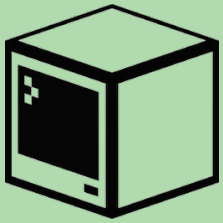
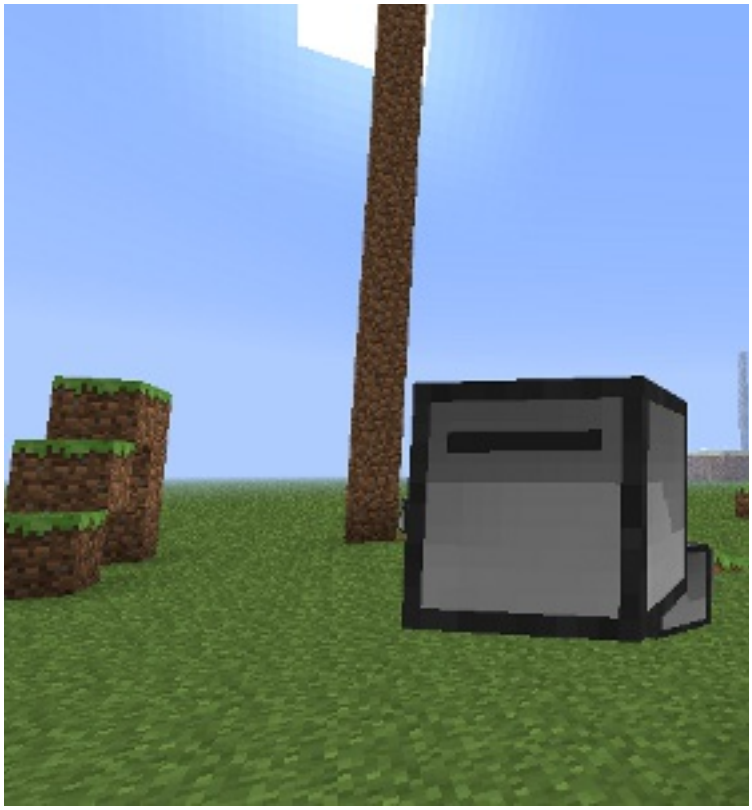


Introduksjon til Compu

Introduksjon

ComputerCraft er en mod til **Minecraft**, som gir deg muligheten til inne i Minecraft-verdenen.

Vi begynner med å bygge en enkel datamaskin. Etter å ha brukt litt ti bruker vi den til å låse opp en dør ved hjelp av passord. Etter at vi har bevege seg, altså roboter! Disse kan vi bruke til både å grave og bygg



Steg 1: Vår første datamas

Vi begynner med å lage en datamaskin.

I utgangspunktet er det mye enklere å starte i *Creative Mode* når vi sk
bruke tid på å samle materiale og slåss mot monstre. Vi vil derfor i for
akkurat det du trenger i inventory'et ditt.

Men først, for å vise at en datamaskin kan bygges på vanlig måte i Mi

Sjekkliste

- [ ] Trykk `E` for å åpne inventory'et ditt. Finn frem 7 **Stone**, 1 **Reds**
- [ ] Åpne et **Crafting table**, og legg ut materialet slik:

- [ ] Legg den nye datamaskinen i den nederste raden i inventory'et
  inventory'et.
- [ ] Bruk talltastene til å velge datamaskinen, og høyreklikk for å lag

I *Creative Mode* kan du også få tak i datamaskiner ved å trykke `E`, de
fanen med datamaskinsymbolet.

ComputerCraft-datamaskiner

Høyreklikk en datamaskin for å starte den opp. Den vil åpne en sva
er datamaskinens *kommandolinje*, og vi vil bruke den til å styre da

Sjekkliste

- [ ] Prøv å skriv `help` og trykk enter.

  Du får nå se noen tips om hvordan du kan finne ut mer om data
  se en liste over hvilke programmer som er på datamaskinen, ell
  man programmerer datamaskinen.

- [ ] Skriv `programs` og trykk enter.

  Mange av disse programmene er enkle programmer som lar deg
  du prøve å utforske flere av disse programmene.

Steg 2: Vårt første progran

Det er nå på tide at vi skriver vårt første program.

Det er en lang tradisjon blant programmerere at det første programm
trivelig melding til skjermen. Vi følger den tradisjonen og begynner m

Sjekkliste

- [ ] Start en datamaskin.

- [ ] Skriv `edit heiverden` for å begynne å skrive på et nytt prograr

- [ ] I det nye vinduet, skriv

  ```
  print('Hei verden!')
  ```

Etter at du har skrevet dette så trykker du på *Ctrl*-tasten og velg
`Exit`.

