2. Beadandó feladat dokumentáció

Készítette: Gábris Attila

Email: ggloe7@inf.elte.hu

Feladat:

Menekülj:

Készítsünk programot, amellyel a következő játékot játszhatjuk. Adott egy $n \times n$ elemből álló játékpálya, ahol a játékos két üldöző elől próbál menekülni, illetve próbálja őket aknára csalni. Kezdetben a játékos játékpálya felső sorának közepén helyezkedik el, a két üldöző pedig az alsó két sarokban. Az ellenfelek adott időközönként lépnek egy mezőt a játékos felé haladva úgy, hogy ha a függőleges távolság a nagyobb, akkor függőlegesen, ellenkező esetben vízszintesen mozognak a játékos felé. A pályán véletlenszerű pozíciókban aknák is elhelyezkednek, amelyekbe az ellenfelek könnyen beleléphetnek, ekkor eltűnnek (az akna megmarad). A játékos vízszintesen, illetve függőlegesen mozoghat (egyesével) a pályán, és célja, hogy az ellenfeleket aknára csalja, miközben ő nem lép aknára. Ha sikerül minden üldözőt aknára csalnia, akkor győzött, ha valamely ellenfél elkapja (egy pozíciót foglal el vele), vagy aknára lép, akkor veszített. A program biztosítson lehetőséget új játék kezdésére a pályaméret megadásával (11 × 11, 15 × 15, 21 × 21), valamint játék szüneteltetésére (ekkor nem telik az idő, és nem léphet senki). Ismerje fel, ha vége a játéknak, és jelenítse meg, hogy győzött, vagy veszített-e a játékos. Ezen felül szüneteltetés alatt legyen lehetőség a játék elmentésére, valamint betöltésére. A program játék közben folyamatosan jelezze ki a játékidőt.

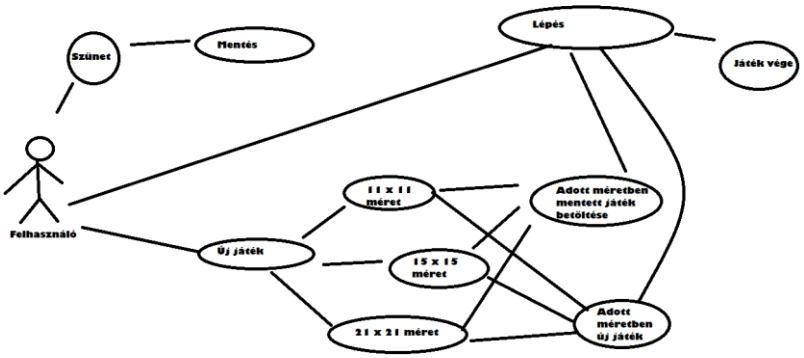
Elemzés:

- A játék három méretű táblával játszható: 11 x 11; 15 x 15; 21 x 21. A program a játék indításakor default a 15 x 15-ös táblaméretre állítódik és rögtön új játékot kezd.
- A feladatot egyablakos asztali alkalmazásként Windows Forms grafikus felülettel valósítjuk meg.
- Az ablakban elhelyezünk egy menüt a következő menüpontokkal:
    - Menü (Új játék (11 x 11 (Saved Game, New Game))
      15 x 15 (Saved Game, New Game))
      21 x 21 (Saved Game, New Game)),

      Játék mentése,

      Szünet)
    - Valamint a kijelző alján egy feliratot, mely folyamatosan kijelzi az eltelt időt.
- A játéktáblán kezdetben aknák (feketével jelölve) random pozíciókon, a játékos (narancs sárgával jelölve) a legfelső sor közepén, és a két üldöző (kékkel megjelenítve) a két alsó sarokban találhatóak. Az üldözők minden másodpercben közelítenek egy négyzetet a játékos felé a leírásban meghatározott módon, míg a játékos a megfelelő irányba tett egér kattintással mozogva igyekszik őket aknára csalni, úgy, hogy ne kapják el, és ne lépjen ő maga sem aknára.
- Amennyiben mind a kettő üldöző aknára lépett a játékos nyer vagy elkapták a játékos vagy az aknára lépett, és egy dialógus ablak jelenik meg jelezve a játék végét. Lehetőség van továbbá Szünetet tartani és közben elmenteni a játékot, vagy kilépni, és előtte elmenteni a játékot.

