3. beadandó feladat: háromrétegű grafikus felületű alkalmazás

Közös követelmények:

- A programot háromrétegű (modell/nézet/perzisztencia) architektúrában kell felépíteni, amelyben a megjelenítés rétege, és az adatkezelést végző réteg is elkülönül a játéklogikától. A modell nem tartalmazhat semmilyen grafikus felületbeli osztályra történő hivatkozást, sem perzisztenciabeli adatot (pl. fájlnév), csak eseményeket küldhet a grafikus felületnek, illetve hívhatja a perzisztencia betöltés és mentés metódusait. A nézet nem tartalmazhat semmilyen játékbeli adatot, és nem hívhatja perzisztencia betöltés és mentés metódusait. A perzisztencia nem tartalmazhat sem a grafikus felületbeli osztályra, sem a játéklogikára történő hivatkozást, csak a játéklogikának küldhet eseményeket.

- A program játékfelületét dinamikusan kell létrehozni futási időben. A megjelenítéshez lehet vezérlőket használni, elemi grafikát, vagy grafikus képernyőt. Egyes feladatoknál különböző méretű játéktábla létrehozását kell megvalósítani, ekkor ügyelni kell arra, hogy az ablakméret mindig alkalmazkodjon a játéktábla méretéhez.

- Azon feladatoknál, ahol szüneteltetni is lehet, szünet alatt ne menjen a játék, és a játékos se tudjon tevékenységet végezni.

- A dokumentációnak tartalmaznia kell a feladat elemzését, felhasználói eseteinek (WHEN-GIVEN-THEN szerkezetű) leírását (UML felhasználói esetek diagrammal kiegészítve), a program statikus szerkezetének leírását (UML osztálydiagrammal), valamint az esemény-eseménykezelő párosításokat és az eseménykezelő tevékenység rövid leírását.

- A program modelljéhez automatikusan futtatható egység-teszteket kell készíteni. A perzisztencia réteg objektumait ilyenkor utánzatokkal (mock objektumokkal) helyettesítsük.

Feladatok:

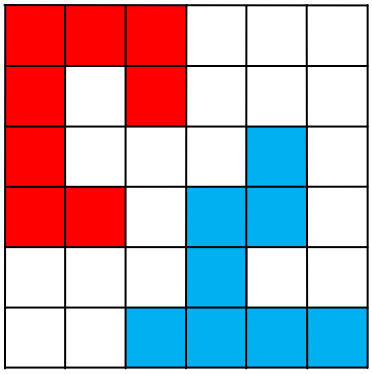
1. **Go**

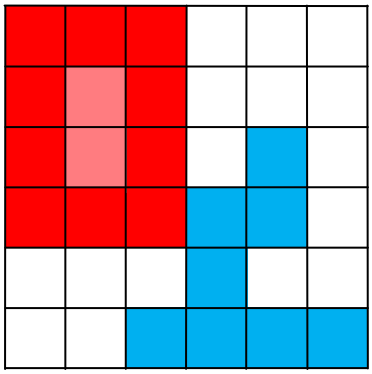
   Készítsünk programot, amellyel a Go játék egyszerűsített változatát játszhatjuk. Adott egy $n \times n$ pontból álló tábla, ahol a pontokra a játékosok felváltva köveket helyezhetnek (tradicionálisan fehér, illetve fekete színűt). A lerakott köveknek élete van, ami a négy szomszéd mezőből a szabad mezők száma. Egy csoport olyan kövek halmaza, amelyek szomszédosan összeérnek. A játékos akkor keríti be a másik játékos egy csoportját, ha azok élete elfogy. Ekkor a kövek fogságba

kerülnek, és levehetőek a tábláról (ekkor a terület újra üres lesz, és lehet oda követ helyezni).
A játék meghatározott körszámig (n) tart, és célja minél több fogoly ejtése. Amennyiben ez egyenlő, a játék döntetlen.
A program biztosítson lehetőséget új játék kezdésére a táblaméret megadásával (5×5, 9×9, 19×19), az aktuális játék mentésére és egy korábban elmentett játék betöltésére. Ismerje fel, ha vége a játéknak, és jelenítse meg, melyik játékos győzött.

