△ Python: Rørsler

PÅ BOKMÅL | LAST NED PDF

Introduksjon

Ein veldig interessant bieffekt av å ha eit akselerometer er at det kan merke rørsler. Med andre ord kan MicroPython oppfatte rørsler du gjer med micro:bit-en.

MicroPython kan oppfatte følgjande rørsler: `up`, `down`, `left`, `right`, `face up`, `face down`, `freefall`, `shake`, `3g`, `6g` og `8g`. Rørslene blir alltid registrert og brukt som tekst i koden. Dei fleste namna er nok enkle å forstå (det er engelske uttrykk for rørslene), men kanskje ikkje dei tre siste. Kodene `3g`, `6g` og `8g` står for når micro:bit-en vert utsett for desse nivåa av g-krefter. Ein g-kraft er forholdet mellom den aktuelle akselerasjonen og standardakselerasjonen 1 g på jordoverflata, altså tyngdekrafta. Til dømes møter astronautar i ei romferge kring 3-4 g i gjennomsnitt når dei tek av.

For å lese kva rørsle som skjer brukar me metoden `accelerometer.current_gesture`. Resultatet er ei av dei rørslene du såg lista opp over. Til dømes vil koden under berre gjere micro:bit-en lykkeleg når den peikar oppover:

```
from microbit import *

while True:
    gesture = accelerometer.current_gesture()
    if gesture == "face up":
        display.show(Image.HAPPY)
    else:
        display.show(Image.ANGRY)
```

Sidan me vil at micro:bit-en skal reagere på endringar i miljøet sitt, så brukar me ei `while`-løkke. Inne i løkka blir det sjekka kva rørsle som skjer, og den blir lagra i variabelen `gesture`. Så sjekkar `if`-setninga om `gesture` er lik `face up` (Python brukar `==` for å sjekke om noko er likt, medan `=` er for å gi innhald til variablar slik som då `gesture` fekk ei rørsle). Viss rørsla er `face up` skal displayet vise eit smilefjer. Elles er micro:bit-en sur på oss!

Magic-8

Ein "Magic-8"-ball er eit leiketøy som vart oppfunne på 1950-talet. Ideen er at ein stiller eit ja/nei-spørsmål, ristar kula, og ventar på at den skal gi eit svar. Det er relativt enkelt å lage eit program som gjer det same:

```
from microbit import *
import random

answers = [
    "It is certain",
    "It is decidedly so",
    "Without a doubt",
    "Yes, definitely",
```

```
    "You may rely on it",
    "As I see it, yes",
    "Most likely",
    "Outlook good",
    "Yes",
    "Signs point to yes",
    "Reply hazy try again",
    "Ask again later",
    "Better not tell you now",
    "Cannot predict now",
    "Concentrate and ask again",
    "Don't count on it"
    "My reply is no",
    "My sources say no",
    "Outlook not so good",
    "Very doubtful",
]

while True:
    display.show("8")
    if accelerometer.was_gesture("shake"):
        display.clear()
        sleep(1000)
        display.scroll(random.choice(answers))
```

Mesteparten av programmet er ei liste som heiter `answers`. Det faktiske spelet er inne i `while`-løkka på slutten.

Standardtilstanden til spelet er å vise talet 8. Men programmet må òg oppdage når det blir rista. Metoden `was_gesture` brukar argumentet sitt (her er det teksten `"shake"` fordi me vil oppdage risting) til å returnere anten `True` eller `False`. Viss einings vart rista, så vil `if`-setninga gå inn i blokka der den fyrst fjernar teksten på skjermen, ventar eitt sekund (så brukaren trur at programmet tenker seg godt om), for så å vise eit tilfeldig valt svar.

Er ikkje dette det beste programmet nokon sinne?

Utfordring: Litt juks

Som ein ekte spåmann eller spåkvinne er det viktig at du kan påverke resultatet til å alltid vere positivt eller negativt.

Sjekkliste

- [ ] Skriv om koden slik at ved å trykke på `A` medan den vert rista gir eit *positivt* resultat.
- [ ] Skriv om koden slik at ved å trykke på `B` medan den vert rista gir eit *negativt* resultat.
- [ ] Skriv om koden slik at ved å trykke både på `A` **og** `B` gir eit nøytralt resultat.

Merk at koden framleis må virke viss ingen knappar trykkast, altså skal berre risting vise eit tilfeldig resultat.