



Robotinvasjon

Introduksjon

Vi har sett enkle datamaskiner. Nå skal vi leke oss med roboter, og fin for oss.



Steg 1: Vår første robot

Vi skal nå bli kjent med roboter og se noe av det de kan brukes til.

En robot er en datamaskin som kan bevege seg. I ComputerCraft kan eksempel kan grave, bygge, slåss og så videre.

Vi begynner likevel med en helt enkel robot:

✓ Sjekkliste

- ☐ Åpne inventory'et ditt ved å trykke 'E'. Finn frem 7 **Iron Ingot**, :
- ☐ Start et **Crafting table**, og lag en robot slik:



- ☐ Legg den nye roboten i hånden din. Lukk inventory'et og lag en

I *Creative Mode* finner du også robotene ved å trykke **E** , deretter **>** c
Robotene heter **Turtle** i ComputerCraft.

Turtles

Navnet **Turtle** betyr *skilpadde* på norsk. Grunnen til at disse robotene siden bygde William Grey Walter et par roboter som kunne bevege seg lave og skallformet. De fikk derfor etterhvert kallenavnet skilpadde.

Senere ble måten disse skilpaddene beveget seg på (vi skal se hvordan programmeringsspråk, spesielt som en måte å tegne på. Språket *L* skilpaddegrafikk, men nesten alle programmeringsspråk støtter de *ComputerCraft*.

Sjekkliste

På samme måte som med datamaskiner starter du roboter ved å høyre roboten.

- ☐ Start en robot. Skriv `programs` og trykk enter.

Dette viser hvilke programmer denne roboten kjenner til. Hvis du kjenner til vil du se at det er mange av de samme programmene datamaskinen ikke kan.

- ☐ Kjør programmet `dance`.

Roboten begynner nå å danse! Trykk *Esc*-knappen for å stenge den imponert?

- ☐ Hvis du vil at roboten skal slutte å danse kan du høyreklikke på som sier at du kan få roboten til å slutte å danse ved å trykke enter. Hvis du vil kan du også la roboten fortsette å danse. Lag da en

Steg 2: Roboter og skilpad

Vi vil nå se hvordan vi kan få robotene våre til å bevege seg rundt.

Som nevnt i boksen *Turtles* ovenfor beveger vi robotene våre på en måte kontrollert for nesten 70 år siden. Dette gjør vi ved å bruke programmet

Sjekkliste

- ☐ Kjør programmet `go forward` i kommandolinjen til en robot.
- ☐ Roboten sier at den er `Out of fuel`.

Roboter bruker *fuel* for å bevege seg. De kan bruke stort sett sa eksempel er **Coal** eller **Blaze Rod** fine å bruke.

- ☐ Finn litt **Coal** i inventory'et ditt. Høyreklikk på roboten. Legg me 4) på høyre side. Dette er robotens inventory. Flytt kullet over ti



- ☐ Skriv **refuel** i kommandolinjen og trykk enter.

Legg merke til at en kull blir borte fra robotens inventory. Robot tallet forteller hvor langt roboten kan bevege seg før den går to

- ☐ Gi roboten litt mer **Coal** og skriv **refuel all**.

Roboten vil nå spise opp alt kullet, og deretter rapportere at der

- ☐ Da prøver vi igjen: Kjør programmet **go forward**.

Flytter roboten din seg? Det kan være litt vanskelig å se hva sor tenke på den lange, smale sprekken som øynene til roboten. Alt

- ☐ Vi kan få roboten til å flytte seg tilbake ved å skrive `go back`.

Finne hjelp

Datamaskiner og roboter har et innebygd hjelpesystem. For å se hjelp skriver du `help` og trykker enter. Dette gir deg en rask introduksjon til nyttige hjelpekommandoer. For å få hjelp om et spesielt program. Da må du bytte ut `<program>` for å få hjelp om et spesielt program. Da må du bytte ut `<program>` med programnavnet. For eksempel kan du skrive `help go` for å se hjelp om `go`-programmet.

