Eseményvezérelt Alkalmazások

1. Beadandó feladat Dokumentáció

Készítette:

Péter Bálint

Neptun kód: gs2xc6

E-mail: gs2xc6@inf.elte.hu

Feladat:

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.

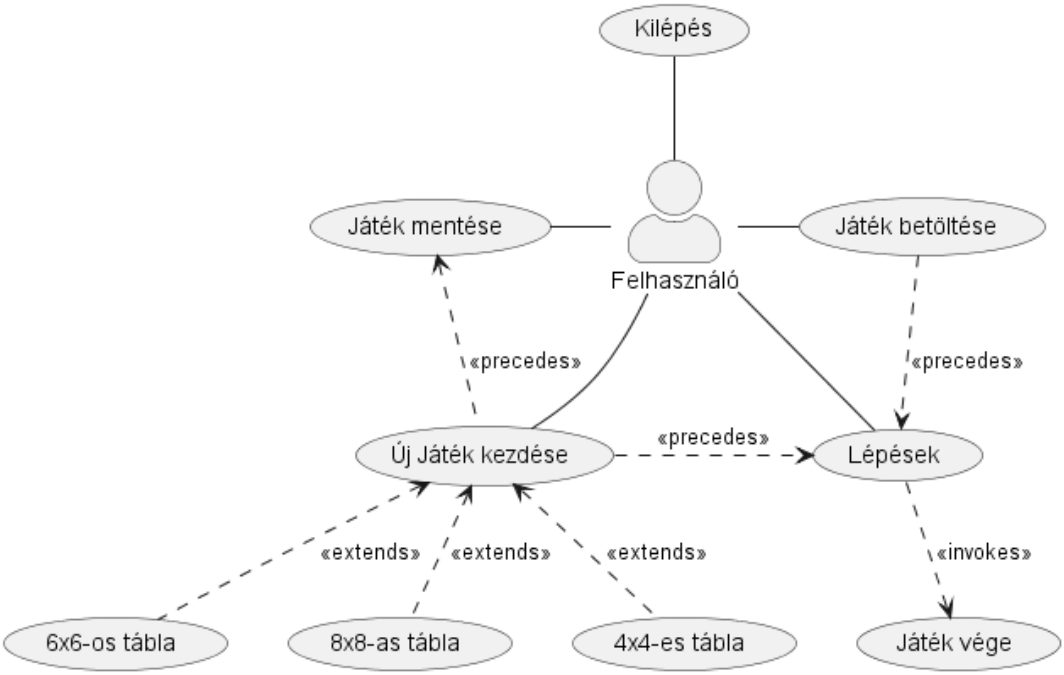
• ütés: támadás a támadóököllel.

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. Ismerje fel, ha vége a játéknak, és jelenítse meg, melyik játékos győzött. Játék közben folyamatosan jelenítse meg a játékosok aktuális sérülésszámait.

Elemzés:

- A feladatot egyablakos asztali alkalmazásként Windows Forms grafikus felülettel valósítjuk meg.
- Indításkor megjelenítjük a főmenüt, 4 opcióval: új játék táblaméretek szerint (4x4, 6x6, 8x8) és játék betöltése.
- Mind a négyen keresztül elkezdődik egy játék, alapértelmezett vagy betöltött helyzetből. A játéktábla középen jelenik meg nxn-es képekből álló rácsként (játékos karakter képe van egy mezőn vagy üres), alsó sarkokban a játékosok életereje. A rendelkezésre álló akciók közüli választást a játéktábla melletti gombok teszik lehetővé, amik jelzik az eddig beadott utasításokat. A tábla alatt szerepel a jelenlegi játékos száma, valamint az utasítások törlésére és a kör befejezésére szolgáló gombok.
- Játék betöltésére és új játék kezdésére szolgáló gombok átkerültek a fölső menüsorba, kiegészítve egy kilépés és egy játék mentése gombbal
- A játék kétszemélyes, miután két darab, öt utasítás hosszú sorozat sikeresen be lett táplálva megkezdődik a szimultán lejátszásuk, ha a végére mindkét játékos életereje nagyobb mint nulla, a játék folytatódik az első játékos körével
- A játék kétféleképpen érhet véget: mindkét játékos életereje elfogy, ez esetben a játék döntetlent hirdet ki; vagy az egyik játékos életben marad míg a másik nem, ebben az esetben természetesen a megfelelő játékos nyert.