2. 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). Csak szabad területre lehet elhelyezni téglalapot. A program külön jelölje meg a lehelyezett téglalapokat, illetve az elfoglalt területeket, és játék közben folyamatosan jelenítse meg az elfoglalt terület méretét játékosonként. A játéknak akkor van vége, ha már nincs 2×1-es méretű szabad hely.



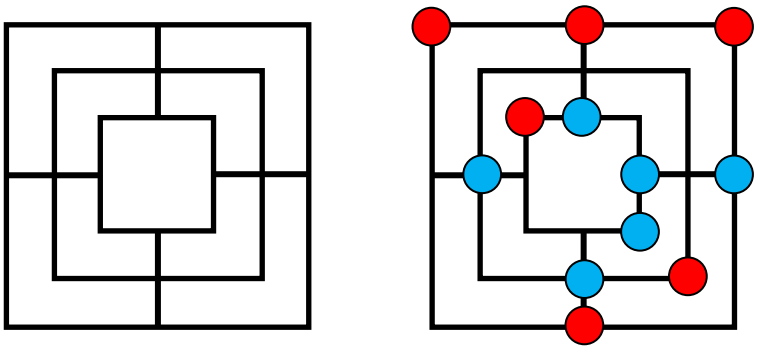


A program biztosítson lehetőséget új játék kezdésére a pályaméret megadásával (6×6, 8×8, 10×10), valamint az aktuális játék mentésére és egy korábban elmentett játék betöltésére. Ismerje fel, ha vége a játéknak, és jelenítse meg, melyik játékos győzött.

3. Malom

Készítsünk programot, amellyel a következő két személyes játékot játszhatja.
A malom játékban két játékos egy 24 mezőből álló speciális játéktáblán játszik 9-9 bábuval, a mezők három egymásba helyezett négyzetben helyezkednek el (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, valamint az aktuális játék mentésére és egy korábban elmentett játék betöltésére. Ismerje fel, ha vége a játéknak, és jelenítse meg, melyik játékos győzött.

4. 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), valamint az aktuális játék mentésére és egy korábban elmentett játék betöltésére. Ismerje fel, ha vége a játéknak, és jelenítse meg, melyik játékos győzött, illetve azt is, ha döntetlen lett a vége. Játék közben a vonalakat, illetve a négyzeteket színezze a játékos színére.

5. 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:

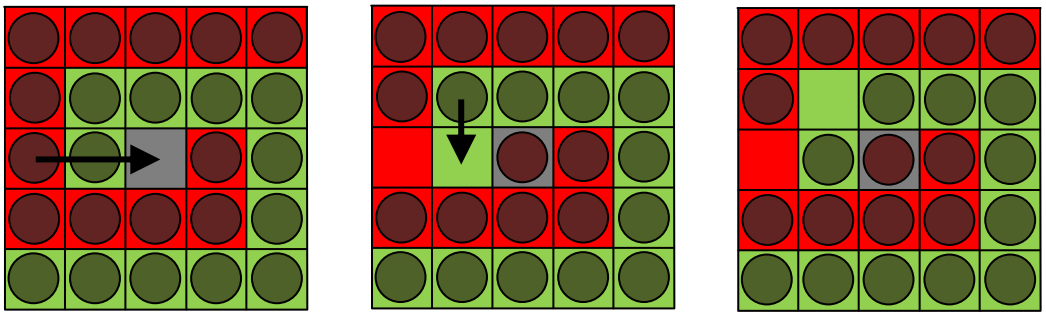
- előre, hátra, balra, jobbra: egy mezőnyi lépés a megadott irányba, közben a robot iránya nem változik.
- fordulás balra, jobbra: a robot nem vált mezőt, de a megadott irányba fordul.
- tűz: támadás előre a lézerágyúval.
- ütés: támadás a támadóököllel.