Det finnes selvsagt også en del hjelp på Internett. Et bra sted å starte er <http://computercraft.info/wiki/>.

✓ Sjekkliste

- ☐ For å få vite mer om hvordan roboten kan flytte seg kan vi skrive `help`. Dette viser oss at vi kan bruke `go forward`, `go back`, `go up`, `go down` og `go around`. I tillegg ser vi at vi kan bruke tall for at roboten skal flytte seg en viss avstand. Prøv `go forward 10` og `go back 5`.
Hvordan kan vi få roboten til å bevege seg sidelengs?
- ☐ Det finnes ingen kommando som får roboten til å bevege seg sideveis. For å få roboten til å gå sidelengs må vi derfor først skrive `go forward`. Skriv `go left` og deretter `go forward 3`.
- ☐ Lek litt mer med `go`-programmet til du skjønner hvordan du flytter roboten. `go right` er litt forvirrende siden roboten ikke går noe sted, den roterer bare. Hva skjer dersom du ber roboten gå gjennom bakken, gjennom

Steg 3: Gruverobot

Hvis vi gir roboter de riktige verktøyene kan de grave, bygge og slåss

Vi skal nå bruke en gruverobot som kan grave for oss.

Sjekkliste

☐ Finn en gruverobot i inventory'et ditt ved å gå til datamaskinfan
gruverobot.

☐ Gi roboten litt **Coal** og kjør `refuel all`.

☐ Vi skal nå bruke et program som heter `excavate`, dette betyr g.
Skriv `excavate 3` og trykk enter.

Ta et steg tilbake og se på mens roboten graver. Roboten vil for
grunnfjellet, **Bedrock**.

☐ Hva tror du tallet `3` i kommandoen vi skrev over betyr? Skriv `h`

☐ Høyreklikk på roboten slik at du ser inventory'et den har. Legg r
Når roboten er ferdig å grave kommer den tilbake dit den starte
slik at du kan plukke det opp om du vil.

☐ Lag flere gruveroboter som kan grave større eller mindre hull.

Steg 4: Robotprogrammer

Vi skal nå lære hvordan vi kan kontrollere roboter i våre egne program

Når vi skriver egne programmer som styrer robotene bruker vi komma

Sjekkliste

- ☐ Start en ny robot. Pass på at den har fått litt kull og blitt `refuel`
- ☐ Vi begynner med å prøve å finne litt mer informasjon om `turtl`
Du får nå se en ganske lang liste med kommandoer som vi kan alle sammen på egen hånd, og se om du skjønner hvordan alle
- ☐ For å gjøre enkle tester vil vi skrive kommandoene direkte i Lua. Husk at du skriver `exit()` for å gå ut av Lua-tolkeren og tilbake
- ☐ Vi begynner med de enkle flytte-kommandoene. Skriv `turtle.f`
Husk at du kan trykke *Esc*-knappen for enklere å se hva roboter Lua-tolkeren.
- ☐ Prøv også de følgende kommandoene. Til sammen gir de oss m
`turtle.forward()`, `turtle.back()`, `turtle.turnLeft()`, `turt`

Prøv selv

Lag en liten kloss litt unna roboten din, omtrent som på bildet under
å plassere roboten på toppen av klossen?



Steg 5: Up, up, up, and aw

Kan vi klare å få roboten til å bygge for oss?

Et av poengene med roboter er at de kan gjøre arbeid for oss. I Minec eksempel grave eller bygge.

✓ Sjekkliste

- ☐ Start en ny robot. Gi den fuel (og kjør `refuel`). Legg også noe k venstre boksen i robotens inventory.
- ☐ Pass på at det ikke er noe foran roboten, og skriv `turtle.place` Bygde roboten en gresskloss foran seg? Da har du gjort alt riktig
1: at du har startet `lua`,

2: at roboten har fuel,

3: at roboten har byggemateriale,

4: at det ikke står noe foran roboten (husk at den smale sprekke

5: at boksen med byggemateriale i robotens inventory er merket enn de andre boksene.