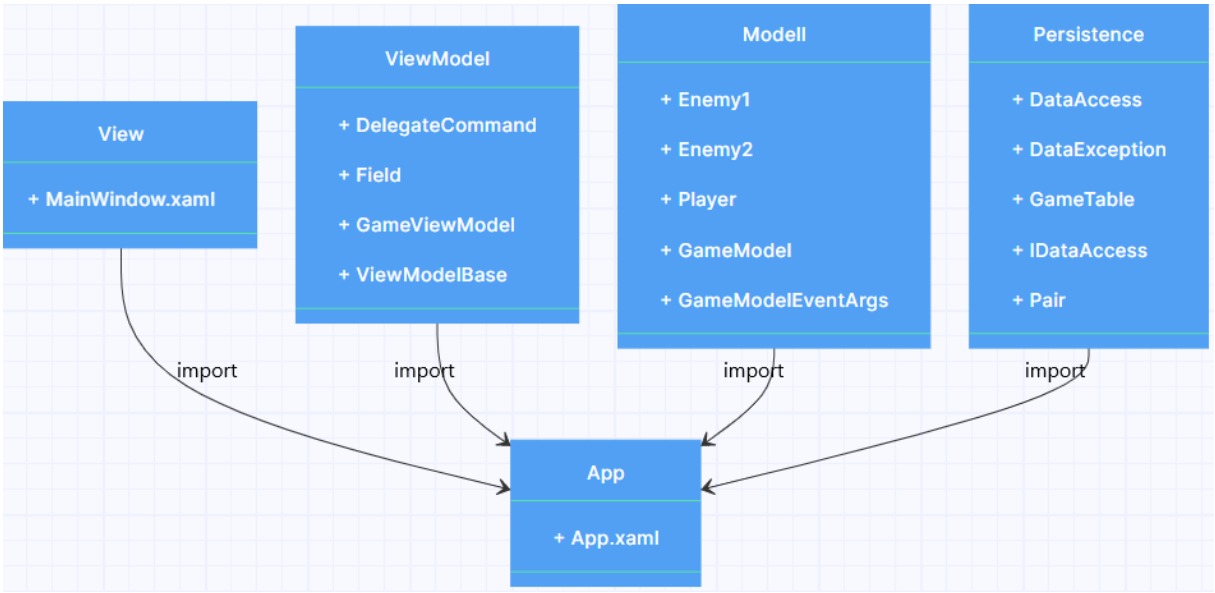
- Új játékot a pályaméret megválasztásával tudunk kezdeni az „Új játék” menüpontban. Ugyanezen helyen tudunk betölteni elmentett játékot.
- A felhasználói esetek az 1. ábrán láthatóak:



1. ábra

Tervezés:

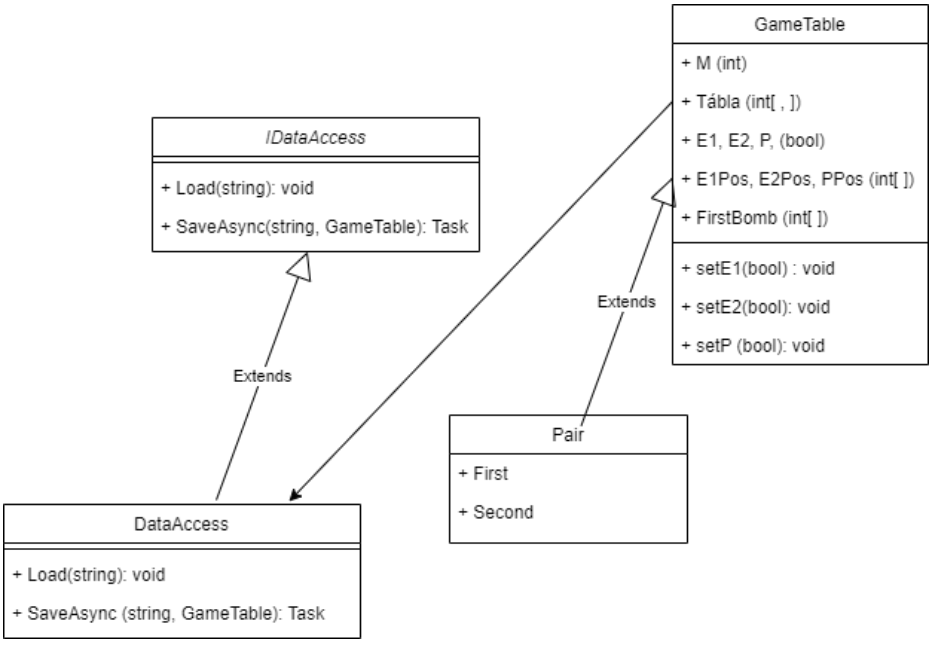
- Programszerkezet:
  - A programot MVVM architektúrában valósítjuk meg, ennek megfelelően View, Model, ViewModel és Persistence névtereket valósítunk meg az alkalmazáson belül. A program környezetét az alkalmazás osztály (App) végzi, amely példányosítja a modellt, a nézetmodell, a perzisztenciát és a nézetet, biztosítja a kommunikációt, valamint felügyeli az adatkezelést



- Perzisztencia:
  - Az adatkezelés feladata a játéktáblával kapcsolatos információk tárolása, valamint a betöltés és a mentés megvalósítása.
  - A GameTable osztály egy érvényes játéktáblát tárol, melynek ismert a mérete (hányszor hányas) és a mező értékei. (m, tabla[i, j]). Ezen felül tárol még 3 bool

változót (p, e1, e2) melyeknek értékei egy adott játék betöltésekor inicializálódnak attól függően, hogy van-e játékosnak / enemy1-nek / enemy2-nek megfelelő értékű mező a táblán. Ezek helyétől függően állítunk a karaktereknek megfelelő pozíciókat. (_ppos, _e1pos, _e2pos) Ezekhez készített segéd fv.-kel. (setP, setE1, setE2)

- A hosszútávú adattárolás lehetőségeit az IDataAcces interfész adja meg, amely lehetőséget ad a tábla betöltésére, (Load) valamint mentésére (SaveAsync) ez utóbbi hatékonyság miatt asszinkron módon van megolva.
- Az interfészt szöveges fájl alapú adatkezelést az DataAccess osztály valósítja meg. A fájlkezelés során fellépő hibákat a DataException kivétel jelzi.
- A program az adatokat szöveges fájlként tudja eltárolni, melyek a .txt kiterjesztést kapják. Ezeket az adatokat a programban bármikor be lehet tölteni, illetve ki lehet menteni az aktuális állást.
- A fájl első sora megadja a tábla méretét (ami alapértelmezés szerint 15). A fájl többi része izomorf leképezése a játéktáblának, azaz összesen 15 sor következik, és minden sor 15 számot tartalmaz szóközökkel választva. A számok 0 és 1 lehetnek, ahol 0 reprezentálja a szabad mezőt, 1 pedig az akna mezőt.

