Feladattípusorientált játékfejlesztés a Scratch-ben

Bernát Péter

bernatp@inf.elte.hu
ELTE IK

Absztrakt. Informatikatanárként a tanulók érdeklődésétől, előismereteitől és képességeitől függően számos nagy programozási témakör közül választhatunk. Ezek egyike a játékfejlesztés, amely nemcsak motiváló lehet, de egy-egy összetettebb játék elkészítése sokféle megoldási módszer és programozási fogalom ismeretét igényelheti. Cikkemben az akció- és kalandjátékok műfaján belül három egymásra épülő feladattípus megvalósításával foglalkozom a Scratch-ben: ismertetem az akadálypályás, a labirintus- és a platformjátékok jellemző részfeladatait és a részfeladatok megoldási lehetőségeit.

Kulcsszavak: Scratch, programozástanítás, játékfejlesztés, akció- és kalandjátékok

1. Bevezetés

Informatikatanárként a tanulók érdeklődésétől, előismereteitől és képességeitől függően számos nagy programozási témakör közül választhatunk. Ezek egyike a játékfejlesztés, amely elsősorban a játékok tipikus megvalósítási módszerei iránt érdeklődő, programozási alapismeretekkel már rendelkező és a programozásra nyitott tanulók számára javasolható.

A számítógépes játékoknak a kezdetektől napjainkig nagyon sokféle műfaja alakult ki. Közülük kezdésnek érdemes az egyszerűbb akció- és kalandjátékokat választani, amelyek a grafikus képernyőn valamilyen szabályok szerint mozgó és interakcióba lépő szereplőkre építenek, megvalósításukhoz ezért kisebb mértékű absztrakcióra van szükség, ugyanakkor látványosak és szórakoztatók.

Ezekben a játékokban a főszereplővel egy leegyszerűsített absztrakt világban kell előre haladni, amelyben jellemzők az áthatolhatatlan falak, az előre haladást nehezítő ellenségek, és az előre jutáshoz szükséges tárgyak, amelyeket fel kell venni és máshol felhasználni. Jellemzően többpályásak, amelyek egy- vagy többképernyősek is lehetnek. A többi játékhoz hasonlóan folyamatosan nyilvántartják a játékos teljesítményét, és a játékos számára a végső cél az összes pálya teljesítése és a minél jobb teljesítmény elérése.

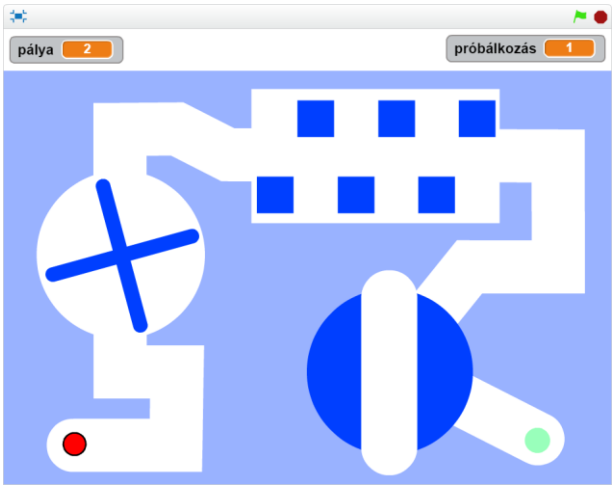
Cikkemben három egyre összetettebb feladattípuson, az akadálypályás, a labirintus- és a platformjátékokon keresztül ismertetem az akció- és a kalandjátékok előzőekben említett fontosabb kellékeinek a megvalósítási lehetőségeit a Scratch-ben. Elképzeléseim szerint ezekkel a feladattípusokkal az animációkészítés témakörét követően célszerű foglalkozni.

A bemutatott példaprogramok és a Scratch-ben alkalmazott megoldási módszerek a sajátjaim. Merítettem ötleteket más programozási nyelvekhez készült és a játékkészítésről szóló leírásokból [1], [2], valamint különböző játékkészítő szoftverek eszközkínálatából [3], [4], nem ismerek azonban olyan irodalmat, amely a módszeres játékkészítéssel hasonló mélységben foglalkozna a Scratch esetén. A Scratch-re építő ismeretterjesztő könyvek jelentős részét a programozási fogalmak tematizálják, és csak egy-két nagyon egyszerű játék elkészítését mutatják be [5]. Pozitív példa azonban Carol Vorderman nemrégen megjelent könyve [6], amely kifejezetten a játékprogramozásról szól a Scratch-ben, és a tematikáját különböző játékok határozzák meg.

