3. beadandó feladat: WPF grafikus felületű alkalmazás adatbáziskezeléssel

Közös követelmények:

- A beadandók dokumentációból, valamint programból állnak, utóbbi csak a megfelelő dokumentáció bemutatásával értékelhető. Csak funkcionálisan teljes, a feladatnak megfelelő, önállóan megvalósított, személyesen bemutatott program fogadható el.

- A megvalósításnak felhasználóbarátnak, és könnyen kezelhetőnek kell lennie. A szerkezetében törekednie kell az objektumorientált szemlélet megtartására.

- A programot MVVM architektúrában kell felépíteni, amelyben a megjelenítésrétege elkülönül a modellnézettől, a modelltől, valamint az adatkezeléstől. A modell és az adatkezelés nem tartalmazhat semmilyen grafikus felületbeli osztályra történő hivatkozást, csak eseményeket küldhet a nézetmodellnek. A nézetmodell nem tartalmazhat semmilyen játékbeli adatot, a nézet pedig semmilyen háttérkódot.

- A prezisztens adatokat adatbázisban kell tárolni, és az *Entity Framework* objektum-relációs adatmodelljének segítségével ábrázolni. A tartós megőrzésre kerülő állapotot entitás alapú leképezéssel kell reprezentálni.

- A modell működését egységtesztek segítségével kell ellenőrizni. Nem kell teljes körű tesztet végezni, azonban a lényeges funkciókat, és azok hatásait ellenőrizni kell. Az adatbetöltés/mentés teszteléséhez a perzisztencia működését szimulálni kell.

- A dokumentációnak jól áttekinthetőnek, megfelelően formázottnak kell lennie, tartalmaznia kell a fejlesztő adatait, a feladatleírást, a feladat elemzését, felhasználói eseteit (UML felhasználói esetek diagrammal), a program szerkezetének leírását (UML osztálydiagrammal), az adatbázis felépítésének leírását (egyedkapcsolati diagrammal), valamint a tesztesetek leírását. A dokumentáció ne tartalmazzon kódrészleteket, illetve képenyőképeket. A megjelenő diagramokat megfelelő szerkesztőeszköz segítségével kell előállítani. A dokumentációt elektronikusan, PDF formátumban kell leadni.

Feladatok:

1. **Maci Laci**

   Készítsünk programot, amellyel a következő játékot játszhatjuk.
   Adott egy $n \times n$ elemből álló játékpálya, amelyben Maci Lacival kell piknikkosarakra vadásznunk. A játékpályán az egyszerű mezők mellett elhelyezkednek akadályok (pl. fa), valamint piknikkosarak. A játék célja, hogy a piknikkosarakat minél gyorsabban begyűjtsük.
   Az erdőben vadőrök is járőröznek, akik adott időközönként lépnek egy mezőt (vízszintesen, vagy függőlegesen). A járőrözés során egy megadott irányba haladnak egészen addig, amíg akadályba (vagy az erdő szélébe) nem ütköznek, ekkor megfordulnak, és visszafelé haladnak (tehát folyamatosan egy vonalban járőröznek). A vadőr járőrözés közben a vele szomszédos mezőket látja (átlósan is, azaz egy 3×3-as négyzetet).
   A játékos kezdetben a bal felső sarokban helyezkedik el, és vízszintesen, illetve függőlegesen mozoghat (egyesével) a pályán, a piknikkosárra való rálépéssel pedig felveheti azt. Ha Maci Lacit meglátja valamelyik vadőr, akkor a játékos veszít.
   A pályák méretét, illetve felépítését (piknikkosarak, akadályok, vadőrök kezdőpozíciója) tároljuk relációs adatbázisban. A program legalább 3 különböző méretű pályát tartalmazzon.
   A program biztosítson lehetőséget új játék kezdésére a pálya kiválasztásával, valamint játék szüneteltetésére (ekkor nem telik az idő, és nem léphet a játékos). Ismerje fel, ha vége a játéknak, és jelezze, győzött, vagy veszített a játékos. A program játék közben folyamatosan jelezze ki a játékidőt, valamint a megszerzett piknikkosarak számát.

2. **Bombázó**

   Készítsünk programot, amellyel a következő játékot játszhatjuk.
   Adott egy $n \times n$ mezőből álló játékpálya, amelyeken falak, illetve járható mezők helyezkednek el, valamint ellenfelek járőröznek. A játékos célja, hogy ellenfeleit bombák segítségével minél gyorsabban legyőzze.
   Az ellenfelek adott időközönként lépnek egy mezőt (vízszintesen, vagy függőlegesen) úgy, hogy folyamatosan előre haladnak egészen addig, amíg falba nem ütköznek. Ekkor véletlenszerűen választanak egy új irányt, és arra haladnak tovább.
   A játékos figurája kezdetben a bal felső sarokban helyezkedik el, és vízszintesen, illetve függőlegesen mozoghat (egyesével) a pályán, de ha találkozik (egy pozíciót foglal el) valamely ellenféllel, akkor meghal.
   A játékos bombát rakhat le az aktuális pozíciójára, amely rövid időn belül robban megsemmisítve a 3 sugáron belül (azaz egy 7×7-es négyzetben) található ellenfeleket (falon át is), illetve magát a játékost is, ha nem menekül onnan.

