Problémamegoldás informatikai eszközökkel

Problémamegoldás, algoritmizálás

Szereted a sütit? Én nagyon! Készítsünk!

Mondjuk kókuszgolyót…
Az a kedvencem.
De hogyan?
Keressünk egy receptet, szerezzük be a hozzávalókat és vágjunk bele!
Olvassuk el a receptet, jegyzeteljük ki magunknak olyan részletességgel, hogy lépésenként, egyszerre csak egy feladatot hajtsunk végre.
A recept:

- A háztartási kekszet elkeverjük a szobahőmérsékletű vajjal, a porcukorral, a vaníliás cukorral, a kakaóval, a tejjel és a rumaromával.
- Nedves kézzel golyókat formálunk a masszából, és egyenként megforgatjuk őket a kókuszreszelékben. A kész kókuszgolyókat fóliával letakarva kb. 2 órára a hűtőszekrénybe tesszük, hogy megkeményedjenek egy kicsit.
- Végül tegyük a hűtőbe, 2 órán keresztül hűtsük.
- Ha jönnek a vendégek, tálaljuk fel nekik, különben megeszem én.

Program, algoritmus

Rendszerezzük a fogalmakat!

Feladatunk, amit végre szeretnénk hajtani, a sütikészítés. Ezt a feladatot programnak nevezzük.
A feladatot apró, pici, egyszerűen végrehajtható lépésekre bontjuk, véges sok lépésben végrehajtjuk a programot/feladatot. Leírhatjuk mondatokkal, egyszerű szöveggel is, vagy egy áttekinthetőbb folyamatábrán.

Algoritmusleíró eszközök

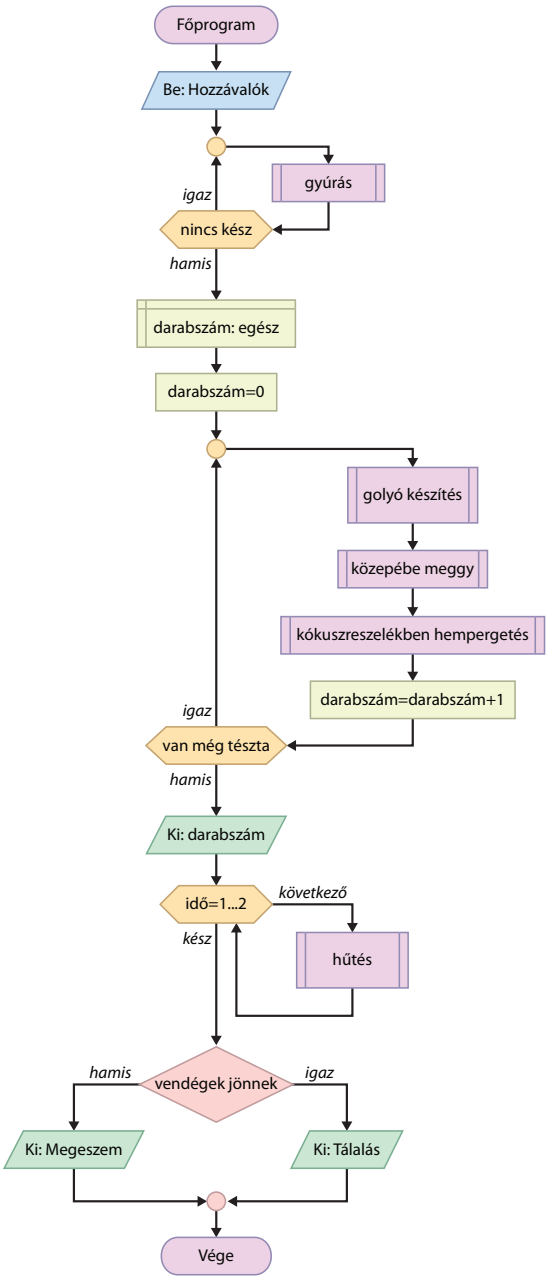
Szöveges leírás

Az algoritmus lépéseit mondatokkal írjuk le.