2. Akadálypályás játék

A bemutatásra kerülő három játéktípus közül az akadálypályás játékok készíthetők el a legkönnyebben. Ezekre az akciójátékokra az egyszerű kivitelezésű, ugyanakkor a játékos koncentrációját maximálisan próbára tevő pályák a jellemzők. Az egyes pályákon a nyílgombokkal irányítható főszereplővel a falakkal határolt folyosókon keresztül és a halálos mozgó akadályokon is áthaladva kell eljutni a pálya kijáratáig. A játék számolja a szükséges próbálkozásokat, a végső cél pedig minél kevesebb próbálkozásból teljesíteni az összes pályát. Ehhez a játéktípushoz egy az interneten népszerű játék adta az ötletet [9].

A mintaprogramban látható pályán világoskékek a falak, a sötétkék akadályok közül kettő forgóajtószerűen mozog, a harmadik pedig a cipzárhoz hasonlóan zár össze és nyílik szét. A piros kör a játékos által irányítható szereplő, a zöld szín pedig a kijáratot jelzi (1. ábra).



1. ábra: Egy akadálypályás játék

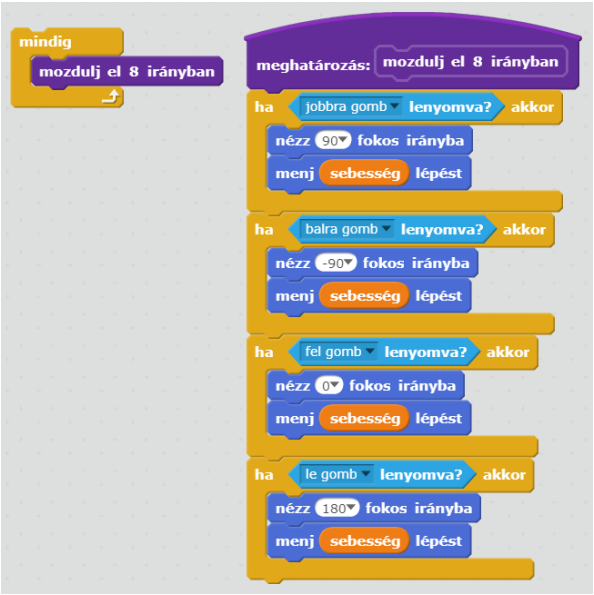
2.1. A játék szereplői

A Scratch-ben mozogni képes szereplőket, valamint a játéktér mozdulatlan hátterét lehet felhasználni a játékok elkészítéséhez. A mintaprogramban a főszereplő és az akadályok mozognak, ezért azokat szereplőkként kell létrehoznunk. A mozdulatlan falakat a háttéren is megrajzolhatjuk, vagy szereplőként is létrehozhatjuk. Most az előbbi lehetőséget választjuk, azonban figyelembe kell vennünk, hogy a háttéren csak a valamely színárnyalattal történő átfedés érzékelésére van mód a Scratch-ben, ezért a falaknak egyszínűeknek kell lenniük.

2.2. A főszereplő irányítása nyolc irányban

A billentyűzettel irányítás egyik legegyszerűbben megvalósítható változata az, amelyben a játékos a játék főszereplőjét a négy nyílbillentyűvel a megfelelő négy alapirányba, két-két nyílbillentyű egyidejű lenyomásával pedig a négy átlós irányba képes elmozdítani egyenletes sebességgel.

A Scratch-ben a szereplők (például a Logo nyelvhez hasonlóan) aktuális hellyel és iránnyal rendelkeznek, és rendelkezésre állnak az ezeket a tulajdonságokat módosító parancsok, valamint a valahány képponttal az aktuális irányba előre mozgató *menj* parancs. Ennek megfelelően a nyolcirányú irányítás megvalósítható az alábbi *mozdulj el 8 irányban* nevű parancs létrehozásával (2. ábra).



2. ábra: A nyolcirányú irányítást lehetővé tevő parancs meghatározása

A parancs felhasználja a szereplő *sebesség* tulajdonságát, amely egy általunk létrehozott saját változója (és a mintaprogramban az értéke 3). A parancsot pedig egy végtelen ciklusba helyezve a főszereplő folyamatosan irányíthatóvá válik.

2.3. A főszereplő megállítása a falnál

Az irányításhoz minden játéktípus esetén szorosan hozzátartozik a falakon való áthaladás korlátozása. A Scratch-ben rendelkezésre állnak a valamely színnel vagy a valamely szereplővel való átfedést ellenőrző feltételek, a használatukhoz viszont elengedhetetlen, hogy az átfedés létrejöjjön. Ezért hagyni kell, hogy a szereplő először belelépjen a falba (3. ábra/1., 2.), majd azt érzékelve javíthatunk a pozícióján. Megtehetjük, hogy egyszerűen visszahelyezzük az előző pontba, ahol még nem érintkezett vele. Mivel azonban a sebességétől függően egy lépésben akár több képponttal is elmozdulhatott, a fallal érintkezés előtt több képpontnyi távolságra is lehetett attól. Ezért inkább egy feltételes ciklussal hátrafelé mozgatjuk képpontonként addig, amíg az átfedés meg nem szűnik (3. ábra/2., 3.).