A pályák méretét, illetve felépítését (falak, ellenfelek kezdőpozíciója) tároljuk relációs adatbázisban. A program legalább 3 különböző méretű pályát tartalmazzon.
A program biztosítson lehetőséget új játék kezdésére a pálya kiválasztásával, valamint játék szüneteltetésére (ekkor nem telik az idő, és nem léphet a játékos). Ismerje fel, ha vége a játéknak, és jelezze, győzött, vagy veszített a játékos. A program játék közben folyamatosan jelezze ki a játékidőt, valamint a felrobbantott ellenfelek számát.

3. Elszabadult robot

Készítsünk programot, amellyel a következő játékot játszhatjuk.
Adott egy $n \times n$ mezőből álló játékpálya, amelyben egy elszabadult robot bolyong, és a feladatunk az, hogy betereljük a pálya közepén található mágnes alá, és így elkapjuk.
A robot véletlenszerű pozícióban kezd, és adott időközönként lép egy mezőt (vízszintesen, vagy függőlegesen) úgy, hogy általában folyamatosan előre halad egészen addig, amíg falba nem ütközik. Ekkor véletlenszerűen választ egy új irányt, és arra halad tovább. Időnként még jobban megkergül, és akkor is irányt vált, amikor nem ütközik falba.
A játékos a robot terelését úgy hajthatja végre, hogy egy mezőt kiválasztva falat emelhet rá. A felhúzott falak azonban nem túl strapabíróak. Ha a robot ütközik a fallal, akkor az utána eldől. A ledőlt falakat már nem lehet újra felhúzni, ott a robot később akadály nélkül áthaladhat.
A program biztosítson lehetőséget új játék kezdésére a pályaméret megadásával (7×7, 11×11, 15×15), valamint játék szüneteltetésére (ekkor nem telik az idő, nem lép a robot, és nem lehet mezőt se kiválasztani). Ismerje fel, ha vége a játéknak, és jelenítse meg, hogy milyen idővel győzött a játékos. A program játék közben folyamatosan jelezze ki a játékidőt. Ezen felül szüneteltetés alatt legyen lehetőség a játék elmentésére, valamint betöltésére relációs adatbázisba.

4. Labirintus

Készítsünk programot, amellyel a következő játékot játszhatjuk.
Adott egy $n \times n$ elemből álló játékpálya, amely labirintusként épül fel, azaz fal, illetve padló mezők találhatóak benne, illetve egy kijárat a jobb felső sarokban. A játékos célja, hogy a bal alsó sarokból indulva minél előbb kijusson a labirintusból.
A labirintusban nincs világítás, csak egy fáklyát visz a játékos, amely a 2 szomszédos mezőt világítja meg (azaz egy 5×5-ös négyzetet), de a falakon nem tud átvilágítani.
A játékos figurája kezdetben a bal alsó sarokban helyezkedik el, és vízszintesen, illetve függőlegesen mozoghat (egyesével) a pályán.

A pályák méretét, illetve felépítését (falak, padlók) tároljuk relációs adatbázisban. A program legalább 3 különböző méretű pályát tartalmazzon.
A program biztosítson lehetőséget új játék kezdésére a pálya kiválasztásával, valamint játék szüneteltetésére (ekkor nem telik az idő, és nem léphet a játékos), továbbá ismerje fel, ha vége a játéknak. A program játék közben folyamatosan jelezze ki a játékidőt.

5. Menekülj

Készítsünk programot, amellyel a következő játékot játszhatjuk.
Adott egy $n \times n$ elemből álló játékpálya, ahol a játékos két üldöző elől próbál menekülni, illetve próbálja őket aknára csalni.
Kezdetben a játékos játékpálya felső sorának közepén helyezkedik el, a két üldöző pedig az alsó két sarokban. Az ellenfelek adott időközönként lépnek egy mezőt a játékos felé haladva úgy, hogy ha a függőleges távolság a nagyobb, akkor függőlegesen, ellenkező esetben vízszintesen mozognak a játékos felé.
A pályán véletlenszerű pozíciókban aknák is elhelyezkednek, amelyekbe az ellenfelek könnyen beleléphetnek, ekkor eltűnnek (az akna megmarad).
A játékos vízszintesen, illetve függőlegesen mozoghat (egyesével) a pályán, és célja, hogy az ellenfeleket aknára csalja, miközben ő nem lép aknára. Ha sikerül minden üldözőt aknára csalnia, akkor győzött, ha valamely ellenfél elkapja (egy pozíciót foglal el vele), vagy aknára lép, akkor veszített.
A program biztosítson lehetőséget új játék kezdésére a pályaméret megadásával (11×11, 15×15, 21×21), valamint játék szüneteltetésére (ekkor nem telik az idő, és nem léphet senki). Ismerje fel, ha vége a játéknak, és jelenítse meg, hogy győzött, vagy veszített-e a játékos. Ezen felül szüneteltetés alatt legyen lehetőség a játék elmentésére, valamint betöltésére relációs adatbázisba. A program játék közben folyamatosan jelezze ki a játékidőt.