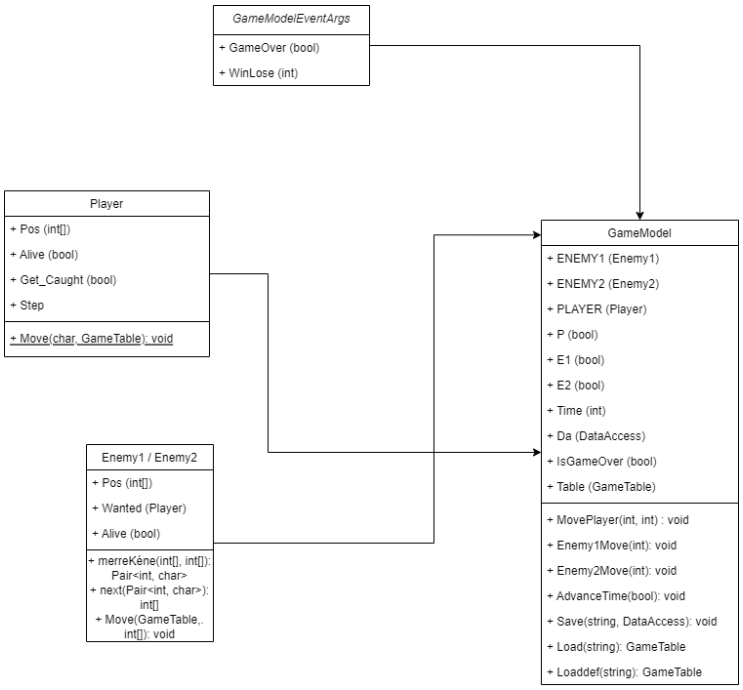


- 2.ábra: Az alkalmazás csomag diagrammja

- **Modell:**
  - A modell lényegi részét a három karakter osztály (Player, Enemy1, Enemy2) valósítja meg, (azok lépéseit) melyeket a GameModel osztály hasznosít és foglal egységbe. A három felsorolt típus metódusai a következők:
    - Player:
      - Move: a játékos mozgatása
      - Valamint a játékosnak van egy jelenlegi pozíciója: pos
      - Egy igaz / hamis értéke amely igaz, ha a játékos még él: _isAlive
      - Még egy arra, hogy elkapták-e már: _caught
      - És még egy arra, hogy éppen lép-e: _step
    - Enemy1 és Enemy2:

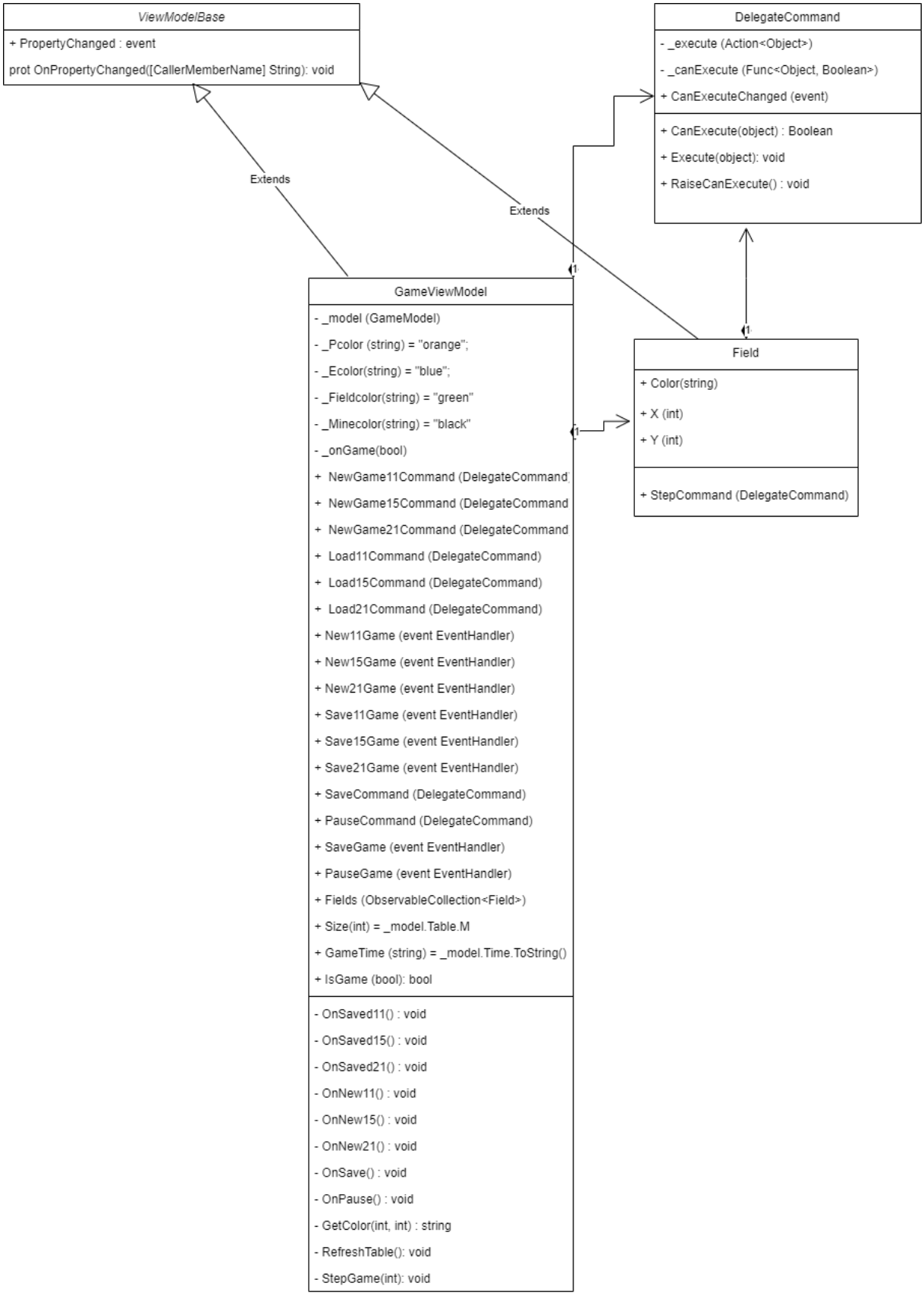
- merreKéne: egy sor/oszlop és abban pozitív/negatív irányt visszaadó metódus mely segéd fv.-e a következő metódusnak
- next: megadja a következő mezőt amerre az üldöző mozog, felhasználva az előző metódust
- Move: mozgatja az üldözőt
- Az üldözőnek van egy pozíciója: pos
- Egy játékost reprezentáló adattagja: player
- És egy igaz/hamis értéke mely igaz, ha él még az üldöző: _isAlive