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 az aktuális játék mentésére és egy korábban elmentett játék betöltésére. Ismerje fel, ha vége a játéknak, és jelenítse meg, melyik játékos győzött. Játék közben folyamatosan jelenítse meg a játékosok aktuális sérülésszámait.

6. 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. Kezdetben minden mezőn, kivéve a középsőn egy kaméleon helyezkedik el (lásd a lenti ábra első tábláját), amelynek színe megegyezik a mezővel, így minden játékos $(n^2 - 1)/2$ kaméleonnal rendelkezik.
A játékosok felváltva léphetnek. Egy saját kaméleonnal léphetnek 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.

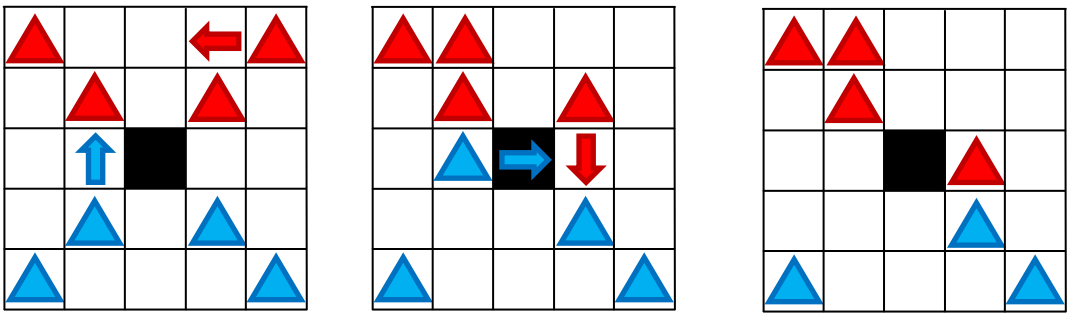


A program biztosítson lehetőséget új játék kezdésére a pályaméret megadásával (3×3, 5×5, 7×7), valamint az aktuális játék mentésére és egy korábban elmentett játék betöltésére. Ismerje fel, ha vége a játéknak, és jelenítse meg, melyik játékos győzött.

7. 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 táblaméret megadásával (5×5, 7×7, 9×9), valamint az aktuális játék mentésére és egy korábban elmentett játék betöltésére. Ismerje fel, ha vége a játéknak. Ekkor jelenítse meg, melyik játékos győzött, majd automatikusan kezdjen új játékot.

8. Kiszúrós amőba

Készítsünk programot, amellyel a közismert amőba játék következő változatát játszhatjuk.
Adott egy $n \times n$ -es tábla, amelyen a két játékos felváltva *X*, illetve *O* jeleket helyez el. Csak olyan mezőre tehetünk jelet, amely még üres. A játék akkor ér véget, ha betelik a tábla (döntetlen), vagy valamelyik játékos kirak 5 egymással szomszédos jelet vízszintesen, függőlegesen vagy átlósan. A program minden lépésnél jelezze, hogy melyik játékos következik, és a tábla egy üres mezőjét kijelölve helyezhessük el a megfelelő jelet.
A kiszúrás a játékban az, hogy ha egy játékos eléri a 3 egymással szomszédos jelet, akkor a program automatikusan törli egy jelét egy véletlenszerűen kiválasztott pozícióról (nem biztos, hogy a hármasból), ha 4 egymással szomszédos jelet ér el, akkor pedig kettőt.
A program biztosítson lehetőséget új játék kezdésére a táblaméret megadásával (6×6, 10×10, 14×14), valamint az aktuális játék mentésére és egy korábban elmentett játék betöltésére. Ismerje fel, ha vége a játéknak, és jelenítse meg, melyik játékos győzött, illetve azt is, ha döntetlen lett a vége, majd automatikusan kezdjen új játékot.