6. Tetris

Készítsünk programot a közismert Tetris játékra.
Adott egy $n \times m$ pontból álló tábla, amely kezdetben üres. A tábla tetejéről egymás után új, 4 kockából álló építőelemek hullanak, amelyek különböző formájúak lehetnek (kocka, egyenes, L alak, tető, rombusz). Az elemek rögzített sebességgel esnek lefelé, és az első, nem telített helyen megállnak. Amennyiben egy sor teljesen megtelik, az eltűnik a játékmezőről, és minden felette lévő kocka eggyel lejjebb esik.
A játékosnak lehetősége van az alakzatokat balra, jobbra mozgatni, valamint forgatni óramutató járásával megegyező irányba, így befolyásolhatja azok mozgását. A játék addig tart, amíg a kockák nem érik el a tábla tetejét.
A program biztosítson lehetőséget új játék kezdésére a pályaméret megadásával (4×16, 8×16, 12×16), valamint játék szüneteltetésére (ekkor nem telik az idő,

és nem mozognak az elemek). Ismerje fel, ha vége a játéknak, és jelenítse meg, mennyi volt a játékidő. Ezen felül szüneteltetés alatt legyen lehetőség a játék elmentésére, valamint betöltésére relációs adatbázisba.

7. Lopakodó

Készítsünk programot, amellyel a következő játékot játszhatjuk.
Adott egy $n \times n$ elemből álló játékpálya, amely falakból és padlóból áll, valamint őrök járőröznek rajta. A játékos feladata, hogy a kiindulási pontból eljusson a kijáratig úgy, hogy közben az őrök nem látják meg. Természetesen a játékos, illetve az őrök csak a padlón tudnak járni.
Az őrök adott időközönként lépnek egy mezőt (vízszintesen, vagy függőlegesen) úgy, hogy folyamatosan előre haladnak egészen addig, amíg falba nem ütköznek. Ekkor véletlenszerűen választanak egy új irányt, és arra haladnak tovább. Az őr járőrözés közben egy 2 sugarú körben lát (azaz egy 5×5-ös négyzetet), ám a falon nem képes átlátni.
A játékos a pálya előre megadott pontján kezd, és vízszintesen, illetve függőlegesen mozoghat (egyesével) a pályán.
A pályák méretét, illetve felépítését (falak és kijárat helyzete, játékos és őrök kezdőpozíciója) tároljuk relációs adatbázisban. A program legalább 3 különböző méretű pályát tartalmazzon.
A program biztosítson lehetőséget új játék kezdésére a pálya kiválasztásával, valamint játék szüneteltetésére (ekkor nem telik az idő, és nem léphet a játékos). Továbbá ismerje fel, ha vége a játéknak. Ekkor jelenítse meg, hogy győzött, vagy veszített-e a játékos.

8. Aszteroidák

Készítsünk programot, amellyel az aszteroidák játékot játszhatjuk.
A feladatunk az, hogy egy űrhajó segítségével átnavigáljuk egy aszteroidamezőn. Az űrhajóval a képernyő alsó sorában tudunk balra, illetve jobbra navigálni. A képernyő felső sorában meghatározott időközönként véletlenszerű pozícióban jelennek meg az aszteroidák, amelyek folyamatosan közelednek állandó sebességgel a képernyő alja felé. Az idő múlásával egyre több aszteroida jelenik meg egyszerre, így idővel elkerülhetetlenné válik az ütközés. A játék célja az, hogy az űrhajó minél tovább elkerülje az ütközést.
A program biztosítson lehetőséget új játék kezdésére, valamint játék szüneteltetésére (ekkor nem telik az idő, és nem mozog semmi a játékban). Ismerje fel, ha vége a játéknak, és jelenítse meg, mennyi volt a játékidő. Ezen felül szüneteltetés alatt legyen lehetőség a játék elmentésére, valamint betöltésére relációs adatbázisba.

9. Gyorsulás

Készítsünk programot, amellyel az alábbi motoros játékot játszhatjuk.
A feladatunk, hogy egy gyorsuló motorral minél tovább tudjunk haladni. A gyorsuláshoz a motor üzemanyagot fogyaszt, egyre többet. Adott egy kezdeti mennyiség, amelyet a játék során üzemanyagcellák felvételével tudunk növelni.
A motorral a képernyő alsó sorában tudunk balra, illetve jobbra navigálni. A képernyő felső sorában meghatározott időközönként véletlenszerű pozícióban jelennek meg üzemanyagcellák, amelyek folyamatosan közelednek a képernyő alja felé. Mivel a motor gyorsul, ezért a cellák egyre gyorsabban fognak közeledni, és mivel a motor oldalazó sebessége nem változik, idővel egyre nehezebb lesz felvenni őket, így egyszer biztosan kifogyunk üzemanyagból. A játék célja az, hogy a kifogyás minél később következzen be.
A program biztosítson lehetőséget új játék kezdésére, valamint játék szüneteltetésére (ekkor nem telik az idő, és nem mozog semmi a játékban). Ismerje fel, ha vége a játéknak, és jelenítse meg, mennyi volt a játékidő. Ezen felül szüneteltetés alatt legyen lehetőség a játék elmentésére, valamint betöltésére relációs adatbázisba.

