2.Beadandó feladat dokumentáció

Készítette:

Amamou Martin

W3Q74H

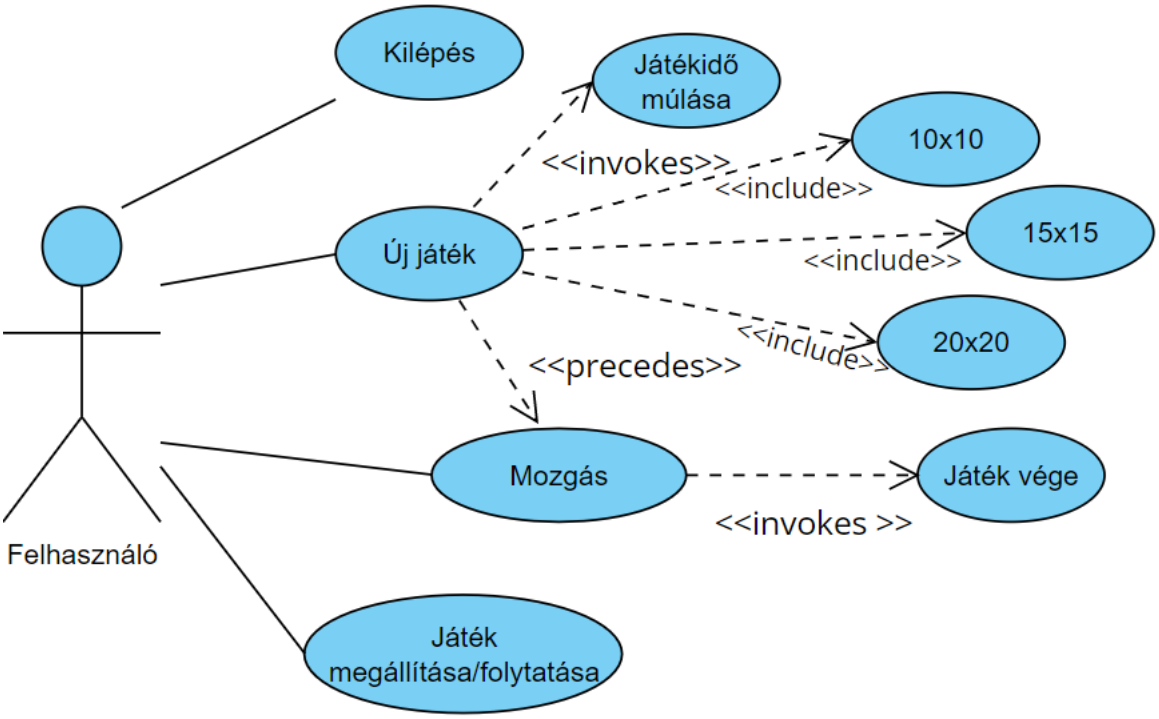
E-mail: amamoumartin@gmail.com

Feladat:

Készítsünk programot, amellyel a klasszikus kígyó játékot játszhatjuk. Adott egy $n \times n$ elemből álló játékpálya, amelyben akadályok (falak) találhatóak. A játékos egy kezdetben 5 hosszú kígyóval indul a képernyő közepén, amely vízszintesen, illetve függőlegesen halad rögzített időközönként a legutoljára beállított irányba. A kígyóval elfordulhatunk balra, illetve jobbra. A pályán véletlenszerű pozícióban mindig megjelenik egy tojás, amelyet a kígyóval meg kell etetni. Minden etetéssel eggyel nagyobb lesz a kígyó. A játék célja, hogy a kígyó minél tovább elkerülje az ütközést az akadályokkal, a pálya szélével, illetve saját magával. A pályák méretét, illetve felépítését (falak helyzete) 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 mozog a kígyó). Továbbá ismerje fel, ha vége a játéknak. Ekkor jelenítse meg, hány tojást sikerült elfogyasztania a játékosnak.

Elemzés:

- A játékot három fajta pályamérettel játszhatjuk: 10x10(6 fal), 15x15(12 fal), 20x20(18 fal)-as táblán. A program indításakor a 15x15-ös pálya töltődik be.
- A feladatot egyablakos asztali alkalmazásként Windows Presentation Foundation grafikus felülettel valósítjuk meg.
- Az ablakban elhelyezünk egy menüt a következő menüpontokkal: File (New Game(10x10,15x15,20x20),Exit) továbbá egy, az aktuális pontot mutató menüponttal. Az ablak alján megjelenítünk egy státuszsort, ami a súgót tartalmazza.
- A program indításakor a játékmentet áll, a játékot az ENTER/SPACE billentyűvel lehet szüneteltetni/újra elindítani. A játékban a kígyót a nyíl billentyűkkel, vagy a W A S D billentyűkkel lehet irányítani, ahogy az a felhasználónak kényelmesebb. A fel nyíl/W lenyomásakor a kígyó felfele, a bal nyíl/A lenyomásakor balra, a le nyíl/S lenyomásakor lefele, végül a jobb nyíl/D lenyomásakor a kígyó jobbra fog menni. A kígyó 0.5 másodpercenként mozog egy egységnyit előre a táblán.
- A játék automatikusan feldob egy dialógusablakot, amikor vége a játéknak (kiraktuk a táblát, vagy letelt az idő), illetve amikor a felhasználó ki szeretne lépni.