- Szerezzük be a hozzávalókat.
- Gyúrjuk össze a tészta alapanyagait, amíg egy jól gyúrható, jól elkevert tésztát kapunk.
- Egy nagy gombócot kaptunk, de nincsenek kis golyóink.
- Addig gyártsuk a golyókat, amíg van tészta.
- Vegyünk egy kis tésztát a kezünkbe, és gyúrjunk belőle golyót.
- Tegyünk a közepére egy meggyet.
- Hempergessük meg kókuszreszelékben.
- Számoljuk közben, hány darab jön ki belőle.

Folyamatábra

Az algoritmust a megoldás lépéseit ábrázoló diagrammal szemléltethetjük meg. Alakzatokat rajzolunk, majd vonalakkal kötjük össze őket. Ilyet legkönnyebben a Flowgorithm programmal készíthetünk, melyben programozási nyelv ismerete nélkül is futtathatunk programot, de megrajzolhatjuk akár egy fejlett szövegszerkesztőben is.



A folyamatábránkon láthatunk paralelogramma alakú adatbeviteli továbbá kiíró, téglalap alakú, változó értékeket kezelő utasításokat. Vannak rombusz alakú feltételes elágazások és hatszögbe kerülnek az ismétlések, a ciklusok feltételei. Vannak dupla téglalap alakú összetett utasítások, melyeket többször is használhatunk (meghívhatunk).
A folyamatábra előnye, hogy átláthatóvá teszi a program szerkezetét, és nem igényli egy konkrét programozási nyelv ismeretét. Viszont bármelyik nyelvre (pl. Scratch, Java, C#, Python) lefordítható, kódolható és futtatható programot készíthetünk belőle.

Bevezetés a Scratch-be, vizuális nyelvű programozás

A cica alig várja, hogy játsszál vele! Vele sok kalandban, játékban lesz részed, és közben észre se veszed, hogy megtanulsz programozni. Mindezt könnyedén, játékosan!
A Scratchet Mitchel Resnick és Andrés Monroy-Hernández találta ki, hogy – főleg a gyerekek – könnyen tanulhassanak programozni.


Mitchel Resnick


Andrés Monroy-Hernández

A program online verzióját használjuk, ezért indításhoz a *scratch.mit.edu*-t írjuk be a böngészőnkbe.

Ha nem magyarul jelenik meg az oldal, a honlap alján kiválaszthatjuk a magyar nyelvet.
Egyébként a programnak van letölthető, offline változata is.

Regisztráció

Mielőtt játszanánk a cicával, regisztrálni kell magunkat. Ehhez válaszd a *Regisztrálj* gombot. Ennek az a célja, hogy elmenthessük munkánkat a Scratch szerverére, így legközelebb bárhol, akár a suliban vagy otthon folytathatjuk, vagy éppen

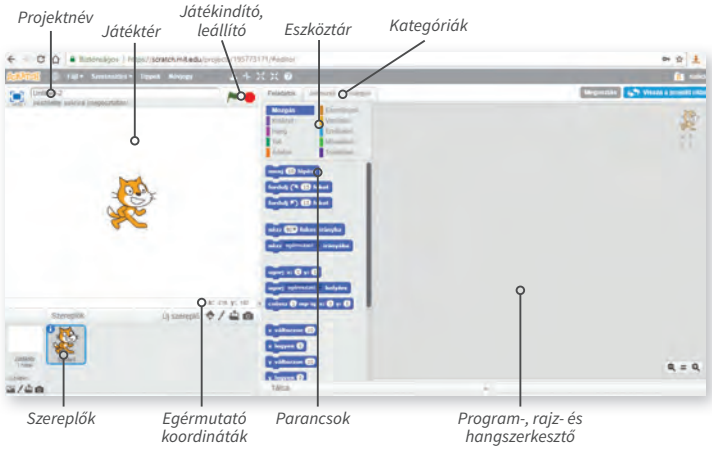
megoszthatjuk másokkal. Megnézhetjük továbbá mások munkáját, és sokat tanulhatunk belőle. A regisztrációkor meg kell adnunk egy e-mail címet, amire kapunk egy megerősítő levelet.
Kész is vagyunk, kezdődhet a nagy kaland, beléptünk a programba! Legközelebb már a *Jelentkezz be* gombbal kezdd a játékot.
A *Böngéssz* gombot választva, nézzük meg, mit készítettek ügyes gyerekek a Scratchben!
Ha kicsodálkoztad magad, alkoss te is ilyeneket! Hihetetlennek tűnik? Az, de képes vagy rá! Kezdődjön a játék!
Válaszd az *Alkoss* gombot, és ismerkedjünk meg a felülettel, amivel dolgozni fogunk!
A cica készen áll!