☐ Roboten kan også sjekke om den har noe foran seg: Skriv `turtle.detect()`. Du skal få svaret `true` som betyr at roboten ser at den har noe foran seg.

☐ Prøv så `turtle.back()` etterfulgt av `turtle.detect()`. Siden roboten nå ikke har noe rett foran seg får du svaret `false`.

I Steg 6 skal vi se hvordan vi kan bruke `place()` og `detect()` sammen.

Men først, en ting vi så når vi brukte `go` var at hvis vi ville at roboten skulle gå fram 3 steg, kunne vi skrive `go forward 3`. Det samme fungerer ikke med `turtle`-biblioteket.

Sjekkliste

☐ En enkel måte å gjøre noe et bestemt antall ganger er å bruke `for`-løkke. Skriv `for i = 1, 5 do turtle.back(); end` i Lua-tolkeren. Flytter roboten din seg 5 steg bakover?

☐ Hvis vi vil kombinere flere kommandoer inne i en løkke i Lua-tolkeren, kan vi prøve `for i = 1, 5 do turtle.back(); turtle.place(); end`.

☐ Pass på at roboten din har mye byggemateriale, for eksempel 6.

☐ La oss bygge et høyt tårn!

Skriv `for i = 1, 60 do turtle.up(); turtle.place(); end`.



Men oops! Vi glemte å fortelle roboten at den skulle komme ned når c den igjen?

Steg 6: Bygg en trapp

Kan vi skrive et program som kan hjelpe oss til å hente ned den forsv

Når vi skal gjøre ting som er litt kompliserte er det som regel enklere i enkeltkommandoer i Lua-tolkeren. La oss prøve å lage et program som tårnet.

Før vi begynner på utfordringen det er å bygge en kjempehøy trapp, l

✓ Sjekkliste

- ☐ Bygg et tårn som er tre klosser høyt. Dette kan du bygge enten



- ☐ Lag en ny robot inntil det lille tårnet du nettopp bygde. Gi den f
- ☐ Begynn å skriv et nytt program ved å skrive `edit byggtrapp`. S

```
turtle.detect()
```

Lagre og avslutt ved å bruke *Ctrl*-tasten.

- ☐ Kjør programmet ved å skrive `byggtrapp`. Dette programmet by ikke gjør noe som helst. Det eneste som skjer er at roboten mer fortalt den hva den skal gjøre etterpå.

Vi kan bruke `detect` til å finne toppen av tårnet.

- ☐ Endre på programmet ditt ved å skrive `edit byggtrapp` igjen. V lenge den merker at tårnet er høyere.

```
while turtle.detect() do  
    turtle.up()
```

end

Lagre og kjør programmet ditt. Klatrer roboten til toppen av det



Vi har lært av feilen vi gjorde tidligere, så nå vil vi passe på at roboten ikke klatrer ned. Denne gangen bruker vi `detectDown()` som `detect()`, bortsett fra at den merker om roboten har en kloss under seg.

Utvid programmet ditt slik:

```
while turtle.detect() do
    turtle.up()
end

while not turtle.detectDown() do
    turtle.down()
end
```

Vi sier at så lenge roboten *ikke* har en kloss under seg kan den klatre opp. Virker programmet?



Nå er vi klar til å la roboten bygge selve trappen. Det gjør vi ved å bruke `place()` i tillegg til `down()` og `back()`.

```
while turtle.detect() do
    turtle.up()
end

while not turtle.detectDown do
    turtle.down()
    turtle.back()
    turtle.place()
end
```

Virker det? Lager roboten en trapp?



Nå er vi klare for den store testen. Klarer vi å sende trappebygg

Pass på at roboten fortsatt har nok fuel, og fyll opp med byggen
byggtrapp !



Gratulerer! Du har nå programmert en robot! Legg merke til at siden
kommandoer kan det bygge trapper opp alle slags tårn og bratte fjells

Lisens: [CC BY-SA 4.0](#) **Forfatter:** Geir Arne Hjelle