Synopsis eksamen programmering C

Indledning

I projektet vil jeg lave retro spillet “Space invaders”.
Space invaders er et klassisk spil fra 1978, som er relativt simpelt.
Det går ud på at man skal nå at skyde nogle rumvæsner før de rammer bunden af skærmen. Rumvæsnerne kan selv skyde, og de rykker sig horisontalt på skærmen medmindre de rammer kanten af skærmen så rykker de en gang ned og fortsætte i den modsatte retning.
Jeg har valgt at have et benspænd som er at alt grafik skal være lavet med kode, så jeg må derfor ikke inkludere billeder. Derudover vil jeg holde mig til den gamle “tickrate” (opdateringshastigheden) på 2 gange per sekund. Dog ikke for spilleren ellers føles spillet for langsomt.

Listen af ting der skal laves:

- I forhold til spillets User Interface (udseende og oplevelse)
  Jeg skal finde ud af at tage sprites fra et sprite sheet, og en smart måde at konvertere tekst og tal om til billeder. Det bruges til at lave HUD elementerne.
- I forhold til spiller, fjender og bullets
  Jeg skal finde ud af at lave spillet i samme pixel stil som det var dengang. Og hvordan man får spillet til at køre med den valgte tickrate uden at sætte framerate i p5 ned.
- I forhold til spillets gameplay
  Der skal laves 3 classes; Player, Enemies og Bullets.
  Player:
  Spilleren skal kunne rykke sig horisontalt, uden at gå ud over kanten.
  Spilleren kunne skyde.
  Spilleren skal være tegnet med pixels (bruger rect funktionen)
  Enemies:
  Fjenderne skal kunne rykke sig horisontalt men når de rammer en væg (kanten af skærmen) skal kollektivt rykke sig en gang ned (på y aksen) og derefter fortsætte i den modsatte x retning.
  De skal kunne skyde.

Til sidst skal de tegnes med pixels og skifte mellem de 2 frames fra original tegningerne.
Hvis de rammer bunden af skærmen skal de fjerne et liv fra spilleren, og derefter slette sig selv.
Bullets:
Bullets er delt op i 2, så spilleren og fjender har 2 forskellige bullet versioner.
Spilleren har en type skud, som skal gå opad. Her skal den kun tjekke om den rammer en fjende.
Fjenderne har 2 typer skud, de skal begge gå nedad og kun tjekke om de rammer spilleren.
Der er 2 forskellige looks til fjendernes skud. Der er en som ligner et lyn (her skal den tegnes med en pixeleret sinus bølje som pixelseres) og den anden som ligner et kors hvor den horisontale streg løber op og ned af skuddet.
skuddene skal slette sig selv hvis de rammer enten bunden (for enemies skud) eller toppen (for player skud) af skærmen.

Programmeringsmiljø

Hvad er Javascript?

Html

Html er det basiske sprog som sørger for at der er ting på skærmen. Det står for at man fx kan lave en knap eller noget tekst. Funktioner er meget begrænset i html og generelt bruger man det ikke til funktionshåndtering eller noget logisk.

CSS

CSS står kun for styling af html. Som gør at fx en knap ser flot ud, eller noget tekst.

JavaScript

JavaScript bruges til mange ting men kort fortalt er det en funktionelt sprog modsat css og html, som giver udviklere en nem måde at lave funktioner, variabler og meget mere. Det bruges ofte til fx at lave modulære hjemmesider, eller få en knapper til at have reele funktioner. I vores tilfælde bruges det sammen med P5.js biblioteket som er rettet mod at lave spil.

Grundlæggende konstruktioner i JavaScript

Variabler

- Hvad er en variabel?
  En variabel er data der gemmes med et navn. Så hvis vi fx har en spiller, så kan vi sige at spillerens x koordinat er en variabel da det er noget der kan gemmes, læses og manipuleres. Dog er det ikke alle variabler der kan manipuleres, som fx const x = 10; her står const for constant som altså betyder at den ikke må ændres.
- Hvilke typer primitive variabler findes der?
  boolean:
  1/0, true/false. Dette er den mest primitive variabel der findes. Det er den der står for at logic gates i computeren fungere. Det er her hvor vi spørger om noget er rigtigt eller falsk.

  string:
  Det er simpelthen bare tekst, der kan man ikke indsætte tal og manipulere med tallene.

  integer:
  Dette er et integer, som betyder at det er et helt tal. Her kan man ikke skive kommatal.

  float:
  Dette er et kommatal. Man kan også skrive hele tal men man bruger generelt float til beregninger da det er mere præcist end int.

```
// Dog kan der forekomme rounding fejl med float som fx:
console.log(0.1+0.2) // 0.30000000000000004
```

  Character:
  Char er den numeriske værdi for tegn såsom bogstaver, tal og særlige tegn fx {(/ osv.

- Kodeeksempler med de forskellige typer af primitive variabler

```
function objCollision(obj1, obj2) {
  if (obj1.x+obj1.w >= obj2.x && obj1.x <= obj2.x+obj2.w) { // horizontal collision
    if (obj1.y+obj1.h >= obj2.y && obj1.y <= obj2.y+obj2.h) { // vertical collision
```

```
      return true
    }
  }
  return false
}
```

I disse if statements bruger vi integer variabler til at se om et objekt er røre et andet objekt.
Vi returnere en boolean værdi da vi skal vi om de røre hinanden eller ej.

Arrays

- Hvad er et array?
  Et array er en liste med variabler.
  Eksempel på en variabel:

```
food = ["banana", "pineapple", "bread", "cheese"];
```

For at søge i et array og fx. få den første værdi skal man skrive:
food[0]
Vi ser altså at et array indexes fra 0.
Problemet med arrays er at (i et low level sprog), er det svært at fjerne den allokerede data, her kan man bruge linked lists, som har en værdi og en key til den næste variabel.

- Hvordan opretter indsætter og fjerner man elementer fra et array?

```
// opret et tomt array
food = [];
// indsæt
// arrayNavn.push(variabel);
food.push("honey");
// fjern
// arrayNavn.splice(index, antal der skal fjernes);
food.splice(0, 1);
```

- Giv et kodeeksempel hvor du har brugt arrays

```
function setEnemySprites() {
  // Vi laver et array af enemy sprites ud fra typen via given spritesheet
```

```
  // enemySprites[ typen af enemy ] = [spritesheet.get(x, y, w
idth, height) for første frame, det samme for andet frame]
  enemySprites[0] = [img.get(39, 1, 12, 7), img.get(39, 11, 1
2, 7)]
  enemySprites[1] = [img.get(22, 1, 11, 7), img.get(22, 11, 1
1, 7)]
  enemySprites[2] = [img.get(5, 1, 8, 7), img.get(5, 11, 8,
7)]
}


class Enemy {
  ...
  show() {
    fill(0, 255, 0)
    //rect(this.x, this.y, this.w, this.h) // brugt til debug
for at se dimentioner
    // Normal modulo bruges ikke da counter vil forstyrre det
viste sprite
    // Counter vil altså tælle op 60 gange i sekundet og derfo
r skifte frames for hurtigt
    this.imgCounter++
    if (this.imgCounter > this.imgMod) {
      this.imgCounter = 0
      this.imgNum = (this.imgNum+1) % enemySprites[this.type].
length
    }
    image(enemySprites[this.type][this.imgNum], this.x, this.
y, this.w, this.h)
  }
  ...
}
```

JSON

- Hvad er JSON objekter?
  Et JSON object er som navnet JavaScript Object Notation. Altså er det en måde at gemme data på, som er dynamisk. Her har vi en variabel som er JSON objektets navn. Under det kan man tilføje atributter til det object. Det kan fx. være player.x, hvor det første der skrives er navnet på variablen "player", og derefter ".x" som altså er spillerens x koordinat. Grunden til at det er godt at bruge json er at man kan genbruge variabel navne, og det gør det nemmere at generalisere til hvis man fx. vil lave en funktion. Så skal den funktion jo bruge et generaliseret variabelnavn.
- Giv et kodeeksempel hvor du har brug et JSON objekt
  Da jeg laver koden objekt orienteret bruger jeg classes som også er en form for JSON, dog ikke skrevet på samme måde. Til andre projekter som jeg har lavet (fx. Boks App) bruger JSON som databehandling. I eksemplet under vises et eksempel på hvordan man kan få dummy data via JSON:

```
// Get data (fetch)
fetch('https://www.boredapi.com/api/activity')
.then(res => res.json()) // vent på response (.then) og omdan til JSON
  .then(json => { // Når det er gjordt log svaret
    console.log(json)
})
// console.log(json) giver os:
activity: "Write a short story"
type: "recreational"
participants: 1
price: 0
link: ""
key: "6301585"
accessibility: 0.1
```

Betingelser

- Hvad er en betingelse i programmering?
  En betingelse i programmering er ligesom at stille et spørgsmål. Altså HVIS du står på højre side af skærmen, så ryk spilleren tilbage på midten.

if (player.x > width) {
player.x = width/2;
} Her ser vi altså at der bliver spurgt et spørgsmål og hvis det er korrekt så køre den koden.

