# ◆ Frantic Felix

*Skrevet av: Oversatt fra Code Club UK (//codeclub.org.uk)*

*Oversatt av: Lars-Erik Wollan*

*Kurs: Scratch*
*Språk: Norsk bokmål*

# Introduksjon:

I dette prosjektet skal du lage et komplett spill. Det er et plattformspill, hvor katten Felix hopper rundt, unnslipper slemminger og samler nøkler for å slippe ut av hulen. Når han kommer seg ut, fortsetter spillet i neste hule.

I del 1 av dette prosjektet kommer du til å lære hvordan vi får Felix til å flytte seg og samhandle med forskjellige ting. I del 2 skal du designe egne brett og nivåer. Til slutt skal du sette brettene sammen til et ferdig spill.

# Del 1: Lag byggeklossene

Plattformspill, som for eksempel **Manic Miner** og **Mario Bros**, handler om en figur som beveger seg rundt i en verden og hopper inn i ting. Noen ting, som vegger, stopper deg. Andre ting, som slemminger, tar livet av deg. Ting som nøkler må du samle på. Andre ting er bakgrunner og påvirker ikke spillet i det heletatt.

Det betyr at det å vite når du har kommet bort i noe er viktig. Scratch har noen klosser for å oppdage at noe berører noe annet: `berører sprite`, `berører farge?` -kloss, `farge berører?` -kloss, og `avstand til` -kloss. Men for å lage spillet, må du vite mer enn at to figurer har kommet borti hverandre; du trenger å vite hvilken **side** som berøres. Hvis du går mot en vegg til venstre, kan du ikke gå lengre til venstre. Men du kan fortsatt gå til høyre, eller hoppe, eller falle ned hvis det ikke er noe gulv å stå på.

Hvis du berører en slemming, kan dette skade deg; men hvis du berører slemmingen med beina, kan du skade den. Ingen av de innebygde Scratchklossene kan si noe om **retningen av berøringen**. Vi må derfor bygge våre egne berøringssensorer.

## Steg 1: Berøringssensor

Vi skal ha fire figurer som følger etter Frantic Felix. **Hver figur oppdager berøring i én retning og setter en variabel hvis det oppstår en berøring**. Hovedfiguren Felix bruker disse variablene for å kontrollere hvordan Felix kan bevege seg. Hver berøringssensor har en farget linje og bruker `farge berører?` -klossen for å oppdage ting. Vi bruker **svart** for gulvet og **grønn** for hinder. Vi bruker **rød** for berøringssensorene.

# Sjekkliste

- [ ] **Åpne et nytt prosjekt**. Legg til **frantic-felix/testlevel** bakgrunnen og slett den hvite **background1**.

- [ ] **Endre navn på figur Sprite1 til Felix**. Kontroller at drakten bare kan flippe venstre-høyre.

- [ ] Lag fire nye figurer fra bildene (be om hjelp hvis du ikke har disse) **frantic-felix/top**, **frantic-felix/bottom**, **frantic-felix/left**, og **frantic-felix/right**. Disse figurene skal bli berøringssensor. Gi de nye figurene navnene **topp**, **bunn**, **venstre**, og **høyre**. Kontroller at draktene deres ikke kan flippes eller roteres.

- [ ] Lag fire variabler (for hver av figurene): `blokkert topp`, `blokkert bunn`, `blokkert høyre` og `blokkert venstre`.

- [ ] Lag dette skriptet for hver av sensorene:

```
når grønt flagg klikkes // eksempel for høyre sensor
for alltid
  gå til [Felix v]
    hvis <farge [rød v] berører [grønn v] ?>
    sett [blokkert høyre v] til [1]
    ellers
    sett [blokkert høyre v] til [0]
    slutt
slutt
```

- [ ] Endre variablene for hver av sensorene. Den nederste sensoren trenger en **eller** kloss slik at den setter **blokkert bunn** hvis den berører grønn eller svart.