3. ábra: A szereplő a falba lépés előtti pillanatban (1.), a falba lépéskor (2.), és ahová helyezni kell (3.)

A megoldáshoz a *mozdulj el 8 irányban* parancs meghatározásában az összes *menj* parancsot helyettesíteni kell az alábbi *menj amíg szabad* paranccsal (4. ábra). A Scratch-ben beállítható, hogy egy parancsot a szereplő képernyőfrissítés nélkül hajtson végre, amit ezen parancs esetén meg is kell tenni.

4. ábra: A menj parancsot helyettesítő parancs meghatározása, amely szükséges esetben kimozdítja a szereplőt a falból

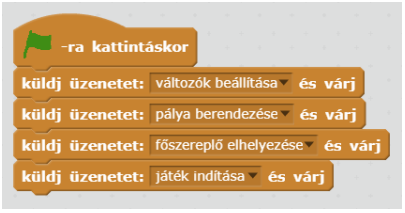
A mozgó akadályokat szintén egy-egy végtelen ciklusban lehet folyamatos forgásra vagy mozgásra bírni; ennek a megvalósításához elegendők az animációkészítésből már ismert módszerek és utasítások.

2.4. A játék előkészítése és indítása

Azután, hogy a főszereplő irányítható, és az akadályok is mozognak, megszervezhető a játék egypályás változata. Működését a következő egymás után végrehajtandó lépésekre bonthatjuk:

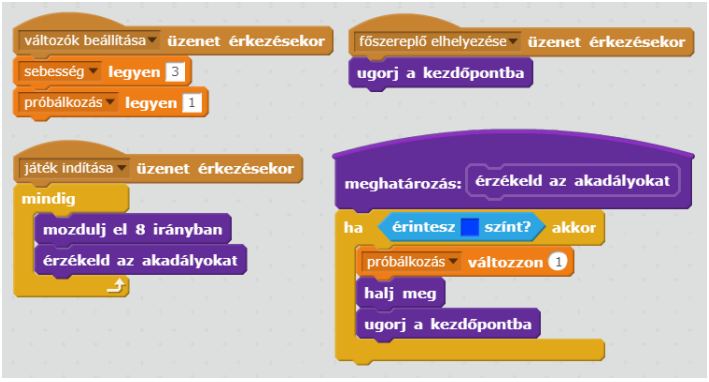
- *a változók beállítása* – kezdőértékkel látjuk el a játék során használt változókat, jelenesetben a főszereplő sebességét és a próbálkozások számát tároló változót;
- *a pálya berendezése* – a kezdőállapotukba hozzuk a pálya valamennyi összetevőjét, most a mozgó akadályokat;
- *a főszereplő elhelyezése* – a főszereplőt elhelyezzük a már berendezett pálya kezdőpontjában;
- *a játék indítása* – működésbe hozzuk a játék összetevőit, elindul a tényleges játék.

Mivel az egyes lépésekben tipikusan több szereplőnek is feladata van, azokra a játéktér szólíthatja fel a szereplőket egy-egy üzenet elküldésével (5. ábra).



5. ábra: A játéktér üzenetei, amelyek előkészítik majd elindítják a játékot

A játék indításától kezdődően a főszereplőnek részben az irányításra, részben az akadályokkal történő esetleges ütközésekre kell figyelnie (6. ábra). Az utóbbit megvalósító *érzékeld az akadályokat* parancs a valamely akadállyal való átfedésbe kerülés esetén eggyel növeli a próbálkozások számát, a *halj meg* parancs hatására valamilyen látványos animációval és hanggal eltünteti a főszereplőt, majd az *ugorj a kezdőpontba* paranccsal visszaállítja annak az eredeti kinézetét és egyúttal visszahelyezi a pálya kezdőpontjába.



6. ábra: A főszereplő feladatai az egypályás játékban

2.5. Többpályás játék készítése

Ahhoz, hogy a játék többpályás legyen, további pályák háttereit és akadályait kell elkészíteni, és minden pályán elhelyezni egy kijáratot. Sajnos a Scratch-ben nincsen lehetőség pályánként külön játéktereket használni és azokon külön szereplőket, ezért az aktuális pálya minden egyes berendezésekor el kell tüntetni, illetve meg kell jeleníteni az éppen nem szükséges, illetve szükséges szereplőket.