- [ ] Vi har nå laget vårt første program. Prøv å skriv `programs` og du

- [ ] For å kjøre programmet vi har laget, skriver vi `heiverden` og try

Prøv selv

Klarer du å endre på programmet slik at det for eksempel sier hei t
deg?

Prøv å skriv `edit heiverden` en gang til. Da åpner programmet dit
noe annet. Som tidligere må du bruke *Ctrl*-tasten for å gå til menye

Steg 3: Datamaskiner og R

Vi skal nå bruke datamaskinen til å åpne en dør for oss, og kanskje et

Datamaskinene i ComputerCraft bruker et programmeringsspråk som
språk som ofte brukes inne i andre programmer. For eksempel kan og
programmeres med Lua.

Lua

Lua ble opprinnelig laget i Brasil på begynnelsen av 1990-tallet. På
språk som het *Simple Object Language* (SOL). Ordet *lua* er portugis
som passet sammen med SOL.

Sjekkliste

Vi vil bruke **Redstone** for å kontrollere døren. Redstone er et materia
sette opp en datamaskin ved siden av en dør.

- [ ] Gå ut av datamaskinen ved å trykke `Esc`-knappen.

- [ ] Samle sammen litt **Stone**, en **Iron door** og en **Computer**, og l
datamaskinen rett ved siden av døren. Det skal se omtrent slik

- [ ] Start datamaskinen.

Vi skal nå prøve å skrive noen kommandoer direkte i Lua. Dette er en

- [ ] Skriv `lua` og trykk enter. Dette starter en *Lua-tolker* som vil utf

- [ ] Vi begynner med kommandoen fra det første programmet vårt.

samme som tidligere?

- [ ] Som de aller fleste programmeringsspråk er Lua glad i å regne.

  Lua kjenner alle de vanlige matematikk-operasjonene. Prøv for
  igjen hva hver av disse betyr?

- [ ] Skriv `redstone.setOutput('left', true)` og trykk enter.

  Denne kommandoen skal åpne døren til venstre for datamaskin
  døren står på. Du kan også bruke for eksempel `right`, `top` elle

  Dette er et eksempel på å kalle en *funksjon*, noe vi gjør ofte når
  `setOutput` og den hører hjemme i `redstone`-*biblioteket*.

Funksjoner

Alle programmeringsspråk lar deg lage noe som kalles *funksjoner*.
sammen slik at det blir enklere å gjøre vanskelige ting. Lua komme
`print` og `redstone.setOutput` som eksempler på dette. Det går

Sjekkliste

- [ ] Trykk pil opp-tasten slik at du kan endre `redstone.setOutput('`
  `false)`. Trykk enter. Nå lukker døren seg igjen fordi vi skrur av

- [ ] Avslutt Lua-tolkeren ved å skrive `exit()` og trykk enter.

Steg 4: Passordlås på en d

Vi skal nå bruke redstone-biblioteket til å lage en passordlås på døra.

Sjekkliste

- [ ] Start et nytt program ved å skrive `edit passord` og trykk enter

- [ ] Skriv inn følgende program nøyaktig slik det står

```
local passord = 'kodeklubben'
print('Passordet er ' .. passord .. '!')
```

  Pass på at du skriver de to punktumene `..` riktig. Disse betyr a

- [ ] Lagre og avslutt editoren. Kjør programmet ved å skrive `passor`

  Programmet forteller deg hva passordet er. Klarer du å endre pa
  til navnet ditt?

Variabler

Vi har nå laget en variabel. Variabelen `passord` husker hva passor
endre passordet senere. Ordet `local` foran variabelen sier at vi ba
(lokalt).

Sjekkliste

- [ ] Vi skal nå jobbe videre med programmet. Vi vil jo at datamaskir
  det er. Skriv `edit passord` igjen, og endre programmet slik at d

```
local passord = 'kodeklubben'
print('Hva er passordet?')                    -- endret
```

```
svar = read()                                    -- ny linj
```

Lagre, avslutt, og kjør programmet på nytt. Nå vil programmet s
Funksjonen `read` brukes for å lese ting du skriver på tastaturet,
ditt.

