„Eseményvezérelt Alkalmazások”

2.Beadandó dokumentáció

4.feladat

Készítette: Székely Regő

Neptun-azonosító: H5LJFS

E-mail: h5ljfs@inf.elte.hu

2023.december 3.

Feladat

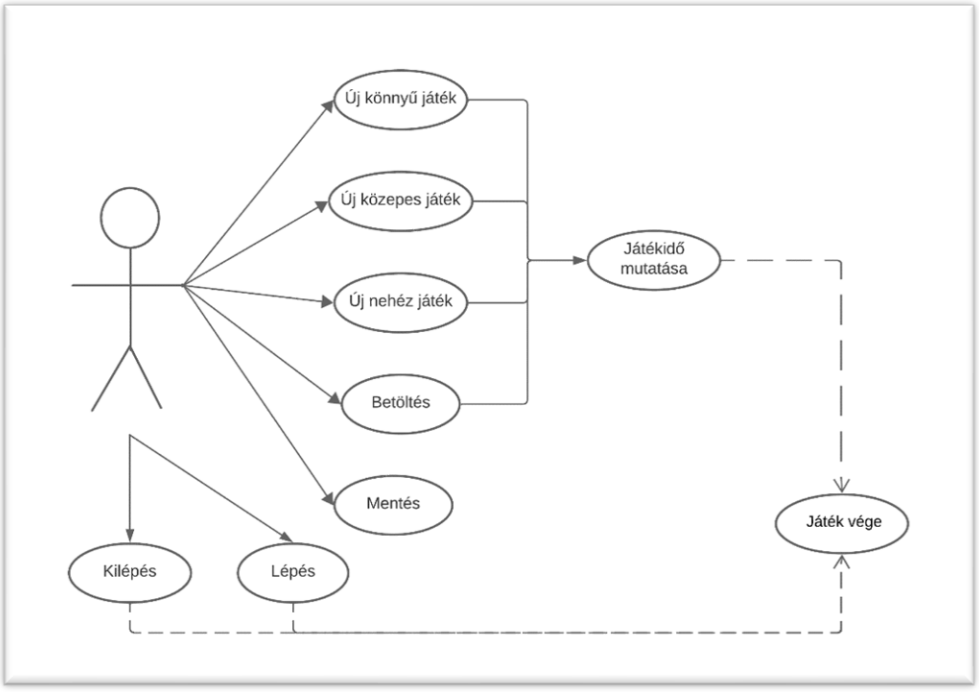
4. Labirintus

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 játékos célja, hogy a bal alsó sarokból indulva minél előbb kijusson a labirintusból.
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 program legalább 3 különböző méretű pályát tartalmazzon.
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.

Elemzés:

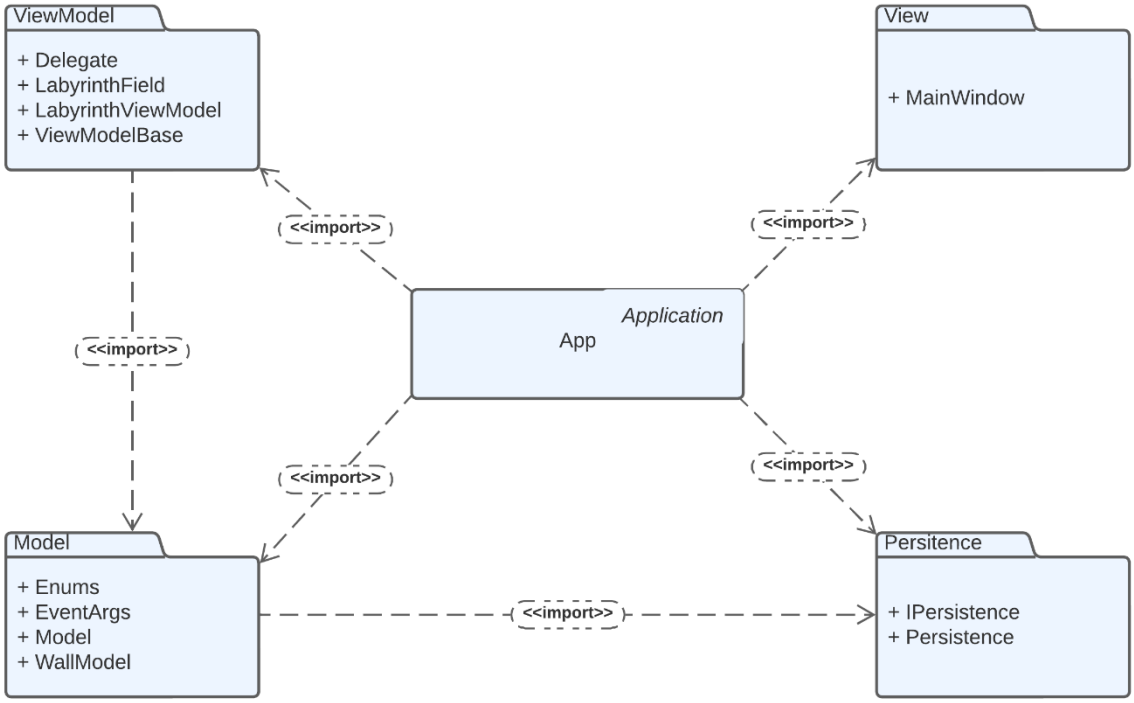
- A játékot három kiválasztott pályamérettel használhatjuk: 10 X 10, 15 X 15, 20 X 20. Ezeket az „Easy”, „Medium” és „Hard” jelzőkkel választhatjuk ki. Érdemes megjegyezni azt, hogy ebben az esetben minél nagyobb a pálya annál nehezebb a játék hiszen a több fallal rendelkező pályák bonyolultabb utakat eredményeznek. A program indításkor New Game gombbal indítható a játék.
- A feladatot egyablakos asztali alkalmazásként Windows Presentation Foundation grafikus felülettel valósítjuk meg.
- Az ablak aljában elhelyezem státuszsort, ami tartalmazza a szüneteltetés (Pause) gombot és az eltel idő jelzésére szolgáló számlálót. A felső menüsávban található a lenyíló menüpont, ahol a következők találhatóak: új játék (New Game), betöltés (Load Game), mentés (Save Game) és kilépés (Exit Game).
- A játéktábla a kiválasztott méretű gombra kattintva generálódik le.
- A játékos mozgása a W, A, S, D gombokkal történik (Fel-Bal-Le-Jobb).
- A játék automatikusan feldob egy dialógusablakot, amikor vége a játéknak rajta megfelelő üzenettel.
- A mentés és betöltés helyét előre meg kell adni a programba változóként. Betöltés után rögtön elindul a játék.
- Az ábrán láthatóak a felhasználói esetek:



1. ábra: Felhasználói esetek diagramja

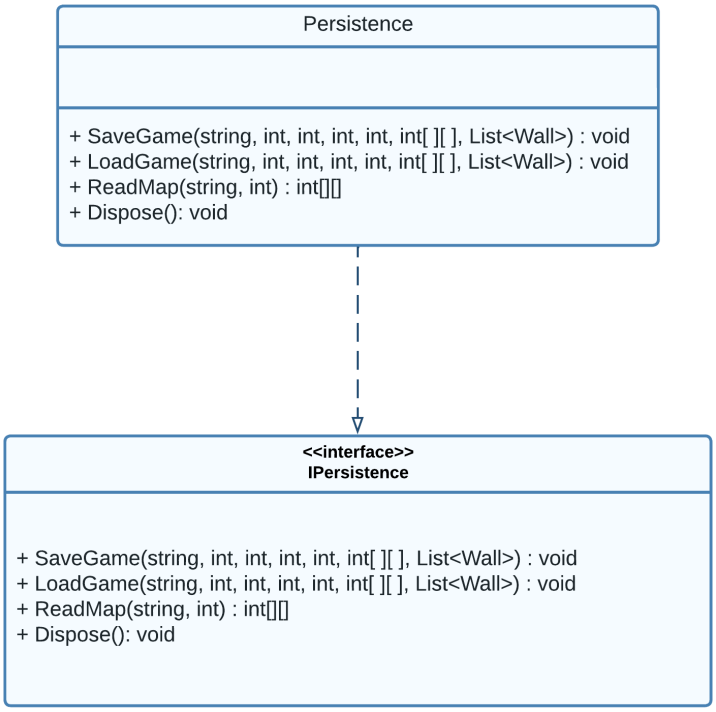
Tervezés:

- *Programszerkezet:*
  - A program MVVM architektúrában van megvalósítva. 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.



