

2.Beadandó feladat dokumentáció

Készítette: Ferencz Tamás, G0820E

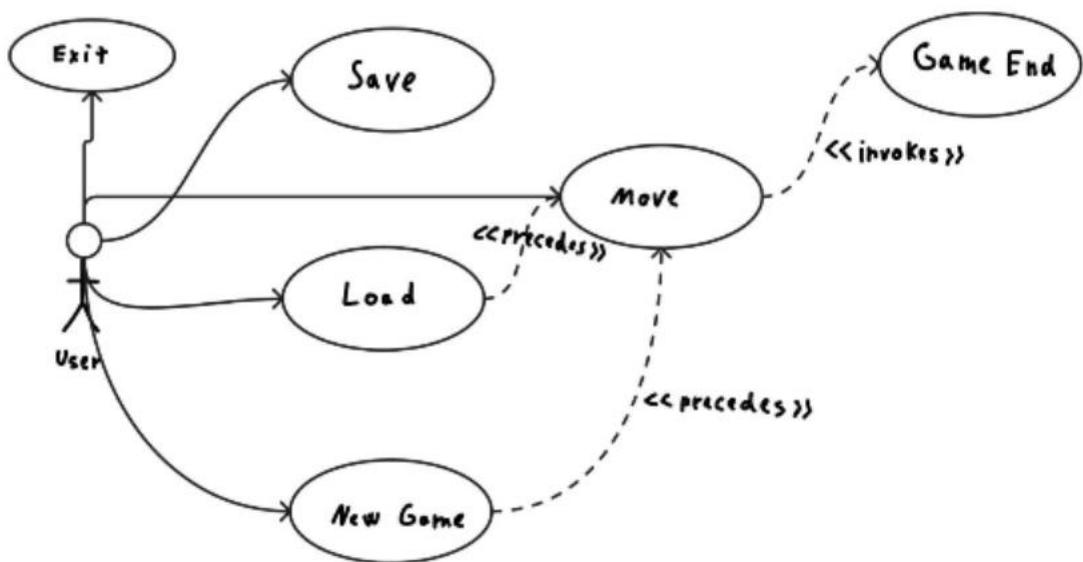
E-mail: ferencztamas@student.elte.hu

Feladat:

Készítsünk programot, amellyel a következő két személyes játékot játszhatjuk. A malomjátékban két játékos egy 24 mezőből álló speciális játéktáblán játszhatja, ahol a mezők három egymásba helyezett négyzetben helyezkednek (mindegyikben 8, a sarkoknál és a felezőpontknál), 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, aki először megy 3 alá a bábuk száma a mozgatási fázis alatt.

Elemzés:

- A program indításkor automatikusan egy üres malom táblát rajzol ki, és egy új játékot indít.
- A feladatot egyablakos asztali alkalmazásként Windows Presentation Foundation grafikus felülettel valósítjuk meg.
- Az ablakban elhelyezünk egy menüt a következő menüpontokkal: Új játék, Mentés és betöltés. A menüpontok fölött helyezkedik el a Fázis (aktuális játékmenet állapotát jelzi), aktív játékos (a soron következő játékost mutatja), illetve a játékmezőn elhelyezett bábuk számát.
- A játéktáblát egy 24 nyomógombkból álló rácsos szerkezet reprezentálja. A nyomógomb egérkattintás hatására a játékosnak zöld jelzésű, lehetséges szomszédos lépéseiit mutatja.
- A játék automatikusan feldob egy dialógusablakot, amikor vége a játéknak (nyertes fél kiírtatása), amikor játék mentés, illetve betöltés történik.