- Maga a GameModel pedig mint már írtam egybefoglalja ezeket a perzisztenciával.
  - Tárol egy perzisztencia típusú adattagot: da
  - Egy játéktáblát: _table
  - Egy játékost: _player
  - Két üldözőt: _enemy1, _enemy2
  - És 3 igaz/hamis értéket, melyek értéke igaz, ha a megfelelő karakter éppen létezik: van_p, van_e1, van_e2
  - A eseményei melyekkel jelzéseket ad a View-nak:
    - GameOver: értelemszerűen akkor váltódik ki, ha vége a játéknak
    - Step: Ha a játékos lép
  - Végül a metódusok melyeket használ:
    - OnGameOver, OnStep: a megfelelő események kiváltásait szolgálják
    - Save: A perzisztencia SaveAsync fv.-ét hívja meg, és kezeli a játék aktuális állását.
    - Loaddef: A perzisztencia Load fv.-ét felhasználva tölti be az Új játékhoz felhasznált default pályákat a megfelelő méretben.
    - Load: A perzisztencia Load fv.-ét felhasználva tölt be egy elmentett játékot.
    - MovePlayer: A Player adattag Move fv.-ét használva kezeli a játékos mozgását.
    - Enemy1Move és Enemy2Move: A két üldözőt reprezentáló adattag Move fv.-eit meghívva mozgatja az üldözőket minden eltelt másodperc után.