Próbáljátok ki a programot, mit lehet a cicával alkotni. Amelyik utasítást akarjátok adni, húzzátok be a *Szerkesztőfelületre*! Törölni úgy lehet, hogy visszahúzzátok a parancsot a menübe.

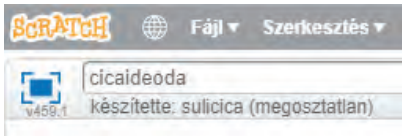
Mit is csinál egy macska? Mászkál...

Mozgatás

Próbáljuk ki! A macska menjen ide-oda a két fal között!



Először adjunk nevet a projektnek, ezután a program időnként automatikusan menti, úgyhogy biztonságban lesz programunk.



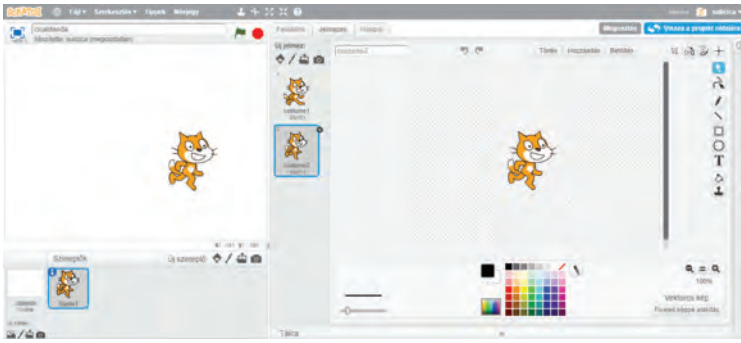
Menni úgy fog a program, hogy behúzzuk a *Szerkesztőfelületre* a *Menj* parancsot, és beállítjuk, mekkorát lépjen. Ha rákattintunk a parancsra, a cica végrehajtja. A cicát egérrel kezdőpontba helyezhetjük, hogy honnan kezdje a mozgást. Ha túl nagy értéket adunk meg a *menj... lépést* parancsban, pl. 1000, akkor sem lép ki a Játéktérről.

Valójában a lépéshosszt állítjuk.

Ha nagyobb útvonalon akarunk mozogni, akkor kisebb lépéshoszt választunk járáskor. Ezt a *Vezérlés* kategóriánál találjuk.
Egy ciklusnál (ismétlés) meg kell adni, hogy mit ismételgessünk. Ezt foglaljuk be egy sárga blokkba, húzzuk bele az utasításokat, és meg kell adnunk, hogy hányszor ismételjük meg.
Be kell állítanunk, hogy a Játéktér szélénél forduljon vissza. Ez a *Mozgás* kategóriában van. És ha a ciklusok közül a *Mindig*et választjuk, már megy is ide-oda...
Bár a mozgás így olyan, mintha jégen csúszna! A rajzfilmeknél több mozgásfázist is megrajzolnak, így élethűbb lesz a mozgás. Állítsuk be mi is!

Jelmezek

Meg lehet rajzolnunk, de előbb nézzük meg, mit takar a *Jelmezek* fülecske. Egy szereplőnek több jelmeze is lehet, azokat váltogathatja. Szerencsére cicánknak már megrajzolták egy másik mozgásfázisát. Ha szeretnénk még néhányat megrajzolni, megtehetjük a rajzfelületen.
Egészítsük ki programunkat a *Kinézet* kategóriában levő *következő jelmez* utasítással! Így lépésenként cserélgetni fogja az alakját a macska. Már egészen élethű lesz!



Még valami nagyon hiányzik, amit a macskák szoktak tenni. Igen, a nyávogás!

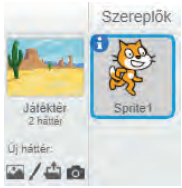
Hang

Ezen könnyen segíthetünk. Ha a *Hangok* fülecskére lépünk, láthatjuk, hogy a cicánkhoz tartozik egy hang is. Ha másikat akarunk, a Scratch könyvtárában sokfélét találunk, de választhatunk egy tetszőleges hangfájlt, vagy akár felvehetjük igazi cicánk hangját is.
Már csak az a kérdés, hogy hová illesszük be a hangot. Ha minden lépés után tesszük, nagyon sokszor és gyakran szólal meg. Az lenne jó, ha csak időnként nyávogna!

