Pong Micropython

Singleplayer:

Hvordan spiller man singleplayer pong?

Singleplayer pong bruker en microbit og en 5x5 display på microbiten selv. Displayet går fra 0 til 4, så -1 og 5 et utenfor spillefeltet. Spilleren er en slags "vegg" som man kan bevege med a og b knappene på microbiten, og det finnes en ball som beveger seg rundt på spillefeltet. Målet er å treffe ballen med spilleren sånn at den ikke går på den bakerste veggen. Man kan tape hvis ballen går bak spilleren, men det finnes ingen motstander eller en måte å vinne på.

Hvordan funker spilleren?

Spilleren er en slags vegg som består av to pixler. Den har egenskapen *'wallBrightness'* som bestemmer hvor lys spilleren er. Veggen er laget av et bilde med bruk av en array.

- **X-verdien:** Først skriver man hvor bred veggen er, altså 5 pixler.
- **Y-verdien:** Etter det kommer høyden, altså 1 pixel.
- **Lysstyrke på alle pixler:** Her bruker vi en slags array. Spilleren er jo egentlig to pixler bred, så tre av verdiene i arrayen må være 0, og to av dem må være "wall Brightness". Vi vil at veggen er ca i midten av den bakerste rekken, så dette er resultatet vårt: [0,wallBrightness,wallBrightness,0,0].

Og sånn kan man lage bildet som funker som en spiller.

```
4  wallBrightness = 9
5  wall = Image(5,1, bytearray([0,wallBrightness,wallBrightness,0,0]))
```

Man kan bevege spilleren ved bruk av a og b knappen. Den kan bare bevege seg hvis den ikke er på kanten av spillefeltet (sånn at den ikke går ut av microbiten).

```
28     if button_a.was_pressed() and wall.get_pixel(0,0) == 0:
29         wall = wall.shift_left(1)
30     if button_b.was_pressed() and wall.get_pixel(4,0) == 0:
31         wall = wall.shift_right(1)
32     display.show(wall)
```

Hvordan beveger ballen seg?

Ballen har mange forskjellige variabler som har en effekt på bevegelsen og funksjonen til ballen.

```
7  bBr = 7
8  xPos = 1
9  yPos = 5
10 deltaX = 1
11 deltaY = -1
```

- **Ball brightness (bBr)** som bestemmer hvor lys ballen er,
- **y-Posisjon (yPos)** som bestemmer posisjonen til ballen på y-aksen,
- **x-Posisjon (xPos)** som bestemmer posisjonen til ballen på x-aksen,
- **deltaX** som bestemmer hvor ballen kommer til å ende opp ved neste skritt på x-aksen,
- **deltaY** som bestemmer hvor ballen kommer til å ende opp ved neste skritt på y-aksen.

For å vite hvor ballen skal gå i neste skritt, må man regne ut den nåværende posisjonen plus deltaX og -Y. Dette blir brukt for å regne ut om ballen treffer en vegg eller en spiller osv. (xPosTest og yPosTest).

```
33     xPosTest = xPos + deltaX
34     yPosTest = yPos + deltaY
```

Den sjekker først om ballen er bak veggen. Dette gjør at spilleren har tapt og at ballen går tilbake til sin opprinnelig posisjon.

```
35     if yPos == 0:
36         losses += 1
37         resetBall()
```

```
13 def resetBall():
14     global xPos
15     global yPos
16     global deltaX
17     global deltaY
18     xPos = 1
19     yPos = 5
20     deltaX = 1
21     deltaY = -1
```

Etter det sjekker den om ballen kommer til å være utenfor microbiten (at den treffer på en vegg). Hvis den er det blir deltaX eller deltaY multiplisert med -1, dvs. ballen går i motsatt retning.

```
39          if xPosTest >= 5 or xPosTest <= -1:
40              deltaX *= -1
41              xPosTest = xPos + deltaX
42          if yPosTest >= 5:
43              deltaY *= -1
44              yPosTest = yPos + deltaY
```