- Hvorfor kaldes en betingelse for et "boolean expression"?
  Grunden til at en betingelse kaldes for en boolean expression, er fordi det kan kun besvares som enten korrekt eller forkert. Der er ikke noget midt i mellem. Boolean er som forklaret før true or false, 1 or 0. Det kan kun være en af delene.

- if..else - beskriv hvordan betingelser kan udvides med else statements
  Grunden til at man vil bruge et else statement er fordi at man vil køre noget kode hvis den betingelse ikke opfyldes. Altså som i eksemplet før kan vi sige at HVIS spilleren er over skærmens længde så rykkes den tilbage på midten af skærmen. ELLERS skal spilleren rykke til højre.

```
if (player.x > width) {
    player.x = width/2
} else {
    player.x++
}
```

- Hvilke **Numeriske Operatorer** findes der, og hvordan bruges de i boolean expressions?
  De numeriske operatorer er dem som der bruges i matematik, altså plus (+), minus(-), gange (*), dividere (/), modulo (%), ekponentiel (**). De bruges til at udvide boolean expressions så der ikke skal laves variabler for hvert if statement.

```
if (player.x + player.w > width) {
    player.x = width/2
}
```

  Her tager vi altså spillerens x koordinat plus spillerens bredde og ser om det er mere end skærmens længde.

- Hvilke **Logiske Operatorer** findes der, og hvordan bruges de i boolean expressions?
  Der er and, or, not. Som generelt bruges, dog er der også nand og nor.
  And gate

værdi 1	værdi 2	resultat
0	0	0
1	0	0

0	1	0
1	1	1

or gate

værdi 1	værdi 2	resultat
0	0	0
1	0	1
0	1	1
1	1	1

not gate er bare om værdien er forkert. altså er resultatet bare omvendt værdien.
nand gate er når enten begge værdier er korrekte eller forkerte

værdi 1	værdi 2	resultat
0	0	1
1	0	0
0	1	0
1	1	1

and operationen bruges til at spørge om begge operationer er korrekte.
or operationen bruges til at spørge om enten den ene eller den anden er korrekt.
nor operationen bruges til at spørge om begge operationer er forkete.
nand bruges til at spørge om begge operationer er forkerte, om en af dem er rigtige men ikke begge på samme tid.

- Kodeeksempel - giv mindst to eksempler på betingelser og beskriv hvilket problem de løser
  Enemies skal IKKE opdateres hvis spillet er slut. Her bruger vi not gate.

```
class Enemy() {
  ...
  update() {
    if (game.gameState != "game over") {
      this.move()
      this.show()
```

```
      this.shoot()
    }
  }
  ...
}
```

Spilleren skal skyde hvis der trykkes på spacebar OG spilleren må skyde. Her bruger vi and gate.

```
class Player() {
  ...
  // SHOOTING
  if (keyIsDown(32) && this.canShoot) { // Key 32 = spacebar
    this.canShoot = false
    bullets[bullets.length] = new Bullet(this.x+(this.w/2), th
is.y, -1)
  }
  ...
}
```

Loops

- Hvad er et loop - hvad er det de kan gøre nemmere når man programmerer?
  Et loop bruges når der skal køres det samme kode flere gange.
  Det er for eksempel hvis man skal opdatere alle bullets, så bruger man et for-loop som vist i eksemplet.
  - Kodeeksempel - while-loop

```
while(game.state != "paused") {
  game.run()
}
```

  - Kodeeksempel - for-loop

  bullets.length siger hvor mange elementer der er i arrayet bullets

```
for (var i = 0; i < bullets.length; i++) {
  bullets[i].update()
  if (bullets[i].checkCol()) {
      bullets.splice(i, 1)
```

```
      waveUpdater() // Check if we need to update the wave onl
y when someone dies
  }
}
```

Funktioner

- Hvad er funktioner?
  En funktion i programmering er lidt det samme som i matematik. I matematik vil vi skrive en funktion som for eksempel f(x) = 2x, hvor vi har et input x og får et output y. Det er også det vi gør med programmering, det skrives dog anderledes. I JavaScript vil vi skrive den samme funktion således:

```
function f(x) {
  return 2*x // returnere med det samme
}
// Eller
function f(x) {
  var y = 2*x
  return y // det gør det samme
}
```

  Dog kan vi i programmering også få funktioner til at returnere mere end i konventionel matematik.
  Vi bruger ofte funktioner som indirekte returnere noget, fx:

```
class Enemy {
  ...
  move() { // da det er en funktion i en class skal man ikke s
krive function først
      this.x += enemyDir
  }
  ...
}
```

  Her returneres der ikke noget dog ændre det stadig data.
  Funktioner er smarte da de kan genbruges (hvis de er skrevet korrekt!)
  Funktioner kan også have default værdier hvis der ikke er givet argumenter.

```
function f(x=2) {
  return 2*x
}
console.log(f()) // 4
console.log(f(2)) // 4
console.log(f(3)) // 6
```

- Beskriv følgende:
  - Navngivning af funktion
    Når man laver en funktion skriver man function og derefter navnet.

```
function navnPåFunktion() {
  ...
}
```

  - Parametre og argumenter i funktioner
    Med det tideligere eksempel f(x) = 2x

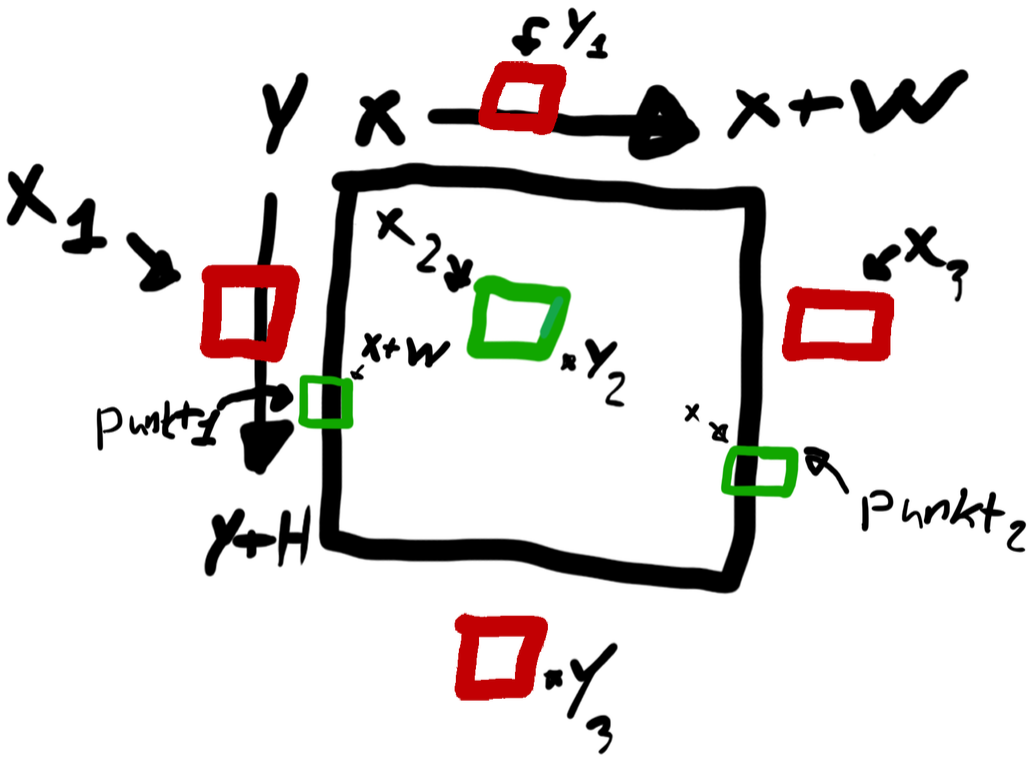
```
function f(x) { // Her er x et parameter
  return 2*x
}
for (var x = 0; x < 10; x++) {
  console.log(f(x)) // Her er x et argument
}
```

    Et parameter er det funktionen er defineret med.
    Et argument er det man giver funktionen som input.
  - Return-value i funktioner
    En funktion kan returnere alle variabel typer.
    I den funktion vist før returnere den et integar. Men det er ofte boolean værdier. Her skriver man return true. Men funktioner behøver nødvendigvis ikke at returnere noget (som vist i move()).