10. Aknamező

Készítsünk programot a következő játékra.
A játékban egy tengeralattjárót kell irányítanunk a képernyőn (balra, jobbra, fel, illetve le), amely felett ellenséges hajók köröznek, és folyamatosan aknákat dobnak a tengerbe. Az aknáknak három típusa van (könnyű, közepes, nehéz), amely meghatározza, hogy milyen gyorsan süllyednek a vízben (minél nehezebb, annál gyorsabban).
Az aknákat véletlenszerűen dobják a tengerbe, ám mivel a hajóskapitányok egyre türelmetlenebbek, egyre gyorsabban kerül egyre több akna a vízbe. A játékos célja az, hogy minél tovább elkerülje az aknákat. A játék addig tart, ameddig a tengeralattjárót el nem találta egy akna.
A program biztosítson lehetőséget új játék kezdésére, valamint játék szüneteltetésére (ekkor nem telik az idő, és nem mozog semmi a játékban). Ismerje fel, ha vége a játéknak, és jelenítse meg, mennyi volt a játékidő. Ezen felül szüneteltetés alatt legyen lehetőség a játék elmentésére, valamint betöltésére relációs adatbázisba.

11. Snake

Készítsük programot, amellyel a klasszikus kígyó játékot játszhatjuk.
Adott egy $n \times n$ elemből álló játékpálya, amelyben akadályok (falak) találhatóak. A játékos egy kezdetben 5 hosszú kígyóval indul a képernyő közepén, amely vízszintesen, illetve függőlegesen halad rögzített időközönként a legutoljára beállított irányba. A kígyóval elfordulhatunk balra, illetve jobbra. A pályán

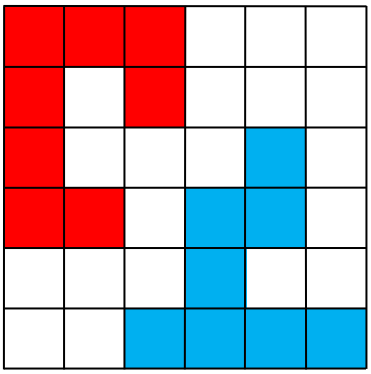
véletlenszerű pozícióban mindig megjelenik egy tojás, amelyet a kígyóval meg kell etetni. Minden etetéssel eggyel nagyobb lesz a kígyó. A játék célja, hogy a kígyó minél tovább elkerülje az ütközést az akadályokkal, a pálya szélével, illetve saját magával.
A pályák méretét, illetve felépítését (falak helyzete) tároljuk relációs adatbázisban. A program legalább 3 különböző méretű pályát tartalmazzon.
A program biztosítson lehetőséget új játék kezdésére a pálya kiválasztásával, valamint játék szüneteltetésére (ekkor nem telik az idő, és nem mozog a kígyó). Továbbá ismerje fel, ha vége a játéknak. Ekkor jelenítse meg, hány tojást sikerült elfogyasztania a játékosnak.

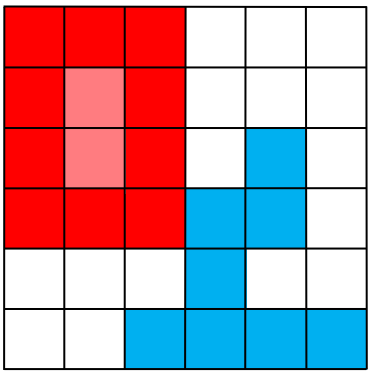
12. Fénymotor párbaj

Készítsük programot, amellyel a Tronból ismert fénymotor párbajt játszhatjuk.
Adott egy $n \times n$ elemből álló játékpálya. A két játékos a bal, illetve jobb oldal közepén indul egy-egy fénymotorral, amely egyenesen halad (rögzített időközönként) a legutoljára beállított irányba (függőlegesen, vagy vízszintesen). A motorokkal lehetőség van balra, illetve jobbra fordulni. A fénymotor mozgás közben fénycsíkot húz, ami a játék végéig ott marad. Az a játékos veszít, aki előbb nekiütközik a másik játékos motorjának, bármelyikük fénycsíkjának vagy a pálya szélének.
A program biztosítson lehetőséget új játék kezdésére a pályaméret megadásával (12×12, 24×24, 36×36), valamint játék szüneteltetésére (ekkor nem telik az idő, és nem mozognak a motorok). Ismerje fel, ha vége a játéknak, és jelenítse meg, melyik játékos győzött. Ezen felül szüneteltetés alatt legyen lehetőség a játék elmentésére, valamint betöltésére relációs adatbázisba.