I neste skritt sjekker den om ballen treffer spilleren. Den gjør det idet den sjekker om pixelen over ballen eller ved neste skritt har likens *'brightness'* som spilleren. Dette gjør også at ballen endrer retning. Hvis spilleren er rett over ballen (ballen treffer spilleren direkte) blir bare deltaX endret, men hvis den treffer spilleren på neste skritt (ballen treffer hjørnet av spilleren) blir deltaX *og* deltaY endret.

```
45      if display.get_pixel(xPos, yPos - 1) == wallBrightness:
46          deltaY *= -1
47      if display.get_pixel(xPosTest, yPosTest) == wallBrightness and not display.get_pixel(xPos, yPos - 1) == wallBrightness:
48          deltaY *= -1
49          deltaX *= -1
50          xPosTest = xPos + deltaX
```

På grunn av noen spesialfall hvor ballen treffer veggen, spilleren og veggen igjen på et skritt må xPosTest sjekkes igjen.

```
50              xPosTest = xPos + deltaX
51              if xPosTest >= 5 or xPosTest <= -1:
52                  deltaX *= -1
```

Etter at alt har blitt testet er den neste posisjonen til ballen klar og den vises på microbiten. Så blir alt gjentatt helt man slutter spillet.

```
53      xPos += deltaX
54      yPos += deltaY
55      display.set_pixel(xPos,yPos,bBr)
56      sleep(750)
```

Multiplayer:

Hvordan spiller man multiplayer pong?

I motsetning til singleplayer, bruker denne versjonen tre microbits, to spillere og et scoreboard som kan starte og slutte spillet.

Hvordan er spilleren forskjellig i multiplayer?

Spilleren endrer seg ikke i multiplayer. Den eneste forskjellen er selvfølgelig at det finnes to i stedet for en spiller, men måten de funker er akkurat lik som hos singleplayer.

Hvordan er ballen forskjellig i multiplayer?

Som hos spilleren finnes det ikke mange forskjeller i hvordan ballen funker mellom singleplayer og multiplayer, men det finnes to ting som er veldig annerledes. Først, når ballen går bak spilleren sender den et signal til scoreboarden og den andre spilleren at den har tapt. F.eks. Hvis spiller en (p1) taper, sender det “P1 lost” til de andre microbits og det viser L (for lose). Spiller to tar det opp og viser W (for win). Hva den tredje microbiten gjør med denne informasjonen blir forklart senere.

P1:

```
62          if yPos == 0:
63              losses += 1
64              radio.send("P1 lost")
65              display.show("L")
66              sleep(1000)
67              resetBall()
68              isBallOwner = False
69              isNextBallOwner = True
```

P2:

```
61          if yPos == 0:
62              losses += 1
63              radio.send("P2 lost")
64              display.show("L")
65              sleep(1000)
66              resetBall()
67              isBallOwner = False
68              isNextBallOwner = True
```

Den andre forskjellen er at ballen går til det andre spillefeltet hvis den treffer veggen mellom begge spillerne. Hvis den treffer denne veggen sender spilleren faktorene til ballen (X-posisjonen, Y-posisjonen, deltaX og deltaY) med et komma mellom hver av dem og spilleren har ikke ballen lenger.

```
71              if yPosTest >= 5:
72                  radio.send(str(xPos) + "," + str(yPosTest) + "," + str(deltaX) + "," + str(deltaY))
73                  isBallOwner = False
```

Den andre microbiten tar opp denne informasjonen og deler opp meldingen med “,”. Dette gjør at man har tilgang til all informasjon som trengs fra bare en melding.

```
47          elif received != "P2 lost":
48              isBallOwner = True
49              receivedValues = received.split(",")
50              xPos = 4 - int(receivedValues[0])
51              yPos = int(receivedValues[1])
52              deltaX = 0 - int(receivedValues[2])
53              deltaY = 0 - int(receivedValues[3])
```

‘ReceivedValues’ er da en type liste av alt som spilleren får. Som man kan se kan man få adgang til xPos ved å bruke *‘receivedValues[0]’* (den første verdien). Fordi den andre spilleren sitt koordinatsystem er omvendt av det av den første må verdiene bli endret litt (som blir vist i bildet oppe). Etter det er ballen hos den andre spilleren og spillet går videre.