- Kodeeksempler hvor du har brugt funktioner i din egen kode

Her er et eksempel på en funktion. Vi skal tjekke om objekter kollidere (bullet og andet objekt), i stedet for at skrive kollisionskoden for fjender og derefter spilleren kan vi lave en samlet funktion. Denne funktion tager 2 parametre, det er de 2 objekter. Derefter tjekker den om det første objekt er indenfor det andet objekt.

```
function checkColObj(obj1, obj2) {
  if (obj1.x + obj1.w >= obj2.x && obj1.x <= obj2.x + obj2.w)
{ // Check x
    if (obj1.y + obj1.h >= obj2.y && obj1.y <= obj2.y + obj2.
h) { // Check y
      return true // Objekterne kollidere
    }
  }
  return false // Objekterne kollidere IKKE
}
```



Tegning af checkColObj

Jeg har prøvet at forklare hvad der sker med en tegning, hvor den sorte boks er en fjende, og de grønne (og røde) bokse er skud.
Når vi kigger på den første linje af koden i funktionen ser vi:

```
if (obj1.x + obj1.w >= obj2.x && obj1.x <= obj2.x + obj2.w) {
// Check x
```

Der kollidere punkt 1 med fjenden fordi at dens x + w (width) er mere end fjendens x koordinat. Dens x også mindre end fjendens x + w. Derfor kollidere den på x-aksen. Og det samme tjek er for y-aksen.

Hvad er P5.js?

P5.js er et bibliotek til javascript.
Det er lavet til at køre på et HTML canvas element.
Det er smart fordi det gør det nemt at lave spil, visualiseringer, og meget mere. Vi bruger det til at lave spil, her gør vi brug af mange af de indbyggede elementer, og funktioner.
For eksempel har den funktionen setup. Det er en funktion som kun køre en gang. Det er smart fordi så kan vi definere variablernes værdier inde i den funktion, og så bliver de ikke definere 60 gange i sekundet. Funktionen draw køres 60 gange i sekundet. Det er et loop der kun slutter når vi stopper programmet. Her har vi alle objekters update funktioner og tegner baggrunden. Vi bruger også preload funktionen til at load billeder som er smart da det nogle gange tager noget tid at load billeder fra memory eller servere, derfor køre resten af koden først efter preload funktionen.

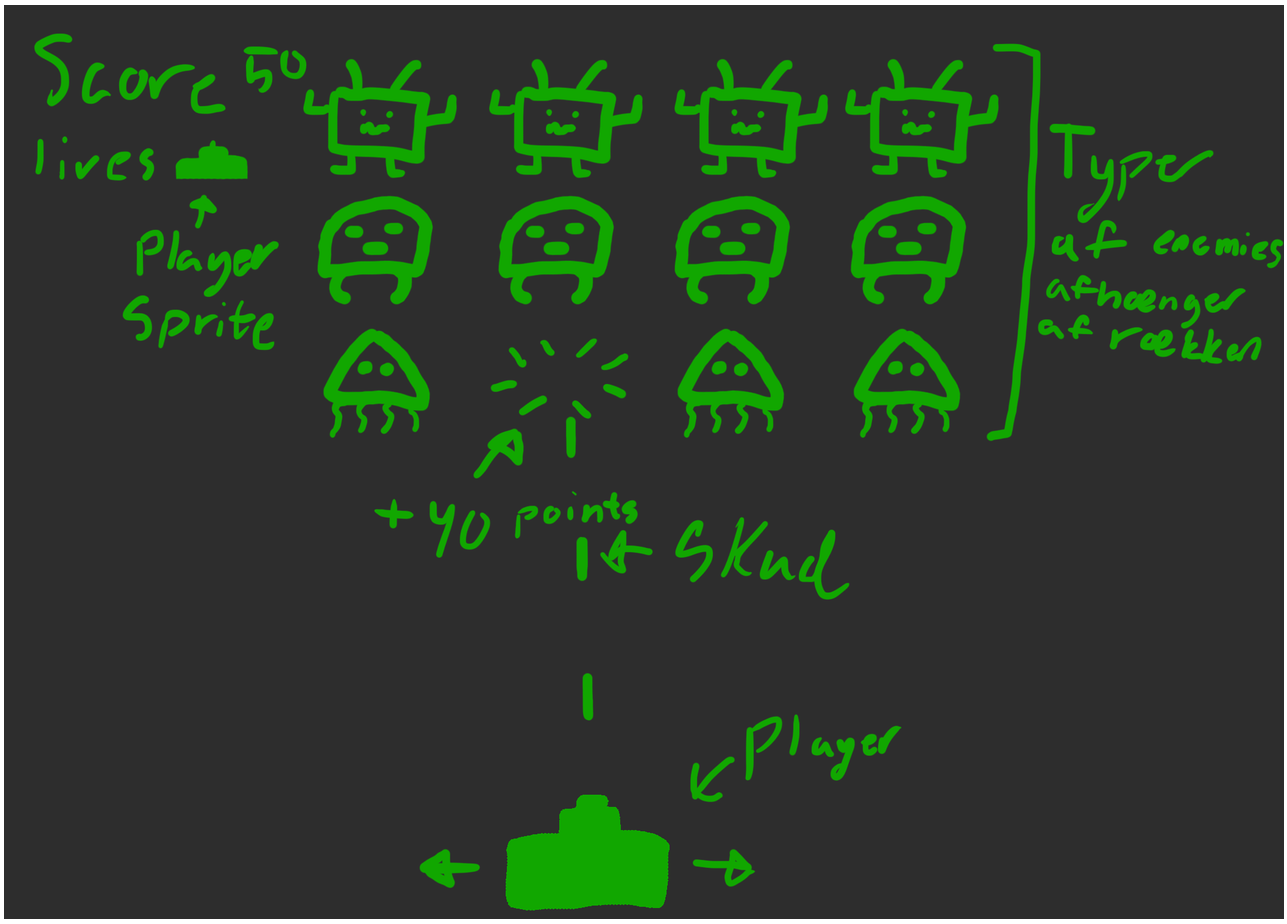
Produktudvikling - forberedelse

Jeg vil lave det klassiske spil "Space Invaders", da spillet originalt er til arkademaskiner har jeg taget udgangspunkt i en replika fra "freeinvaders".
Det er for at se hvordan spillet fungere og så kan jeg derefter lave det selv.
Der er også brugt et sprite sheet fra "The Spriters Resource"

Link til min kode (ellers download i billag):
https://editor.p5js.org/Rufluenza/sketches/RPdxU9PyM

Jeg startede med at lave spillet hvor alt blev tegnet med pixels, dog blev det hurtigt langsomt at køre da P5.js ikke er det mest optimale sprog. For at give den gamle effekt af arkade maskine versionen havde jeg også sat spillet til at køre med 2 FPS (frames per second) men det føles også for langsomt. Derfor har jeg valgt at starte fra ny, hvor jeg bruger sprites og 60 FPS.