13. Bekerítés

Készítsünk programot, amellyel a következő két személyes játékot játszhatjuk.
Adott egy $n \times n$ mezőből álló tábla, amelyre a játékosok 2×1-es méretű téglalapokat helyezhetnek el (vízszintesen, vagy függőlegesen).
A játékosok felváltva léphetnek. A játék célja, hogy a téglalapokkal elhatároljuk a tábla egy részét (teljesen körbevéve téglalapokkal), amelyben így minden mező a játékosé lesz (beleértve az ellenfél által korábban elfoglalt mezőket is). A program külön jelölje meg a lehelyezett téglalapokat, illetve az elfoglalt területeket, és játék közben folyamatosan jelenítse meg az elfoglalt terület méretét játékosonként.



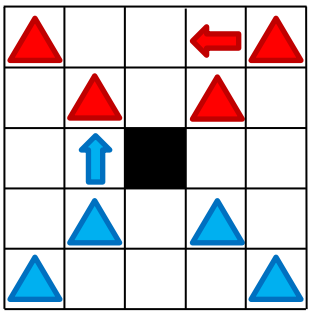


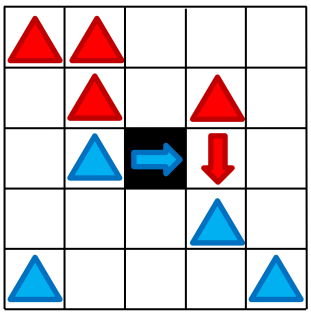
A program biztosítson lehetőséget új játék kezdésére a pályaméret megadásával (6×6, 8×8, 10×10), valamint játék mentésére és betöltésére relációs adatbázisba. Ismerje fel, ha vége a játéknak, és jelenítse meg, melyik játékos győzött.

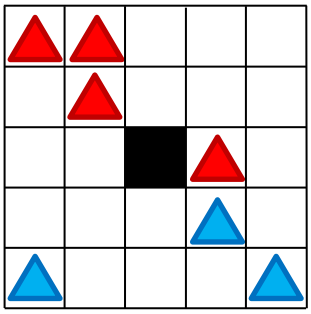
14. Fekete lyuk

Készítsünk programot, amellyel a következő két személyes játékot lehet játszani. Adott egy $n \times n$ mezőből álló tábla, amelyen két játékos űrhajói helyezkednek el, középen pedig egy fekete lyuk. A játékos $n - 1$ űrhajóval rendelkezik, amelyek átlóban helyezkednek el a táblán (az azonos színűek egymás mellett, ugyanazon az oldalon).

A játékosok felváltva léphetnek. Az űrhajók vízszintesen, illetve függőlegesen mozoghatnak a táblán, de a fekete lyuk megzavarja a navigációjukat, így nem egy mezőt lépnek, hanem egészen addig haladnak a megadott irányba, amíg a tábla széle, a fekete lyuk, vagy egy másik, előtte lévő űrhajó meg nem állítja őket (tehát másik űrhajót átlépni nem lehet). Az a játékos győz, akinek sikerül űrhajóinak felét eljuttatnia a fekete lyukba.







A program biztosítson lehetőséget új játék kezdésére a pályaméret megadásával (5×5, 7×7, 9×9), valamint játék mentésére és betöltésére relációs adatbázisba. Ismerje fel, ha vége a játéknak, és jelenítse meg, melyik játékos győzött.

15. Aknakereső

Készítsünk programot, amellyel az aknakereső játék két személyes változatát játszhatjuk.
Adott egy $n \times n$ mezőből álló tábla, amelyen rejtett aknákat helyezünk el. A többi mező szintén elrejtve tárolják, hogy a velük szomszédos 8 mezőn hány akna helyezkedik el.
A játékosok felváltva léphetnek. Egy mező felfedjük annak tartalmát. Ha az akna, a játékos veszített. Amennyiben a mező nullát rejt, akkor a vele szomszédos mezők is automatikusan felfedésre kerülnek (és ha a szomszédos is nulla, akkor annak a szomszédai is, és így tovább). A játék addig tart, amíg valamelyik játékos aknára nem lép, vagy fel nem fedték az összes nem akna mezőt (ekkor döntetlen lesz a játék).
A program biztosítson lehetőséget új játék kezdésére a pályaméret megadásával (6×6, 10×10, 16×16), valamint játék mentésére és betöltésére relációs adatbázisba. Ismerje fel, ha vége a játéknak, és jelenítse meg, melyik játékos győzött (ha nem döntetlen).

16. Harcos robotmalacok csatája

Készítsünk programot, amellyel a következő két személyes játékot játszhatjuk.
Adott egy $n \times n$ elemből álló játékpálya, ahol két harcos robotmalac helyezkedik el, kezdetben a két ellentétes oldalon, a középvonaltól eggyel jobbra, és mindkettő előre néz. A malacok lézerágyúval és egy támadóököllel vannak felszerelve.
A játék körökből áll, minden körben a játékosok egy programot futtathatnak a malacokon, amely öt utasításból állhat (csak ennyi fér a malac memóriájába). A két játékos először leírja a programot (úgy, hogy azt a másik játékos ne lássa), majd egyszerre futtatják le őket, azaz a robotok szimultán teszik meg a programjuk által előírt 5 lépést.
A program az alábbi utasításokat tartalmazhatja:

- előre, hátra, balra, jobbra: egy mezőnyi lépés a megadott irányba, közben a robot iránya nem változik.
- fordulás balra, jobbra: a robot nem vált mezőt, de a megadott irányba fordul.
- tűz: támadás előre a lézerágyúval.
- ütés: támadás a támadóököllel.