**Tips**: Det er enklere å velge fargene i én figur, for så å kopiere skriptet over til de andre figurene og oppdatere variablene som blir satt. Da slipper du å finne de riktige fargene fire ganger. Inntil videre trenger Felix bare et skript for å følge musepekeren evig.

---

## Test prosjektet

**Trykk på det grønne flagget.**

Du bør se at Felix følger etter musepekeren, omringet av et rødt rektangel. Rektangelet er en berøringssensor. Hvis du følger med på variablene, ser du at de endrer seg når du drar Felix rundt og når han berører ulike deler av skjermen. For øyeblikket beveger Felix seg gjennom plattformene og grønne hindre. Det skal vi fikse nå.

# Steg 2: Bevegelse, gulv og falling

## Sjekkliste

- [ ] Neste steg er å få Felix til å gå. Vi bruker **venstre** og **høre piltast** for å flytte ham til venstre og høyre. Hvis han ikke står på et gulv, faller han ned (vi fikser hopping etterpå). Vi vil at når man trykker på venstre piltast, så skal Felix peke til venstre og flytter seg litt. Men vi vil ikke at han skal flytte seg hvis det er en hindring i veien.

- [ ] Vi kunne brukt `når tast trykkes` for å flytte Felix, men det gir en hakkete bevegelse. Det blir en mykere bevegelse hvis du bruker `hvis tast pil venstre trykket?` -klosser i en `for alltid` -løkke. Berøringssensorer betyr at vi må teste **blokkert venstre** variabelen i `hvis` , og bruke en **og kloss** slik at Felix bare beveger seg til venstre når man trykker på en tast og **blokkert venstre** er 0. Vi må gjøre tilsvarende for bevegelse til høyre.

- [ ] Vi kunne animere Felix sine bein i samme kloss, men da beveger de seg for raskt. Gjør animasjonen i en egen `for alltid` -løkke under en annen `når grønt flagg klikkes` -hatt.

- [ ] Det siste vi må gjøre er å ordne **falling**. Hvis Felix ikke står på fast grunn vil vi at han skal falle ned. Det ordner vi med en `for alltid` -løkke under en `når grønt flagg klikkes` -hatt.

```
når grønt flagg klikkes // håndter flytting
for alltid
    hvis < <tast [pil venstre v] trykket?> og ((blokkert venstre)=(0
    pek i retning (-90)
        gå (2) steg
    slutt
    hvis < <tast [pil høyre v] trykket?> og ((blokkert høyre)=(0))>
    pek i retning (90)
        gå (2) steg
    slutt
slutt

når grønt flagg klikkes // beveg Felix
for alltid
  hvis < <tast [pil venstre v] trykket?> eller <tast [pil høyre v] t
      neste drakt
        vent (0.1) sekunder
    slutt
slutt

når grønt flagg klikkes // håndter falling
for alltid
    hvis <(blokkert bunn)=(0)>
        endre y med (-2)
    slutt
slutt
```

#  Test prosjektet ditt

**Bruk musen for å dra Felix et sted på scenen og trykk så på det grønne flagget**. Hvis Felix hopper til muspekeren, må du fjerne det skriptet! Du bør kunne bruke venstre og høyre knapper for å få Felix til å gå fra side til side. Hvis han ikke står på gulvet, bør han falle sakte nedover.

- [ ] Vi vil også at berøringssensorene ikke skal vises. Det er ikke mulig å bruke en skjult blokk, for da vil ingen berøringer bli oppdaget. Istedet, legg til en `sett gjennomsiktig effekt til 100` -kloss rett under den grønn flag hatt'en i hver kollisjonssensor. Dette gjør figuren gjennomsiktig, uten å å skjule den.

## Test prosjektet ditt

**Nå du klikker på det grønne flagget**, skal kollisjonssensorene forsvinne. De dukker opp igjen når du trykker på det røde stoppskiltet.

