Közös követelmények:

- A programot kétrétegű (modell/nézet) architektúrában kell felépíteni, amelyben a megjelenítés rétege elkülönül a játéklogikától. A modell nem tartalmazhat semmilyen grafikus felületbeli osztályra történő hivatkozást, csak eseményeket küldhet a grafikus felületnek. A nézet nem tartalmazhat semmilyen játékbeli adatot.

- 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.

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. **Harcos robotmalacok csatája**

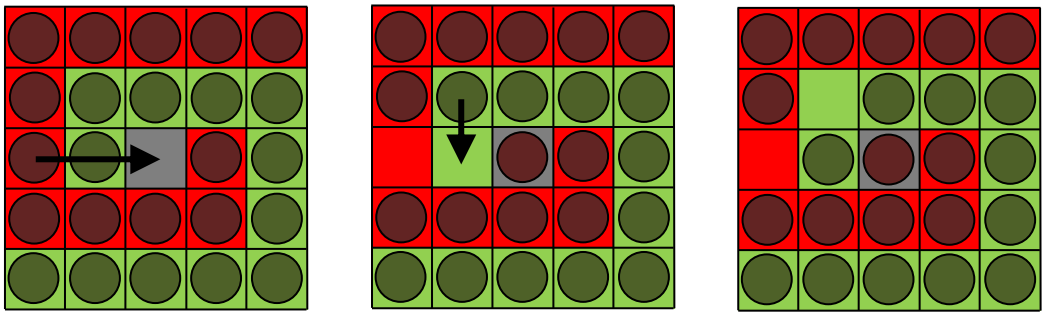
   Készítsünk programot, amellyel a következő két személyes játékot játszhatjuk.
   Adott egy $n \times n$ elemből álló játékpálya, ahol két harcos robotmalac helyezkedik el, kezdetben a két ellentétes oldalon, a középvonaltól eggyel jobbra, és mindkettő előre néz. A malacok lézerágyúval és egy támadóököllel 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 ($4 \times 4,\ 6 \times 6,\ 8 \times 8$), 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.

3. Kaméleonok

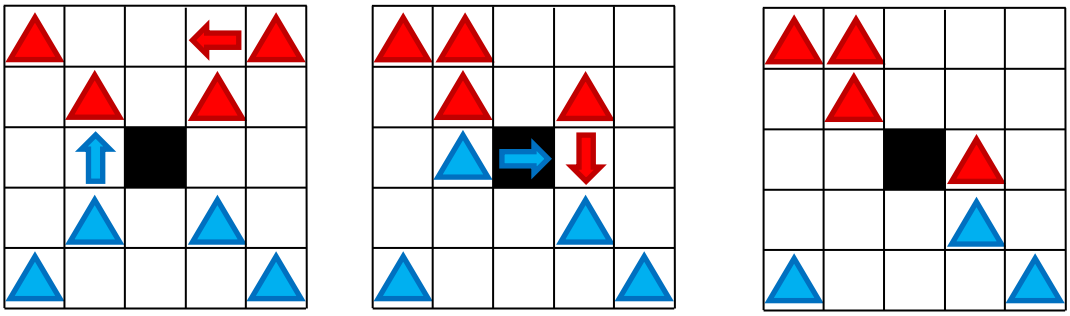
Készítsünk programot, amellyel a következő két személyes játékot játszhatjuk.
Adott egy $n \times n$ mezőből álló tábla, amelyen a mezők két színt vehetnek fel spirális alakban (tradicionálisan pirosat, illetve zöldet), továbbá a középső mező szürke. Kezdetben minden mezőn, kivéve a középsőn egy kaméleon 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 ($3 \times 3,\ 5 \times 5,\ 7 \times 7$), valamint az aktuális játék mentésére és egy korábban elmentett játék betöltésére. Ismerje fel, ha vége a játéknak, és jelenítse meg, melyik játékos győzött.

4. 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.

5. 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.

6. 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.

7. 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 ($\boldsymbol{5n}$) 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.

8. 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.

9. Maci Laci

Készítsünk programot, amellyel a következő játékot játszhatjuk.
Adott egy $n \times n$ mezőből álló erdő, amelyben Maci Lacival kell piknikkosarakra vadásznunk, amelyek a játékpályán helyezkednek el. A játék célja, hogy Maci Laci irányításával a piknikkosarakat minél gyorsabban begyűjtsük.
A játékpályán a piknikkosarak mellett akadályok (pl. fa) is elhelyezkedhetnek, amelyekre nem léphetünk. A pályán emellett 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 pálya szélébe) nem ütköznek, ekkor megfordulnak, és visszafelé haladnak (tehát folyamatosan egy vonalban járőröznek). Egy vadőr a 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) tárolhatjuk fájlban, vagy létrehozhatjuk véletlenszerűen (előre rögzített paraméterek mellett). A programot legalább 3 különböző méretű pályával lehessen használni.
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.

10. 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) folyamatosan előre haladva egészen addig, amíg falba nem ütközik. Ekkor véletlenszerűen választ egy új irányt, és arra halad tovább.
A játékos a robot terelését úgy hajthatja végre, hogy egy mezőt kiválasztva falat emelhet rá. A felhúzott falak sajnos 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.

11. 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 játék célja az, hogy egy játékost minél előbb kivezessünk a labirintusból a bal alsó sarokból indulva.
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 fájlban. A programot legalább 3 különböző méretű pályával lehessen használni.
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.

12. Snake

Készítsük programot, amellyel a klasszikus kígyó játékot játszhatjuk.
Adott egy $n \times n$ mezőből 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) tárolhatjuk fájlban, vagy létrehozhatjuk véletlenszerűen (előre rögzített paraméterek mellett). A programot legalább 3 különböző méretű pályával lehessen használni.
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.