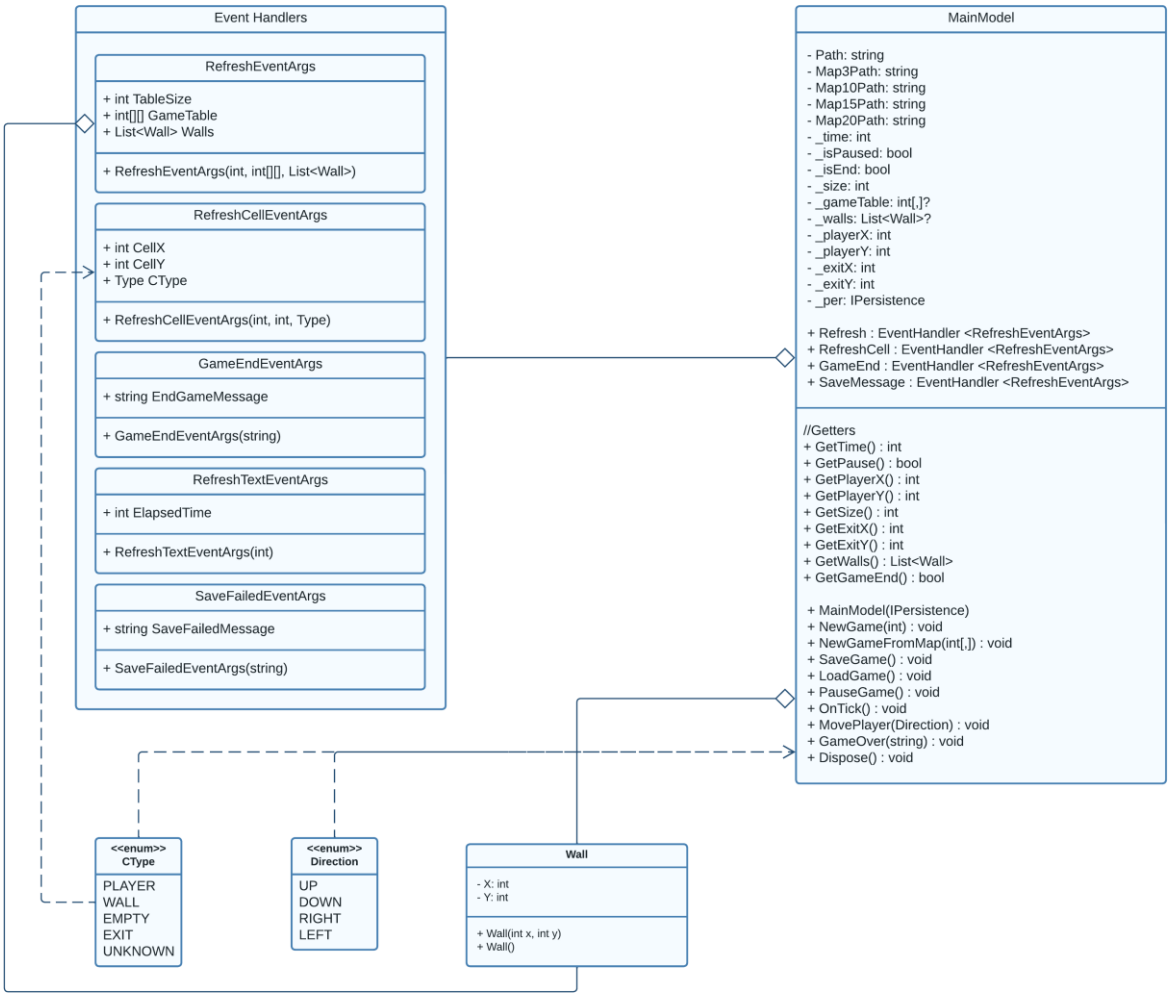
2. ábra: Az alkalmazás csomagdiagramja

- *Perzisztencia:*
  - A betöltés és a mentés feladatát látja el. Illetve saját hibát is dob, ha sikertelen a mentés vagy a betöltés.
  - A program az adatokat egy szövegfájlban tárolja, amit a megadott mentési útvonalon bármikor el lehet menteni. Hiba lép fel, ha nem létezik a megadott útvonal.
  - A mentési fájl felépítése:
    - Eltelt idő
    - Játékos X pozíciója
    - Játékos Y pozíciója
    - Tábla mérete
    - A tábla az utak és falak eloszlásával
    - Jelen lévő falak pozíciója (X, Y) egymás alatt szóközzel elválasztva.