Det neste vi vil er derfor at programmet skal sjekke om du svare
som heter `if`-tester. Disse kan teste om noe er sant, og vi vil b
likt med det faktiske passordet. Legg til en `if`-test nederst i pro

```
local passord = 'kodeklubben'
print('Hva er passordet?')
svar = read()

if svar == passord then                          -- ny linj
    print('Riktig, passordet er ' .. passord)    -- ny linj
else                                             -- ny linj
    print('Feil, passordet er ikke ' .. svar)    -- ny linj
end                                              -- ny linj
```

Kjør programmet igjen. Hva skjer når du svarer riktig? Hva skjer

Nå kan vi koble sammen passordet med `redstone`-kommandoe
riktig. Etter 5 sekunder kan vi lukke døren igjen. Endre program

```
local passord = 'kodeklubben'
print('Hva er passordet?')
svar = read()

if svar == passord then
    redstone.setOutput('left', true)             -- endret
    sleep(5)                                     -- endret
    redstone.setOutput('left', false)            -- endret
end
```

Kjør programmet. Hva skjer nå når du skriver riktig passord?

Funksjonen `sleep` gjør at datamaskinen sover, det vil si gjør ing
sekunder, før energien skrus av igjen og døren lukker seg.

- [ ] Det er kjedelig at vi hele tiden må starte programmet på nytt. V
igjen og igjen.

```
local passord = 'kodeklubben'

while true do                                          -- ny linj
    print('Hva er passordet?')
    svar = read()

    if svar == passord then
        redstone.setOutput('left', true)
        sleep(5)
        redstone.setOutput('left', false)
    end
end                                                    -- ny linj
```

Det er bare èn ny ting her som vi ikke har sett tidligere, nemlig
om igjen så lenge noe er sant. I vårt tilfelle er dette *noe* verdien
evig løkke. Dette programmet vil fortsette å spørre oss om pass

- [ ] Kjør programmet. Oppfører programmet seg slik du hadde trodd

Avslutte programmer

For å avslutte dette programmet holder du inne *Ctrl* og *T* samtidig
teksten `Terminated` skrives på skjermen. Dette fungerer både her

Sjekkliste

- [ ] Før vi er helt fornøyde med dette programmet vil vi gjøre noen s
`term`-biblioteket. Den ene vil rense skjermen, mens den andre s
som betyr øverst til venstre. I tillegg forteller vi `read` at vi ikke
stedet vil vi at `*`-tegn skal vises. Programmet ser da slik ut:

```
local passord = 'kodeklubben'

while true do
    term.clear()                                -- ny linj
    term.setCursorPos(1, 1)                     -- ny linj
    print('Hva er passordet?')
    svar = read('*')                            -- endret

    if svar == passord then
        redstone.setOutput('left', true)
        sleep(5)
        redstone.setOutput('left', false)
    end
end
```

Gratulerer, du har allerede lært ganske mye om hvordan man prograr
Prøv gjerne å forandre noen av programmene vi har laget. Kan du lag

Steg 5: 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**, 1
- [ ] 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 robot
siden bygde William Grey Walter et par roboter som kunne bevege
lave og skallformet. De fikk derfor etterhvert kallenavnet skilpadde

Senere ble måten disse skilpaddene beveget seg på (vi skal se hvo
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 høyr

roboten.

- [ ] Start en robot. Skriv `programs` og trykk enter.

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

- [ ] Kjør programmet `dance`.

  Roboten begynner nå å danse! Trykk *Esc*-knappen for å stenge l
  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 e

  Hvis du vil kan du også la roboten fortsette å danse. Lag i så fal

Steg 6: 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 m
kontrollert for nesten 70 år siden. Dette gjør vi ved å bruke programm

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`. Dette skal flyt

  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 bakover ved å skrive `go back`.

Finne hjelp

Datamaskiner og roboter har et innebygd hjelpesystem. For å se h
enter. Dette gir deg en rask introduksjon til nyttige hjelpekomman
`<program>` for å få hjelp om et spesielt program. Da må du bytte u
eksempel kan du skrive `help go` for å finne hjelp om `go`-program

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

Sjekkliste

- [ ] For å få vite mer om hvordan roboten kan flytte seg kan vi skriv

  Dette viser oss at vi kan bruke `go forward`, `go back`, `go up`, `g`
  rundt. I tillegg ser vi at vi kan bruke tall for at roboten skal flytte

- [ ] Prøv `go up 2`, `go forward 10`, `go down` og lignende kommand

  Hvordan kan vi få roboten til å bevege seg sidelengs?

- [ ] Det finnes ingen kommando som får roboten til å bevege seg si
  snur roboten. For å få roboten til å gå sidelengs må vi derfor før
  `forward`. Skriv `go left` og deretter `go forward 3`.

- [ ] Vi kan også kombinere flere kommandoer i et kall. For eksempe
  roboten først går fremover 3 steg, så snur den seg mot venstre,
  oppover.

