

○ Lærerveiledning - Labyrint

[✎ TIL OPPGAVE](#)[↓ LAST NED PDF](#)

Om oppgaven

I denne oppgaven skal elevene lage et enkelt spill hvor man kontrollerer en liten utforsker som leter etter skatten gjemt inne i en labyrint.



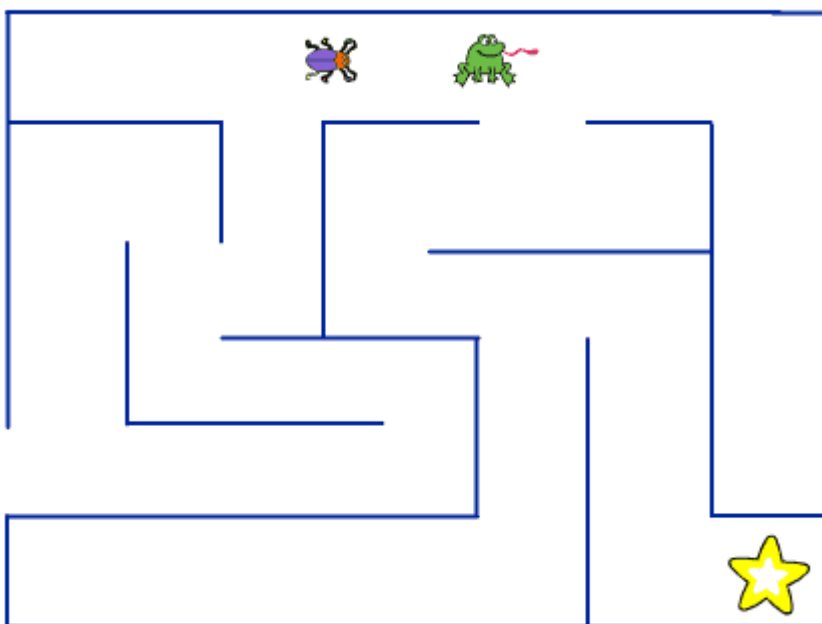
Oppgaven passer til:

Fag: Programmering

Trinn: 3.-10. trinn

Tema: Løkker, tester, variabler, tilfeldighet

Tidsbruk: Dobbeltime eller mer



Kompetansemål

- ☐ **Programmering, valgfag:** bruke grunnleggende prinsipper i programmering, slik som løkker, tester, variabler, funksjoner og enkel brukerinteraksjon

Forslag til læringsmål

- ☐ Elevene kan forklare hvordan løkker, tester og variabler fungerer, og hvorfor de er hensiktsmessige å bruke i denne oppgaven.
- ☐ Eleven kan forklare hvordan brukeren kan styre figuren med piltastene, og hvordan figuren interagerer med labyrinthen.
- ☐ Eleven kan forklare hvordan tilfeldighetsgeneratoren fungerer.

Forslag til vurderingskriterier

- ☐ Eleven oppnår måloppnåelse ved å fullføre oppgaven slik det er beskrevet.
- ☐ Eleven oppnår høy måloppnåelse ved å legge til flere momenter, og utvikle en mer utfordrende labyrinth.
- ☐ Dette er en oppgave hvor elevene fint kan prøve hverandres labyrinth og vurdere hverandre.

Forutsetninger og utstyr

- ☐ **Forutsetninger:** Ingen
- ☐ **Utstyr:** Datamaskiner med Scratch installert. Eventuelt kan elevene bruke Scratch i nettleseren dersom de har en bruker (eller registrerer seg) på scratch.mit.edu/.

Fremgangsmåte

Her kommer tips, erfaring og utfordringer til de ulike stegene i den faktiske oppgaven. [Klikk her for å se oppgaveteksten.](#)

Steg 3: Utforskeren kan ikke gå gjennom veggen

- ☐ Et vanlig problem her er at **utforskeren kan gå rett gjennom veggen**. Typisk vil dette være fordi man ikke har rett farge i `<berører fargen [#cc0000]>`-klossen. Eventuelt at man har brukt flere farger på veggene i labyrinthen. Det er viktig at alle veggene er tegnet i samme farge.

