Lærerveiledning - Straffespark

Scratch Oppgave

Informasjon til veiledere

Straffespark er et enkelt fotballspill, hvor man skal prøve å score på så mange straffespark som mulig. Dette prosjektet passer bra som en introduksjon til Scratch.

Forberedelser

- [ ] **Antatt tidbruk**: 1.5 - 2 timer for hele prosjektet.
- [ ] **Nødvendige forkunnskaper**: Ingen.

Dersom dette prosjektet brukes som en introduksjon til Scratch anbefaler vi at du følger veiledningen Kom i gang med Scratch.

Om elevene allerede er komfortable med Scratch kan du benytte anledningen til å snakke om hvordan man programmerer ulik oppførsel på forskjellige figurer.

Typiske utfordringer

Nedenfor er en liste over utfordringer vi har opplevd at noen elever kommer borti.

- [ ] I Straffespark må man holde styr på tre figurer i tillegg til scenen, og passe på at hvert skript kodes på riktig sted. Vær nøye med at skriptene ligger på riktig figur som beskrevet i oppgaven.
- [ ] Katten skyter ballen før man klikker på den, eller den må gå flere steg før den når frem til ballen. Om dette skjer bør man flytte på hvor `Leo` og `Ball` plasseres ved å endre på `gå til x: () y: ()`-klossene. Om problemet er at katten må gå flere steg kan man også endre på hvor langt `Leo` `går` når han klikkes. Dersom elevene allerede kan litt om koordinatsystemet er det en fin øvelse å tenke på hvilke koordinater man bør endre for å flytte figurene. Alternativt kan man flytte på figurene ved å klikke og dra, og deretter se på koordinatene øverst til høyre i skriptvinduet.
- [ ] I steg 4 jobber vi videre med skriptet som ble skrevet på `Ball` i steg 2. Pass på at elevene ikke lager to forskjellige skript. Om de lager to forskjellige skript vil effekten stort sett være at ballen beveger seg fortere enn normalt fordi begge skriptene flytter ballen.
- [ ] I steg 4 og 5 lages det mange små skript som starter på meldingene `Mål` og `Redning`. Pass på at disse havner på

korrekt figur. Det vil si at `Ball` og `Keeper` har skript med `stopp [andre skript i figuren v] :: control`, `Leo` har skript hvor han `sier` noe, mens Scenen har skript som teller `(Mål)` og `(Redninger)`

Dersom elevene vil at også `Ball` eller `Keeper` skal si noe kan det være utfordrende på grunn av `stopp`-klossen. En mulig løsning er som følger:

```
når jeg mottar [Redning v]
si [Hurra, jeg reddet!]
stopp [andre skript i figuren v] :: control
```

Det er her viktig å *ikke* bruke `si [Hello!] i (2) sekunder` siden den klossen vil gjøre at `Ball` eller `Keeper` ikke slutter å bevege seg før etter 2 sekunder. For at snakkeboblen skal bli borte kan man bruke en `si []`-kloss (uten tekst). Denne kan legges øverst i `når grønt flagg klikkes` - eller `når jeg mottar [Nytt spark v]`-skriptet.

- [ ] For enkelhets skyld settes aldri retningen på `Ball` i dette prosjektet. Siden ballen aldri forandrer retning - den beveger seg alltid horisontalt fra venstre mot høyre - er dette sjelden et problem. Men dersom elevene har endret retning på `Ball` slik at den spretter på skrå over skjermen må retningen tilbakestilles. Dette gjøres enklest ved å klikke på klossen `pek i retning (90 v)` (eller ved å legge denne klossen øverst i `Ball` sitt hovedskript).

Variasjoner

Dette er et introduksjonsprosjekt, og elevene ledes derfor ganske detaljert gjennom hvordan spillet skal programmeres. Det er likevel rom for en del kreativitet. Elevene kan gjerne oppfordres til å

- [ ] velge sine egne figurer og bakgrunner. `Leo` trenger absolutt ikke å være en katt, og det har blitt scoret mange mål med noe annet enn en fotball.