Amennyiben a robot olyan mezőre akar lépni, ahol a másik robot helyezkedik, akkor nem léphet (átugorja az utasítást), amennyiben a két robot ugyanoda akar lépni, akkor egyikük se lép (mindkettő átugorja az utasítást).
A két malac a lézerrel és az ököllel támadhatja egymást. A lézer előre lő, és függetlenül a távolságtól eltalálja a másikat. Az ütés pedig valamennyi

szomszédos mezőn (azaz egy 3 × 3-as négyzetben) eltalálja a másikat. A csatának akkor van vége, ha egy robotot háromszor eltaláltak.
A program biztosítson lehetőséget új játék kezdésére a pályaméret megadásával (4×4, 6×6, 8×8), valamint játék mentésére és betöltésére relációs adatbázisba. Ismerje fel, ha vége a játéknak, és jelenítse meg, melyik játékos győzött. Játék közben folyamatosan jelenítse meg a játékosok aktuális sérülésszámait.

17. Reversi

Készítsünk programot, amellyel az alábbi Reversi játékot játszhatjuk.
A játékot két játékos játssza $n \times n$-es négyzetrácsos táblán fekete és fehér korongokkal. Kezdéskor a tábla közepén X alakban két-két korong van elhelyezve mindkét színből. A játékosok felváltva tesznek le újabb korongokat. A játék lényege, hogy a lépés befejezéseként az ellenfél ollóba fogott, azaz két oldalról (vízszintesen, függőlegesen vagy átlósan) közrezárt bábuit (egy lépésben akár több irányban is) a saját színünkre cseréljük.
Mindkét játékosnak, minden lépésben ütnie kell. Ha egy állásban nincs olyan lépés, amivel a játékos ollóba tudna fogni legalább egy ellenséges korongot, passzolnia kell és újra ellenfele lép. A játékosok célja, hogy a játék végére minél több saját színű korongjuk legyen a táblán.
A játék akkor ér véget, ha a tábla megtelik, vagy ha mindkét játékos passzol. A játék győztese az a játékos, akinek a játék végén több korongja van a táblán. A játék döntetlen, ha mindkét játékosnak ugyanannyi korongja van a játék végén.
A program biztosítson lehetőséget új játék kezdésére a táblaméret megadásával (10×10, 20×20, 30×30), játék szüneteltetésére, valamint játék mentésére és betöltésére relációs adatbázisba. Ismerje fel, ha vége a játéknak, és jelenítse meg, melyik játékos győzött. A program folyamatosan jelezze külön-külön a két játékos gondolkodási idejét (azon idők összessége, ami az előző játékos lépésétől a saját lépéséig tart, ezt is mentsük el és töltsük be).

18. Vadászat

Készítsünk programot, amellyel a következő két személyes játékot lehet játszani. Adott egy $n \times n$ mezőből álló tábla, ahol egy menekülő és egy támadó játékos helyezkedik el.
Kezdetben a menekülő játékos figurája középen van, míg a támadó figurái a négy sarokban helyezkednek el. A játékosok felváltva lépnek. A figurák vízszintesen, illetve függőlegesen mozoghatnak 1-1 mezőt, de egymásra nem léphetnek. A támadó játékos célja, hogy adott lépésszámon ($4n$) belül bekerítse a menekülő figurát, azaz a menekülő ne tudjon lépni.
A program biztosítson lehetőséget új játék kezdésére a táblaméret (3×3, 5×5, 7×7) és így a lépésszám (12, 20, 28) megadásával, folyamatosan jelenítse meg a

lépések számát, és ismerje fel, ha vége a játéknak. Ekkor jelenítse meg, melyik játékos győzött, majd kezdjen automatikusan új játékot. Ezen felül legyen lehetőség a játék elmentésére, valamint betöltésére relációs adatbázisba.