# Steg 3: Hopping

**Det er et par utfordringer med hopping.**

- Vi vil ikke at Felix **faller** hvis han er **på vei opp**.
- Vi vil ikke at Felix skal hindres av gulv når han er på vei opp, men vi vil fortsatt at han skal stoppes av gulvet når han **er på vei ned**.
- Vi vil ikke at Felix skal *_hoppe opp* i undersiden av et **grønt hinder**.
- Vi vil at hoppet skal være en myk bevegelse, så vi vil ikke at Felix flytter seg for raskt.
- Felix skal bare kunne hoppe hvis han **står på gulvet**. Spillet blir litt for enkelt hvis Felix kan hoppe fra løse luften.

## Sjekkliste

- [ ] Det som gjenstår nå, er å få Felix til å hoppe. La oss bruke **mellomromstasten** for å få Felix til å hoppe.
- [ ] For å kontrollere hoppet, bruker vi en ny variabel, `hopphøyde` . Hvis denne er høyere enn 0, er Felix på vei opp. Hvis den er 0, faller han (eller har falt), som beskrevet over.

- [ ] Vi vil at Felix skal hoppe opp til **100 pixler**. Legg til en ny `hvis` -kloss inni `for alltid` -klossen som håndterer tastetrykk. Hvis vi trykker **mellomromstasten** og Felix står på gulvet (variablen `blokkert under` er satt til 1, så setter vi `hopphøyde` til 100.

- [ ] Vi må endre falle-skriptet. I `for alltid` , trenger vi en `hvis–ellers` -kloss som oppdater hvis Felix hopper opp eller ikke. Kravet for denne `hvis–ellers` -klossen er at hvis `hopphøyde` er **større enn 0**. Den eksisterende faller ned `hvis` -klossen går inn i ellers-delen av den nye `hvis–ellers` -klossen.

- [ ] Når vi vet at Felix hopper opp, må vi sjekke om hodet hans treffer noe. Hvis `blokkert topp` er **1**, sett `hopphøyde` til **0**. (Dette gjør at Felix ikke kan hoppe inn i hindringer). **Ellers, flytt Felix opp 10 og reduser hopphøyde med 10.**

- [ ] Du ender opp med noe tilsvarende dette:

```
når grønt flagg klikkes // håndtere falling
for alltid
  hvis <(hopphøyde)=(0)>
    hvis <(blokkert topp)=(1)>
      sett (hopphøyde) til (0)
        ellers
      endre y med (10)
            endre [hopphøyde v] med (-10)
        slutt
    ellers
    hvis <(blokkert bunn)=(0)>
      endre y med (-2)
        slutt
    slutt
slutt
```

#  Test prosjektet ditt

**Trykk på det grønne flagget**. Kan Felix hoppe? Hopper han fra en plattform til en annen? Faller han hvis han går over kanten? Hva om han hopper oppover over kanten av en plattform? Hva om han prøver å hoppe under den grønne blokken til høyre? Hva skjer når du trykker på mellomromstasten mens Felix faller?

# Steg 4: Nøkler og mål

**Vi har klart å få Felix til å bevege seg rundt i verdenen. Nå må vi få han til å klare ett nivå**.

Vi plasserer tre nøkler rundt i hulen. Felix kan plukke dem opp ved å gå til dem. Når han har samlet alle tre, kan han klatre til en redningskapsel og redde seg selv ut av hulen.

- [ ] Vi lager en ny variabel, `nøkler igjen`, som holder orden på hvor mange nøkler som gjenstår. Et nytt skript på **scenen** skal sette denne til **3** når det **grønne flagget** klikkes.

- [ ] Både nøklene og redningskapselen må være figurer. (Bruk **frantic-felix/key** for nøklene og **frantic-felix/escape-pod** for kapselen.)

- [ ] Hver nøkkel trenger **to skript**: det første plasserer nøklen på riktig sted, med størrelse og vinkel, og en `for alltid`-løkke for å endre fargen (som gjør det enklere å se den på skjermen). Det andre skriptet er en `for alltid`-løkke som venter til Felix kommer bort til nøkkelen. Når dette skjer, skjuler skiptet nøkkelen og reduserer antall nøkler.

