10. alkalom (játékkészítés)

Ezúttal a diákok saját projektjei előtt közösen fogunk elkészíteni egyet, méghozzá egy olyan témában, ami eddig nem került szóba: a játékkészítés. A micro:biten lehetőségünk van jónéhány ismert játék elkészítésére, emellett a saját ötleteink megvalósítására is van mód. Ebben a fejezetben egy jól ismert okostelefonos játék, a Flappy Bird micro:bites verzióját fogjuk leprogramozni. Ebben a játékban egy kis madárral kell csövek közt átrepülnünk. A haladást nehezíti, hogy a madár folyamatosan zuhan, egy-egy kattintással csap egyet a szárnyával, aminek hatására egy kicsit emelkedik. Ez a játék sokáig a legnépszerűbb volt mindkét okostelefonos platformon, ám azóta az alkotó törölte azt[19], így jelenleg hivatalos forrásból az eredeti programot nem tudjuk beszerezni. Szerencsére létezik egy böngészőben játszható verziója[20], így ki tudjuk próbálni az eredeti verzió mását a programozás előtt. Természetesen szükségünk lesz apróbb változásokra, hiszen a grafikus felület helyett csupán egy 5x5-ös LED mátrix áll a rendelkezésünkre. Alkossuk meg a játékot, lépésről lépésre haladva![21]

1. Üdvözlő üzenet, a játék elkezdése

Nagyon nehézzé válna a játék, ha egyből a micro:bit áram alá helyezésénél elindulna, ezért célszerű valamilyen üdvözlő üzenetet, esetleg visszaszámlálást megjeleníteni a játék kezdete előtt. A szokásos import után jelenítsük meg ezt:

```
display.scroll("3...2...1...GO!", 75)
```

A második paraméterként szereplő számmal tudjuk a szöveg görgetésének sebességét változtatni. Ennek az alapértelmezett értéke 150, így itt az alap sebesség duplájával pörög a szöveg a kijelzőn. Ezt az értéket személyes preferenciáinknak megfelelően szabadon változtathatjuk, akárcsak a megjelenő szöveget.

2. A madár megjelenítése

Kezdjük el a kijelzőn megjeleníteni a játék elemeit, elsőként a madarat! Mivel a madár vízszintesen nem, csak függőlegesen mozog, ezért csak azt kell eltárolnunk, hogy az y tengelyen hol található. Habár csak 5 soros a kijelző, a madár pozícióját 0 és 99 között

[19] https://twitter.com/dongatory/status/432227971173068800 Elérés dátuma: 2019.02.05.
[20] https://flappybird.io/ Elérés dátuma: 2019.02.05.
[21] https://blog.withcode.uk/2016/05/flappy-bird-microbit-python-tutorial-for-beginners/ Elérés dátuma: 2019. 02. 05.

fogjuk eltárolni, majd ebből számoljuk ki, hogy melyik sorban legyen. Ezáltal realisztikusabb mozgás érhető el, mivel nemcsak 5 féle értékkel tudunk számolni. Egészítsük ki a kódot a madár megjelenítésével!

```
y = 50
led_y = int(y / 20)
display.set_pixel(1, led_y, 9)
```

Az `y` változóba megadjuk a madár kezdőpozícióját, a képernyő közepén, hiszen itt kezdi a játékot. A `led_y` változóban fogjuk tárolni, hogy melyik LED-en kell szerepelnie a madárnak. Ehhez az `y` értékét 20-szal osztjuk, mivel egy 0-99 közti értékből szeretnénk egy 0 és 4 köztit előállítani (tehát 100 féle értékből 5 félét). Az `int()` függvénnyel ezt a számot egészre kerekítjük, hiszen a LED koordinátája csak egész szám lehet. Végül, az utolsó sorban felvillantjuk a madár helyét a LED mátrixon. Az `x` koordináta 1 lesz, így a második oszlopban lesz a madarunk, jobban követhető lesz, mintha a képernyő legszélén lenne. Az `y` koordinátát korábban kiszámoltuk, a világosság értéke pedig a maximális, 9 lesz. A csövek ennél halványabbak lesznek, így könnyű lesz megkülönböztetni a szereplőnket a pálya részeitől.