Felhasználói eset diagram

Tervezés:

Programszerkezet

- Három rétegű megvalósítás: megjelenés a **RobotFight_WinForms** projekt **View** mappájában, modell és perzisztencia pedig a **RobotFight** projekt **Model** és **Persistence** almappáiban

Perzisztencia

- A perzisztencia feladata a játékállapot hosszú távú elmentésének valamint elmentett játékállapot betöltésének biztosítása
- A hosszú távú adattárolás lehetőségeit az **IRobotFightDataAccess** interfész adja meg, amely lehetőséget ad a tábla betöltésére (**LoadAsync**), valamint mentésére (**SaveAsync**), aszinkron módon
- **PlayerData** osztály: tartalmazza egy játékosra vonatkozóan az x és y koordinátáit a táblán (**int,int**), életerő pontjainak számát (**int**) , pályájának méretét (**int**) valamint irányát (**RobotFight.Model.Orientation**)
- Az említett interfészt megvalósítja a **RobotFightFileAccess** osztály szöveges fájlba való mentésre és betöltésre. A szöveges fájlok .rf kiterjesztéssel rendelkeznek, és a **PlayerData** osztály mezőihez tartozó értékeket tartalmazzák, a következő módon:
    - 1.sor:”x y életerő pályaméret”,
    - 2.sor:”orientáció”

  két példányban

Modell:

- Tartalmazza a:
    - **RobotFightModel** osztály
    - **Player** osztály
    - **PlayerEventArgs** osztály
    - **GameAction** enum
    - **Orientation** enum
    - **GameEnd** enum

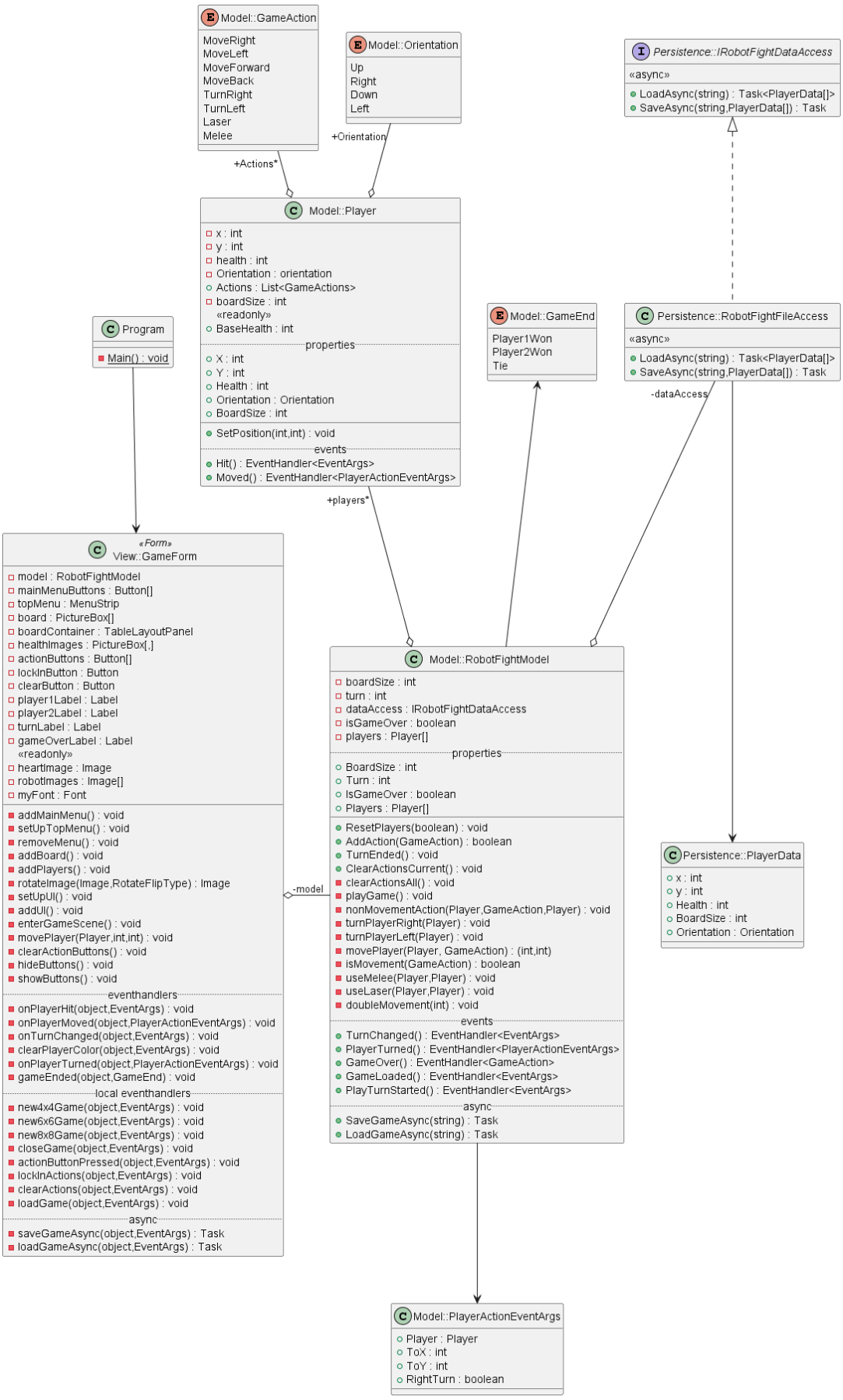
  tagokat (enum-ok nem saját fájlban)