A programban ezért a következő módosításokat kell végrehajtani:

- be kell vezetni egy *pálya* változót, amelyet a *változók beállítása* lépésben 1-re kell állítani, és érdemes bevezetni és beállítani egy *pályák száma* változót is, hogy a pályák mennyiségét a programkód jól látható helyén lehessen később módosítani;
- a *pálya berendezése* lépés során a *pálya* változó aktuális értékétől függően kell a hátteret beállítani, az akadályokat pedig megjeleníteni vagy eltüntetni;
- az *ugorj a kezdőpontba* parancsnak úgyszintén a *pálya* változótól függően kell a főszereplőt visszahelyeznie az aktuális pálya kezdőpontjába;
- végül pedig ki kell egészíteni a főszereplő végtelen ciklusát az alábbi *érzékeld a kijáratot* paranccsal (7. ábra).

7. ábra: A főszereplő a kijáratban vagy megjeleníti a következő pályát, vagy véget ér a játék

Az *érzékeld a kijáratot* parancs, amennyiben a főszereplő megérintette a zöld színű kijáratot és még nem az utolsó pályán volt, valamilyen látványos animációval és hanggal eltűnteti azt *(örülj),* eggyel növeli a *pálya* változót, felszólítja a hátteret és az akadályokat a *pálya* változó értékétől függően a következő pálya berendezésére, a játékost az új pálya kezdőpontjába helyezi *(ugorj a kezdőpontba),* majd a vezérlést visszaadja a végtelen ciklusnak és folytatódik a játék. Ha pedig az utolsó pályán sikerült elérnünk a kijárathoz, a főszereplő valamilyen látványos módon gratulál nekünk *(nyerj),* és leállítja a játékot.

A teljes mintaprogram kipróbálható és a forráskódja megtekinthető a következő címen:

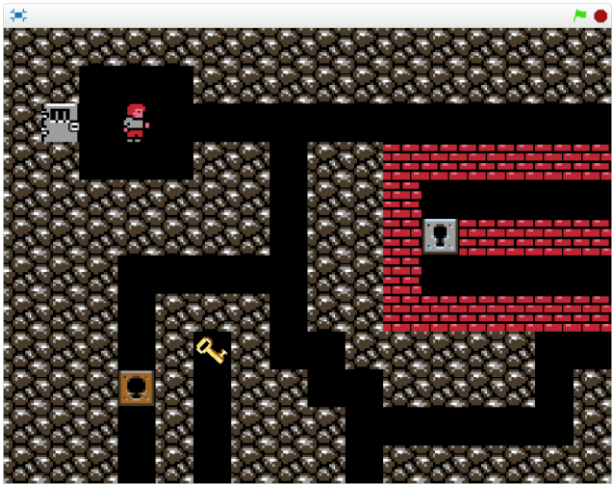
https://scratch.mit.edu/projects/183214473/

A játékok megszervezése a további két feladattípus esetén is hasonlóan történhet, ezért a továbbiakban csak az eltéréseket ismertetem.

3. Labirintusjáték

Ebben a játéktípusban a feladat megtalálni a nyílgombokkal irányítható főszereplővel a kivezető utat egy négyzetrácsos felépítésű labirintusból. A folyosókon nincsenek ellenségek, azonban a játékos dolgát így is két körülmény nehezíti: egyrészt a labirintus többképernyős, és ezáltal sosem látható egyben, másrészt útközben tárgyakat kell felszedni és azokat máshol felhasználni ahhoz, hogy egyes utak szabaddá váljanak. Ehhez a játéktípushoz egy a Scratch-ben mások által készített játék adta az alapötletet [10].

Az alábbi mintaprogramban egy arany-, egy ezüst- és egy bronzzár zárja le a labirintus egyes folyosóit, amelyek csak a labirintus más-más pontján megszerezhető ugyanolyan színű kulccsal nyithatók (8. ábra).



8. ábra: Egy labirintusjáték

3.1. A játék szereplői

A többképernyős labirintus minden egyes képernyője felfogható egy-egy pályaként, amelyek között akkor kell váltani, ha a főszereplő a képernyő valamely szélét elérte.

Az egy adott pályán előforduló kulcsokat és zárakat szereplőkként kell létrehoznunk, ugyanis a kulcsoknak a megszerzésük, a záraknak pedig a kinyitásuk pillanatában el kell tűnniük.