Man kan også oppleve at utforskeren går rett gjennom veggen om `(hastighet)` er for høy. Dette er fordi utforskeren *hopper* `(hastighet)` steg hver gang man trykker en piltast, og den kan da hoppe over en vegg.

- ☐ Et annet problem er at **utforskeren hopper gjennom vegger når den snur**. Alle figurer har et definert senterpunkt som de roterer rundt (se Steg 2 i prosjektet [Soloball](#) for et bra eksempel på hvordan dette virker). Dersom dette senterpunktet ikke er midt på `Utforsker`-figuren vil det se ut som om den hopper rundt når den snur.
- ☐ For å sette senterpunktet riktig kan du klikke på `Drakter`-fanen, og = ikonet nede til høyre for tegningen for å se hele tegningen samtidig. Hvis du ser nøye etter vil senteret i tegningen ha et lite blink-ikon i seg. Du må legge figuren midt på denne for at figuren skal være på midten.
- ☐ Det kan også forekomme at **det er vanskelig å bevege seg i labyrinthen**. Om gangene er for smale eller veggene for skrå blir det vanskelig for utforskeren og froskekongen å bevege seg. Se [nedenfor](#) for flere tips til hvordan man bruker tegneverktøyet effektivt som kan hjelpe elevene.

Steg 5: Froskekongen vokter i gangene

- ☐ Et vanlig problem her kan være at **utforskeren eller froskekongen setter seg fast i veggen**. I dette prosjektet har vi fokusert på å holde koden så enkel som mulig. Spesielt er koden som passer på at figurene ikke går gjennom veggene litt *for enkel*. Noen enkle tips som vil motvirke problemet er presentert i tipsboksen nederst i [steg 5 i oppgaven](#).

Stopp alle

Vi bruker klossen `stopp alle` for å stoppe alle skript i programmet når froskekongen tar utforskeren. Dette stopper alle skriptene som startet da vi klikket på det grønne flagget, men det hindrer ikke nye skript å starte. Derfor kan fortsatt utforskeren flyttes rundt med piltastene etterpå.

Siden oppgaven er et introduksjonsprosjekt er det ikke gjort noe med dette. For elever som har lyst å prøve seg på en løsning kan du foreslå følgende:

- ☐ Lag en `game_over`-variabel som er satt til `false` eller `0` mens spillet pågår, som sjekkes hver gang spilleren trykker en piltast, før utforskeren flytter seg. Sett variabelen til `true` eller `1` når spillet er over.
- ☐ En mer vanlig (og bedre) løsning er å bruke en `for alltid`-løkke med `hvis ... trykket?`-klosser. Disse blir da stoppet av `stopp alle`-klossen.

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**. De kan fritt velge figurene som brukes for `Utforsker`, `Skatt` og `Froskekonge` uten at det har noen effekt på programmeringen.
 - ☐ **tegne sin helt egne labyrinth**. I oppgaven vises et eksempel (eller to om man også ser på tegningen først i oppgaven) på en labyrinth, men elevene kan gjerne tegne denne annerledes. Be gjerne elevene likevel tenke på at det skal være enkelt for utforskeren og froskekongen å bevege seg rundt, slik at de bør ha rette vegger og brede nok ganger.
 - ☐ **eksperimentere med hastighet**. I [steg 1 i oppgaven](#) vises det hvordan man lett kan endre hvor raskt en figur flytter seg ved å bruke en `(hastighet)`-variabel. La elevene eksperimentere med denne og `(hastighet)`-variabelen for froskekongen, og spør dem hvordan det forandrer vanskelighetsgraden i spillet.
- ☐ Om elevene allerede er komfortable med Scratch kan du benytte anledningen til å snakke om hvordan tegneverktøyet i Scratch fungerer og gi dem noen tips til hvordan man bruker det effektivt.
- ☐ For de mer avanserte elevene kan du også vise frem følgende kode som gjør en bedre sjekk av kollisjon med veggen.