Papirskitser

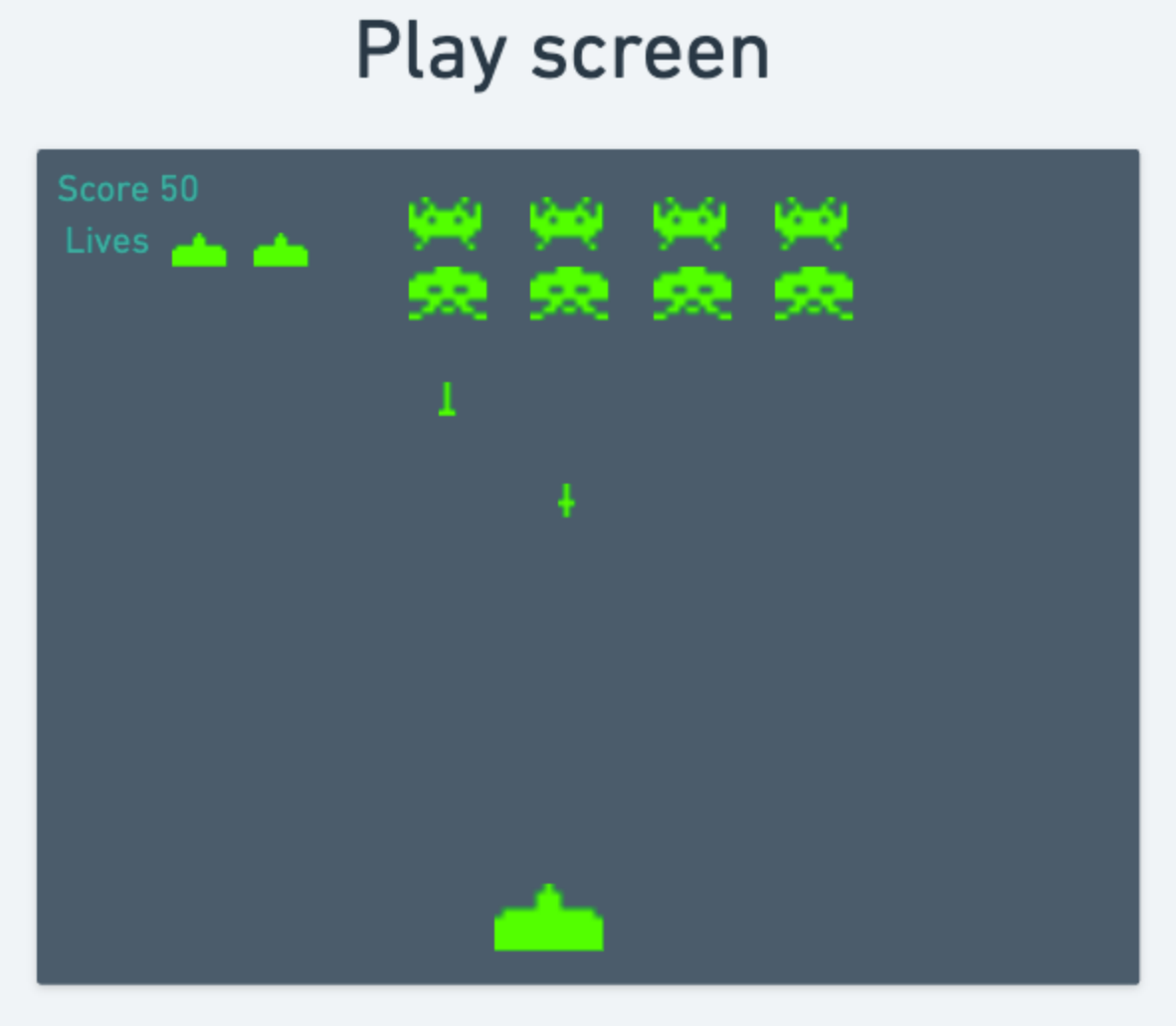


Tegning af gameState "Play"

Jeg har tegnet hvordan det vil se ud når spillet er igang. Spilleren skal kunne bevæge sig til højre og venstre, samt at kunne skyde. Jeg har også tegnet alle typer af fjender, bemærk at de skal være på samme række. Så er der 2 HUD elementer; score som viser hvor mange point man har. De afhænger af hvilke typer fjender man skyder, da de har forskellige point. Og lives, som viser spillerens sprite for hvert liv man har tilbage. Det var svært at vise på en pæn måde på tegningen men alle fjenderne bevæger sig i samme retning, med samme hastighed.

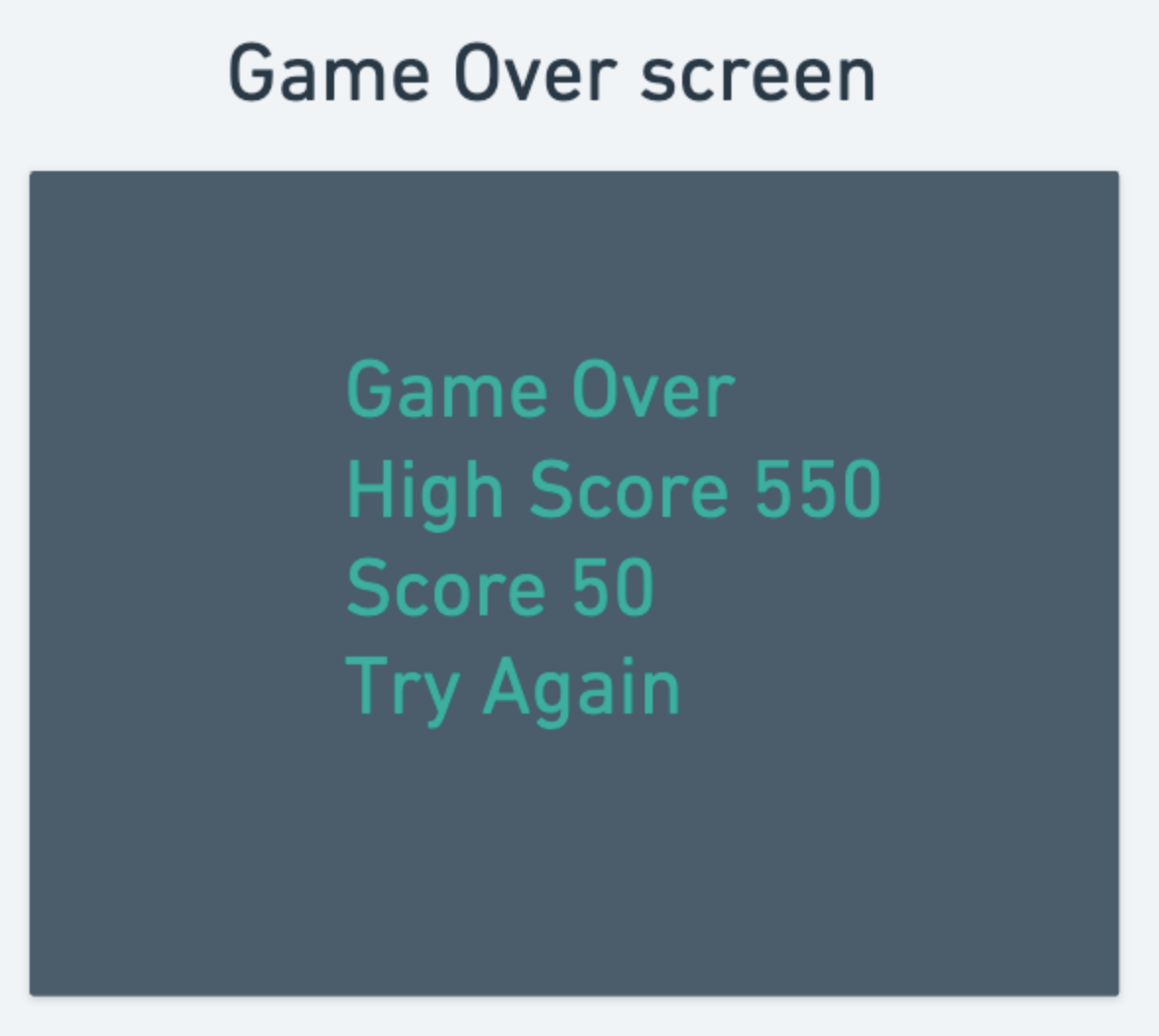
Wireframes

Spillet er delt op i 2 game states, play og game over.