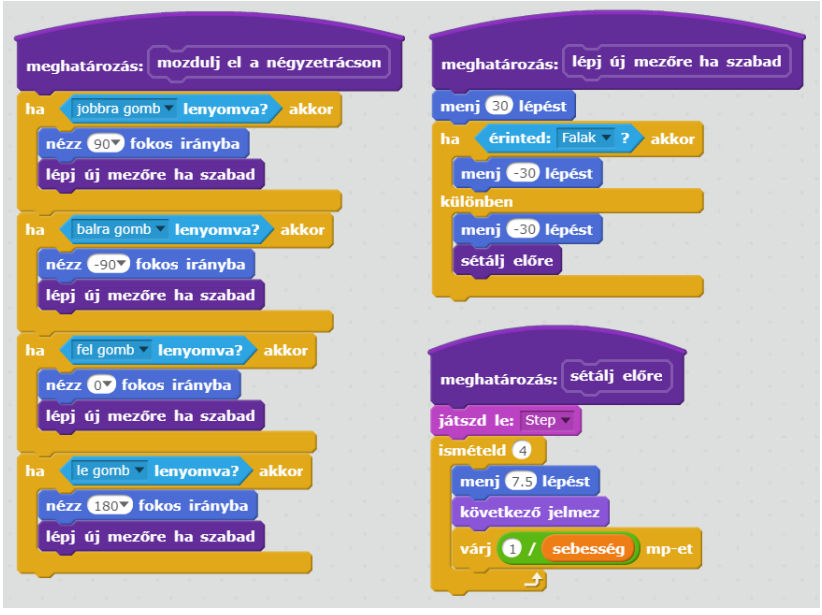
A pályák falai ezúttal is mozdulatlanok, viszont sok különböző színárnyalatot tartalmaznak, így a főszereplő nem lenne képes azokat a színük alapján a háttéren érzékelni. Ezért a játéktér egyes hátterei helyett ezúttal egy például *Falak* nevű szereplő egyes jelmezei tartalmazhatják az egyes pályák falait, amely szereplővel az esetleges átfedést már képes lesz a főszereplő érzékelni. A jelmezek mérete meg fog egyezni a játéktér méretével, a falakat nem tartalmazó helyeken pedig átlátszók lesznek. Mindösszesen arra kell majd figyelni, hogy az új pálya berendezésénél nem a játéktér hátterét, hanem a *Falak* szereplő jelmezét kell leváltani.

Sajnos a Scratch beépített rajzolóeszköze nem támogatja a négyzetrácsos jelmezek (és hátterek) megrajzolását, vannak azonban kifejezetten erre a célra kifejlesztett ingyenes és egyszerűen használható képszerkesztőprogramok. Ezek egyike a *Tiled* [7], amelyet a mintaprogram elkészítése során magam is használtam. Az egységméretű grafikákhoz pedig az *OpenGameArt* [8] oldaláról jutottam hozzá.

3.2. A főszereplő irányítása a négyzetrácson

Négyzetrácsos pálya esetén is felmerülhet a főszereplő nyolcirányú irányítása, könnyebb azonban egy olyan szereplővel közlekedni rajta, amely mindig egy-egy egész mezőnyit mozdul el a négy alapirány valamelyikébe.

A főszereplő ezúttal sem érkezhet falra. Az előző játéktípusnál tárgyaltakhoz hasonlóan megtehetjük most is, hogy először egy mezőmérettel a nyílgomb szerinti irányba mozgatjuk, majd ha falra érkezett, visszavonjuk a lépést. Ehhez a művelethez a *mozdulj el a négyzetrácson* parancson belül bevezethetjük a *lépj új mezőre ha szabad* parancsot. A 9. ábrán látható változatban a főszereplő mozgását azzal tettük látványosabbá, hogy a szomszédos mező ellenőrzése után mindenképpen visszahelyezzük őt az előzőre, és amennyiben a szomszédos mezőn nem volt fal, folyamatosan mozgatjuk át az új helyre. Erre a folyamatosnak tűnő mozgásra a *sétálj előre* parancs utasítja, amely a lépések közötti kismértékű várakozást a szereplő *sebesség* tulajdonságából (általunk létrehozott saját változójából) számolja ki.

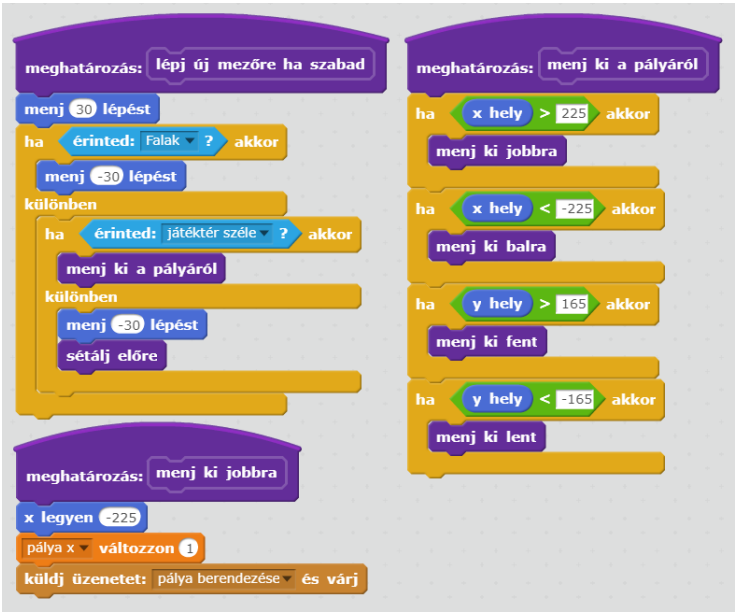


9. ábra: A négyzetrácson irányítást megvalósító parancs és a hozzá tartozó parancsok

