Készítette:

Beregi Bence Zsolt
E-mail: dqk6te@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 játék táblázata háromféle méretű lehet: kicsi (4x4), közepes (6x6) és nagy (8x8). A program indításakor egy új játék indul közepes méretű táblázattal.

A feladatot egy ablakos (+1 ablak a méret kiválasztásra) asztali alkalmazásként Windows Forms grafikus felülettel valósítjuk meg.

Az ablakban elhelyezünk egy menüt a következő menüponttal: File → New Game, Save és Load. Az ablak alján pedig status sorban láthatjuk a hátralévő lépéseket.

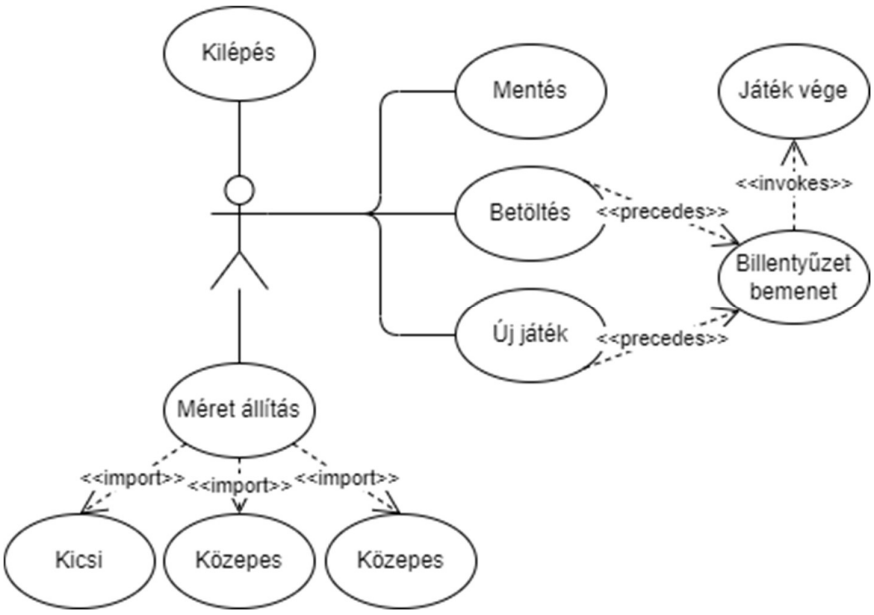
A játéktáblát rajzolt rács ábrázolja az előre megadott méretben és a piros pötty az első játékost, a kék meg a másodikat. Billentyűzet nyomásra adhatjuk meg a lépéseket., Amint

megadjuk az 5 rendelkezésre álló lépést, a másik játékos következik. Amikor mindkettő játékos megadta a lépéseit a program végrehajtja sorba azokat.

Jobb alsó sarokban láthatjuk, hogy kinek mennyi élete van, bal alsóban azt, hogy ki van épp soron és a betáplált lépéseket.

A játék automatikusan feldob egy dialógus ablakot a játék végén jelezve, hogy ki nyert. Automatikusan jön létre új játék.

A felhasználói esetek az 1. ábrán láthatók.



1. ábra

Tervezés:

A programszerkezet:

A programot háromrétegű architektúrában valósíjuk meg. A megjelenítés a View, a modell a Model, míg a perzisztencia a Persistence névtérben helyezkedik el. A program csomagszerkezete a 2. ábárán található

Perzisztencia

Az adatkezelés feladata a játék táblával kapcsolatos adatok tárolása, valamint mentés/betöltés biztosítása.