Kicsit át kell alakítanunk a programunkat. Menjen egy darabig, csak utána nyávogjon! Például lépjen 50-szer, akkor nyávogjon, és ezt ismételgesse befejezésig!
Itt már két ciklusunk (ismétlés) van, egyik a másikban. A belső ciklust 50-szer végrehajtja, lejátssza a hangfájlt, és ezt ismételgeti a külső ciklusban megadott lépésben.

Háttér

Csinosítsuk programunkat egy szép háttérrel! Választhatunk a Scratch saját képtárából, magunk is rajzolhatunk, tetszőleges képfájlt letölthetünk az internetről, de magunk is fotózhatunk.



Kezdőpozíció

Eddig onnan indult a cicánk, ahol az előző játékban leállítottuk. Esetleg az egérrel arrébb húztuk. De jobb, ha meghatározzuk, hogy honnan induljon a szereplőnk a játék kezdetekor. Ezt az *ugorj* paranccsal adhatjuk meg, amely a *Mozgás* kategóriában van. Érdemes megfigyelni, hogy amikor a cicát mozgatjuk az egérrel, az ugorj parancs megjegyzi a pozícióját, és ha behúzzuk a parancsot a Játéktérre, ez a hely lesz beállítva. Ettől még indulhat olyan irányba, ahogy az előzőekben le lett állítva. Ha azt akarjuk, hogy mindig jobbra kezdjen, még a *nézz 90 fokos irányba* utasítást ki kell adnunk.

A játék indítása

Már csak az indítás van hátra. Általában a zöld zászlóra kattintással indulnak a játékok. Ezt az *Események* kategóriában állíthatjuk be. De megadhatunk másfajta indítást is, illetve majd később a különféle szereplők indulhatnak sokféleképpen. Kész vagyunk, gratulálok, elkészült az első játékunk! Ha megmutatnád másoknak, oszd meg! Ha csak egy ismerősödnek szeretnéd elküldeni, a böngésző címsorában található azonosítókódot küldd el!

Rendszerezzük végül az itt használt fogalmainkat!

Egyszerűen végrehajtható utasítások: *ugorj*, *menj*, *nézz*, *játszd le*.
Feltételhez kötött utasítások, elágazások, amikor kétféle irányban folytatódhat a program, pl. *ha szélen vagy, pattanj vissza*. Amennyiben teljesül a feltétel, az egyik irányba megy tovább a folyamat, különben a másikba.
Ismétlések, ciklusok: *mindig*, *ismételd*. Ilyenkor ugyanazokat az utasításokat többször hajtja végre a program.
Vizuális nyelvű programozás: grafikai elemeket használó programozási nyelv, ahol blokkok behúzogatásával lehet programot készíteni.

- Cicánk egyedül érzi magát. Válasszunk egy másik helyszínt, és tegyünk mellé még néhány társat!
- Próbálgassuk, milyen lesz a mozgás, ha a lépéshosszat változtatjuk.
- Változtassuk meg a belső ciklus ismétlésszámát, figyeljük meg, milyen lesz a nyávogássűrűség!
- Keresd meg a Wikipédián a Scratch kitalálóinak tudományos életrajzát!

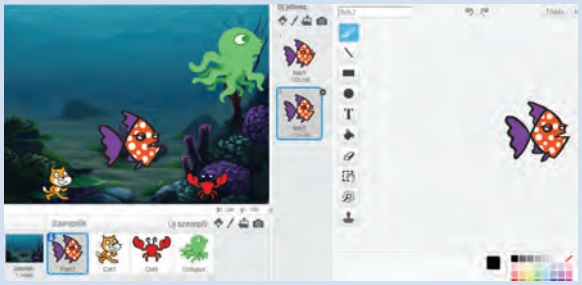
Az Akvárium

Nézzük meg az előző oldalon feladott házi feladatot!

