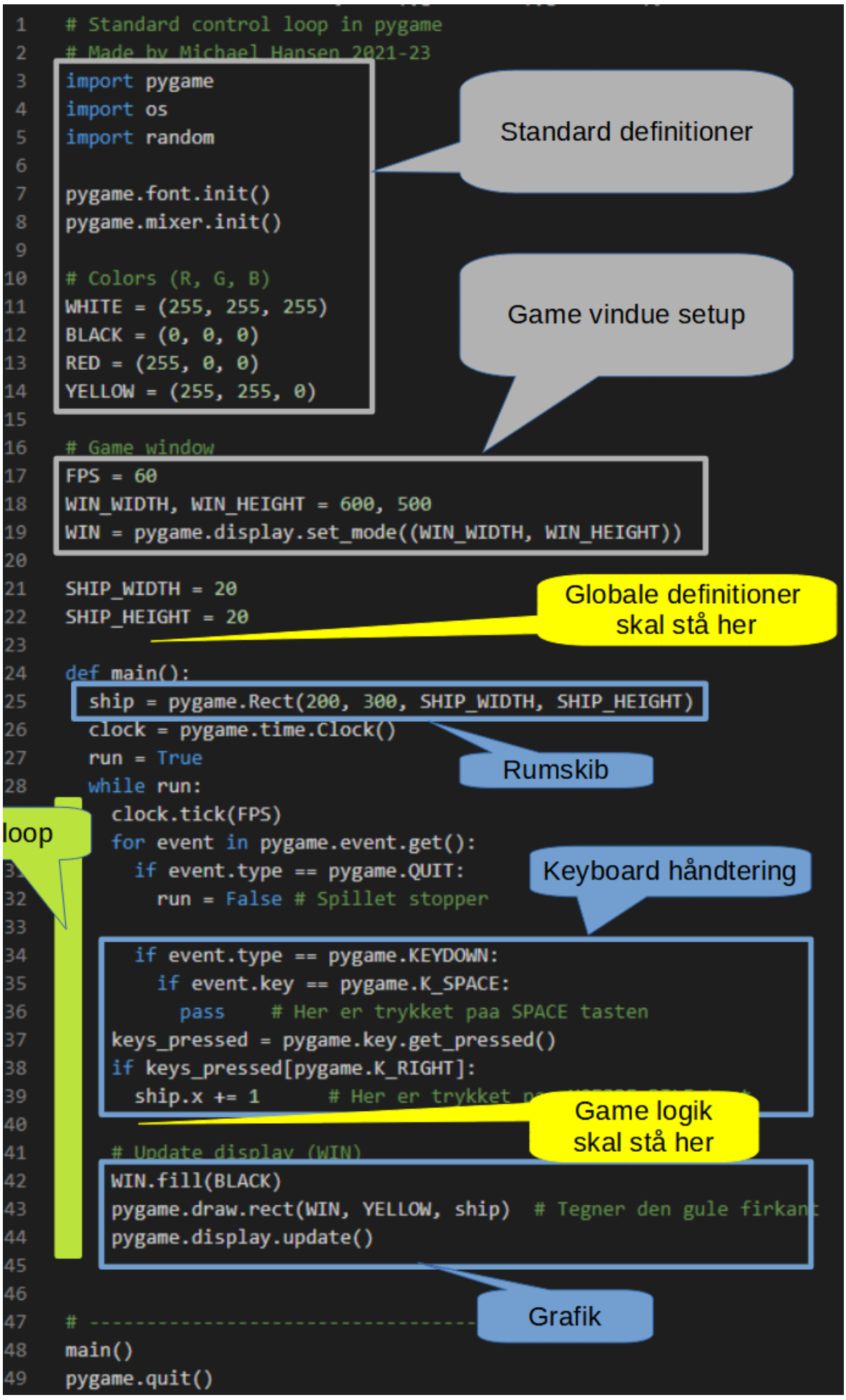
```
python3 space_inv.py
```

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. Luk vinduet igen. Det er vigtigt at lukke vinduet efter brug, da man ellers ikke kan starte spillet igen næste gang).

3.1.1 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!


Standard definitioner
Game vindue setup
Globale definitioner skal stå her
Rumskib
loop
Keyboard håndtering
Game logik skal stå her
Grafik

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
```

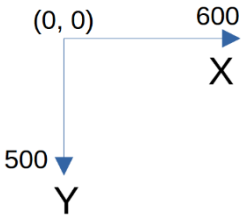
Følgende linje flytter rumskibet en til højre

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

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.

3.2 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)**

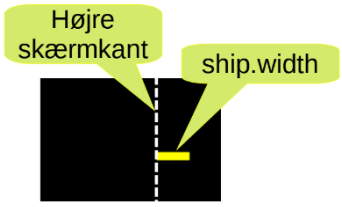
Hvis man hellere vil bruge tasterne A og D så hedder de: `K_a` og `K_d`.

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.

3.3 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. Bruge 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.

3.4 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 gange, 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.

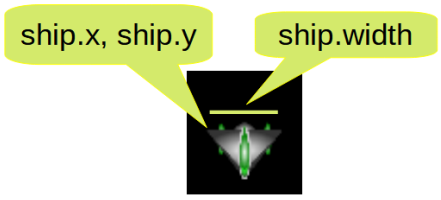
5. Kør spillet og se hvad der sker.

Der er flere ting galt: Skuddene bevæger sig ikke, men stå 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 slettet 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.

3.5 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. Tilføje 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 tættes 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`.)

3.6 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.colliderect(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.colliderect(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.colliderect(ship):
```

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

3.7 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.

3.8 Næste skridt

Der er mange måder at udvide spillet på – alt lige fra små tempo ændringer til flere fjender og bedre eksplosioner. De følgende ideer kan laves i vilkårlig rækkefølge, men de letteste står først.

3.8.1 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 ad skærmen.
Hint ændrer `alienMoveX` over tid.

3.8.2 Point tæller

Man kan tilføje 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.

3.8.3 Aliens skyder igen

Man kan få 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 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 antallet af skud ved at ændre på `10` og `5` tallene.

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.
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.

3.8.4 Flere ideer

- Når spillet er slut så skriv 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.
- Flere rækker af aliens
  Tilføj en linje mere inde i den loop som opretter alien'sne, men med en anden y værdi.
- Tilføj et baggrundsbillede. Brug følgende linjer:
  `BACKGROUND_IMAGE = pygame.image.load('background.png')`
  `WIN.blit(BACKGROUND_IMAGE, (0, 0))`

4 Avanceret

4.1 UFO'er

Lav nogle UFO'er, som flyver lidt mere tilfældigt rundt og som målrettet skyder efter os.

4.2 Eksplosionsanimationer

Lav en eksplosionsanimation, når en alien bliver ramt.

5 Frivillig info

På github ligger der en fuld version af Space Invarders 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 Inverders 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.

6 Links

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