Alkalmazott modul III
3. feladatcsoport

Közös követelmények:

- A program játékfelületét dinamikusan kell létrehozni futási időben. Egyes feladatoknál különböző méretű játékmezők létrehozását kell megvalósítani, ekkor az ablak méretének alkalmazkodnia kell a táblamérethez.

- A programnak vagy támogatnia kell új játék kezdését, mentését, illetve betöltését, amelyre adott szöveges fájlformátumot használ, vagy számolnia a játékidőt (másodpercben), ekkor lehessen szüneteltetni a játékot (szünet alatt semmilyen tevékenység nem végezhető a játékban).

Feladatok:

1. **Kiszúrós amőba**

   Készítsünk programot, amellyel a közismert amőba játék alábbi 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 átlós irányban. 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, és ismerje fel, ha a játék véget ért. Ilyenkor jelezze ki a győzelmet jelentő 5 jelet (ha több lehetőség van, akkor egy tetszőlegeset), illetve írja ki, hogy döntetlen, ha betelt a tábla.
   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.
   Legyen lehetőség új játék kezdésére a táblaméret megadásával (10×10, 20×20, 30×30), valamint játék betöltésére és mentésére.

2. **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, mindig jelezze a következő játékost, és

ismerje fel, ha a játék véget ért. Ilyenkor jelezze ki a győzelmet jelentő 4 jelet (ha több lehetőség van, akkor egy tetszőlegeset), illetve írja ki, hogy döntetlen, ha betelt a tábla (játék vége után ne lehessen tovább játszani).
Legyen lehetőség új játék kezdésére a táblaméret megadásával (10×10, 20×20, 30×30), valamint játék betöltésére és mentésére.

3. Aknakereső

Készítsünk programot, amellyel az aknakereső játékot játszhatjuk.
A játékosok feladata egy $n \times n$-es játéktábla felderítése az aknák kihagyása mellett úgy, hogy egymás után felfedi a mezőket.
Az aknákat véletlenszerűen helyezzük el a játéktáblán. Amennyiben a felfedezett mező nem akna, akkor megjelenik rajta a szomszédos 8 mezőben található aknák száma. Amennyiben ez nulla, akkor nem csak a játékmező kerül felfedezésre, hanem az a teljes terület, amelyet nem nulla értékű mezők határolnak (ekkor ezek a mezők is felfedésre kerülnek).
A játék addig tart, amíg aknára nem lép, vagy nem sikerül az összes nem akna mezőt felfednie. A program ismerje fel, ha véget ért a játék, és közölje, hogy sikerült-e megoldani a feladatot (utána ne lehessen tovább játszani).
Legyen lehetőség új játék kezdésére, játék mentésére és betöltésére a táblaméret (10×10, 20×20, 30×30), valamint az aknák számának (10, 30, 50) megadásával.

4. Ugrálós

Készítsünk programot, amellyel a következő játékot lehet játszani. A játékos egy n×n-es tábla mezőire kattintva ugrándozik a táblán. A táblán szereplő mezők között véletlenszerűen elhelyezünk valamennyi gödröt, de a játékos nem tudja, hogy ezek a gödrök hol helyezkednek el. A játékos minden mezőre csak egyszer tud ugrani. (Bejárt mezőre kattintva nem történik semmi.) A játék során jelezzük, mely mezőket járta már be a játékos. A játék célja, hogy a játékos minél több ugrást tegyen meg anélkül, hogy „beleesne” valamelyik gödörbe. Egy játékmenetnek akkor van vége, ha a játékos „gödörbe esett”. A játékmenet végén tájékoztassuk a játékost arról, hány mezőt járt be (utána ne lehessen tovább játszani).
A program játék közben is jelezze a megtett ugrások számát. A program kényelmes fejlesztése és tesztelése érdekében helyezzen el egy Mutat/Elrejt gombot, amellyel játék közben „meglesheti” a gödrök helyét. Amikor „látjuk” a gödröket, akkor ne lehessen a mezőkön tovább ugrálni.
Legyen lehetőség új játék kezdésére a táblaméret (10×10, 20×20, 30×30), valamint a gödrök számának (10, 30, 50) megadásával, valamint játék mentésére és betöltésére.