3. A madár mozgatása

Sikeresen megjelenítettük a madarunkat, azonban ez még csak egy égő pont a LED-ek közt. Kezdjük el hát mozgatni! Első lépésként a gravitációt fogjuk szimulálni, hogy a madár elkezdjen zuhanni a föld felé. Ehhez elő kell vennünk a végtelen ciklust, hiszen a játék folyamatosan fog játszódni. Vezessük be egy változót a sebesség tárolására, ennek az értéke legyen 1. A cikluson belül a madár pozícióját mindig a sebességnek megfelelően változtassuk! Ha így lefuttatjuk a programot, két hibát vehetünk észre: az egyik, hogy a madár ugyan esik a föld felé, azonban az előző pozíciója is felvillantva marad, így egy vonal rajzolódik ki. A másik hiba pedig az, hogy a madár „ki tud esni” a kijelzőről, így az y értékét maximalizálnunk kell. Kérdezzük meg a tanulókat arról, hogy hogyan javítanák ki ezeket a hibákat! A jó válaszok megtalálása után kódoljuk is le őket. A végeredménynek így kell kinéznie:

```
1.  display.scroll("3...2...1...GO!",75)
2.
3.  y = 50
4.  speed = 1
5.
```

```
6.  while True:
7.      display.clear()
8.      y += speed
9.      if y > 99:
10.         y = 99
11.     led_y = int(y / 20)
12.     display.set_pixel(1, led_y, 9)
13.     sleep(20)
```

Ezzel a madár már esik ugyan, azonban nem túl valósághű, hiszen konstans 1 sebességgel teszi ezt. A valóságban a tárgyak esés közben egyre gyorsabbak. Növeljük hát esés közben a sebességet! Kezdetben állítsuk be 0-ra, a ciklus első lépéseként növeljük, és ne hagyjuk, hogy 2-nél több legyen. A feladat önálló munka keretében megoldható.

A madár lefelé mozgásával elkészültünk, azonban játszhatatlan lenne a játék, ha nem tudna csapkodni a szárnyaival, vagyis készítsük el a felfelé mozgatást! A madár az „A” gomb lenyomására csapjon egyet a szárnyával, ezáltal kis löketet kapva felfelé!

```
1.  display.scroll("3...2...1...GO!",75)
2.
3.  y = 50
4.  speed = 0
5.
6.  while True:
7.      display.clear()
8.      if button_a.was_pressed():
9.          speed = -8
10.
11.     speed += 1
12.     if speed > 2:
13.         speed = 2
14.     y += speed
15.     if y > 99:
16.         y = 99
17.     if y < 0:
18.         y = 0
19.     led_y = int(y / 20)
20.     display.set_pixel(1, led_y, 9)
21.     sleep(20)
```

A 8. sortól kezdődően egészítettük ki a programot: amennyiben az „A” gombot lenyomtuk, a sebességet állítsuk -8-ra! Ezáltal a madár felrepül, de a gravitáció máris elkezd dolgozni rajta, és megkezdi a zuhanást. Mivel most már felfelé is tud menni a madár, meg kell akadályozni, hogy felül kirepüljön a kijelzőről, ezt a 17-18. sorban tettük meg. Ezzel a madár megjelenítésével és mozgásával el is készültünk.

4. A csövek megjelenítése

Most pedig térjünk át a játék másik szereplőjére, a csövekre. Ezek fognak majd egyre közeledni a madárhoz, aminek az ezeken lévő lyukakon kell keresztül repülnie. A programon belüli megvalósításuk úgy fog történni, hogy a kijelző jobb szélére kirajzolunk egy függőleges vonalat, amibe véletlenszerűen egy 2 pixel magas lyukat fogunk elhelyezni. Kérdezzük meg a diákoktól az ehhez szükséges modul nevét!