- [ ] Redningskapselen er et litt vanskeligere skript. Den bruker en `for alltid`-løkke for å vente på at `nøkler igjen` blir **0**. Med en gang dette skjer, begynner kapselen å blinke (for å vise spilleren at de kan komme seg ut). Så kan vi bruke en annen `hvis` for å oppdage når Felix berører den blinkende redningskapselen. Så snart han gjør det, sender kapselen en `seier`-melding og sier “Du vant!”. Felix svarer på meldingen ved å gjemme seg.

```
når grønt flagg klikkes
gå til x:(220) y:(-125)
for alltid
  hvis <(nøkler igjen) = [0]>
    endre [farge v] effekt med (25)
        hvis <berører [Felix v]?>
      send melding [seier v]
            si [Du vant!]
        slutt
    slutt
slutt
```

## Test prosjektet ditt

**Trykk på det grønne flagget**.

# Steg 5: Slemminger og dødelige omgivelser

**Nå er det på tide med slemminger!**

Det skal være to typer farlige ting. En type vil være slemminger som går rundt og skader Felix hvis kan krasjer inn i dem. Den andre typen er farlige ting i bakgrunnen.

**La oss først lage en slemming. Den vil bare bevege seg langs en fast sti.**

## Sjekkliste

- [ ] **Lag en ny figur**, bruk hvilken som helst drakt. Den bør være samme størrelse som Felix (Vi brukte **things/flower-vase** drakt, og gjorde den fire steg mindre). Slemmingen trenger bare et enkelt skript som beveger den og oppdager om den berører Felix.

- [ ] Lag tre `hvis` -klosser inni en `for alltid` -løkke. Den første sjekker om en slemming berører Felix; hvis den gjør det, sender den en `tap` -melding. De to andre `hvis` -klossene sjekker om slemmingen har nådd slutten av stien; hvis den har det, snur slemmingen. Tilslutt, tar slemmingen to steg. (Ved å bruke `gå` isteden for `gli` -klosser blir det enklere å kontrollere hvor raskt slemmingen går.)

  **Vi trenger ikke å bruke kollisjonssensor her, da vi ikke bryr oss om på hvilken side Felix berører slemmingen.**

```
når grønt flagg klikkes
gå til x:(-50) y:(47)
pek i retning (-90)
for alltid
  hvis <berører [Felix v]?>
    send melding [tap v]
    slutt
    hvis < (x-posisjon) > [-200] >
    pek i retning (90)
    slutt
    hvis < (x-posisjon) > [-50] >
    pek i retning (-90)
    slutt
    gå (2) steg
slutt
```

- [ ] Legg skript til både Felix og redningskapselen slik at den svarer på `tap` -meldingen. Felix skal bare skjule seg selv. Kapselen skal si "Du tapte!".

---

#  Test prosjektet ditt

**Trykk på det grønne flagget**. Beveger vasen seg? Stopper den og snur ved kanten? Hva skjer når Felix går inn i den? Hva skjer hvis Felix hopper på den, fra oversiden eller undersiden? Forsvinner Felix? Sier kapselen ifra når du har tapt? Kan du forstatt vinne spillet?

- [ ] **Neste, farlige ting!** La oss si at alt som er lyseblå er dødlig for Felix. Last inn bakgrunnen **frantic-felix/level2**, som har en blå rose på det øverste nivået. Legg til enda et skript på Felix, under et grønt flagg hatt:

```
for alltid
    hvis <berører fargen [blå v]?>
        send melding [spill slutt v]
    slutt
slutt
```

# Test prosjektet ditt

**Trykk på det grønne flagget**. Dør Felix hvis han kommer borti den blå rosen? Hva skjer når han berører andre ting?

# Oppsummering

**Det du har laget nå er et veldig enkelt plattformspill. Akkurat nå, er det et ganske kjedelig spill. Men det er ikke poenget. Dette spillet er en verktøykasse som du kan bruke til å lage egne spill og brett. I neste del skal du lage egne brett.**

# Del 2: Brettdesign

**Forrige gang bygde du alle delene et plattformspill består av. Nå skal du bruke disse delene for å lage egne brett..**

Kort oppsummert gjorde du:

- Felix kan gå til venstre og høyre og han kan hoppe.
- Felix faller ned hvis han ikke står på gulvet.
- Felix kan ikke gå gjennom grønne hindere.
- Blå ting i bakgrunnen og slemminger tar livet av Felix.
- Slemminger beveger seg i faste ruter.
- Slemminger kan ta livet av Felix hvis han kommer borti dem.
- Felix kan samle nøkler ved å gå bort til dem.
- Når Felix har samlet alle nøklene, kommer en redningskapsel til syne og han kan rømme i sikkerhet (eller til neste nivå).