Wireframe af Play Screen

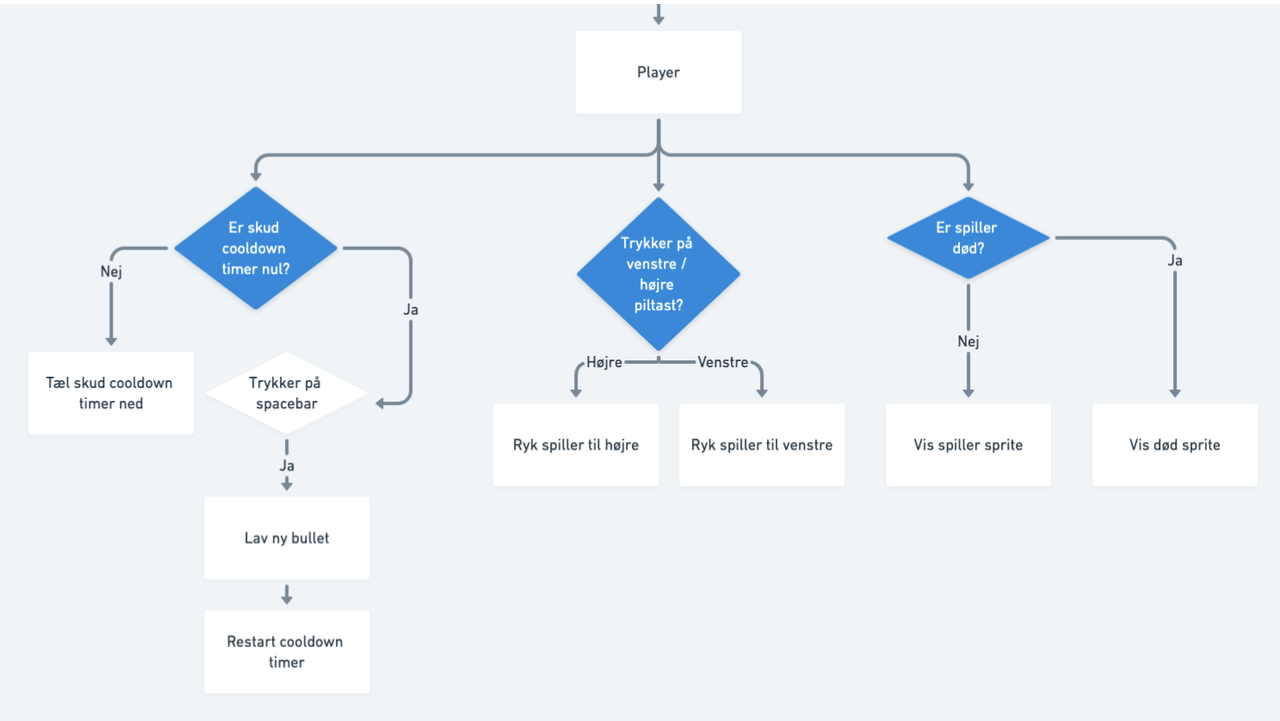
Der skal være 2 tekst elementer, som skal være i venstre hjørne af skærmen.
Score skal vises med tekst og tal sprites fra et spriteSheet (The Spriters Resource, 2021).
Lives skal også vises med tekst sprites dog skal der også vises de antal liv spilleren har med spillerens egen sprite. Fjenderne skal vises i et matrix, og de skal alle bevæge sig i samme retning. De kan skyde hvert 10. sekund plus et tilfældigt tal fra 0 til 10 for at de ikke alle skyder på samme tid.
Spilleren kan kun bevæge sig horisontalt.
Spilleren har 2 forskellige skyde mekanikker;
Når player.shootMechanic er "time" er den tidsbaseret, hvor der er en cooldown timer der tæller ned.
Når player.shootMechanic er "original" kan spilleren skyde når spillerens skud har ramt enten toppen af skærmen eller en fjende.



Wireframe af Game Over

Når spilleren dør vises game over skærmen. Her er der 4 tekst felter;
"Game Over", "High Score + {tal}", "Score + {tal}", "Try Again".
High score tallet tjekker først om scoren er højere end den tideligere high score, hvis den er så vis den nye score, ellers vis den gamle. Score viser spillerens score for det sidste spil, og try again er en knap som vises som tekst, her bruges funktionen showText til både at vise teksten og få dens dimensioner.

Flowcharts
For at få en grundig forståelse for hvad der skal laves, er det en god idé at lave et flowchart.
Programmet er objekt orienteret, derfor har jeg delt det op i classes.
Der er Player, Enemy, Bullet og Game