- [ ] eksperimentere med hastigheten til `Ball` og `Keeper`. Ved å endre på tallene i `gå () steg`-klossene vil figurene flytte seg saktere eller raskere. Det er nyttig læring å teste effekten av slike endringer, og observere hvordan vanskelighetsgraden i spillet forandrer seg (se også boksen **Endre farten** på slutten av steg 4).

- [ ] forandre på tekstene i snakkeboblene til `Leo`, eller tekstene som vises når man vinner eller taper spillet.

- [ ] legge på passende lydeffekter. Dette nevnes i oppgaven mot slutten av steg 5, men om elevene har litt erfaring med Scratch fra tidligere kan de gjerne gjøre dette underveis i programmeringen også.

Tema: Skript tilhører figuren

Dersom elevene allerede er komfortable med Scratch er dette prosjektet en bra anledning for å snakke om hvordan man gir forskjellige figurer en unik oppførsel ved å legge ulike skript på dem.

Et viktig konsept i Scratch er at man koder ved å beskrive egenskapene (utseende, posisjon, retning, osv.) og oppførselen (skript) til figurer. På fagspråket kalles dette **objektorientert programmering** (mer presist er Scratch *prototypeorientert programmering*, men forskjellen er ikke relevant her). Dette virker så naturlig at elevene sjelden bevisst tenker på dette, og samtidig skaper det sjelden problemer.

Presentasjon

- [ ] Start et nytt Scratchprosjekt ved å klikke `Programmering` fra hovedsiden, eller `Ny` i `Fil`-menyen.

- [ ] Legg til en ekstra figur - for eksempel `Bat1` - slik at det er to figurer i prosjektet. Dra dem rundt på scenen slik figurene er i hvert sitt hjørne.

- [ ] Spør elevene hvordan de vil kode at katten beveger seg mot flaggermusa (den andre figuren)? Spesielt, pass på at de er bevisst hvilken figur som må programmeres (*Katten*). Spør om det det samme kan programmeres ved å legge et skript på den andre figuren (*Nei, siden Katten beveger seg er det Kattens oppførsel vi må beskrive*).

```
for alltid
   pek mot [Bat1 v]
   gå (10) steg
slutt
```

- [ ] Hvordan kan vi programmere at flaggermusa rømmer fra katten når katten tar (berører) den? Igjen, hvilken figur må programmeres? Kanskje begge? *Vi må programmere flaggermusa siden den rømmer (oppførsel).* Her trenger vi ikke noe nytt program for katten så lenge den ikke reagerer på at den berører flaggermusa (*ingen ny oppførsel å beskrive*).

  Det er mange måter å skrive kode for at flaggermusa rømmer. Det følgende er et eksempel (husk at koden hører til flaggermusa):

```
for alltid
   vent til <berører [Sprite1 v]>
   gli (0.2) sekunder til x: (tilfeldig tall fra (-240) til (240)) y: (tilfeldig tall fra (-180) til (180))
slutt
```

- [ ] Spør elevene om de kan tenke seg noen annen måte (enn objektorientert) å programmere på? Hvor man ikke knytter skriptene til figurene?

  Et eksempel på en annen type programmering er **imperativ programmering** hvor programmer skrives som en serie kommandoer uten at det skilles mellom hvilken figur som kommanderes. I et slikt språk ville de to skriptene over skrives som *ett* skript omtrent som dette (ikke alle disse klossene eksisterer i Scratch):

```
for alltid
   flytt [katten v] mot [flaggermusa v] :: motion
   hvis <[katten v] berører [flaggermusa v] :: sensing>
      flytt [flaggermusa v] til x: (tilfeldig tall fra (-240) til (240)) y: (tilfeldig tall fra (-180) til (180)) :: motion
   slutt
slutt
```

  Vis gjerne denne koden til elevene. I tillegg til at det bare er ett skript, hvilke andre forskjeller ser de? *Den andre store forskjellen er at man alltid må fortelle hvilken figur som skal utføre kommandoene. Dette er underforstått i Scratch*.