Dette kan være byggeklossene dine. **Bruk dem til å lage egne brett.**

Du kan lage en hel rekke med brett som Felix må klare. Neste gang skal vi se på hvordan vi kan knytte brettene sammen.

Brett kan ha store eller små plattformer, mange eller få plattformer. Det kan være mange slemminger eller ingen. Det kan være mange hindringer eller dødelige ting i bakgrunnen. Prøv å lage flere måter man kan klare brettet på, selv om et kanskje er enklere enn et annet. Tenk over hvor vanskelig eller enkelt brettet er.

Du kan endre de spesielle fargene (svart, grønn og blå), men da må du **oppdatere fargeberøringsblokkene** i alle skriptene. Det må være samme farge i alle brett. (Du kunne hatt forskjellige farger i forskjellige brett, men det betyr at du måtte legge inn mange `eller` -klosser rundt fargeberøringsblokkene.)

**Test brettene dine**. Hvis du har tid, opprett brett i Scratch og spill dem. Sjekk at de ikke er **for vanskelige** og **ikke for enkle**. Hvis du programmerer brettene, må du **lagre bakgrunnen du lager** og notere **start posisjon** for Felix, nøkler, og eventuelle slemminger. I tillegg må du huske retningen slemmingene beveger seg i og hvor langt de går.

Hvis du har laget et par brett og lagt dem inn i Scratch, prøv disse ekstraoppgavene:

## Utvidelse: Tramping

**Hva om slemmingene døde hvis de ble trampet på?** Kanskje du kan legge til et skript på slemmingene som gjør noe hvis slemmingen berører den nederste kollisjonssensoren.

## Utvidelse: Kraftpiller

**Du kan lage en "kraftpille" som gjør at Felix kan ødelegge slemmingene**. Når Felix tar kraftpillen, ødelegger Felix slemmingene han berører. Effekten går over etterhvert

Det er opp til deg hvordan du får dette til å fungere. Kanskje vil du at slemmingene skal endre utseende når Felix kan ta dem?

# Test prosjektet ditt

**Trykk på det grønne flagget**.

# Del 3: Sett sammen spillet

**Nå har du noen brett. Du har nok verktøy til å få dem til å fungere. Det siste vi må gjøre er å sette delene sammen til et ferdig spill!** Hvis du klarer å sette sammen brettene kjapt, kan du ta en titt på aktivitet 2 and 3 før du begynner å spille spillet ditt.

# Steg 1: Vise et nytt brett

**Når Felix klarer ett brett, må vi flytte til neste**. Det betyr at vi må få redningskapselen til å sende en `start brett` -melding (istedet for `seier` -meldingen) når Felix har funnet alle nøklene og klatret inn i kapselen. Vi kan bruke start brett melding for å klargjøre neste brett. Vi trenger også en variabel, `nåværende nivå` , som settes av redningskapselen før den sender `start brett` -melding.

Alt må stort sett svare på `start brett` -meldinger for å klargjøre neste brett.

**Scenen** må vise riktig bakgrunn. Hver av slemmingene, alle nøkler, og redningskapselen trenger å flytte til **riktig start posisjon**. Slemmingenes rute trenger **oppdatering**. Felix må flyttes til sin **nye startposisjon**. Først da er brettet klart til start.

De fleste figurene må svare på `start brett` -meldingen istedet for det grønne flagget. Det betyr at du må **endre hatt-klossene** i de fleste skript.