3.3. Többképernyős játék készítése

Mint ahogyan a játék szereplőkre bontásánál már szó volt róla, a többképernyős labirintus egyképernyős részleteire külön pályákként gondolhatunk. Az előző játékhoz képest azonban ezek a pályák nem időben követik egymást, hanem a kétdimenziós síkon oldalszomszédosan kapcsolódnak egymáshoz. Ezért az aktuális pálya nyilvántartásához ezúttal nemcsak egyetlen *pálya* változóra, hanem egy *pálya x* és egy *pálya y* változóra lesz szükség. Amennyiben a szereplő eléri a képernyő valamelyik szélét, a két változó valamelyikét 1-gyel növelni vagy csökkenteni kell, majd a két aktualizált pályakoordináta alapján kiválasztható a *Falak* szereplő megfelelő jelmeze.

A 10. ábrán látható változatban a *lépj új mezőre ha szabad* parancsot kiegészítettük azzal az esettel, ha a főszereplő elhagyja a játékteret. Ekkor a *menj ki a pályáról* parancs ellenőrzi, hogy melyik irányban történt a kilépés, majd meghívja a megfelelő újabb parancsot. Az ábrán a jobbra kilépés esete látható *(menj ki jobbra)*: ekkor a szereplőt változatlan y koordinátával a képernyő bal szélére kell helyezni, a *pálya x* változót meg kell növelni 1-gyel, végül pedig el kell küldeni a *pálya berendezése* üzenetet, amelynek a hatására a szereplők a *pálya x* és a *pálya y* változótól függően jelenítik meg a szükséges pályát.



10. ábra: Ha a játékos elérte a játéktér szélét,
a másik oldalon kell feltűnnie és új pályát kell berendezni

3.4. Kulcsok és ajtók hozzáadása a játékhoz

Amíg a kulcsokat nem szereztük meg, illetve a zárakat nem nyitottuk ki, addig azokat a pályák berendezésekor a két pályakoordinátától függően kell elrejteni, vagy a megfelelő helyen megjeleníteni.

A főszereplőnek az egyes kulcsokkal való lehetséges átfedését elegendő akkor ellenőrizni, amikor az éppen egy szomszédos mezőre átsétált. Ha a megérkezését követően az arany-, az ezüst- vagy a bronzkulcsot érinti, a megfelelő kulcsot egy üzenet elküldésével fel kell szólítania az eltűnésre, továbbá egy a megfelelő kulcshoz tartozó, például *megvan* nevű segédváltozó értékének a beállításával el kell könyvelnie, hogy az adott kulcsot a játékos megszerezte.

Amikor ugyanis a főszereplő a mozgatása során egy zárral kerül ideiglenesen átfedésbe, attól függően kell kinyitnia a zárat, vagy visszalépnie az előző mezőre, hogy a zárral megegyező színű kulcs

megvan segédváltozója szerint a játékos birtokolja-e már a kulcsot. Ha a zárat ki szabad nyitni, a zárat a főszereplő egy üzenet elküldésével szólíthatja fel az eltűnésre.

A zárak esetén is egy-egy, például *nyitva* nevű segédváltozóban tárolni kell a nyitott vagy csukott állapotukat, ugyanis a későbbiekben a pálya újbóli berendezésekor ezen segédváltozó értékétől függően kell a zárat megjeleníteni vagy elrejtve hagyni.

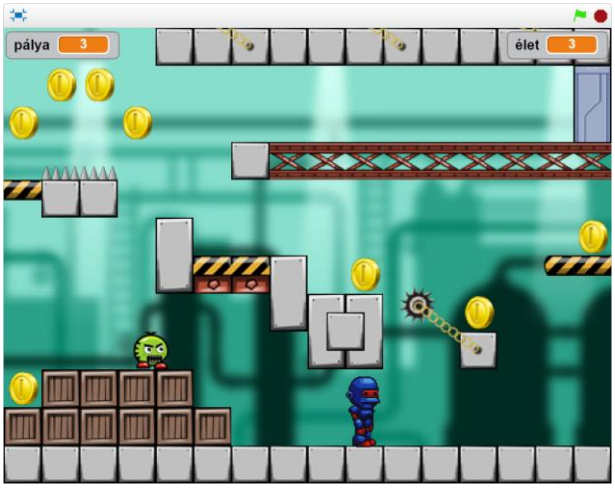
A teljes mintaprogram kipróbálható és a forráskódja megtekinthető a következő címen:

https://scratch.mit.edu/projects/184526679/

4. Platformjáték

A platformjátékok onnan kapták a nevüket, hogy a pályákat különböző magasságokban elhelyezett platformok alkotják, amelyeken az ugrálni vagy mászni képes főszereplővel haladni kell. A pályákon jellemzők a pontot érő vagy a továbbhaladáshoz szükséges érmék vagy más gyűjthető tárgyak, valamint a mozdulatlan vagy mozgó ellenségek.