Módosítsuk! A háttér egy akvárium, amelyben három halacska úszkál. Még mit tehetnek? Mi lenne, ha az egyik beszélgetne a cicával? De legalábbis tátogna?

Mivel a halacskáknak, csak egy jelmeze van, ezért készítünk egy másikat! Rajzoljuk át a száját, mintha kinyitná.

Ehhez készítsünk másolatot a jelmezéről (duplikálás), majd a rajzeszközökkel módosítsuk! Ha kész, egészítsük ki a programot, hogy váltogassa a jelmezét! Már egészen élethű a rajzfilmünk…



A Kastély

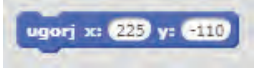
Milyen volt eddig a mozgatás? A *menj* parancsot használtuk. Hogy hívjuk ezt a mozgást? Térugrás! Az egyik helyről eltűnik, a másikon adott lépéssel távolabb megjelenik.

Van másfajta is, például a *csúszás*.

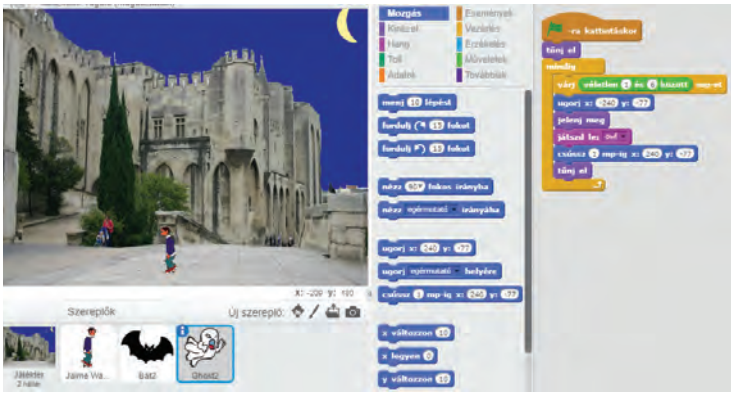
Itt a közbülső pontokon is végig látszik a mozgás. Megadjuk, melyik pontba jusson el, és mennyi idő alatt. Ekkor a mozgás közben nem változhatnak a jelmezek, mint a jégen való csúszáskor.

Próbáljuk ki, nézzük meg egy szellemkastély életét! A szellem, és a denevér a kép egyik szélétől a másikig száguld, de mindig csak egy irányban haladjon, ne pattanjon vissza. Ha kimegy a képből, valamivel később ismételje meg újra, ugyanarra!

Használnunk kell az *ugorj* parancsot, mellyel bárhonnan egy adott helyre vihetjük a szereplőket. Hogyan kerül vissza újra a kezdőoldalra? Úgy, hogy amikor a kép szélén eltűnik, visszavisszük a kezdőhelyére, ott megjelenik, és újra mehet.

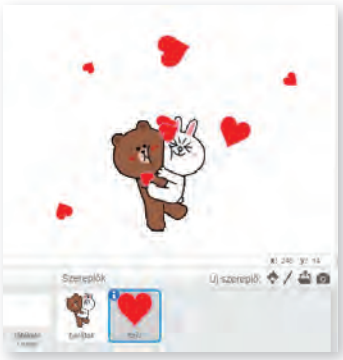


Ahhoz hogy ne legyen monoton, kiszámítható a felbukkanásuk, a megjelenésük legyen véletlenszerű. *Várj*anak egy bizonyos ideig (1 és 6 másodperc között), utána jelenjenek meg. A háttér is változzon bizonyos időnként nappaliról éjszakaira, majd vissza.



Jóbarátok

Maci és nyuszi rég nem látták egymást. Amikor végre találkoztak, nagyon megörültek egymásnak. Ennek a boldogságnak a hőfokát jelzik a röpködő szívecskék. Készítsünk sok-sok piros képet!



Ezt létrehozhatjuk úgy is, hogy nagyon sok hasonló szereplőt hozunk létre. Előtűnnek, repkednek, elszállnak, a széleken eltűnnek.

Csináljuk okosabban! Hozzunk létre egyet, és rengeteg másolatát!