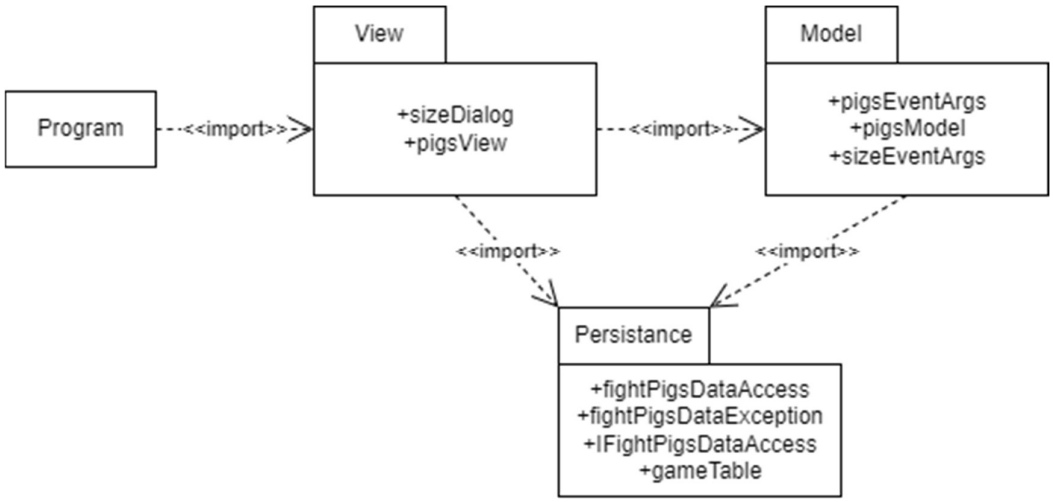
A **gameTable** osztály egy érvényes játék táblát biztosít, ahol minden mezőnek ismert az értéke az az, hogy a játékosok hol helyezeknek el. Ezt lekérhetjük a **GetPlayer**-el. A tábla alapból 6x6os méretben jön létre, de ezt a konstruktorban megváltoztathatjuk. Illetve a bejött adatok alapján végrehajtja a lépést (**Step**)

A hosszú távú adattárolás lehetőségeit az **IFightPigsDataAccess** interfész adja meg, amely lehetőséget biztosít a tábla betöltésére (**LoadAsync**), valamint a tábla mentésére (**SaveAsync**). A műveleteket hatákonysági okokból async módon valósítjuk meg.

Az interfész szöveges fájl alapú adatkezelését a **FightPigsDataAccess** osztály valósítja meg. A fájlkezelés során fellépő hibákat a **FightPigsDataException** kivétel jelzi.

A program az adatok egyedi fájlként tudja eltárolni, melyek a .pig kiterjesztést kapják. Ezeket az adatokat bármikor be lehet tölteni, illetve ki lehet menteni az aktuális állást.

A fájl első sora tartalmazza a tábla méretét, az életerőket, a játékosok elhelyzkedéseit és orientációját, ami egy szám, ami a Direction enummal kap értelmet.



2. ábra

Modell:

A modell lényegi részét a **pigsModel** osztály valósítja meg, amely szabályozza a tábla tevékenységeit, valamint a játék egyéb paramétereit. Úgymint a méret (**size**), soron következő játékos (**Player**) véget ért-e a játék (**whoWon**). A típus lehetőséget ad az új játék létrehozásra (**NewGame**). Billentyűzet bemenet lekezelése, és adott esetben léptetés elindítása (**KeyInputHandler**). Illetve amennyiben vége a játéknak a győztes megállapítását (**whoWon**).

A játékállapot változásról a **GameAdvanced** esemény. A **GameAdvanced** argumentuma a **PigsEventArgs** tárolja a győztest (amennyiben nincsen ennek értéke 0), valamint a hátralévő lépések számát.

A modell példányosításkor megkapja Az adatkezelés felületét, amelynek segítségével lehetőséget ad betöltésre (**LoadGameAsync**) és mentésre (**SaveGameAsync**).