19. Potyogós amőba

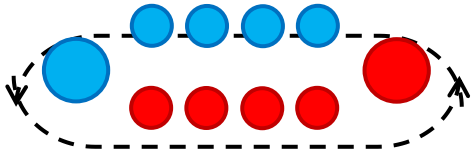
Készítsünk programot, amellyel a potyogós amőba játékot lehet játszani, vagyis az amőba azon változatát, ahol a jeleket felülről lefelé lehet beejteni a játékmezőre. A játékmező itt is $n \times n$ -es tábla, és ugyanúgy X, illetve O jeleket potyogtathatunk a mezőre. A játék akkor ér véget, ha betelik a tábla (döntetlen), vagy valamelyik játékos kirak 4 egymás melletti jelet (vízszintesen, vagy átlósan).
A program minden lépésnél jelezze, hogy melyik játékos következik, és a tábla egy üres mezőjére kattintva helyezhessük el a megfelelő jelet. Természetesen csak a szabályos lépéseket engedje meg a program.
A program biztosítson lehetőséget új játék kezdésére a táblaméret megadásával (10×10, 20×20, 30×30), játék szüneteltetésére, valamint játék mentésére és betöltésére relációs adatbázisba. Ismerje fel, ha vége a játéknak, és jelenítse meg, melyik játékos győzött (a táblán jelölje meg a győztes 4 karaktert). A program folyamatosan jelezze külön-külön a két játékos gondolkodási idejét (azon idők összessége, ami az előző játékos lépésétől a saját lépéséig tart, ezt is mentsük el és töltsük be).

20. Négyzetek

Készítsünk programot, amellyel az alábbi két személyes játékot játszhatjuk.
Adott egy $n \times n$ pontból álló játéktábla, amelyen a játékosok két szomszédos pont között vonalakat húzhatnak (vízszintesen, vagy függőlegesen). A játék célja, hogy a játékosok a húzogatással négyzetet tudjanak rajzolni (azaz ők húzzák be a negyedik vonalat, független attól, hogy az eddigieket melyikük húzta). Ilyen módon egyszerre akár két négyzet is elkészülhet. A játék addig tart, amíg lehet húzni vonalat a táblán.
A játékosok felváltva húzhatnak egy-egy vonalat, de ha egy játékos berajzolt egy négyzetet, akkor ismét ő következik.
A program biztosítson lehetőséget új játék kezdésére a pályaméret megadásával (3×3, 5×5, 9×9), játék mentésére és betöltésére relációs adatbázisba. Ismerje fel, ha vége a játéknak, és jelenítse meg, melyik játékos győzött (ha nem döntetlen). Játék közben a vonalakat, illetve a négyzeteket színezze a játékos színére.

21. Awari

Készítsünk programot, amellyel a következő két személyes játékot lehet játszani. Két játékos egymással szemben helyezkedik el, közöttük pedig (paraméterként megadható) páros számú tálka és két gyűjtőtál az alábbi lerendezésben.



(8 tálkával)

Mindkét játékos a hozzá közelebbi tálkákat és a tőle jobb kézre eső gyűjtőtálat mondhatja sajátjának. (Így az ellenfél gyűjtőtálja baloldalra esik.) Kezdetben mindegyik tálkában 6-6 kavics van, a gyűjtőtálak pedig üresek.

A játékban a soron következő játékos kiválasztja egyik saját tálkáját (ez nem lehet a gyűjtőtál), hogy azt kiürítse úgy, hogy tartalmát az óramutató járásával ellentétes irányban egyesével beledobálja, majd ismét a saját tálkáiba, de azt a tálkát kihagyva, amelyiknek a kiürítését végezzük, amíg el nem fogynak a kavicsok. Ha az egyik játékos rákattint valamelyik tálkájára, akkor a tálkában lévő kavicsok áthelyezése automatikusan történjen meg. Ha az utolsó kavics a játékos saját üres tálkáinak egyikébe kerül, akkor ezt a kavicsot, valamint a szemközti tálka tartalmát a saját gyűjtőládába teszi. Viszont, ha az utolsó kavics a játékos saját gyűjtőtálkájába esik, akkor újra ő következik, de ezt csak egyszer teheti meg, hogy ellenfele is szóhoz juthasson.

A játéknak akkor van vége, ha az egyik térfél kiürült, azaz az egyik játékos tálkái mind kiürülnek. Ekkor az a játékos nyeri a játékot, akinek a gyűjtőtáljában több kavics van.

A program biztosítson lehetőséget új játék kezdésére a tálkák számának megadásával $(4, 8, 12)$, játék mentésére és betöltésére relációs adatbázisba. Ismerje fel, ha vége a játéknak, és jelenítse meg, melyik játékos győzött.

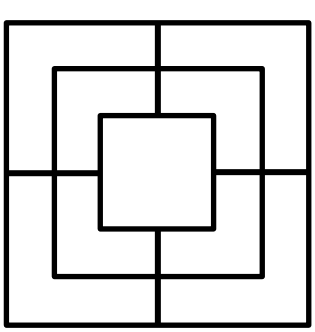
22. Malom

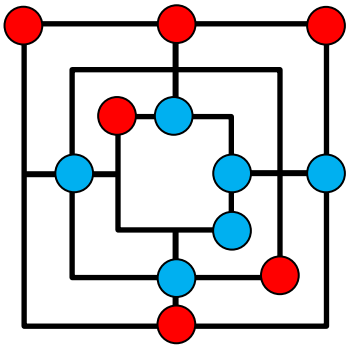
Készítsünk programot, amellyel a következő két személyes játékot játszhatjuk.

A malom játékban két játékos egy 24 mezőből álló speciális játéktáblán játszhatja, ahol a mezők három egymásba helyezett négyzetben helyezkednek (mindegyikben 8, a sarkoknál és a felezőpontoknál), melyek a felezőpontok mentén össze vannak kötve.