- A modell megvalósításának fő komponense a **RobotFightModel** osztály, ami példányosít két játékost (**Player**), számon tartja a játéktábla méretét (**boardSize**), a játék állapotát (**turn**, **isGameOver**), változásakor kommunikál a nézet felé (**TurnChanged, PlayerTurnStarted**)
- A játék egy körének (különböző egy játékosnak a körétől) lejátszását a **playGame** metódus végzi, ez hívja meg a megfelelő metódusokat a játékosok akcióitól (**Player.Actions**) függően; ha mindkettő mozgás (**isMovement** segítségével) valamilyen irányba akkor doubleMovement-et, egyébként

először a mozgást, aztán a másikat (**nonMovementAction**), vagy játékosok sorrendjében ha egyiksem mozgás, így tudja fenntartani a mozgások szimultánságát a mozgási szabályokra ügyelve
- **movePlayer** metódussal nyeri ki a célkoordinátákat, majd az előbb említett metódusok validálják a helyességét
- **AddAction**, **clearActionsAll** és **ClearActionsCurrent** segítségével menedzseli a játékosok beérkező akció inputjait
- **ResetPlayers** állítja a játékosokat alaphelyzetbe új játék indításakor, vagy más paraméterrel egy játék betöltésekor
- El lehet menteni a játék helyzetét (ami tulajdonképpen a játékosok bizonyos adattagjaiból áll) aszinkron módon a **SaveGameAsync** függvénnyel, betölteni egy másik játékállapotot **LoadGameAsync**-el (amihez a **GameLoaded** esemény is tartozik)
- Játéknak egy féleképpen lehet vége (leszámítva a programból való kilépést, de az nem a modellre tartozik): egy vagy mindkét játékos életereje nullára csökken, ezt a **GameOver** eseménnyel kommunikálja a nézet felé, felparaméterezve a nyertessel

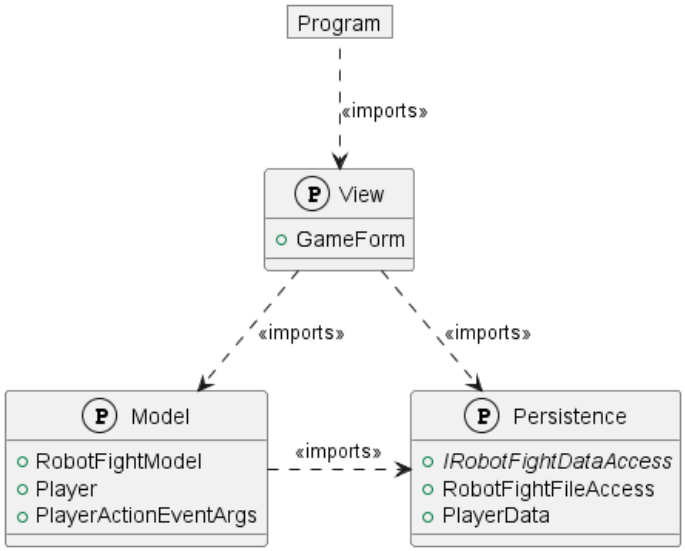
Nézet:

- A nézetet **WinForms** keretrendszerben a **GameForm** osztály valósítja meg, amely tárolja a modell egy példányát (**model**), valamint az adatelérés aktuális példányát (**dataAccess**, ami egy **IRobotFightDataAccess** intefészt megvalósító osztály)
- Program indításakor a főmenü fogad (**mainMenu**), ahonnan új játékot lehet indítani, amikor is megváltozik az ablak mérete, megjelenik a játéktábla, játékosok információi (**healthImages, player1/2Label, turnLabel**), játékos karakterek, az adatbevitelre szolgáló gombok (**actionButtons**, **lockInButton**, **clearButton**) és a felső menüsor (**topMenu**)
- A játéktábla (**board**, a **boardContainer TableLayoutPanelen**) egy **PictureBox** tömbbel van ábrázolva, a játékosok ezen mozognak (leképezve a koordináta párjaikat az egydimenziós megvalósításra), a játékos karaktert nem tartalmazó mezők üresek (**Image** property null), játékosok képei a **robotImages** mezőben vannak tárolva
- Ezeknek az előkészitését az **enterGameScene** metódus végzi, az hívja meg az őket új helyzetbe állító metódusokat
- Új játékot innentől a felső menüsorról lehet indítani, ugyanitt elérhető egy kilépés gomb (**Exit Game**), valamint a **File** almenüben lehet elmenteni a játékállapotot (**Save game**) vagy betölteni másik játékot (**Load game**)
- Egy játék végén a **gameOverLabel** tudósít az eredményről, az akciók gombjai nem lesznek elérhetőek, felső menü továbbra is használható, kilépésre vagy új játszma kezdésére



Osztálydiagram

Csomagdiagram



Tesztelés:

A modellre vonatkozó egységtesztek a RobotFightModelTest osztályban lettek elhelyezve, Moq segítségével szimulált adateléréssel

A tesztesetek:

- **TestLoadGameAsync**: mock-olt betöltés után a modell **Player** adattagjaiban tárolt információk helyességének tesztje
- **TestResetPlayers**: helyes értékeket vesznek fel a re-inicializált **Player**-ek
- TestTurnChange: megadható-e helytelen érték a turn mezőnek (**Turn** propertyn keresztül), helyes értékadás kiváltja-e a megfelelő eseményt (**TurnChanged**)
- **TestSetPosition**: **Player** osztály **SetPosition** metódusának tesztelése a táblához képest helyes és helytelen értékekkel, esemény (**PlayerMoved**) megfelelő helyen történő kiváltása megtörténik-e
- **TestAddAction**: **AddAction** metódus tesztelése alaphelyzetben, és mikor már tartalmaz az aktuális **Player Action** listája 5 tagot
- **TestPlayTurn**: **TurnEnded** metódus tesztelése a benne meghívott **playGame** eseményére fókuszálva (**PlayTurnStarted**)
- **TestMoveToEachOther**: mozgás edgecase-ének a vizsgálata, amit a **SetPosition** önmagában nem ellenőriz, mert a táblán helyes: a játékosok egymásra akarnak lépni
- **TestTurnPlayers**: **turnPlayerLeft**, **turnPlayerRight** és eseményük (**PlayerTurned**) tesztelése
- **TestAttacks**: **useLaser** és **useMelee** tesztelése, a játékosok életerejével együtt
- **TestGameOver1Won**: **GameOver** esemény tesztelése az egyik játékos megölésén keresztül, esemény paraméterének ellenőrzése
- **TestGameOverTie**: **GameOver** esemény tesztelése játékos karakterek egyszerre történő halálakor