Ezúttal egy új funkcióval is megismerkedünk a programozás során, a saját függvény készítésével. Beszéljünk a tanulókkal arról, hogy ez miért célszerű a programozás során! A Pythonban ezt a `def` kulcsszóval érhetjük el, ezután adhatjuk meg a függvény nevét és a paramétereit, a visszatérési érték típusa természetesen nem szükséges. Nézzük a kódot:

```
1.  import random
2.
3.  display.scroll("3...2...1...GO!",75)
4.
5.  y = 50
6.  speed = 0
7.
8.  def make_pipe():
9.      pipe = Image("00003:00003:00003:00003:00003")
10.     gap = random.randint(0,3)
11.     pipe.set_pixel(4, gap, 0)
12.     pipe.set_pixel(4, gap+1, 0)
13.     return pipe
14.
15. pipe = make_pipe()
16.
17. while True:
18.     display.show(pipe)
19.     if button_a.was_pressed():
20.         speed = -8
21.
22.     speed += 1
23.     if speed > 2:
24.         speed = 2
25.     y += speed
26.     if y > 99:
27.         y = 99
28.     if y < 0:
29.         y = 0
30.     led_y = int(y / 20)
31.     display.set_pixel(1, led_y, 9)
32.     sleep(20)
```

A programba importáltuk a `random` modult. A csőkészítő függvényünk a 8-13. sorban található. Először létrehozzuk a vonalat a kijelző jobb szélén, a teljes fényerősség harmadán. Ahogy már korábban említettük, erre azért van szükség, hogy a cső könnyen megkülönböztethető legyen a madártól. Ezután választunk egy 0 és 3 közti véletlenszámot. Azért 3 a felső határ, mert a választott számú és az eggyel nagyobb koordinátájú LED-et fogjuk lekapcsolni, így ha a 4-et is megengednénk, már a kijelzőn kívülre hivatkoznánk. A választás után a `set_pixel()` metódussal lekapcsoljuk (0 fényerősségre állítjuk) a kiválasztott LED-eket. Végül a `return` utasítással visszaadjuk a kész képet, ami majd a csőként fog funkcionálni. További módosításként még a ciklus előtt „elkérünk” egy csövet a függvénytől, majd a cikluson belül a képernyő törlése helyett kirajzoljuk a csövet. A törlésre azért nincs szükség, mert a cső kirajzolása letöröl mindent a kijelzőről, így a madarunk előző pozíciója sem fog látszódni már. Minden szereplőnk ott van a kijelzőn, így jöhet a következő lépés.

5. A csövek mozgatása, az ütközés vizsgálata

Ebben a lépésben tehát elkezdjük mozgatni a csöveket, és megvizsgáljuk, hogy a madarunk beleütközött-e valamelyikbe.

Először is szükségünk lesz arra, hogy számolni tudjuk, hányszor futott már le a játék fő ciklusa. Erre bevezetjük a `frame` (képkocka) nevű változót, ami 0-ról indul, és minden futásnál eggyel növekszik. Ezeknek a számolásával tudjuk majd beállítani a játék sebességét, nehézségét. Ehhez természetesen konstansokra is szükségünk van. Ezen felül a ciklust ki kell egészítenünk a cső mozgatásával, valamint az új cső készítésével. A kód kezd kicsit meghízni, az új sorokat félkövéren jelöljük, továbbá a főciklus nagysága miatt kommenteket is bevezetünk (a saját függvények fontossága máris megmutatkozik!):

```
1.  import random
2.
3.  DELAY = 20
4.  FRAMES_PER_PIPE_MOVE= 20
5.  FRAMES_PER_NEW_PIPE = 100
6.
7.  display.scroll("3...2...1...GO!",75)
8.
9.  y = 50
10. speed = 0
11. frame = 0
12.
```

```
13. def make_pipe():
14.     pipe = Image("00003:00003:00003:00003:00003")
15.     gap = random.randint(0,3)
16.     pipe.set_pixel(4, gap, 0)
17.     pipe.set_pixel(4, gap+1, 0)
18.     return pipe
19.
20. pipe = make_pipe()
21.
22. while True:
23.     frame += 1
24.
25.     display.show(pipe)
26.
27.     # szarnycsapas a gombra
28.     if button_a.was_pressed():
29.         speed = -8
30.
31.     # gravitacio
32.     speed += 1
33.     if speed > 2:
34.         speed = 2
35.
36.     # madar mozgatasa, de maradjon a kijelzon
37.     y += speed
38.     if y > 99:
39.         y = 99
40.     if y < 0:
41.         y = 0
42.
43.     # madar kirajzolasa
44.     led_y = int(y / 20)
45.     display.set_pixel(1, led_y, 9)
46.
47.     # cso mozgatasa balra
48.     if (frame % FRAMES_PER_PIPE_MOVE == 0):
49.         pipe = pipe.shift_left(1)
50.
51.     # uj cso letrehozasa
52.     if (frame % FRAMES_PER_NEW_PIPE == 0):
53.         pipe = make_pipe()
54.
55.     sleep(20)
```