9. Potyogós amőba

Készítsünk programot, amellyel a következő két személyes játékot lehet játszani.
Adott egy $n \times m$ mezőből álló tábla (n az oszlopok, m a sorok száma), amelyre a játékosok *X*, illetve *O* jeleket potyogtatnak (azaz egy adott oszlopban a karakter mindig „leesik” a legalsó üres sorba, függetlenül attól, melyik sorban helyeztük le).
A játékosok felváltva lépnek, és egy oszlopban csak akkor helyezhetnek el új jelet, ha az még nem telt meg. A játékot az nyeri, aki előbb elhelyez vízszintesen, vagy átlósan négy szomszédos jelet. A játék döntetlennel ér véget, ha betelik a tábla.
A program biztosítson lehetőséget új játék kezdésére a táblaméret megadásával (8×5, 10×6, 12×7), valamint az aktuális játék mentésére és egy korábban elmentett játék betöltésére. Ismerje fel, ha vége a játéknak, és jelenítse meg, melyik játékos győzött, illetve azt is, ha döntetlen lett a vége, majd automatikusan kezdjen új játékot.

10. Kitolás

Készítsünk programot, amellyel a következő két személyes játékot lehet játszani.
Adott egy $n \times n$ mezőből álló tábla, amelyen kezdetben a játékosoknak $\boldsymbol{n}$ fehér, illetve $\boldsymbol{n}$ 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

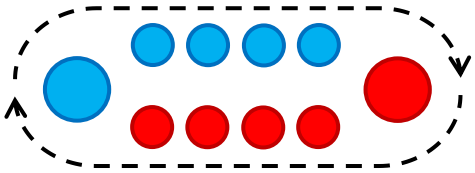
pedig lekerülnek a játéktábláról. A játék célja, hogy adott körszámon belül ($\mathbf{5n}$) az ellenfél minél több kavicsát letoljuk a pályáról (azaz nekünk maradjon több kavicsunk a végére). Ha mindkét játékosnak ugyanannyi marad, akkor a játék döntetlen.
A program biztosítson lehetőséget új játék kezdésére a táblaméret (3×3, 4×4, 6×6) és így a lépésszám (15, 20, 30) megadásával, valamint az aktuális játék mentésére és egy korábban elmentett játék betöltésére. Ismerje fel, ha vége a játéknak, és jelenítse meg, melyik játékos győzött, illetve azt is, ha döntetlen lett a vége, majd automatikusan kezdjen új játékot.

11. 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, valamint az aktuális játék mentésére és egy korábban elmentett játék betöltésére. Ismerje fel, ha vége a játéknak, és jelenítse meg, melyik játékos győzött, illetve azt is, ha döntetlen lett a vége, majd automatikusan kezdjen új játékot.

12. 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 elrendezésben.


(8 tálkával)

A játékosok az északi, illetve a déli oldalán ülnek a játéknak. Mindkét játékos a hozzá közelebbi tálkákat és a tőle jobb kézre eső gyűjtőtálat mondhatja sajátjának. (Így az ellenfél gyűjtőtálja bal oldalra 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 annak tartalmát az óramutató járásával

azonos irányban haladva egyesével beledobálja a többi tálkába, amíg el nem fogynak a kavicsok. Ha körbe érne, folytatja tovább, de azt a tálkát kihagyja, amelyiknek a kiürítését végzi. Ez a kiürítés automatikusan történjen meg, amikor az egyik játékos rákattint valamelyik tálkájára. Ha az utolsó kavics a játékos saját üres tálkáinak egyikébe kerül, akkor ezt a kavicsot, valamint a szemközti tálka tartalmát a saját gyűjtőládájába 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 $(4, 8, 12)$, valamint az aktuális játék mentésére és egy korábban elmentett játék betöltésére. Ismerje fel, ha vége a játéknak. Ekkor jelenítse meg, melyik játékos győzött, majd kezdjen automatikusan új játékot.