A mintaprogramban egy bekapcsolva felejtett robottal kell összegyűjteni a gyárban napközben szétszóródott pénzérméket. A sétálni és ugrani is képes robotot a három megfelelő nyílgombbal lehet irányítani. Az aktuális pálya kijárata csak akkor nyílik ki, ha a robot a pályán található összes érmét felszedte. A magasabb szintű pályákon előforduló mozdulatlan tüskék és mozgó ellenségek érintése életvesztéssel jár (és a robot visszakerül a pálya kezdőpontjába). A játék végső célja teljesíteni az összes pályát az összesen rendelkezésre álló 3 élettel (11. ábra).



11. ábra: Egy platformjáték

4.1. A játék szereplői

A főszereplő mellett a falak is egy külön szereplőt kell, hogy alkossanak az előző játéktípusnál megbeszélt okok miatt. Az egy pályán előforduló mozdulatlan tüskék szintén egy például *Tüskék* nevű szereplő jelmezei kell, hogy legyenek, amelynek az érintése halálos lesz a főszereplő számára.

Az érmék és az ellenségek nagyobb mennyiségben fordulnak elő az egyes pályákon, ezért azokat egy-egy szereplő másolataiként (klónjaiként) érdemes a pálya berendezésekor létrehozni. Egy szereplő másolatai mind tökéletesen egyformán működnek, így elegendő csak az eredeti szereplő feladatait elkészíteni.

Ebben a játéktípusban a legfontosabb újítás a főszereplő irányítása, akire ezúttal hatni fog a gravitáció, viszont képes lesz egy bizonyos magasságra felugrani. Az alábbiakban ennek az irányításnak egy megvalósítási lehetőségét mutatom be.

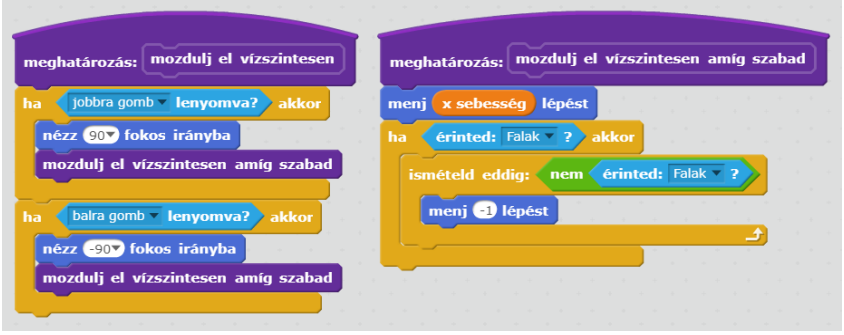
4.2. A főszereplő irányítása platformjátékosként

Ebben a játéktípusban elkerülhetetlen, hogy a főszereplő bonyolult és jellemzően animált alakját egy egyszerűbbel helyettesítsük a falakkal történő átfedések vizsgálatakor. Máskülönben például az egyébként már a platform széléről lelépő szereplőt valamely kiálló része megakadályozhatná az esésben. A játékfejlesztésre specializálódott környezetekben és játékmotorokban (például [3]) külön ütközésészlelő alakzatokat lehet hozzárendelni az egyes grafikus objektumokhoz. A Scratch-ben erre nincsen lehetőség, ezért a szükséges funkciókat két külön szereplővel tudjuk csak ellátni: egy téglalap alakú de láthatatlan szereplő fog ténylegesen a pályán mozogni és az átfedéseket érzékelni, és lesz egy a játékos által is látható, és a megfelelő módon animálódó másik szereplő, amelynek minden pillanatban az előző szereplővel azonos helyre kell kerülnie.

A platformjátékok főszereplői vízszintesen és függőlegesen különbözőképpen mozognak. Vízszintesen csak akkor mozdulnak el, ha a játékos a két megfelelő nyílgomb valamelyikét lenyomja. Függőlegesen viszont a változó irányú és nagyságú függőleges sebességüknek megfelelően mozdulnak el, amelyet a gravitáció és az ugrás is befolyásolhat. Érdemes ezért a mozgásukat megvalósító *mozdulj el platformjátékosként* parancs meghatározását is két részre, a *mozdulj el vízszintesen* és a *mozdulj el függőlegesen* parancsokra bontani (12. ábra).