5. Hanoi tornyai

Készítsünk programot, amellyel a közismert Hanoi tornyai játékot lehet játszani. Adott három rúd, és az első rúdon n korong, amelyek alulról felfelé egyre kisebb méretűek.
A játék célja, hogy az összes korongot helyezzük át az első rúdról a másodikra úgy, hogy minden lépésben csak egy korongot mozgathatunk, és egy korongot mindig csak egy nála nagyobb korongra vagy üres rúdra helyezhetünk. A programban a korongok áthelyezése történjen úgy, hogy először rákattintunk arra a rúdra, amelyikről a legfelső korongot mozgatni akarjuk, aztán pedig arra a rúdra, amelyikre át akarjuk tenni a korongot (és csak a szabályos lépéseket engedje). Az első kattintás után jelöljük meg valahogyan a kiválasztott rudat, illetve korongot.
Legyen lehetőség új játék kezdésére a korongok számának megadásával (3, 5, 8), valamint játék szüneteltetésére. A program folyamatosan jelezze az eltelt időt (másodperceket), valamint a megtett mozgatások számát.

6. Kancsók

Készítsünk programot, amellyel a következő játékot lehet játszani. Adott három kancsó, amelyek mérete $x \leq y \leq z$. Kezdetben a z méretű kancsó tele van, a másik kettő üres. Adott továbbá egy $s \leq z$ mennyiség. A játék célja, hogy a kancsók között a vizet ide-oda töltögetve kimérjünk pontosan s mennyiségű vizet. Amikor egy kancsóból vizet töltünk egy másikba, akkor a forráskancsót teljesen ki kell üríteni, vagy a célkancsót teljesen meg kell tölteni. Víz nem kerülhet a kancsókon kívülre. A programban a kancsók méretét, és a bennük lévő víz mennyiségét ábrázoljuk vizuálisan és írjuk is ki. A töltögetés úgy történjen, hogy először rákattintunk a forráskancsóra, majd a célkancsóra. Az első kattintás után a forráskancsót jelöljük meg valahogyan. A program ismerje fel, ha vége a játéknak, vagyis ha az egyik kancsóban pontosan s mennyiségű víz jelenik meg, és ekkor kérje be a játékos nevét. Legyen lehetőség új játék kezdésére az x, y, z és s értékek megadásával (legalább 3 kombináció közül lehessen választani, pl. 3, 4, 5, 2), valamint játék szüneteltetésére. A program folyamatosan jelezze az eltelt időt (másodperceket), valamint a megtett mozgatások számát.

7. Képes reflex teszt

Készítsünk programot, amellyel a következő játékot lehet játszani.
A játék egy n×n-es felületű táblán játszható. A játék elkészítéséhez különféle képekre lesz szüksége. Ezek között háromban illetve ötben van valami közös (például három kép egy-egy repülőt ábrázol, ötön pedig különféle felhők vannak). A kilencedik kép legyen egyedi (például egy maci integet egy repülőből). A játéktábla minden egyes pozícióján másodpercenként véletlenszerűen jelennek meg képek. A képekre kattintva pontokat gyűjthetünk vagy veszíthetünk az alábbiak szerint:

- Ha olyan képre kattintunk, amelyiken például csak felhő van, akkor nem történik semmi.

- Ha olyan képre kattintunk, amelyiken repülő van, akkor a kép lecserélődik egy zuhanó repülőre (ez már a tizedik kép), kapunk egy pontot, majd egy másodperc eltelte után az első kilenc képek véletlenszerű megjelenése folytatódik.
- Ha olyan képre kattintunk, amelyiken repülő van, és egy maci ül benne, akkor megjelenik az „angyalkás maci” (ez a tizenegyedik kép), egy pontot veszítünk, majd egy másodperc eltelte után az első kilenc képek véletlenszerű megjelenése folytatódik.

Legyen lehetőség új játék kezdésére a sebesség megadásával (legalább három különböző fokozat legyen), valamint játék szüneteltetésére. Egy játék hossza 1 perc. A program folyamatosan jelezze az eltelt időt (másodperceket).

