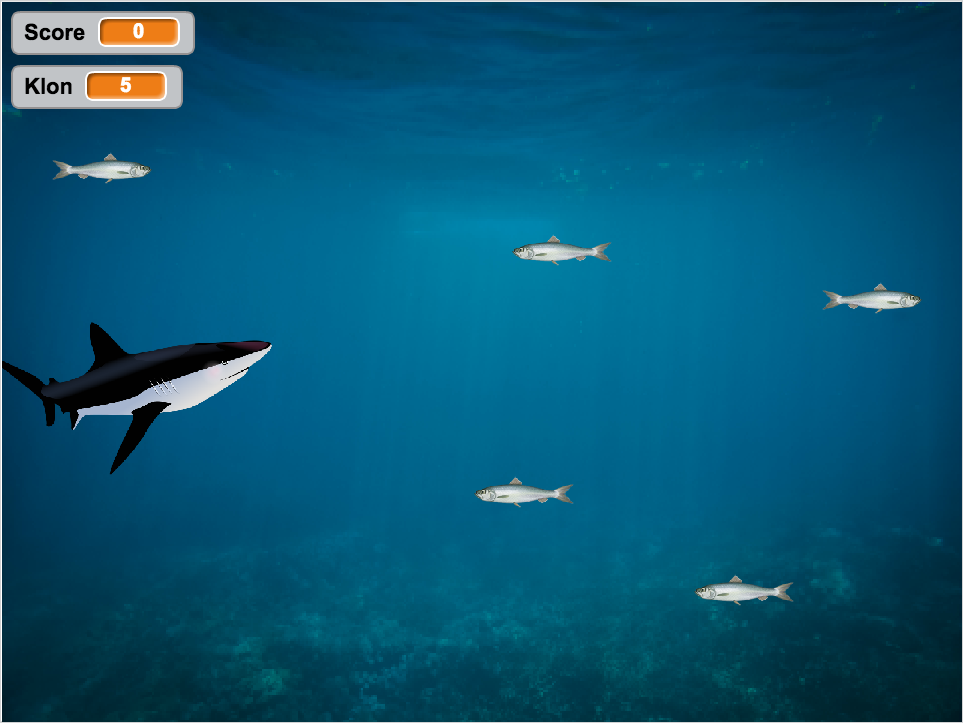
**Opgavetype #6: Tomme felter**

Elev-handout

Du finder et spil, som er næsten færdigt, her:

<https://scratch.mit.edu/projects/143272259/>



**Opgave:**

Spillet går ud på, at hajen skal spise sildene. Hvis der er mere end 5 sild i billedet, er spillet tabt. Start med at afprøv spillet. Kan du regne ud, hvilke funktioner der mangler?

Kig derefter i hajens og sildens kode. Hvor skal du indsætte de kommandoer, som mangler, for at spillet virker?

**Udvidelsesopgave:**

Prøv at modificere spillet, så det bliver sjovere eller mere udfordrende. Du kan fx programmere sildene til at komme med kortere mellemrum, eller lade hajen dø/fryse i x antal sekunder, hvis den rører et nyt objekt, du indsætter. Det kan også være, at der er en fiskerbåd uden for billedet, som prøver at fange hajen på en eller anden måde?

**Til underviseren**

Opgavetype #6: Tomme felter

### Løsning:

### Alle de ”tomme felter” er i hajens kode for at gøre det overskueligt for eleverne. Samtidig er størstedelen af programmet lagt ind i sildens kode, som udover at oprette sig selv som klon, der starter forskellige steder med et tidsinterval mellem 3 og 5 sekunder, også holder øje med antallet af kloner og skifter til ”Game over”-baggrunden, hvis dette overstiger 5.

### Eleverne skal udvide hajens kode med styringskommandoer + angivelse af det parameter, som udløser point (hvis ”berører sild”). Det færdige program findes her: <https://scratch.mit.edu/projects/142850378/#editor>

### 

### Hvad eleven bør vide på forhånd

### Eleven skal have lidt kendskab til brug af variable i Scratch for at kunne forstå programmets logik, og han/hun skal vide, hvordan kommandoer kan sættes ind i andre kommandoer. Derudover skal eleven kende basale kommandoer i Scratch.

**Beskrivelse af opgavetype #6: Tomme felter**

Opgavetypen, som kendes fra mange andre fagområder, er velegnet til let øvede elever, som man gerne vil styrke i den eksperimenterende tilgang. Eleverne præsenteres for et program, som mangler et eller flere væsentlige elementer, for at programmet kan udføre et specifikt mål. Opgavetypen minder om Ret fejlen, men i dette tilfælde, skal eleverne ikke blot modificere en eller flere kommandoer, som allerede findes i programmet, men derimod selv regne ud, hvilke kommandoer der mangler hvor.

***Didaktiske principper:*** *#3: Programmet afprøves, analyseres og modificeres, inden der konstrueres et lignende program. #5: Programmet udvides med delelementer.*

***Computational Thinking:*** *logisk ræsonnement, algoritmisk tænkning, dekomposition, fejlsøgning, eksperimentering, udformning.*

***SOLO-taksonomi****: relationelt/udvidet abstrakt niveau. Eleven ser sammenhænge mellem programmets koder og logik og kan konstruere nye koder og, som følge heraf, nye sammenhænge.*