12. ábra: A platformjátékos mozgása egy vízszintes és egy függőleges irányú elmozdulással állítható elő

A vízszintes irányítása hasonlóan történhez az akadálypályás játéktípus főszereplőjének az irányításához: először elmozdítjuk a nyílgombnak megfelelő irányba a vízszintes irányú sebességének megfelelő számú képponttal, majd a szükséges esetben képpontonként kimozgatjuk a falból (13. ábra).



13. ábra: A platformjátékos vízszintes irányítása a nyolcirányúnál látottakhoz hasonló

A főszereplő függőleges sebessége irányban és nagyságban is változhat, érdemes ezért egy saját *y sebesség* változójában nyilvántartani az aktuális előjeles függőleges sebességét. A beépített *y változzon*

parancsot felhasználva fogjuk tudni a szereplőt az *y sebességnek* megfelelő irányban és mértékben elmozdítani.

Célszerű először a szereplőnek azt a változatát elkészíteni, amelyre folyamatosan hat a gravitáció, de ugrani egyelőre nem képes. Ebben az esetben a *mozdulj el függőlegesen* parancsnak el kell mozgatnia a szereplőt az előjeles függőleges sebességének megfelelő irányban és mértékben, továbbá változtatnia kell ezt a sebességet egy a játék elején beállított gravitációs értékkel (ez a mintaprogramban -1). Amennyiben pedig belezuhant a talajba, a szokásos módon képpontonként kell abból kiemelni, és ráadásként az *y sebességét* nullázni (14. ábra).

14. ábra: A zuhanni és a talajon megállni képes szereplő függőleges mozgása

Ezután az irányítást kiegészíthetjük az ugrás lehetőségével. Ezzel kapcsolatban két meggondolnivaló van.

Egyrészt amint a szereplő ugrani is képes lesz, már nem lesz egyértelmű, hogy a függőleges elmozdulása során a talpával alulról, vagy a tetejével felülről ütközött-e a falba. Az előbbi esetben továbbra is a falból kiemeléssel, az utóbbiban azonban éppen a fal alá süllyesztéssel szabadítható ki. Hogy a két eset melyike áll fenn, arra a függőleges sebesség aktuális irányából, tehát az *y sebesség* előjeléből lehet következtetni (15. ábra).

Az ugrás megvalósítása során arra kell még figyelni, hogy ugrani csak akkor szabad, amikor a főszereplő a talajon áll. Ehhez az *ugorj ha kérik* parancsot (amely az ugrást az *y sebesség* beállításával valósítja meg) közvetlenül az *emelkedj a fal fölé* parancsot követően kell elhelyezni.

15. ábra: A platformjátékos irányításának teljes megvalósítása

A teljes mintaprogram kipróbálható és a forráskódja megtekinthető a következő címen:

https://scratch.mit.edu/projects/184537362/

5 Befejezés

Cikkemmel egyrészt kedvet és konkrét ötleteket kívántam adni a játékkészítéshez a Scratch-ben. Másrészt pedig rá kívántam mutatni arra, hogy miért érdemes félretennünk a számítógépes játékokkal szembeni esetleges ellenérzéseinket. Egy-egy összetettebb játékprogram elkészítése jóval bonyolultabb feladat a közoktatásban előírt típusfeladatoknál, így nagymértékben fejleszthetik a problémamegoldó gondolkodást. A témakör motiválja továbbá az eseményvezérelt és objektumorientált programozást, ezáltal jó szemléletet és továbblépési lehetőséget biztosíthat a professzionális programozási nyelvek, majd később az informatikus szakmák irányába.

Terveim között szerepel további játéktípusok feldolgozása a Scratch-ben, részben a fellelhető (esetleg más programozási nyelvhez kapcsolódó) irodalom, részben a saját elképzeléseim (és a már meglévő játékprogramjaim) alapján.

Irodalom

1. Andy Harris: *Beginning Flash Game Programming For Dummies*, Wiley Publishing, New Jersey (2006)
2. Arjan Egges: *Swift Game Programming for Absolute Beginners*, Apress, New York (2015)
3. *GameMaker Studio 2*
   https://www.yoyogames.com/gamemaker (utoljára megtekintve: 2017.11.01.)
4. *Buildbox*
   https://www.buildbox.com/ (utoljára megtekintve: 2017.11.01.)
5. Sean McManus: *Tanulj meg kódolni 10 lépésben!*, Babilon Kiadó (2017)
6. Carol Vorderman: *Computer Coding Games for Kids*, Dorling Kindersley, London (2015)
7. *OpenGameArt*
   https://opengameart.org/ (utoljára megtekintve: 2017.11.01.)
8. *Tiled*
   http://www.mapeditor.org/ (utoljára megtekintve: 2017.11.01.)
9. *World's Hardest Game*
   https://www.coolmath-games.com/0-worlds-hardest-game (utoljára megtekintve: 2017.11.01.)
10. *Maze Runner*
    https://scratch.mit.edu/projects/77278452/ (utoljára megtekintve: 2017.11.01.)