8. Rubik tábla

Készítsünk programot, amellyel egy Rubik táblát tudunk kirakni.
A Rubik tábla lényegében a Rubik-kocka két dimenziós változata. A játékban adott egy $n \times n$-es tábla, amelyen n különböző színű mező lehet, mindegyik színből n darab, kezdetben véletlenszerűen elhelyezve. A játék célja az egyes sorok, illetve oszlopok mozgatásával (ciklikus tologatásával, azaz ami a tábla egyik végén lecsúszik, az ellentétes végén megjelenik) egyszínűvé alakítani vagy a sorokat, vagy az oszlopokat (azaz vízszintesen, vagy függőlegesen csíkokat kialakítani).
Legyen lehetőség új játék kezdésére a táblaméret megadásával (3×3, 6×6, 10×10), valamint játék szüneteltetésére. A program folyamatosan jelezze az eltelt időt (másodperceket), valamint a szerzett pontokat.

9. Rubik óra

Készítsünk programot, amellyel egy Rubik órát tudunk kirakni.
A játékban 9 óra kap helyet, amely 1-12 közötti értéket mutatnak (kezdetben véletlenszerűen beállítva), és 3×3-as formában jelennek a játéktáblán. Az órák között az átlóknál 4 gomb helyezkedik el, amelyet a szomszédos 4 óra állását tudják eggyel növelni (tehát 4 óra van, amit csak egy gomb növel, 4, amit kettő, és 1, amit mind a négy gomb növel). A játék célja az, hogy a gombokkal történő állítással mind a 9 óra 12-t mutasson.
Legyen lehetőség új játék kezdésére, valamint játék szüneteltetésére. A program folyamatosan jelezze az eltelt időt (másodperceket).

10. Mozaik

Készítsük el az alábbi mozaikrajzoló programot.
A mozaik képet egy $n \times n$-es táblán lehet megrajzolni. A mozaik kockáinak színét a kockára kattintgatva lehet megváltoztatni, összesen 5 különböző szín között váltakozhat körkörösen az aktuális kocka. Egy-egy kocka színe minden egyes kattintásnál „körbe-körbe” járva, rendre megváltozik. A megrajzolt képet lehessen

mozgatni úgy, hogy a képet el lehet indítani mind a négy irányba (fel, le, balra jobbra). Ilyenkor a kép az adott irányba csúszik egy-egy oszloppal, vagy sorral, és az így eltűnt mezők a kép ellenkező végén jelennek meg (pl. ha balra tolunk, akkor az első oszlop a jobb oldalon jelenik meg). Ezt a mozgást bármikor leállíthatjuk, vagy másik irányba, de mozgás közben nem lehet változtatni a színeket.
Legyen lehetőség új rajz kezdésére a táblaméret megadásával (5×5, 10×10, 20×20), valamint a mozgás sebességének szabályozására. A program bezáráskor automatikusan mentse el az aktuális képet, és újraindításnál töltse be.

11. Négyzetek

Készítsünk programot, amellyel az alábbi négyzetek játékot játszhatjuk.
A négyzetek játékban a játéktábla egy $n \times n$-es pontokból álló felület, amelyen a játékosok két pont között vonalakat húzhatnak (vízszintesen, vagy függőlegesen) felváltva két olyan pont között, ahol eddig még nem húztak vonalat. A játék célja, hogy a játékosok a huzogatással négyzetet tudjanak rajzolni (azaz ők húzzák be a negyedik vonalat, független attól, hogy az eddigieket melyikük húzta). Ilyen módon egyszerre két négyzet is elkészülhet. Ha egy játékos berajzolt egy négyzetet, akkor ismét ő következik. A játék addig tart, amíg lehet vonalat húzni a táblán, és az nyer, akinek több négyzete lesz a végén.
Legyen lehetőség új játék kezdésére a táblaméret megadásával (10×10, 20×20, 30×30), valamint játék szüneteltetésére. A program folyamatosan jelezze külön-külön a két játékos gondolkodási idejét (azon idők összessége, ami az előző játékos lépésétől a saját lépéséig tart, ezt nem kell elmenteni).

12. 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.
Legyen lehetőség új játék kezdésére a táblaméret megadásával (10×10, 20×20, 30×30), valamint játék szüneteltetésére. A program folyamatosan jelezze külön-

külön a két játékos gondolkodási idejét (azon idők összessége, ami az előző játékos lépésétől a saját lépéséig tart).