Hva gjør den tredje microbiten?

Den tredje microbiten funker som scoreboard (en som viser poengsummene til spillerne) og den kan starte og slutte spillet. Man kan trykke a på begynnelsen for å starte spillet. Dette sender et “ready” signal til spillerne og den som er *‘isNextBallOwner’* (P1 på begynnelsen) begynner med ballen.

Scoreboard:

```
40      else:
41          if button_a.was_pressed():
42              gameStart = True
43              radio.send("ready")
```

P1 + P2:

```
43          elif received == "ready":
44              if isNextBallOwner:
45                  isBallOwner = True
46                  isNextBallOwner = False
```

Når spillet er i gang venter den på signaler fra spillerne. Hvis en av spillerne sender at de har tapt viser microbiten den oppdaterte poengsummen. Fordi display tar litt tid sender microbiten et signal når den er klar. Etter det går spillet videre.

```
15            if received == "P1 lost":
16                p2Wins += 1
17                display.scroll(str(p1Wins) + "-" + str(p2Wins))
18                radio.send("ready")
19            elif received == "P2 lost":
20                p1Wins += 1
21                display.scroll(str(p1Wins) + "-" + str(p2Wins))
22                radio.send("ready")
```

Man kan også trykke a eller b-knappen i spillet. A-knappen slutter spillet og setter poengsummen til 0 for begge spillerne.

```
23        if button_a.was_pressed():
24            radio.send("end game")
25            p2Wins = 0
26            p1Wins = 0
27            display.scroll(str(p1Wins) + "-" + str(p2Wins))
28            gameStart = False
```

B-knappen gjør det samme men den regner også ut hvem som er vinneren av spillet.

Scoreboard:

```
29        if button_b.was_pressed():
30            radio.send("end game")
31            if p1Wins > p2Wins:
32                display.scroll("P1")
33            elif p2Wins > p1Wins:
34                display.scroll("P2")
35            else:
36                display.scroll("DRAW")
37            p1Wins = 0
38            p2Wins = 0
39            gameStart = False
```

P1:

```
36            if received == "end game":
37                wins = 0
38                losses = 0
39                isBallOwner = False
40                isNextBallOwner = True
41                resetBall()
```

P2:

```
36          if received == "end game":
37              wins = 0
38              losses = 0
39              isBallOwner = False
40              isNextBallOwner = False
```

Måter å gjøre spillet vanskeligere

Hvis man synes at det er for lett eller vanskelig er det veldig enkelt å tilpasse vanskelighetsgraden. Man kan for eksempel gjøre spilleren mindre, sånn at det er vanskeligere å treffe ballen. For å endre dette må man bare endre litt på "wall" bildet. I arrayen må man bare endre en av de to pixelne som har wall Brightness til 0. Da blir spilleren bare en pixel stor.

```
5  wall = Image(5,1, bytearray([0,0,wallBrightness,0,0]))
```

Selvfølgelig kan man gjøre spilleren enda større hvis man vil, men dette kan føre til at man nesten ikke kan tape lenger eller bevege seg. (Det kan også skje noen feil i koden).

En annen måte å endre vanskelighetsgraden på er å endre hvor fort ballen er. På slutten av koden finnes det *'sleep(speed)'*, som betyr at den skal vente så mange millisekunder som står i speed før den går videre.

```
9  speed = 750
```

```
92      sleep(speed)
```

Derfor går ikke ballen utrolig fort. Men dette tallet kan man enkelt justere for å gjøre ballen så fort man vil. Hvis man vil gjøre den fortere må man bare endre tallet til f.eks. 500 eller 200 millisekunder, og hvis man synes det er for fort kan man endre det til 1000 millisekunder eller mer. (Dette påvirker også hvor fort man kan bevege spilleren).