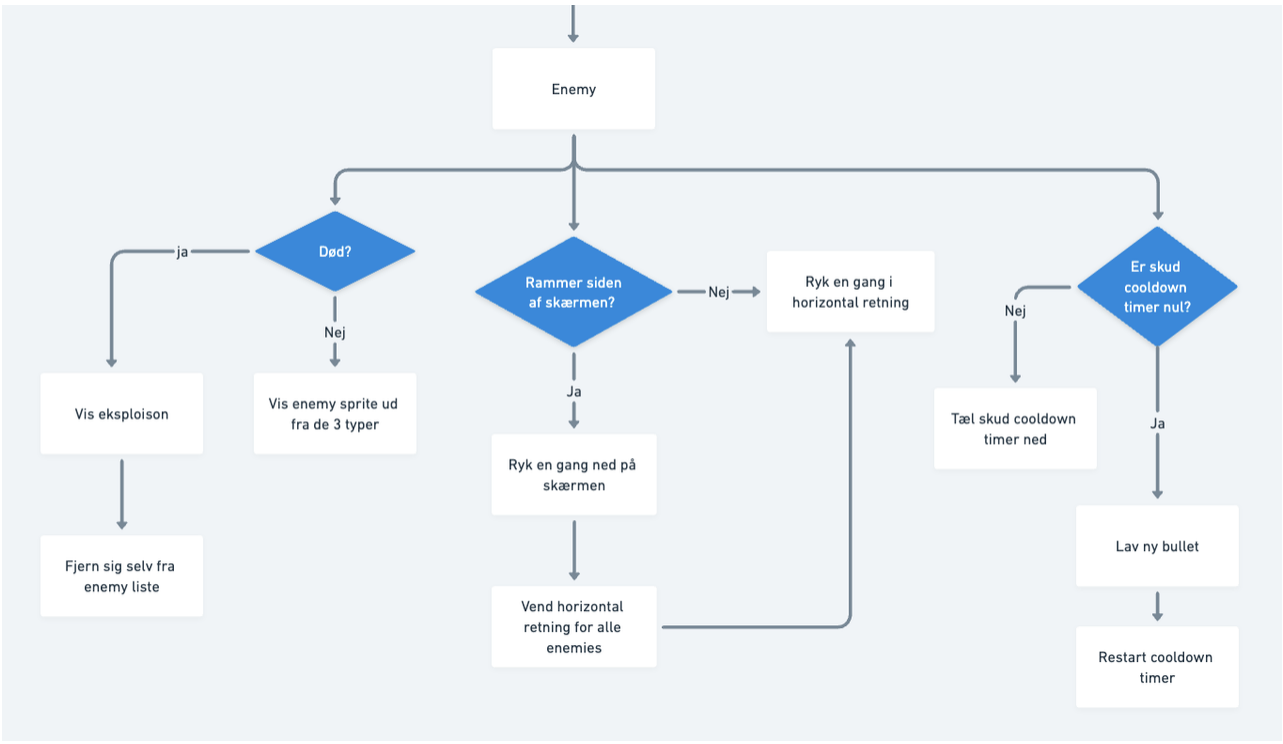


Player class

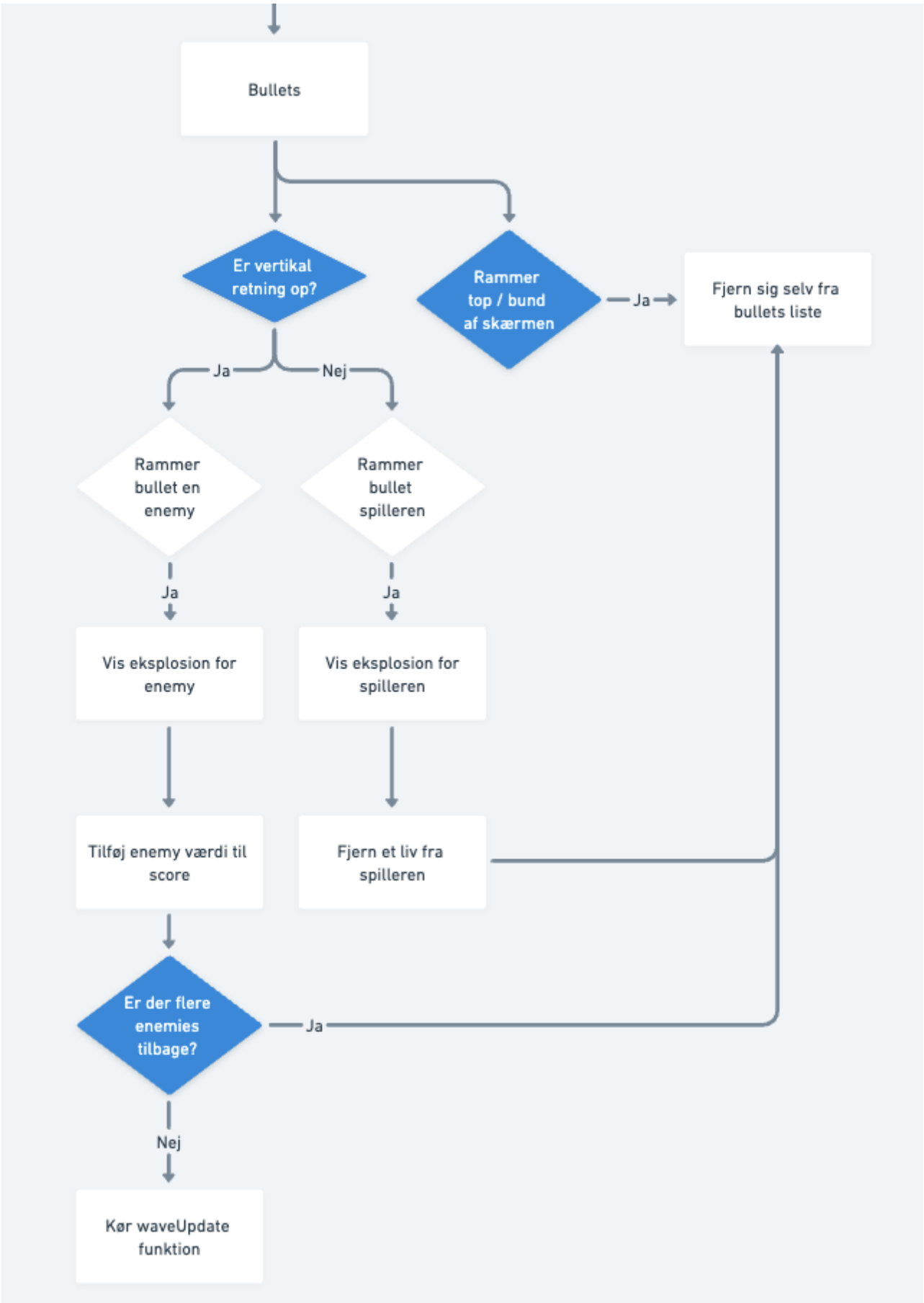
Spilleren har 3 funktioner;

- Move()
  Tager imod venstre og højre piltast, og rykker spilleren i den retning
- Shoot()
  Der er en cooldown timer der tæller ned hvis spilleren ikke kan skyde, hvis spilleren kan skyde og trykker på mellemrum skyder spilleren. Det gøres ved at tilføje et skud til bullets arrayet, med argumenterne; x + halvdelen af spillerens bredde, y, yDirection på -1.
- show()
  Hvis spilleren ikke er død viser den spillerens sprite, ellers viser den spillerens død sprite.

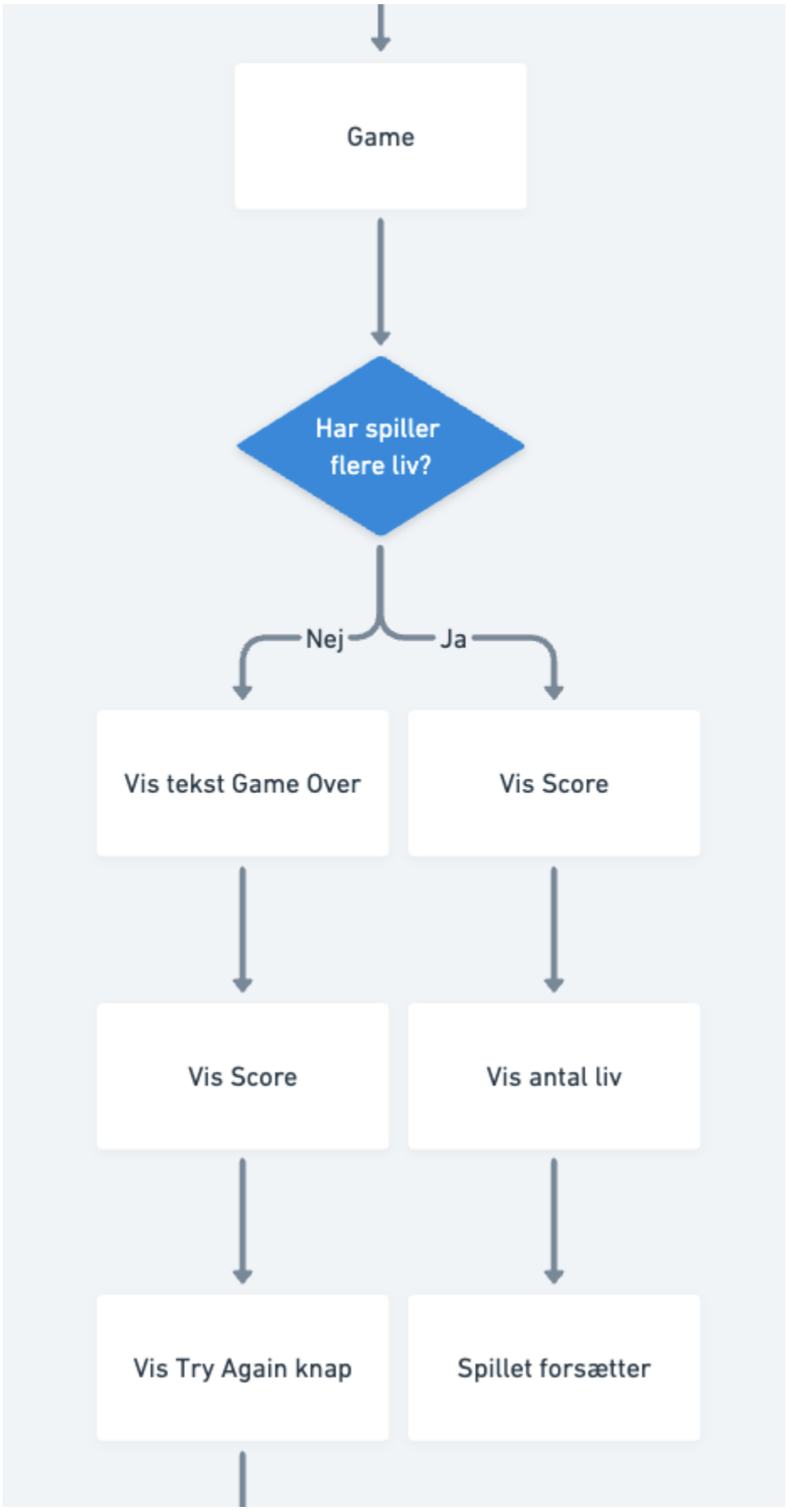
Den sidste funktion er update men den gør ikke andet end at køre alle de andre funktioner hvis spilleren har flere liv tilbage.