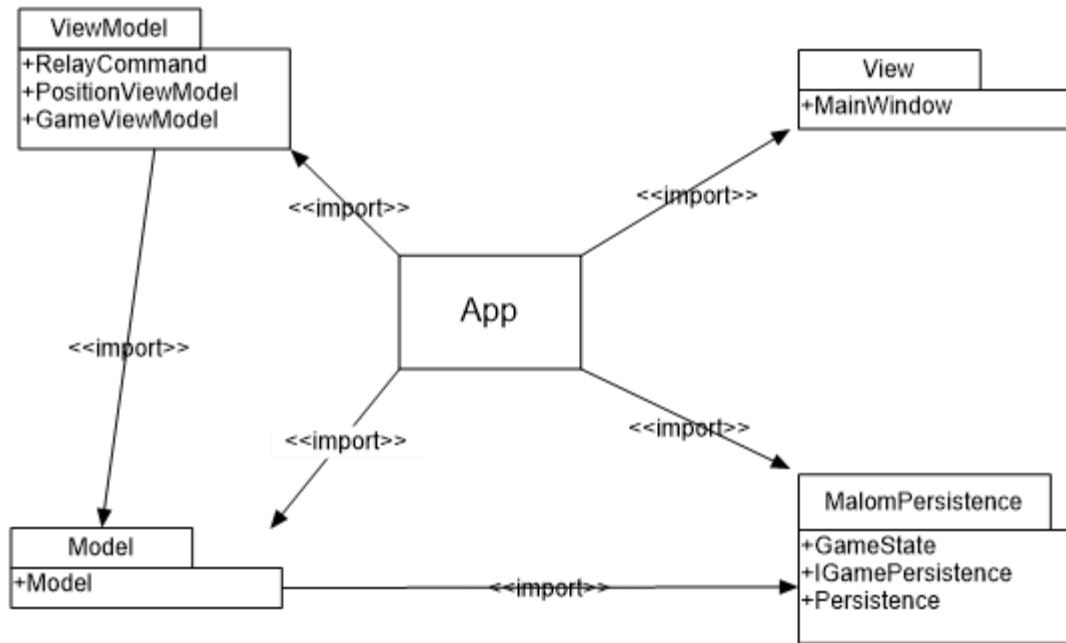


Felhasználati esetek diagramma

Tervezés:

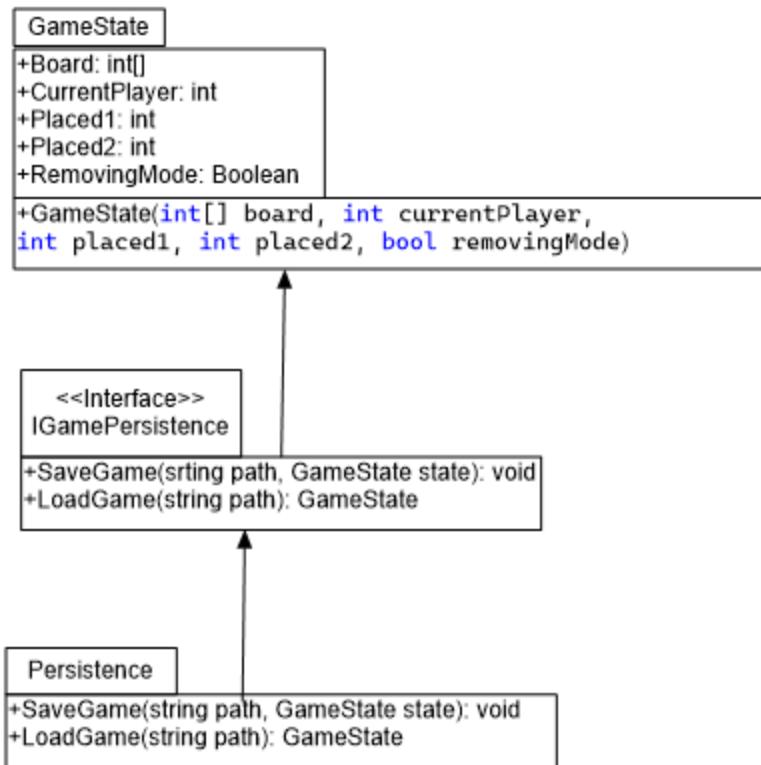
Programszerkezet:

- A programot MVVM architektúrában valósítjuk meg, ennek megfelelően View, Model, ViewModel és Persistence névtereket valósítunk meg az alkalmazáson belül. A program környezetét az alkalmazás osztály (App) végzi, amely példányosítja a modellt, a nézetmodell és a nézetet, biztosítja a kommunikációt, valamint felügyeli az adatkezelést. A program csomagszerkezete a 2. ábrán látható.
- A program szerkezetét két projektre osztjuk implementációs megfontolásból: a Persistence és Model csomagok a program felületfüggetlen projektjében, míg a ViewModel és View csomagok a WPF függő projektjében kapnak helyet.



Perzisztencia:

- A Persistence réteg feladata a játékállás hosszú távú tárolása és visszatöltése. Az adatkezelés a Persistence osztályban történik, amely a játékhoz kapcsolódó adatokat szöveges fájlban tárolja. Ehhez a réteghez tartozik az IGameModelPersistence interfész is, amely az adatok mentéséhez és betöltéséhez szükséges műveleteket definiálja. Az interfész a Persistence osztály valósítja meg.



Model réteg

A **Model** osztály tartalmazza a teljes játéklogikát.

Feladata a játék szabályainak betartatása, a lépések kezelése, valamint az állapotok tárolása.