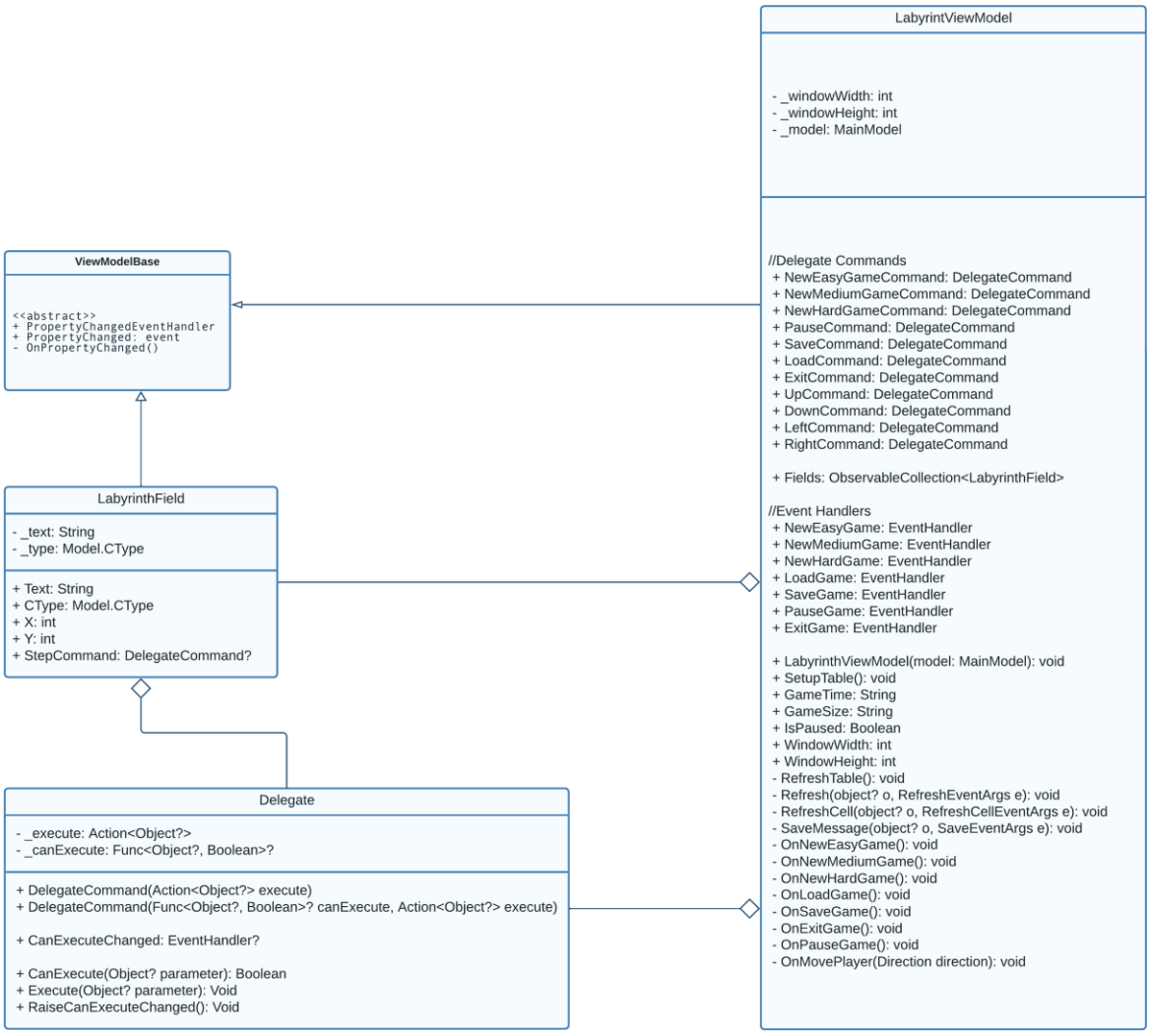
3. ábra: A Persistence csomag osztálydiagramja