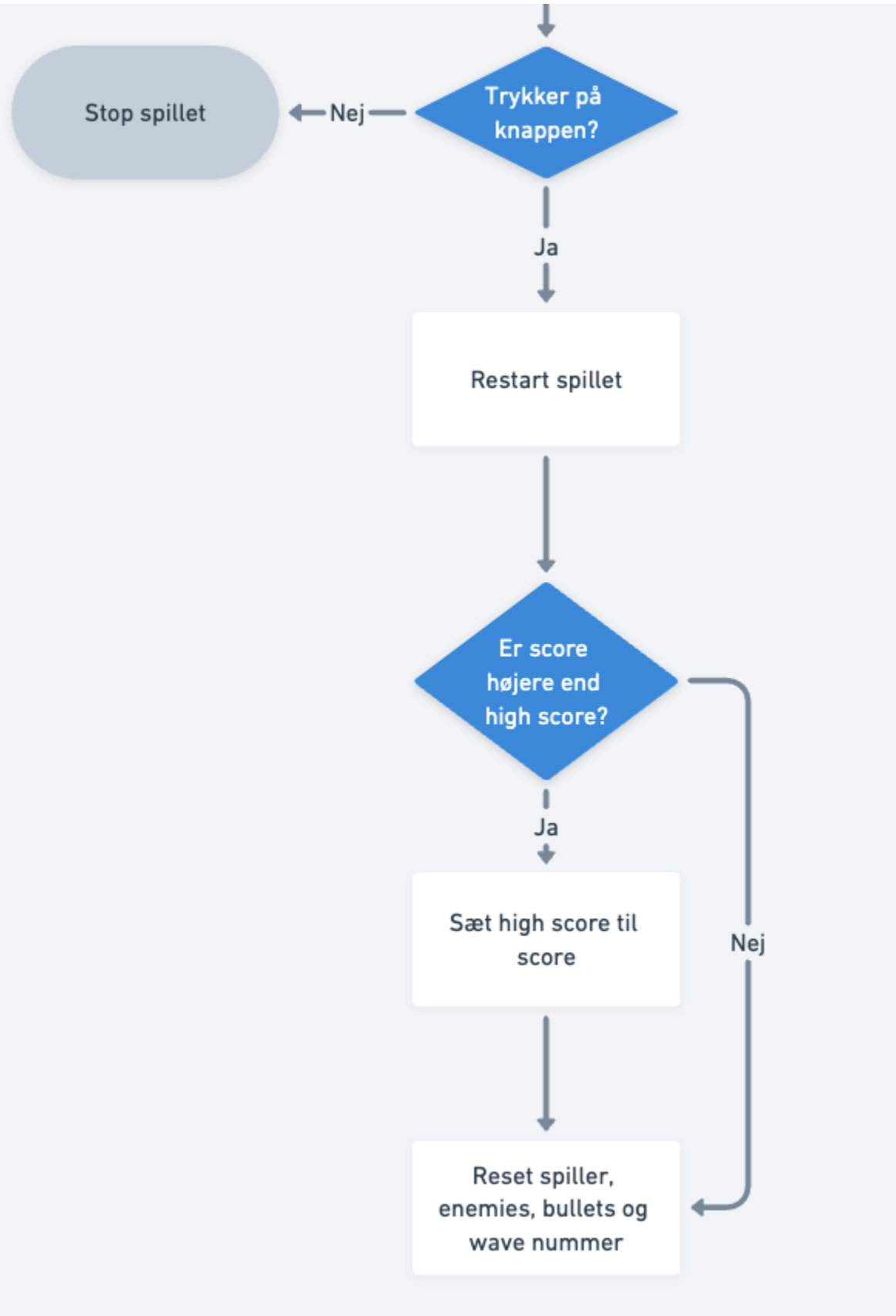

Enemy
Død?
ja
Nej
Vis eksploison
Fjern sig selv fra enemy liste
Vis enemy sprite ud fra de 3 typer
Rammer siden af skærmen?
Nej
Ryk en gang i horizontal retning
Ja
Ryk en gang ned på skærmen
Vend horizontal retning for alle enemies
Er skud cooldown timer nul?
Nej
Tæl skud cooldown timer ned
Ja
Lav ny bullet
Restart cooldown timer

Enemy class


Bullets
Er vertikal retning op?
Rammer top / bund af skærmen
Ja
Fjern sig selv fra bullets liste
Ja
Nej
Rammer bullet en enemy
Rammer bullet spilleren
Ja
Ja
Vis eksplosion for enemy
Vis eksplosion for spilleren
Tilføj enemy værdi til score
Fjern et liv fra spilleren
Er der flere enemies tilbage?
Ja
Nej
Kør waveUpdate funktion

Bullet class


Game
Har spiller flere liv?
Nej
Ja
Vis tekst Game Over
Vis Score
Vis Score
Vis antal liv
Vis Try Again knap
Spillet forsætter


Trykker på knappen?
Nej
Stop spillet
Ja
Restart spillet
Er score højere end high score?
Ja
Sæt high score til score
Nej
Reset spiller, enemies, bullets og wave nummer

Game class

Produktudvikling - kode

Jeg har valgt 3 eksempler på kode brugt i produktet. Der er kommentarer i koden samt beskrivelse. De eksempler jeg har valgt er henholdsvis:

- Funktion til at vise tekst
- Enemy wall collision
- Funktion til at opdatere waves

Funktion til at vise tekst

Noget interessant i min kode er hvordan der vises tekst.
Vi har 2 billede arrays, et for bogstaver (chars) og et for tal (sprNum).
Når der skal vises noget tekst bruger vi denne funktion, den tager imod teksten som en string samt x og y koordinater.
Vi starter med at lave et loop. Det gør vi fordi at vi skal vise alle bogstaver eller tal i teksten.
Derefter skal vi finde ud af hvilket billede der skal vises.
Det første problem består i hvordan vi skal kunne oversætte bogstaver til det korrekte billede i chars arrayet. Det løser vi ved at bruge den indbyggede funktion unchar. Den giver os tal værdien for bogstaver og tal. Unchar(“a”) er 97, unchar(“b”) er 98 osv, det gør at vi kan tage tekstens character værdi minus 97 for at få det korrekte bogstav fra arrayet med bogstavsbilleder.
Vi gør det samme med tal, hvor unchar(“0”) er 48, unchar(“1”) er 49 osv. Derfor kan vi nu sige at hvis vi har et tal skal det først omdannes til char værdien minus 48.
Det særlige er ved mellemrum som har værdien 32, hvis vi har et mellemrum vil vi altså ikke vise et billede da det er ligegyldigt, og derfor går videre til den næste character. Det får stadig loopet’s i værdi til at stige med 1, så en “tom” plads fylder noget.
Til sidst returnere vi dimentionerne som teksten har fyldt, det kan derefter bruges til fx knapper.

```
function showText(text, x, y) { // text = string, x og y er st
art koordinaterne til teksten
  for (var i = 0; i < text.length; i++) { // loop igennem hele
teksten
    if (unchar(text[i]) != 32) { // Hvis det ikke er mellemrum
```

```
      var shownImg = chars[unchar(text[i])-97] // -97 fordi ch
ar for "a" er 97
      if (unchar(text[i]) >= 48 && unchar(text[i]) <= 57) { //
hvis det er et tal
        // char 48 = "0", char 57 = "9"
        shownImg = sprNum[unchar(text[i])-48] // -48 fordi cha
r 48 = "0"
      }
      image(shownImg, x+game.cs*i, y, game.cs, game.cs) // vis
billedet
      // game.cs = character image size (8) * upscale amount
(3)
      // x koordinatet skal stige med start x + game.cs * anta
l af gange loopet er gennemgået
    }
  }
  return { // Bruges til fx "Try Again" knap
    "x": x, // start x
    "y": y, // start y
    "ex": x+game.cs*text.length, // start x + enkelt billede l
ængde * tekstens antal bogstaver eller tal (altså antal af bil
leder der er blevet vist)
    "ey": y+game.cs // start y + billede højde
  }
}

// Funktionen bruges i fx "Try Again" knappen, som laves af ga
me objektet
class Game {
  ...
  // TRY AGAIN BUTTON
```

```
  var tryAgainButton = showText("try again", x, height/2+this.
cs*4)
  if (mouseIsPressed == true && mouseButton == LEFT) {
  // Hvis spilleren trykker på left click
    if (mouseX >= tryAgainButton.x && mouseX <= tryAgainButto
n.ex) {
    // Samme kollisions tjek som før dog har musen ingen højde
og bredde.
    // tryAgainButton.ex er end x koordinat
      if (mouseY >= tryAgainButton.y && mouseY <= tryAgainButt
on.ey) {
        this.restart()
      }
    }
  }
  ...
}
```

Enemy wall collision

Som vist i flowdiagrammet for enemies, er der et if statement for om der er en fjende der rammer kanten af skærmen. I det rigtige space invader spil skifter alle fjenderne retning (og går en gang ned) så snart en enkelt fjende rammer kanten. Vi vil gøre det på samme måde da de ellers vil flyve ind i hinanden. Det vi gør er ret simpelt.
Jeg har delt det op i flere funktioner så det er nemt at se hvad der sker.

1. Det første vi gør er at køre funktionen enemyUpdateDir, det den gør er at loop igennem alle enemies, og køre vi funktionen checkWallCol
2. CheckWallCol er en funktion der skal tjekke om en fjende's x korrdinat plus dens bredde er mere end (eller lig) skærmens bredde, eller om dens x koordinat er mindre (eller lig) 0. Hvis det er korrekt returnere den true, ellers returnere den false.