- [ ] Lek litt mer med `go`-programmet til du skjønner hvordan du flyt
  og `go right` er litt forvirrende siden roboten ikke går noe sted,

  Hva skjer dersom du ber roboten gå gjennom bakken, gjennom

Steg 7: 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 fort 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 8: Robotprogrammer

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

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 teste alle sammen på egen hånd, og se om du skjønner hvordar

  For å komme ut av listen kan du for eksempel holde mellomrom
- [ ] For å gjøre enkle eksperimenter vil vi begynne med å skrive kor enter. Husk at du skriver `exit()` for å gå ut av Lua-tolkeren og
- [ ] 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. Disse tilsvarer `go`-progra roboten dit vi vil ha den:

  `turtle.forward()`, `turtle.back()`, `turtle.turnLeft()`, `turt`

Prøv selv

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

Steg 9: Up, up, up, and aw

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

Ett av poengene med roboter er at de kan gjøre arbeid for oss. I Mine
kunne grave eller bygge.

Sjekkliste

- [ ] Start en ny robot. Gi den fuel (og kjør `refuel`). Legg også noe b
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 sprekk

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

- [ ] Roboten kan også sjekke om den har noe foran seg: Skriv `turtl`

  Du skal få svaret `true` som betyr at roboten merker at den har

- [ ] Prøv så `turtle.back()` etterfulgt av `turtle.detect()`.

  Siden roboten nå ikke har noe rett foran seg får du svaret `false`

I Steg 10 skal vi se hvordan vi kan bruke `place()` og `detect()` samm

Men først, en ting vi kunne gjøre med `go` var å flytte roboten flere ste
samme fungerer ikke med `turtle`-biblioteket. I stedet må vi bruke lø

Sjekkliste

- [ ] En enkel måte å gjøre noe et bestemt antall ganger er å bruke

  Skriv

  ```
  for i = 1, 5 do turtle.back(); end
  ```

  i Lua-tolkeren. Flytter roboten din seg 5 steg bakover?

`for`-løkker

En `for`-løkke brukes ofte når vi vet hvor mange ganger vi vil gjøre

flyttet seg 5 ganger bakover. En nyttig ting er at vi også kan følge en variabel. For eksempel kan vi skrive

```
for i = 1, 5 do print(i); end
```

Dette vil skrive tallene 1, 2, 3, 4, 5 til skjermen på hver sin linje.

Vi kan også bruke andre variabelnavn enn `i`, og vi kan starte på a

```
for tall = 10, 20 do print(tall); end
```

Sjekkliste

- [ ] Hvis vi vil kombinere flere kommandoer inne i en løkke i Lua-toll

  Prøv

  ```
  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
  ```

Oops! Vi glemte å fortelle roboten at den skulle komme ned når den b
igjen?

Steg 10: Bygg en trapp

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

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

- [ ] Begynn et nytt program ved å skrive `edit byggTrapp`. Skriv inn

```
turtle.detect()
```

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

- [ ] Kjør programmet ved å skrive `byggTrapp`. Dette programmet b
ikke gjør noe som helst. Det eneste som skjer er at roboten mer
fortalt den hva den skal gjøre etterpå.

Vi kan bruke `turtle.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                    -- endret
    turtle.up()                             -- ny linj
end                                         -- ny linj
```

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 r programmet, slik at roboten klatrer ned. Denne gangen bruker v som `detect()`, bortsett fra at den merker om roboten har en kl

Utvid programmet ditt slik:

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

while not turtle.detectDown() do            -- ny linj
    turtle.down()                           -- ny linj
end                                         -- ny linj
```

Vi sier at så lenge roboten *ikke* har en kloss under seg kan den t programmet?

Nå er vi klar til å la roboten bygge selve trappen. Det gjør vi ved en kloss.

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

while not turtle.detectDown() do
    turtle.down()
    turtle.back()                           -- ny linj
    turtle.place()                          -- ny linj
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

Prøv selv

Vi har nå laget et program som kan bygge trapper oppover. Kan du
bakken?

Lag et program som heter `gravTrapp`. For å gjøre dette trenger du

Se på kommandoene `turtle.dig()` og `turtle.digDown()`. Disse v
`turtle.digUp()` kan også være nyttig om du vil lage nok plass til a

For å vite hvor langt ned roboten skal grave, er det enkleste å bruk
skal grave 10 trinn nedover.

Hvis du vil ha en utfordring kan du la roboten grave nedover til den
gjøre dette kan du legge en **Bedrock** øverst til venstre i inventory'
`turtle.select(1)` og `turtle.compareDown()` for å finne ut om rob

 Forfatter: Geir Arne Hjelle