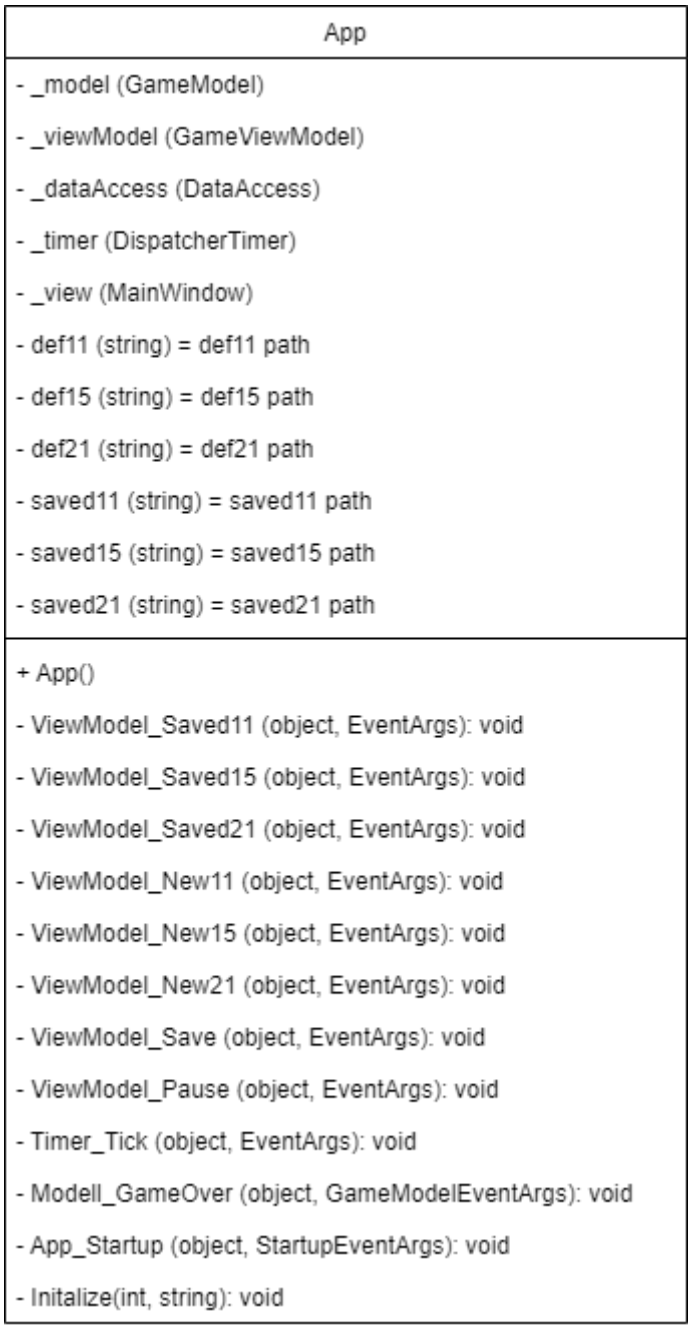


- Nézetmodell:
  - A nézetmodell megvalósításához felhasználunk egy általános utasítás (DelegateCommand), valamint egy ős változásjelző (ViewModelBase) osztályt.
  - A nézetmodell feladatait a GameViewModel osztály látja el, amely parancsokat biztosít az új játék kezdéséhez, játék betöltéséhez, mentéséhez, 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 (Field), amely eltárolja a pozíciót, színt, engedélyezettséget, valamint a lépés parancsát (StepCommand). A mezőket egy felügyelt gyűjteménybe helyezzük a nézetmodellbe (Fields).



- 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.
    - A fájlnév bekérését betöltéskor és mentéskor, valamint a figyelmeztető üzenetek megjelenését beépített dialógusablakok segítségével végezzük.
- Környezet:
    - 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.



Tesztelés:

- A modell funkcionalitása egységtesztek segítségével lett ellenőrizve a Test osztályban.
- Az alábbi tesztesetek kerültek megvalósításra:
    - Initalize11, Initalize15, Initalize21: Megfelelő méretű random játéktáblák inicializása
    - Load11, Load15, Load21: A megfelelő méretű betöltött játéktáblák Player.Pos, Enemy1.Pos, Enemy2.Pos pozíciók értékeinek összevetése a random generált táblák ugyanezen pozíciókon lévő értékeivel
    - EnemiesMove_And_Die_GameOver_Test_(11, 15, 21): Megfelelő méretű táblákon az üldözők mozgásának ellenőrzése pozíció váltás utáni érték ellenőrzésével. Majd aknára lépésük után isGameOver igaz/hamis érték igazra állítódásának ellenőrzése.
    - PlayerMove_And_PlayerDie_GameOver_Test_(11, 15, 21): Megfelelő méretű táblákon a játékos mozgásának ellenőrzése pozícióváltás utáni értékek

ellenőrzésével. Majd aknára lépése után isGameOver igaz/hamis érték igazra állítódásának ellenőrzése.