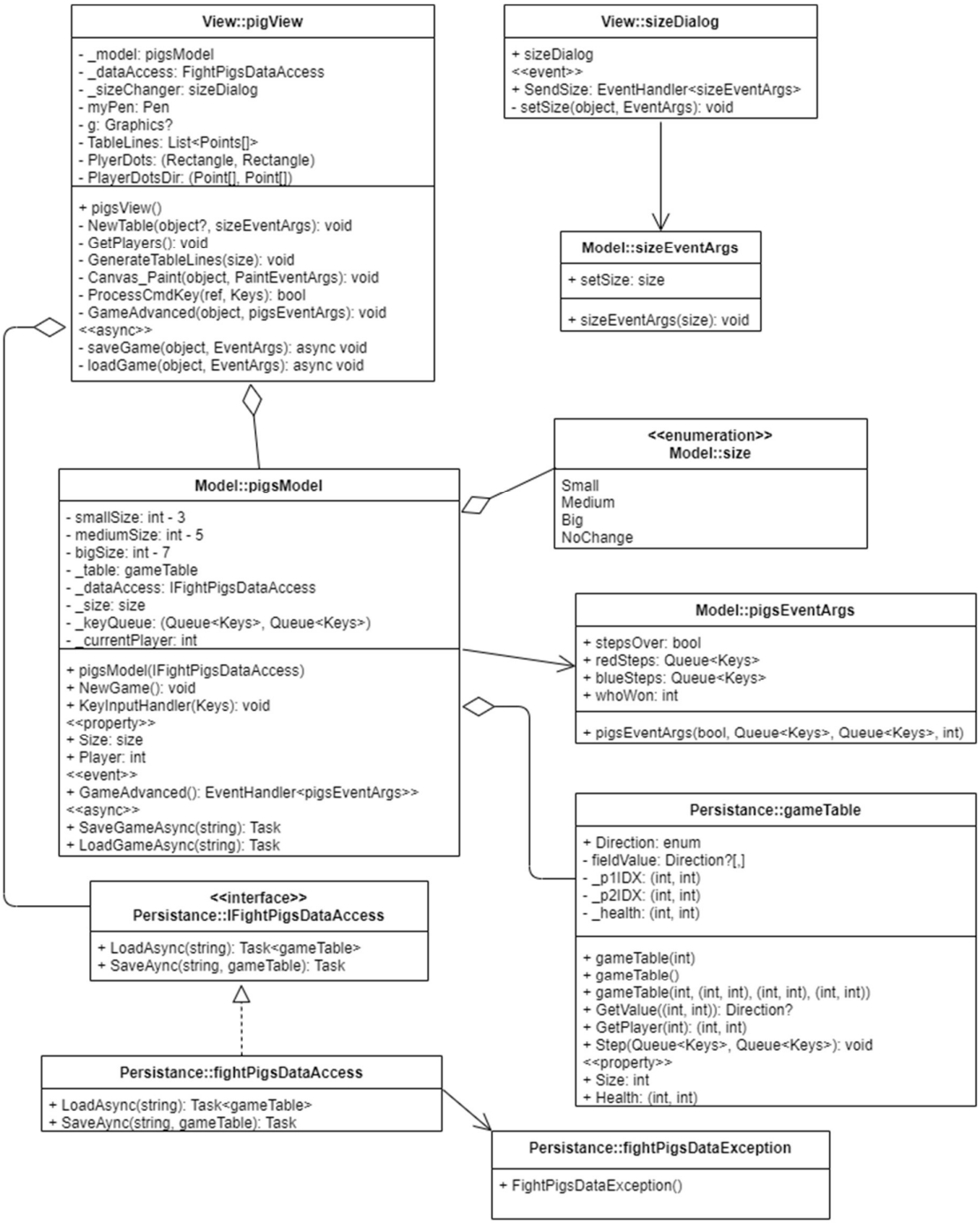
A játék méretét **Size** felsorolási típuson át kezeljük, és a **pigsModel**ben konstansok segítségével tároljuk az egyes méretek paramétereit.

Nézet:

A nézetet a **pigsView** osztály biztosítja, amely tárolja a modell egy példányát (**_model**), valamint az adatelérés konkrét példányát (**_dataAccess**).

A játéktáblát egy grafikusan kirajzolt táblázat ábrázolja. A felületen létrehozzuk a megfelelő menüpontokat, illetve a státuszsort, valamint a dialógusablakoktat. A méret választó dialógusablakot egy külön nézetben (**sizeDialog**) hozzuk létre. A játéktábla vonalainak generálását, játékos pöttyeinek megrajzolását és méret meghatározását (**GenerateTableLines, GetPlayers** és **newTable**), a vonalak kirajzolását (**Canvas_Paint**). A billentyűzet input kezelését a ***ProccessCmdKey*** metódus felülírásával határozzuk meg.

A program teljes statikus szerkezete a 3. ábrán látható.



3. ábra

Tesztelés:

A modell funkcionalitása egységtesztek segítségével lett ellenőrizve a pigsTest osztályban.

Az alábbi tesztesetek kerültek megvalósításra:

- **PigsModelBaseConstruct**, **PigsModelNewGameBig**, **PigsModelNewGameSmal**l: Új játék indítása különböző méretekkel és alapbeállítások ellenőrzése.
- **PigsModelStep**: A Modelben létrehozott játéktáblában léptetjük a játékosokat és ellenőrizzük, hogy megfelelően lépett-e.
- **KeyInputHandler**: Ebben a billentyű lenyomáskor meghívódó metódust teszteljük névszerint a KeyInputHandler-t. Itt megvizsgáljuk, hogy megfelelően lépnek-e a játékosok.
- **HPLoss**: Itt azt vizsgáljuk, hogy amennyiben egy játékos eltalálja a másikat megfelelően levonódik-e az életerő.
- **GameOver**: Itt azt vizsgáljuk, hogy amennyiben teljesítődik valamelyik játék vége feltétel megfelelően véget ér- e a játék.