

Løsningsforslag Eksamen Vår 2020

I dette løsningsforslaget brukes CSS-dokumentet [brukerinput.css](#) som du også finner på elevnettstedet <http://www.lokus.no/direkte/IT2> under kapittel 10 («Brukerinput med HTML og CSS»). Dokumentet inneholder CSS-kodene som blir gjennomgått i kapitlet.

Løsningen bruker også et bildebehandlingsprogram og et lydbehandlingsprogram i Oppgave 1. Du finner mer informasjon om bildebehandling under kapittel 5 på elevnettstedet, og lydbehandling under kapittel 13.

Nedenfor finner du kommentarer til hvordan vi har valgt å løse de tre oppgavene. Du finner også utfyllende kommentarer i hver av kodefilene.

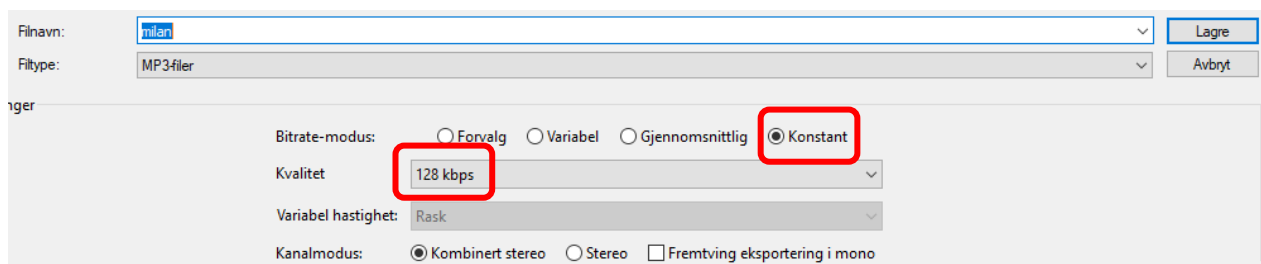
Oppgave 1

Vi valgte her å lage ferdige elementer der vi kunne legge til overskrift, lyd og bilde. En enda mer elegant løsning ville involvert å lage dem når brukeren trykker på knappen. Men denne varianten er mer oversiktlig.

Det er litt kronglete å bytte lydfil ved valg av lag. Man må nemlig bruke funksjonen `load()` for at spilleren skal bli klar med ny fil. Her er det et alternativ å lage tre `<audio>`-elementer, ett for hvert lag, og avgjøre hvilket som skal spilles av.

kamprop.mp3

En enkel løsning her, er å lage tre kopier av originalfilen, og klippe bort to av lagene i hver fil. Vi blir også bedt om å gjøre om kvaliteten til 128 kbps. Det kan gjøres ved eksport av mp3-filen:



Finavn: milan

Filtype: MP3-filer

Lagre

Avbryt

Bitrate-modus: ☐ Forvalg ☐ Variabel ☐ Gjennomsnittlig ☒ Konstant

Kvalitet: 128 kbps

Variabel hastighet: Rask

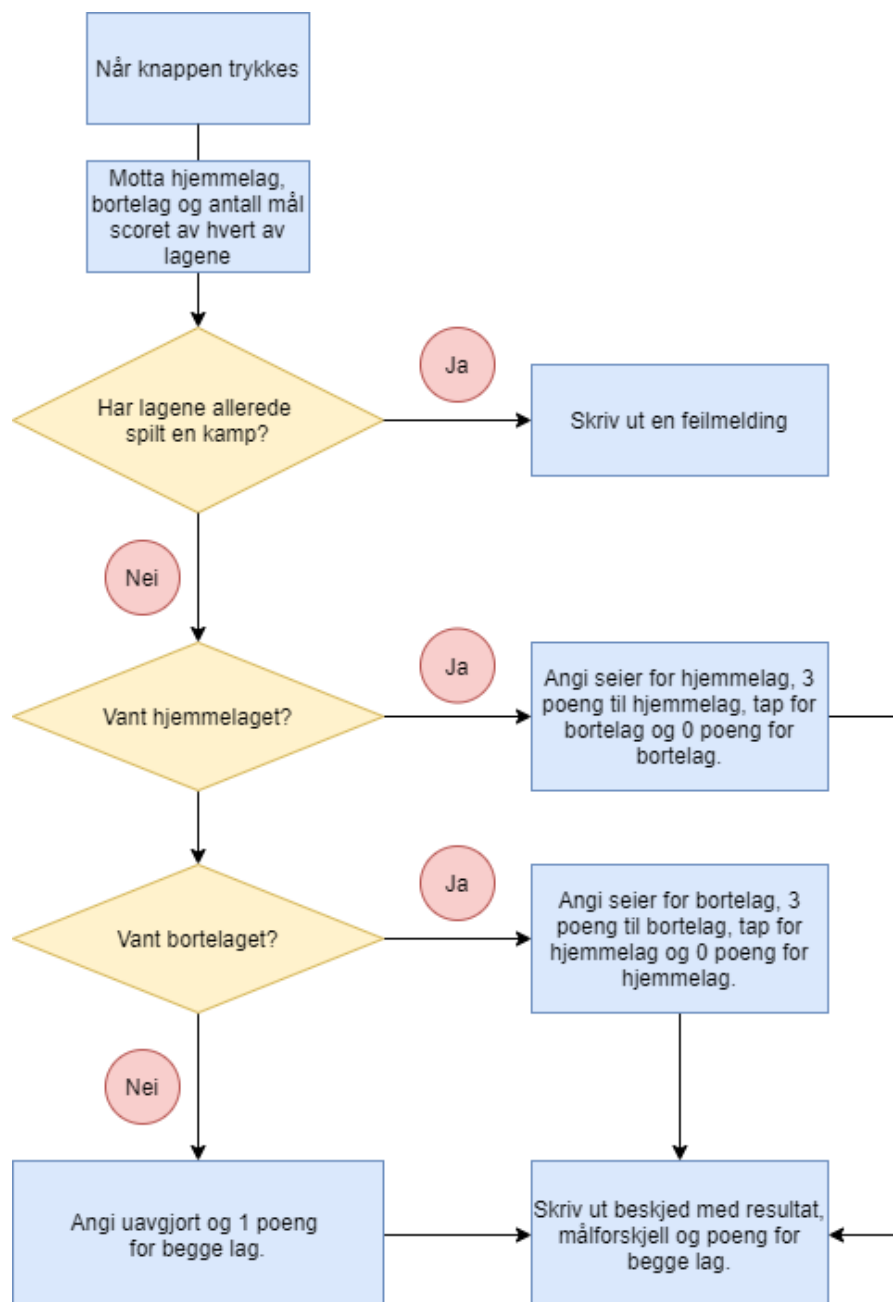
Kanalmodus: ☒ Kombinert stereo ☐ Stereo ☐ Fremtving eksportering i mono

inter.jpg

Den eneste forskjellen er at `inter.jpg` er bredere enn de andre bildene. Vi kan derfor beskjære bredden til `inter.jpg` slik at den blir 400 piksler (like mye på hver side, slik at vi beholder drakten på midten).

Oppgave 2 og 3

Vi har valgt å løse oppgave 2 og 3 samlet.

Flytdiagram for oppgave 2:

Figur 1. Flytdiagram laget med <http://www.draw.io>

Vi har valgt å lage en array med lagobjekter, og en med kampobjekter. På den måten kan programmet utvides med flere lag og kamper.

Den største utfordringen her, er sorteringen i oppgave 3. Det kan løses på mange måter, men vi har valgt å lage en array med lagene, som sorteres fortløpende. Konsekvensen er at lagene ikke har samme plass i arrayen hele tiden, så vi kan ikke bruke indeksen i arrayen som identifikator for lagene. Vi la derfor til en egen egenskap, **lagid**, for hvert lag.

Sortering

Å sortere objekter etter én egenskap er greit, men her skal vi sortere etter flere egenskaper dersom to lag har lik poengsum.

En annen utfordring er sorteringsretningen. Vanligvis kommer minst først når vi vil sortere noe, men her skal størst poengsum, størst målforskjell og størst antall mål scoret komme først.

En pseudokode for hvordan lagene kan sorteres, kan se slik ut (legg merke til at vi returnerer -1 hvis a har større poengsum enn b, fordi da skal a plasseres *foran* b. Her returnerer vi vanligvis 1, fordi a er større enn b, og skal plasseres etter):

```
Sammenlign poengsummene til lag a og b
Hvis a sin poengsum er større enn b sin:
    returner -1
Ellers hvis b sin poengsum er større enn a sin:
    returner 1
Ellers (lagene har like poengsum, vi må fortsette å sortere)
    Hvis a sin målforskjell er større enn b sin:
        returner -1
    Ellers hvis b sin målforskjell er større enn a sin:
        returner 1
    Ellers (vi kan fortsett ikke skille lagene, må sortere mer)
        Hvis a har scoret flere mål enn b:
            returner -1
        Ellers hvis b har scoret flere mål enn a:
            returner 1
    Ellers (lagene kan ikke skilles, vi gir oss her):
        returner 0
```