Kezdetben a tábla üres, és felváltva helyezhetik el rajta bábuikat az üres mezőkre. Az elhelyezés után a játékosok felváltva mozgathatják bábuikat a szomszédos (összekötött) mezőkre. Amennyiben egy játékos nem tud mozgatni, akkor passzolhat a másik játékosnak. Ha valakinek sikerül 3 egymás melletti mezőt elfoglalnia (azaz malmot alakít ki, rakodás, vagy mozgatás közben), akkor leveheti az ellenfél egy általa megjelölt bábuját (kivéve, ha az egy malom része).

Az a játékos veszít, akinek először megy 3 alá a bábuk száma a mozgatási fázis alatt.





A program biztosítson lehetőséget új játék kezdésére, mentésére és betöltésére relációs adatbázisba. Ismerje fel, ha vége a játéknak, és jelenítse meg, melyik játékos győzött.

23. Kitolás

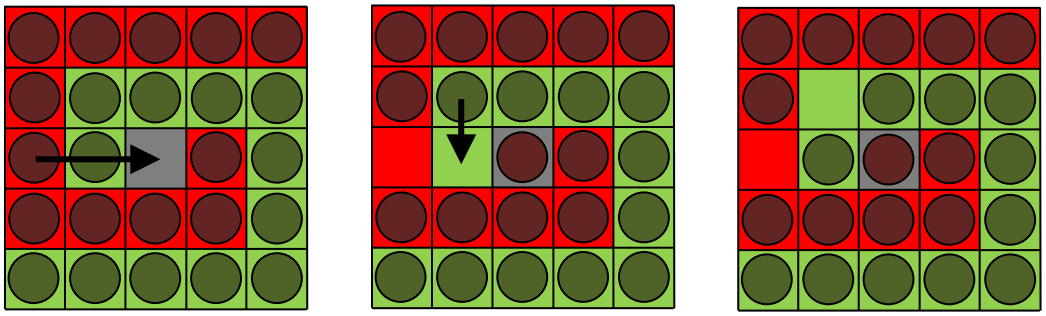
Készítsünk programot, amellyel a következő két személyes játékot lehet játszani. Adott egy $n \times n$ mezőből álló tábla, amelyen kezdetben a játékosoknak n fehér, illetve n fekete kavics áll rendelkezésre, amelyek elhelyezkedése véletlenszerű.
A játékosok kiválaszthat egy saját kavicsot, amelyet függőlegesen, vagy vízszintesen eltolhat. Eltoláskor azonban nem csak az adott kavics, hanem a vele az eltolás irányában szomszédos kavicsok is eltolódnak, a szélső mezőn lévők pedig lekerülnek a játéktábláról. A játék célja, hogy adott körszámon belül ($5n$) az ellenfél minél több kavicsát letoljuk a pályáról (azaz nekünk maradjon több kavicsunk a végére). Ha mindkét játékosnak ugyanannyi marad, akkor a játék döntetlen.
A program biztosítson lehetőséget új játék kezdésére a táblaméret (3×3, 4×4, 6×6) és így a lépésszám (15, 20, 30) megadásával, és ismerje fel, ha vége a játéknak. Ekkor jelenítse meg, melyik játékos győzött (ha nem döntetlen), majd automatikusan kezdjen új játékot. Ezen felül legyen lehetőség a játék elmentésére, valamint betöltésére relációs adatbázisba.

24. Kaméleonok

Készítsünk programot, amellyel a következő két személyes játékot játszhatjuk.
Adott egy $n \times n$ mezőből álló tábla, amelyen a mezők két színt vehetnek fel spirális alakban (tradicionálisan pirosat, illetve zöldet), továbbá a középső mező szürke. Minden mezőn, kivéve a középsőn egy kaméleon helyezkedik el, amelynek színe megegyezik a mezővel, így minden játékos $(n^2 - 1)/2$ kaméleonnal rendelkezik.
A játékosok felváltva léphetnek. Egy kaméleonnal léphetünk egy szomszédos üres mezőre (vízszintesen, illetve függőlegesen), illetve átugorhatjuk az ellenfél

kaméleonját (vízszintesen, illetve függőlegesen), amennyiben a rákövetkező mező üres. Az átugrott kaméleon lekerül a tábláról. A játék célja, hogy a másik játékos elveszítse az összes kaméleonját.

A játékban a csavar, hogy a kaméleonok alkalmazkodnak a környezetükhöz. Amennyiben egy kaméleon egy másik színű mezőre ugrott, vagy lépett, akkor további 1 kör elteltével átszíneződik a másik színre (tehát a másik játékosé lesz). Ez alól kivétel a középső mező.



A program biztosítson lehetőséget új játék kezdésére a pályaméret megadásával (3×3, 5×5, 7×7), valamint játék mentésére és betöltésére relációs adatbázisba. Ismerje fel, ha vége a játéknak, és jelenítse meg, melyik játékos győzött.