A konstansok a 3-5. sorban találhatóak. A Python konvencióinak megfelelően ezek csupa nagybetűvel, a szavak közt alulvonással írandóak. A `DELAY` mondja meg, hogy két frame közt mennyi idő teljen el, a `FRAMES_PER_PIPE_MOVE` értéke azt mutatja, hogy hány frame teljen el amíg a cső eggyel balra mozdul, míg a `FRAME_PER_NEW_PIPE` értéke a

két cső előállítása között eltelt frame-ek számát mutatja. Ezeket az értékeket használjuk a ciklusban. a 48. sorban eldöntjük, hogy szükséges-e ebben a frame-ben mozgatni a csövet. Ezt a maradékos osztás műveletével egyszerűen megtehetjük. Az `Image` osztály `shift_left()` metódusával a balra mozgatást könnyedén megoldhatjuk. Hasonlóképpen járunk el az új cső létrehozásánál is. Ha már sikeresen mozognak a csövek, vizsgáljuk meg az ütközést a madárral! Ehhez egészítsük ki a főciklust, a madár kirajzolása után, a következőkkel:

```
if pipe.get_pixel(1, led_y) != 0:
    display.show(Image.SAD)
    break
```

Amennyiben a csövet tartalmazó kép 1. oszlopában (a kijelző 2. oszlopában, tehát a madarat tartalmazó oszlopban), a `led_y` sorban, azaz a madár aktuális helyén nem 0 a világosságérték, akkor ott bizony a csőnek egy darabja van. Ilyenkor sajnos ütköztünk a csővel. Ezt kísérje egy szomorú arc megjelenítése a kijelzőn, majd álljon le a játék! Ezzel elkészült a saját Flappy Bird-ünk, ami a micro:biten játszható. Azonban egy dolog még hiányzik egy jó játékból. Kérdezzük meg a tanulóktól, hogy mi lehet az!

Feladat a diákok számára

Egészítsd ki a játékot pontszámítással! Minden cső, amin átrepültünk érjen egy pontot! A játék végén kerüljön megjelenítésre az elért pontszám!

Az elkészült játék teljes kódja:

```
1.  from microbit import *
2.  import random
3.
4.  DELAY = 20
5.  FRAMES_PER_PIPE_MOVE= 20
6.  FRAMES_PER_NEW_PIPE = 100
7.
8.  # display.scroll("3...2...1...GO!",75)
9.
10. y = 50
11. speed = 0
12. frame = 0
13. score = 0
14.
15. def make_pipe():
16.     pipe = Image("00003:00003:00003:00003:00003")
```

```
    gap = random.randint(0,3)
    pipe.set_pixel(4, gap, 0)
    pipe.set_pixel(4, gap+1, 0)
    return pipe

pipe = make_pipe()

while True:
    frame += 1

    display.show(pipe)

    # szarnycsapas a gombra
    if button_a.was_pressed():
        speed = -8

    # gravitacio
    speed += 1
    if speed > 2:
        speed = 2

    # madar mozgatasa, de maradjon a kijelzon
    y += speed
    if y > 99:
        y = 99
    if y < 0:
        y = 0

    # madar kirajzolasa
    led_y = int(y / 20)
    display.set_pixel(1, led_y, 9)

    # utkozes vizsgalata
    if pipe.get_pixel(1, led_y) != 0:
        display.show(Image.SAD)
        sleep(500)
        display.scroll(score)
        break

    # cso mozgatasa balra
    if (frame % FRAMES_PER_PIPE_MOVE == 0):
        pipe = pipe.shift_left(1)

    # uj cso letrehozasa
    if (frame % FRAMES_PER_NEW_PIPE == 0):
        pipe = make_pipe()
        score += 1

    sleep(20)
```