A modell a következő fő adattagokat tartalmazza:

- `int[] Board` – a tábla 24 mezőjének állapota (0 = üres, 1 = játékos 1, 2 = játékos 2),
- `int CurrentPlayer` – a soron következő játékos azonosítója,
- `int Placed1, int Placed2` – az eddig lehelyezett bábuk száma játékosonként,
- `bool RemovingMode` – jelzi, hogy az aktuális játékos millt (sor) alkotott-e, és el kell távolítania az ellenfél egyik bábuját.

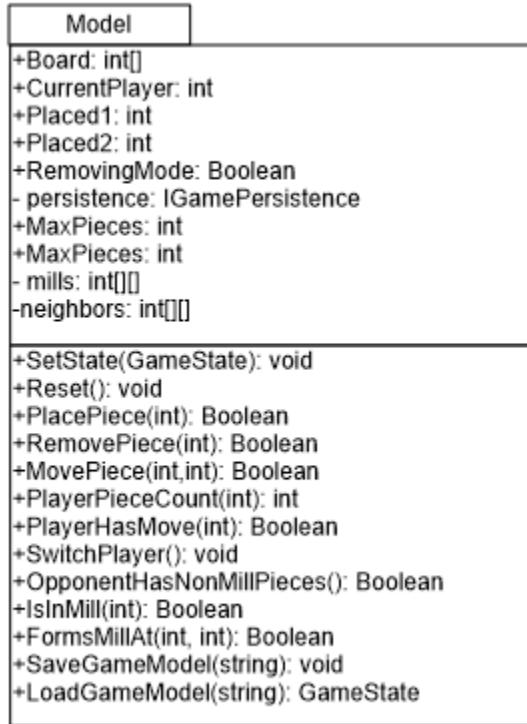
A modell biztosítja a következő főbb funkciókat:

- **PlacePiece(int idx)** – új bábu lehelyezése a megadott mezőre,
- **MovePiece(int from, int to)** – bábu mozgatása, ha a játékos már minden bábuját lehelyezte,
- **RemovePiece(int idx)** – az ellenfél bábu eltávolítása, ha mill alakult ki,
- **FormsMillAt(int idx, int player)** – ellenőrzi, hogy létrejött-e egy új sor (mill),
- **Reset()** – új játék indítása alapállapotból,
- **SetState(...)** – betöltött játékállapot beállítása a fájlból,
- **PlayerPieceCount(...)** – visszaadja a játékos bábuinak a számát

- **PlayerHasMove(...)** – a játékos maradék lépését adja vissza, ha van
- **SwitchPlayer(...)** – a játékosok soron következését módosítja

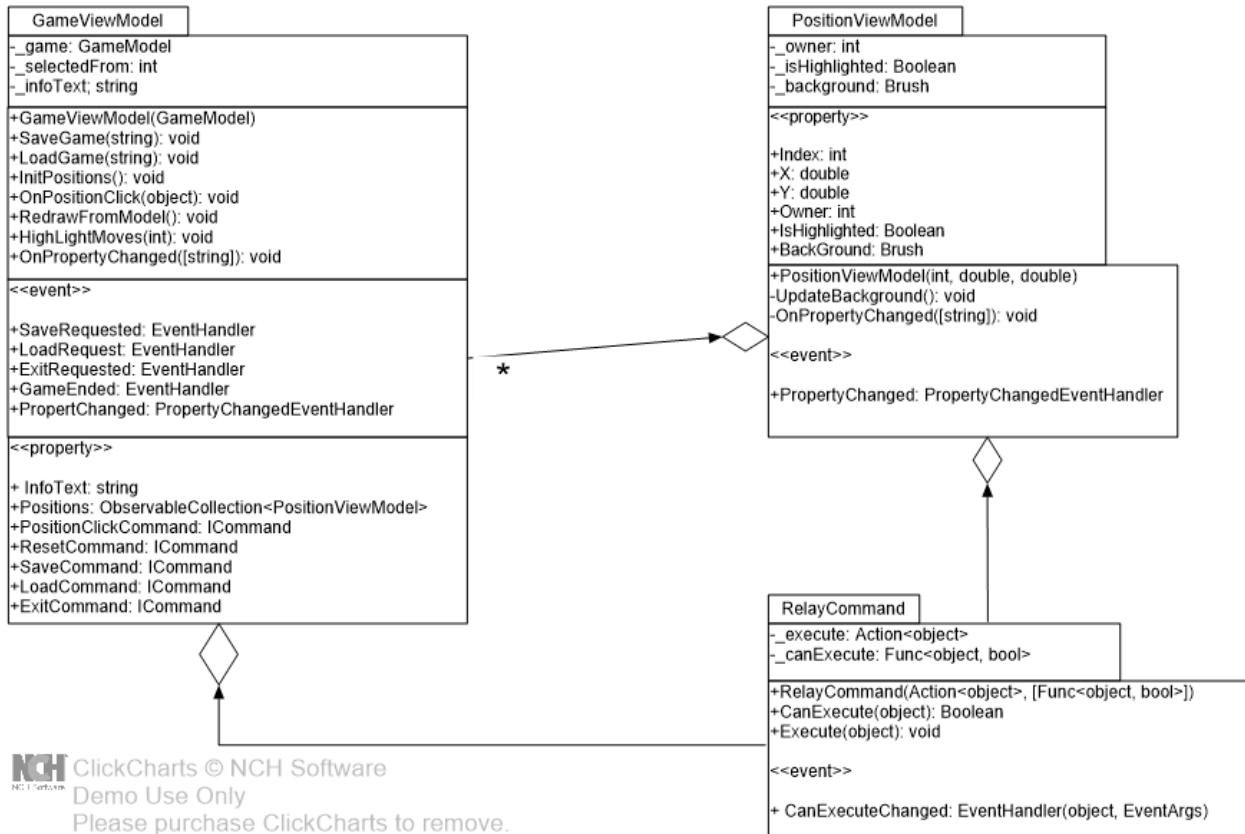
A logika betartja a malom szabályait:

- csak üres mezőre lehet bábut tenni,
- mozgatni csak szomszédos mezőre lehet (kivéve, ha a játékosnak 3 bábuja van – ekkor „repülhet”),
- ha mill alakul ki, az ellenfél egyik bábuja levehető, kivéve, ha minden bábuja millben van.
- Az alábbi diagramm a model csomag diagrammja.



ViewModel réteg

- A ViewModel kapcsolatot teremt a Model és a View között, parancsokkal és propertykkel adja át az adatokat a felületnek, miközben maga UI-független marad.
- Az alábbi ábra a nézetmodell osztálydiagrammja

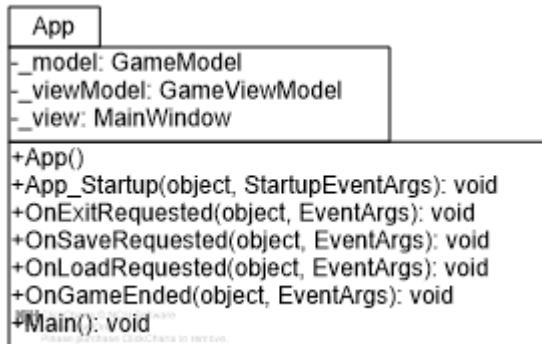


View réteg

- A View a felhasználóval való interakcióért felel, vizuálisan megjeleníti a ViewModelből érkező adatokat és továbbítja a parancsokat.

Környezet réteg

- Az App osztály feladata az egyes rétegek példányosítása (App_Startup), összekötése, a nézetmodell, valamint a modell eseményeinek kezelése, és ezáltal a játék, az adatkezelés, valamint a nézetek szabályozása.
- Az alábbi ábra a vezérlés osztálydiagrammja



Tesztelés

- A **Malom játék modelljének** funkcionalitása egységesztek segítségével került ellenőrzésre a MalomTests projektben.
- A tesztek célja a **játékmenet logikai helyességének** biztosítása: a bábu elhelyezés, mozgatás, levétel, malomképzés, valamint a játékosváltás és állapotkezelés helyes működésének ellenőrzése.

Az alábbi főbb tesztesetek kerültek megvalósításra:

- **Reset_ShouldInitializeGameState:** új játék indításakor az alapállapot ellenőrzése.
- **SetState_ShouldApplyCorrectValues:** a modell állapotának helyes visszaállítása betöltéskor.
- **PlacePiece_ShouldPlacePieceAndSwitchPlayer:** a bábu lerakás, játékosváltás és malomképzés ellenőrzése.

- **RemovePiece_ShouldRemoveOpponentPiece:** ellenfél bábujának levételének tesztelése különböző feltételek mellett.
- **ShouldAllowFlyingIfThreePiecesLeft:** bábumozgatás és „flying” szabály ellenőrzése.
- **PlayerHasMove és PlayerPieceCount tesztek:** a játékos mozgásképességének és bábu-számának helyes számítása.
- **FormsMillAt:** privát függvények tesztelése reflekszióval a malomképzés logikájának validálására.
- **OpponentHasNonMillPieces:** a játékosváltás és az ellenfél állapotának ellenőrzése.