---

#  Test prosjektet ditt

**Fortsett å teste spillet etter hver endring. Husk å teste de delene som du allerede har endret, slik at de fortsatt virker som planlagt.**

Vi kommer ikke beskrive alle endringene, men vi skal vise hvordan Felix sitt oppdaterte skript for å vise hva som må gjøres.

```
når jeg mottar [start brett v]
    gå til x: (element [nåværende nivå v] av [xs v]) y: (element [nåværer
pek i retning (element [nåværende nivå v] av [retning v])
for alltid
  hvis <<tast[pil venstre v] trykket?> og <(blokkert venstre)=(0)>
    pek i retning (-90)
        gå (2) steg
    slutt
    hvis <<tast [pil høyre v] trykket?> og <(blokkert høyre)=(0)>
        pek i retning (90)
        gå (2) steg
    slutt
    hvis <<tast [mellomrom v] trykket?> og <(blokkert bunn)=(1)>
    sett [hopphøyde v] til [100]
    slutt
slutt
```

Du legger kanskje merke til at start verdiene **x**, **y**, og **retning** for Felix settes med lister. Vi laget noen lister for hver figur (hver liste privat til den figuren) for å lagre verdiene vi trenger for den figuren. Du trenger en liste for hver ting du lagrer. Du trenger ikke bruke lister, du kan bruke `hvis` -klosser som sjekker hvilket brett du er på og gjør riktig ting basert på dette.

Og her er redningskapselen, som håndterer all nivå-endringen:

```
når grønt flagg klikkes
sett [nåværende nivå v] til [1]
send melding [start brett v]

når jeg mottar [start brett v]
gå til x: (element [nåværende nivå v] av [xs v]) y: (element [nåværende r
for alltid
  hvis <[nøkler å ta v] = [0]>
      endre [farge v] effekt med (25)
        hvis <berører [Felix v]?>
      hvis <[nåværende nivå v] = <lengden av [nøkler per brett v]>
        si [Du vant!!]
                send melding [seier v]
                stopp [alle v] :: control
            ellers
        endre [nåværende nivå v] med (1)
                send melding [start brett v]
            slutt
        slutt
    slutt
slutt
```

# Steg 2: Spill!

**Du har laget spill**. De andre i kodeklubben har laget spill. Spill de forskjellige spillene! Klarer du å runde deres spill? Kan de andre runde spillet ditt?

## Utvidelse 1: Flere liv

Felix trenger kanskje et par sjanser, eller et par liv for å klare seg igjennom brettene. **Legg til en ny figur** med tre drakter, som viser **ett**, **to**, og **tre hjerter**.

Bruk frantic-felix/1-heart, frantic-felix/2-heart, and frantic-felix/3-heart draktene. Plasser figuren i hjørnet av scenen. Når det **grønne flagget** blir trykket, skal det vises tre hjerter. Hver gang det mottar en tap melding, skal det vises et hjerte mindre. Isteden for å vise ingen hjerter, skal figuren skjules og sende en `slutt over` -melding.

`spill slutt` - og `seier` -meldinger bør håndteres av en **ny figur**, som **skjules** når det grønne flagget trykkes og viser riktig plakat når `spill slutt` - og `seier` -meldinger sendes. Denne figurer skal også stoppe alle skript når den viser seg selv.

Du må endre hvordan redningskapselen håndterer vinning og tap, da dette er ansvaret for liv og ferdigplakat figurene.

Kanskje er slemmingen litt for kjappe på foten, de tar Felix før brettet er kommet igang. Hvis Felix taper mer enn et liv når han kommer borti en slemming, `skjul` slemmingen så snart den sender `tap` -meldingen. Det gir resten av spillet (inkludert Felix) tid å nullstille før slemmingene oppdager en ny kollisjon med Felix.

# Utvidelse 2: Tidsbegrensninger

**Ta tiden på Felix!** Bruk et **Timer Scratchkort** for å legge til en tidsbegrensning. Hvis timeren løper ut, sender den `spill over` -melding. Husk å **nullstille timeren** når et nytt brett starter.