Ehhez két újabb utasítást kell megismernünk. A másolatok létrehozásához a *készíts másolatot magadról* parancsot használd! Azt, hogy mennyi

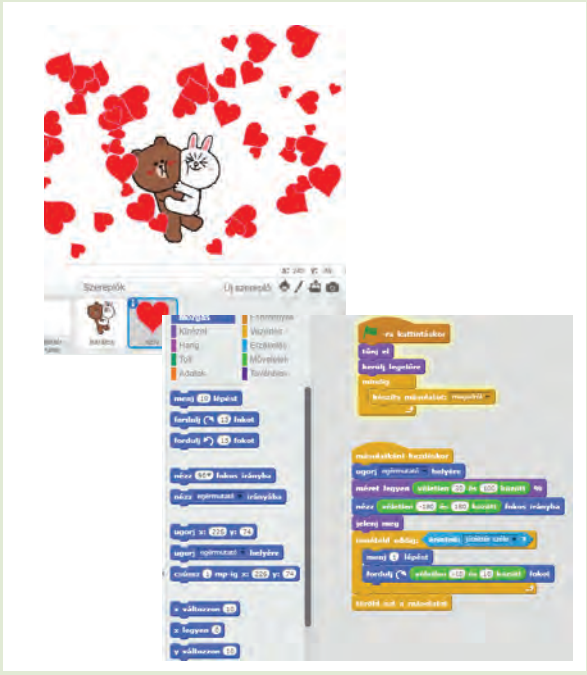
időnként jöjjenek létre a klónok, a *várj* utasítással szabályozhatjuk. Ha véletlen számmal adjuk meg, nem egyenletes, hanem változatos lesz a megjelenés.

A *másolatként kezdéskor*, megadhatjuk, mi történjen megszületéskor. Pl. megadhatjuk, hol jelenjen meg, mekkora legyen, merre induljon el, mikor jelenjen meg, mit csináljon, illetve mikor törlődjön. Példánkban kisebb-nagyobb szívecskék röpködnek összevissza, majd, ha elérik a falat, eltűnnek, megszűnnek. Az összevissza mozgást a véletlenszerű forgatásokkal lehet elérni.

A szülő szereplőt célszerű a legelején letiltani, eltüntetni, ne legyen állandóan látható, ettől még létre tudja hozni a másolatokat.

Ha azt szeretnénk, hogy a figurák elé is bejöjjenek a szívek, ezt a *kerülj legelőre* paranccsal biztosíthatjuk.

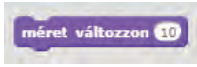
Módosítsuk játékunkat! Változtassunk meg benne sok mindent, alakítsuk kedvünkre! Például jöjjenek gyorsabban a szívecskék, lassabban tűnjenek el! Ne egy helyen jöjjenek létre, hanem amerre éppen az egérmutatót húzzuk.



Ha túl gyorsan jönnek létre a másolatok, vegyük ki a hangot, mert nagy lesz a hangzavar.

Még egy variáció. Játsszunk fizikaórát, keltsünk vízhullámokat! Mintha az ujjunkat húznánk a víz felszínén.
Rajzoljunk magunk egy kört, ami majd egyre nagyobb lesz, mint amikor kavicsot dobunk a vízbe. A másolat itt is kövesse az egérmutatót!

A méretváltoztatáshoz használjuk a *méret változzon* parancsot!

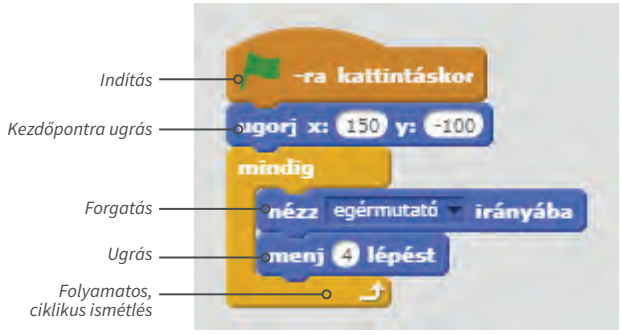


- Készíts korcsolyázókat egy jégpályán, szánkózókat, síelőket egy domboldalon a *csússz* parancs használatával!
- Hozz létre hangyabolyt a *készíts másolatot* parancs felhasználásával!
- Készíts egy járást bemutató programot, amelyben az egérmutatót követve lábnyomok jelennek meg a homokban!