- *Modell:*
  - A modell lényegi részét a MainModel osztály valósítja meg, amely szabályozza a tábla tevékenységeit, valamint a játék paramétereit úgy, mint az idő(_time), játékos pozíciója (_playerX és _playerY) stb. A típus lehetőséget ad új játék kezdésére (NewGame), valamint lépésre (MovePlayer). Új játéknál a kiválasztott méret alapján legenerálódnak a falak, a játékos és a kijárat. Az idő haladását időbeli lépések végzése (OnTick) teszi lehetővé.
  - A játékállapot változásáról az OnTick és a MovePlayer esemény gondoskodik, hiszen a kettő egymástól függetlenül működik. A játék végét a GameOver esemény kezeli.
  - A modell példányosítása során példányosítva lesz a timer is, aminek Tick adattagjához hozzá lesz rendelve az OnTick esemény, ami kiváltja az idő előrehaladását.
  - A játéktér előre el van tárolva.
  - A játék időbeli kezelését egy időzítő végzi (_timer), amelyet mindig aktiválunk játék során, illetve inaktiváljuk, amennyiben a játék megállításra kerül.



4. ábra: A Model csomag osztálydiagramja

- *Nézetmodell:*
  - A nézetmodell megvalósításához felhasználunk egy általános utasítás (DelegateCommand), valamint egy ős változásjelző (ViewModelBase)osztályt.
  - A nézetmodell feladatait a LabyrinthViewModel osztály látja el, amely parancsokat biztosít az új játék kezdéséhez, játék betöltéséhez, mentéséhez, valamint a kilépéshez. A parancsokhoz eseményeket kötünk, amelyek a parancs lefutását jelzik a vezérlőnek. A nézetmodell tárolja a modell egy hivatkozását (_model), de csupán információkat kér le tőle, illetve a játéknehézséget szabályozza. Direkt nem avatkozik a játék futtatásába.
  - A játékmező számára egy külön mezőt biztosítunk (LabyrinthField), amely eltárolja a pozíciót, szöveget, valamint a typusát A mezőket egy felügyelt gyűjteménybe helyezzük a nézetmodellbe (Fields).



5. *ábra: A nézetmodell osztálydiagramja*

- *Nézet:*
  - A nézet csak egy képernyőt tartalmaz, a MainWindow osztályt. A nézet egy rácsban tárolja a játékmezőt, a menüt és a státuszsort. A játékmező egy ItemsControl vezérlő, ahol dinamikusan felépítünk egy rácsot (UniformGrid), amely gombokból áll. Minden adatot adatkötéssel kapcsolunk a felülethez, továbbá azon keresztül szabályozzuk a gombok színét is.
- *Környezet (6. ábra):*
  - • Az App osztály feladata az egyes rétegek példányosítása (App_Startup), összekötése, a nézetmodell, valamint a modell eseményeinek lekezelése, és ezáltal a játék, az adatkezelés, valamint a nézetek szabályozása.
  - • A játék léptetéséhez tárol egy időzítőt is (_timer), amelynek állítását is szabályozza az egyes funkciók hatására.

6. ábra: A vezérlés osztálydiagramja

Tesztelési terv:

- A modell működését egységtesztek segítségével lett vizsgálva (LabyrinthTest osztály).
- Megvalósított tesztesetek:
    - NewGameTest1: Új játék létrehozásának elemzése és annak változói ellenőrzése 10 x 10-es tábla esetén.
    - NewGameTest2: Új játék létrehozásának elemzése és annak változói ellenőrzése 15 x 15-es tábla esetén.
    - NewGameTest3: Új játék létrehozásának elemzése és annak változói ellenőrzése 20 x 20-es tábla esetén.
    - PlayerMovementTest: Játékos mozgásának ellenőrzése minden irányba, illetve, hogy nem mozdul pályán kívülre.
    - GameEndTest: Játék végállapotának vizsgálata indítás előtt.
    - TimerTest: Állapotvizsgálatok, hogy az időzítő tényleges rendben működik.
    - PauseTest: A megállítás állapotvizsgálata.
    - EndGameTest: A játék végállapotának vizsgálata
    - HitWallTest: A játékos pozíciójának vizsgálata falnak ütközés esetén.