Micro:bit

Python: Rotasjon og fall

Skrevet av: Oversatt fra microbit-micropython.readthedocs.io (https://microbit-micropython.readthedocs.io/en/latest/tutorials/gestures.html)

Oversatt av: Øistein Søvik og Susanne Rynning Seip

Kurs: Microbit
Tema: Elektronikk, Tekstbasert
Fag: Programmering, Matematikk
Klassetrinn: 5.-7. klasse, 8.-10. klasse, Videregående skole

Denne oppgaven er en del av oppgavesamlingen *Programmering i micro-python* og bygger videre på Python: Tilfeldig (../python_random/python_random_nb.html).

Vi anbefaler at du laster ned og skriver koden din i mu editor (https://codewith.mu/) når du jobber med disse oppgavene. Instruksjoner for hvordan man laster ned Mu finner du på nettsiden via linken.

Når Mu er installert kan du koble micro:biten din til datamaskinen via en USB-kabel. Skriv koden din i editor-vinduet og trykk på “Flash”-knappen for å laste koden over på micro:biten. Hvis det ikke fungerer, sørg for at micro:biten har dukket opp som en USB-enhet på datamaskinen din.

Introduksjon

En virkelig interessant bieffekt av å ha ett akselerometer er at den kan merke bevegelser, med andre ord dersom du beveger micro:bit'en på en viss måte kan MicroPython oppfatte dette.

MicroPython kan oppfatte følgende bevegelser: `up`, `down`, `left`, `right`, `face up`, `face down`, `freefall`, `3g`, `6g`, `8g`, `shake` og bevegelser er alltid representert som tekststrenger. Mens de fleste av navnene burde være åpenbare, så er `3g`, `6g` og `8g` bevegelser som inntreffer når enheten møter disse nivåene av g-krefter. En g-kraft er forholdet mellom aktuell akselerasjon og standardakselerasjonen 1 g på jordoverflaten. For eksempel møter astronauter i en romferge rundt 3-4g i snitt når de tar av.

For å lese hvilken bevegelse som skjer brukes metoden `accelerometer.current_gesture`. Resultatet er en av de navngitte bevegelsene listet ovenfor. For eksempel så vil koden under bare gjøre enheten din lykkelig når den peker opp:

```
from microbit import *

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

Igjen, siden vi ønsker at enheten skal reagere på endringer i miljøet så bruker vi en `while` -løkke. Inne i løkka så leses bevegelsen som skjer i variabelen `gesture` . Mens `if` -setningen sjekker om `gesture` er likt `face up` (Python bruker `==` for å teste likhet, ett enkelt likhetstegn `=` brukes for å deklarere variabler -- akkurat som vi deklarerte at bevegelsen skulle leses inn i `gesture` variabelen). Dersom bevegelsen er likt `face up` så skal displayet vise et smilefjes. Ellers så er enheten sur på oss!

Magic-8

En Magic-8 ball er et leketøy oppfunnet på 1950-tallet. idéen er å spørre ett ja/nei spørsmål, riste kula og vente for at den skal avsløre sannheten. Det er relativt enkelt å gjøre dette om til et program:

```
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 en liste kalt `answers`. Det faktiske spillet er inne i `while`-løkka på slutten.

Standard tilstanden til spillet er å vise tallet `"8"`. Men programmet må og oppdage når det ristes. Metoden `was_gesture` bruker argumentet sitt (i dette tilfellet tekststrengen `"shake"` fordi vi vil oppdage risting) til å returnere enten `True` eller `False`. Dersom enheten ble ristet så vil `if`-setningen gå inn i blokken hvor den først fjerner all tekst på skjermen, venter ett sekund (så brukeren får en illusjon av at programmet ditt tenker på svaret) for deretter å vise et tilfeldig valgt svar.

Er ikke dette det beste programmet noensinne skrevet?

Utfordring: Litt juks

Som en sann spåmann eller spåkvinne er det viktig at du kan påvirke resultatet til og alltid være positivt eller negativt.

Sjekkliste

- [ ] Skriv om koden slik at ved å trykke på `A` mens den ristes vises ett *positivt* resultat.
- [ ] Skriv om koden slik at ved å trykke på `B` mens den ristes vises ett *negativ* resultat.
- [ ] Skriv om koden slik at ved å trykke både på `A` **og** `B` vises ett nøytralt resultat.

Merk at koden skal fortsatt virke som før om ingen knapper trykkes, med andre ord dersom enheten bare ristes skal den vise et tilfeldig resultat.

Neste oppgave i samlingen er Python: Retninger (../python_direction/python_direction_nb.html). Klikk videre for å fortsette gjennom samlingen.