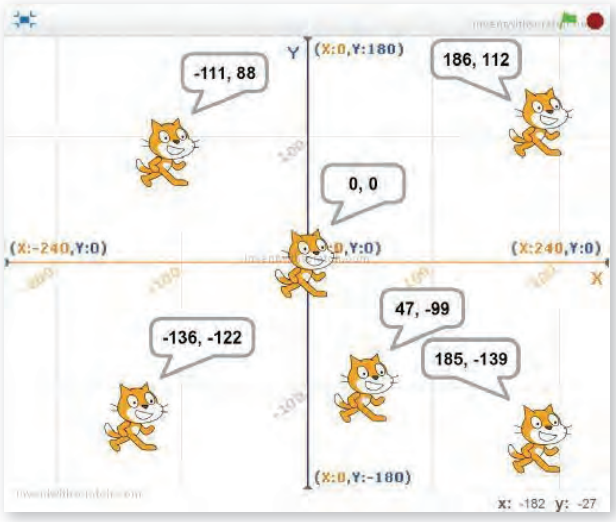
Az Egérfogó elkészítése

Játsszunk egy jót! Biztos találkoztál már valamilyen pontszerző játékkal. Mentél, futottál, valamit össze kellett gyűjteni, el kellett érni, és ha sikerült, jutalompontokat kaptál érte. Ha összegyűlt egy bizonyos mennyiség belőle, megnyerted a játékot, vagy továbbjutottál egy másik pályára. Szeretnél egy ilyen játékot írni? Meg tudod csinálni? Lássuk, vágjunk bele!

Eszközök: Gondoljuk végig, mi kell a játékhoz! Például legyen az a cél, egy kisegér szedegeti az almát, hogy megegye. Ha ötöt összegyűjtött, megnyerte a játékot. De közben előkerül egy macska, aki viszont az egeret szeretné elkapni. Ha sikerül neki, a cica nyert! Az almák véletlenszerűen helyezkedjenek el, az egérkét mi vezessük az egérmutatóval, a macska viszont automatikusan menjen az egér irányába! (Szereplőnek másokat is választhatsz.)
Mi kell hozzá? Szereplők, akikkel játszunk, utasítások, amelyek vezérlik a játékosokat.

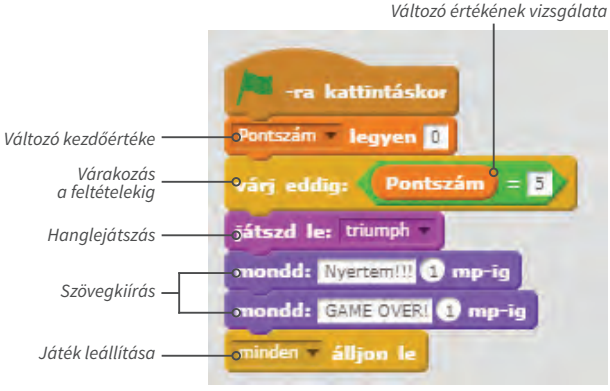


Határozzuk meg a kezdőpozíciót! Nézze meg, merre van az egérmutató, forduljon abba az irányba, majd menjen arra. De mindezt ne csak egyszer, hanem folyamatosan, újra és újra, amíg a játék véget nem ér.



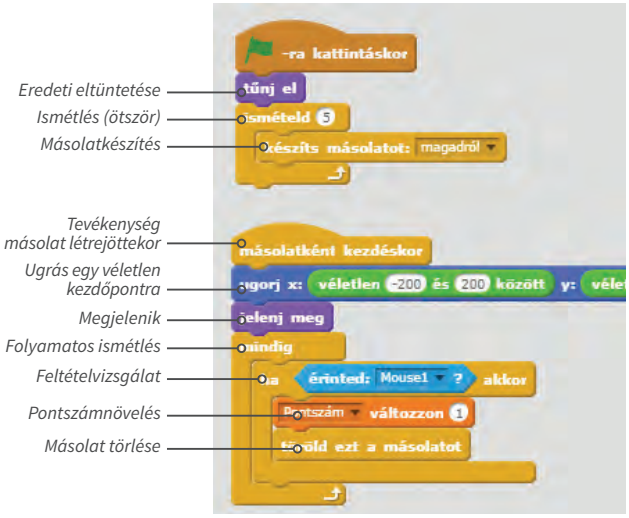
A pozíciót két koordinátával adhatjuk meg, ezeket könnyen leolvashatjuk, hiszen kiírja az egérmutató helyét a program.