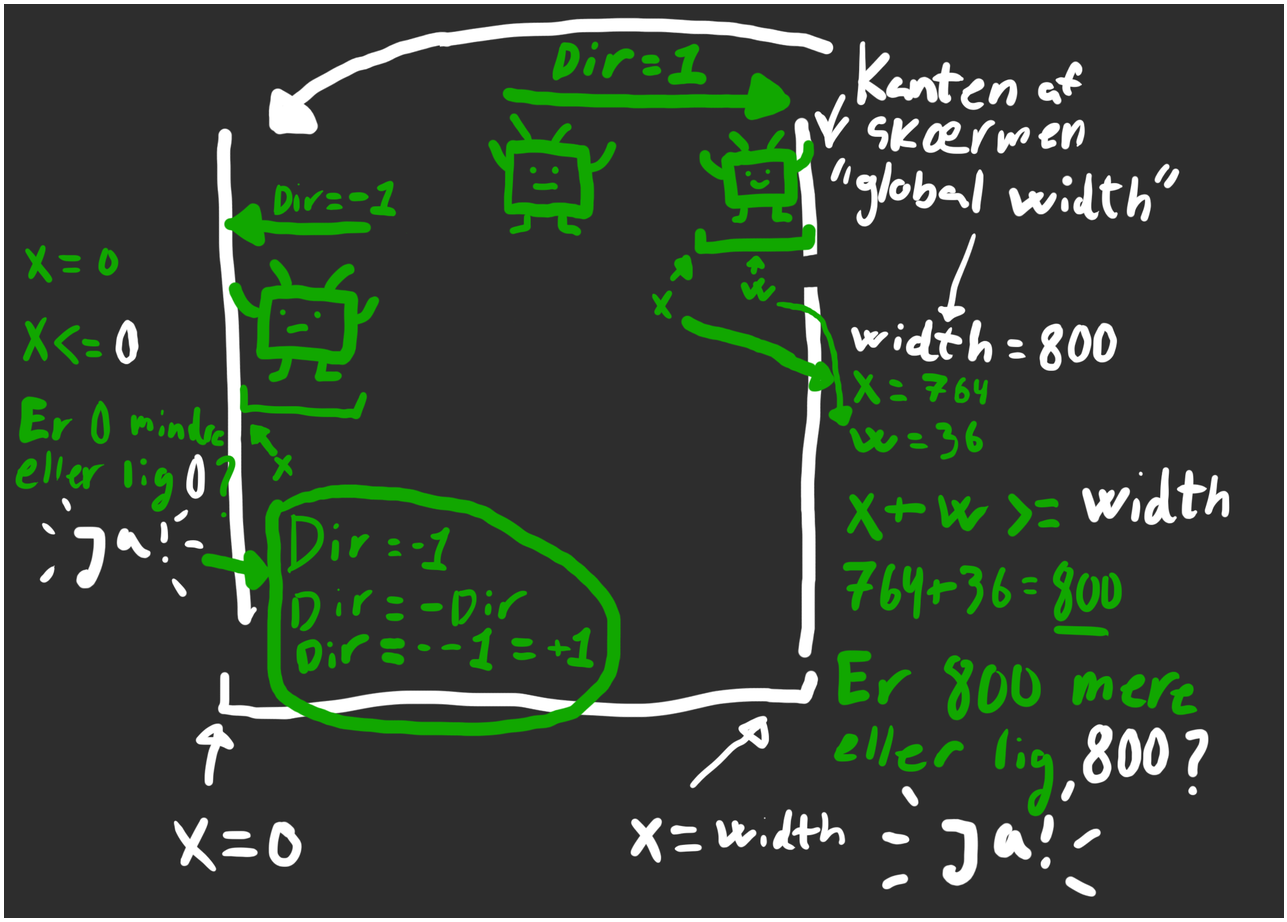
Grunden til at der ikke er brug for et else statement er fordi når der står "return" køre den ikke resten af koden, det trick bruges flere gange i koden.

3. Hvis funktionen returnere true køre vi funktionen enemyYUpdate som køre endnu et loop igennem alle fjenderne og rykker dem en gang ned på skærmen.
4. Til sidst returnere enemyUpdateDir enten den omvendte retning for fjenderne (hvis der er en der har ramt en væg), ellers returnere den enemyDir.

```
class Enemy {
  ...
  checkWallCol() {
    if (this.x + this.w >= width || this.x <= 0) { // hvis den
s x koordinat plus dens bredde er mere end (eller lig) skærmen
s bredde, eller hvis dens x koordinat er mindre end (eller li
g) 0
      return true // "jeg har ramt kanten"
    }
    return false // "jeg har IKKE ramt kanten"
  }
  ...
}
function enemyYUpdate() {
  for (var i = 0; i < enemies.length; i++) {
    enemies[i].y += enemyHeight // += er det samme som at skri
ve y = y + enemyHeight
  }
}
function enemyUpdateDir() {
  for (var i = 0; i < enemies.length; i++) {
    if (enemies[i].checkWallCol()) { // Kør enemies checkWallC
ol og få return værdien
      enemyYUpdate() // opdater y værdien for alle fjender
      return -enemyDir // vend retningen om, altså hvis den va
r 1 vil den være -1, og hvis den var -1 vil den være --1. dobb
elt minus er plus.
```

```
    }
  }
  return enemyDir // returner den eksisterende retning
}
class Game {
  ...
  enemyDir = enemyUpdateDir()
  ...
}
```

Her er en tegning af funktionen.



Tegning af funktionen checkWallCol

Funktion til at opdatere waves

Det næste kode vi skal kigge på er måden vi laver fjenderne.

Når spilleren har skudt alle fjenderne skal der komme en ny “wave” af fjender. Det gør vi med waveUpdater funktionen. For at minimere hvor mange gange funktionen

bliver kaldt, gør vi det kun i starten, ved restart og hvis der er et skud der rammer en fjende.
For at gøre det nemt bruger vi et matrix (2D array) i stedet for et "normal" (1D) array. Her er første værdien antal horisontale fjender og den anden værdi er vertikale. Det totale antal fjender bliver hor*ver (altså col * row i programmet).
Vi looper igennem matrixen, og tilføjer en fjende til enemies arrayet. Som forklaret før bruger vi enemies.length for at tilføje en fjende i enden af arrayet. Vi bruger også modulo, da vi gerne vil have at rækken har samme type. Da der er 3 typer af fjender (0, 1, 2) bruger vi row%3, så gentages typerne hvis vi har flere rækker end der typer af fjender.

```
let wave = [[4, 1], [7, 3], [8, 4]] // wave[0] = [4, 1] hvor 4 er antal fjender på x-aksen og 1 er antal fjender på y-aksen

function waveUpdater() {

  if (enemies.length == 0 && waveNum < wave.length) { // Er der ikke nogle fjender tilbage? Og tjek om vi ikke prøver at index en wave som ikke er lavet.

    for (var col = 0; col < wave[waveNum][0]; col++) { // Loop igennem antal fjender på x-aksen

      for (var row = 0; row < wave[waveNum][1]; row++) { // Loop igennem antal fjender på y-aksen

        enemies[enemies.length] = new Enemy(col*40+30, row * (enemyHeight+10)+20, row%3)

        // Enemy(x, y, type)

        // Vi bruger enemies.length for at sørge for at index arrayet i slutningen hvor der ikke er noget endnu.

      }

    }

    waveNum++

  }

}
```

Kildehenvisninger og bilag

Kildehenvisninger

- "Reference", P5.js, 2024:

https://p5js.org/reference/
- “Space Invaders”, The Spriters Resource, 2021: https://www.spriters-resource.com/arcade/spaceinv/
- “Free Invaders”, The free video games project, Paul Neave, 2023: https://freeinvaders.org/
- “JavaScript Key Code Event Tool”, Toptal, 2024: https://www.toptal.com/developers/keycode
- “Boks App”, Ruben Lau Sonnichsen, 2024: http://www.sonic-server.software (username og password: test)

Bilag

Det er vigtigt at I kigger på den nye version af spillet da det er den som refereres til i opgaven!

Den opdaterede version af spillet:

https://www.dropbox.com/scl/fi/p8eh3rkhfnffja348c7jd/Space_Invaders_NEW_2024_05_20_13_59_03.zip?rlkey=74uvery0alqtl5r83tuyjmkxj&dl=0

Den gamle version af spillet:

https://www.dropbox.com/scl/fi/zvzgynst1xiig7h1tk9k8/Space_invaders_2024_05_20_14_01_22.zip?rlkey=bl5174dmarq5x00m02az3sdw3&dl=0

Sprite sheet brugt til spillet, uden baggrund.