Változó: Mit tegyen még az egér? Össze kell szednie az almákat.
Kell egy változó, amelyben számoljuk, hány almát szedett össze. Az aktuális értéket írjuk ki pl. a bal felső sarokba. Kezdetben legyen az értéke 0. Hagyjuk, hogy szedegesse az almákat. Ha összesedett öt almát, játsszuk le győzelmi zenénket. Végül írjuk ki, hogy a kisegér nyert, majd állítsuk le a programot.

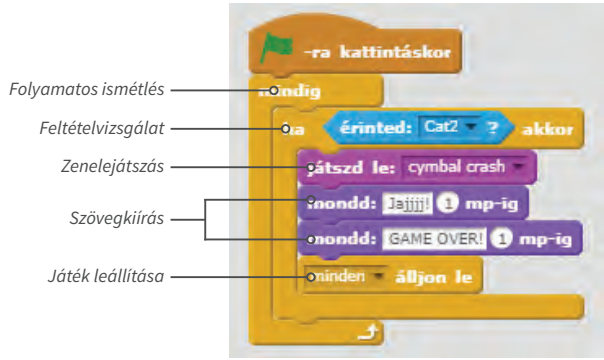


A macskának egyszerűbb a feladata. Neki csak egy dolga van, menjen az egér irányába. Sebességét a lépés hosszával változtathatjuk. Ha azt akarjuk, hogy legyen esélye az egérnek, akkor legyen lassabb a macska. Minél közelebb van a két sebesség egymáshoz, annál nehezebben nyer a kisegér.

Másolatkészítés, véletlen szám, feltétel, elágazás: Most az almák következnek! Szeretnénk sok almát létrehozni, viszont nem akarjuk egyenként megrajzolni, pláne, ha több száz van. Ezért inkább készítsünk másolatokat egyről, és azokkal játsszunk, az eredetit pedig tüntessük el. Miután megszületett a másolat, helyezzük el egy véletlenszerű helyre a színpad alsó felében. A pontszámot az almák fogják változtatni, ha hozzáér az egér, növeljük a darabszámot eggyel, és utána tűnjön el az alma.



Nagyon beleéltük magunkat a győzelembe, de sajnos másképp is alakulhat a játék. Ha a macska elkapja a kisegeret, vesztettünk. Ezt is le kell írnunk a programunkban! Ha a macska hozzáér az egérhez, vége a játéknak.



Ezzel elkészítettük a játékot, megírtuk a programot. Már csak a kipróbálás van hátra, és egy jókedvű ügyességi játék!

Összefoglalás

Megírtuk első komoly programunkat. Elégedetten dőlhetünk hátra. Még sok ötletünk van, amivel jobbá, izgalmasabbá tehetjük, tehát nem kell befejezni, mindig lehet továbbfejleszteni!

Mi is történt? Kitaláltunk egy történetet: egy kisegér almákat gyűjt. Ha eleget összeszedett, megnyerte a játékot. Ha azonban időközben elkapja a macska, vesztett.

Ezt az összetett programot sok kicsi apró lépésre bontottuk, ezeket egyenként kódoltuk. A játékhoz három szereplőt választottunk, akiknek feladatokat adtunk. Eseményeket értékeltünk ki, ha érintkeztek egymással a szereplők, valami történt. Számoltuk a pontokat, ha elértek egy értéket, véget ért a játék.

Rendszerezzük a játék készítése során használt fogalmainkat!

Változó: értékét változtatható, adatot tároló, névvel azonosítható programelem.
Értékadás: a változónak értéket adunk.
Ciklus: többször végrehajtott művelet.
Elágazás: Két vagy több lehetőség közötti választás, hogy melyik legyen a következő lépés. Ez egy feltételtől függ.

- Van ötleted, hogyan lehetne az Egérfogót továbbfejleszteni? Folytasd a fejlesztést!
- Találj ki egy másik történetet, válassz hozzá szereplőket, és készíts játékot! Az egérmutató helyett a kurzormozgató nyilakkal irányítsd a szereplőket!