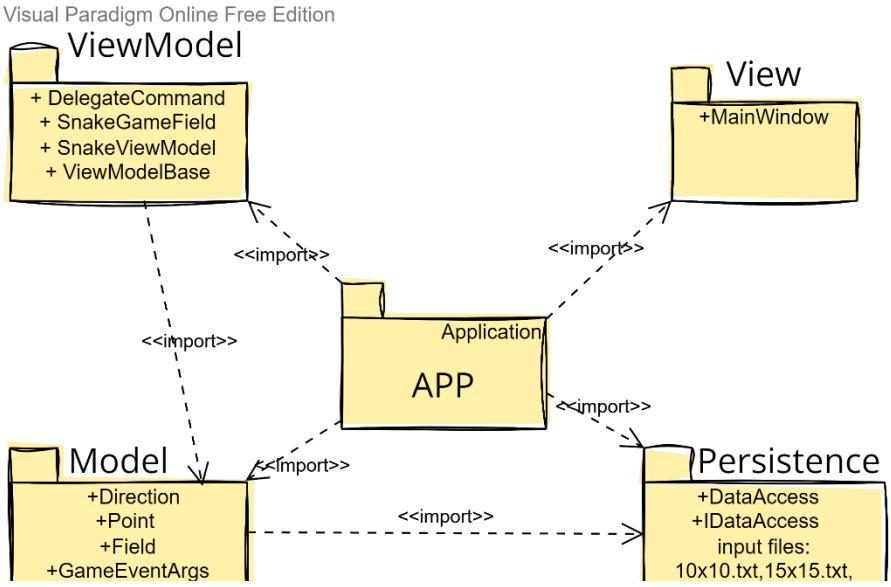


1. ábra: Felhasználói eset diagram

Tervezés

Programszerkezet:

- A programot MVVM architektúrában valósítjuk meg, ennek megfelelően View, Model, ViewModel és Persistence névtereket valósítunk meg az alkalmazáson belül.



2. ábra Az alkalmazás csomagdiagramja

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.

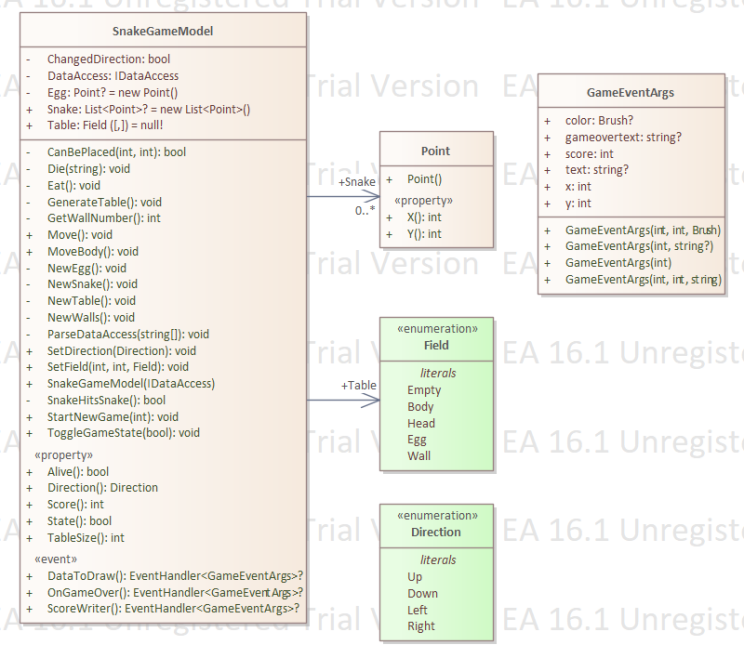
- A program szerkezetét két projektre osztjuk implementációs megfontolásból: a Persistence és Model csomagok a program felületfüggetlen projektjében, míg a ViewModel és View csomagok a WPF függő projektjében kapnak helyet.

Modell:

- A modell fő részét a ***SnakeGameModel*** valósítja meg, a tábla jelentősebb tevékenységeit kezeli, szabályozza a paramétereket: táblaméret(***TableSize***), pontszám(***Score***), irány(***Direction***), stb.
- Lehetőséget ad új játék kezdésére (***StartNewGame***)
- A modell példányosításkor az adatkezelést megkapja, ennek segítségével átveszi az inputfájl tartalmát, ez alapján inicializálja a táblát(***ParseDataAccess***)
- Tartalmazza a játéktáblát, mely egy játékmező(***Field***) felsorolási típusból álló mátrix.
- Létrehozza az elemeket(tábla, kígyó, tojás, falak) (***GenerateTable***)
- Továbbá a kígyó reprezentációját(pontokból álló lista), és folyamatait: mozgását(***MoveSnake***), étkezését(***Eat***), elpusztulását(***Die***) implementálja, emellett az ezeket biztosító ellenőrző- és segédfüggvényeket tárolja.

Eseménykezelés:

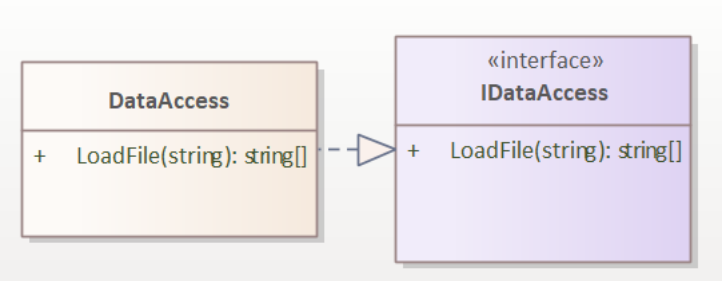
- A játékállapot változásáról a ***ScoreWriter*** esemény tájékoztat, mely az aktuális pontszámot írja ki.
- A tábla kirajzolásához a model a ***DataToDraw*** eseményt használja.
- A játék végét az ***OnGameOver*** esemény jelzi.
- Az ezekhez tartozó adatokat az események argumentuma (***GameEventArgs***) tárolja.



3. ábra: A modell csomag osztálydiagramja

Persistence:

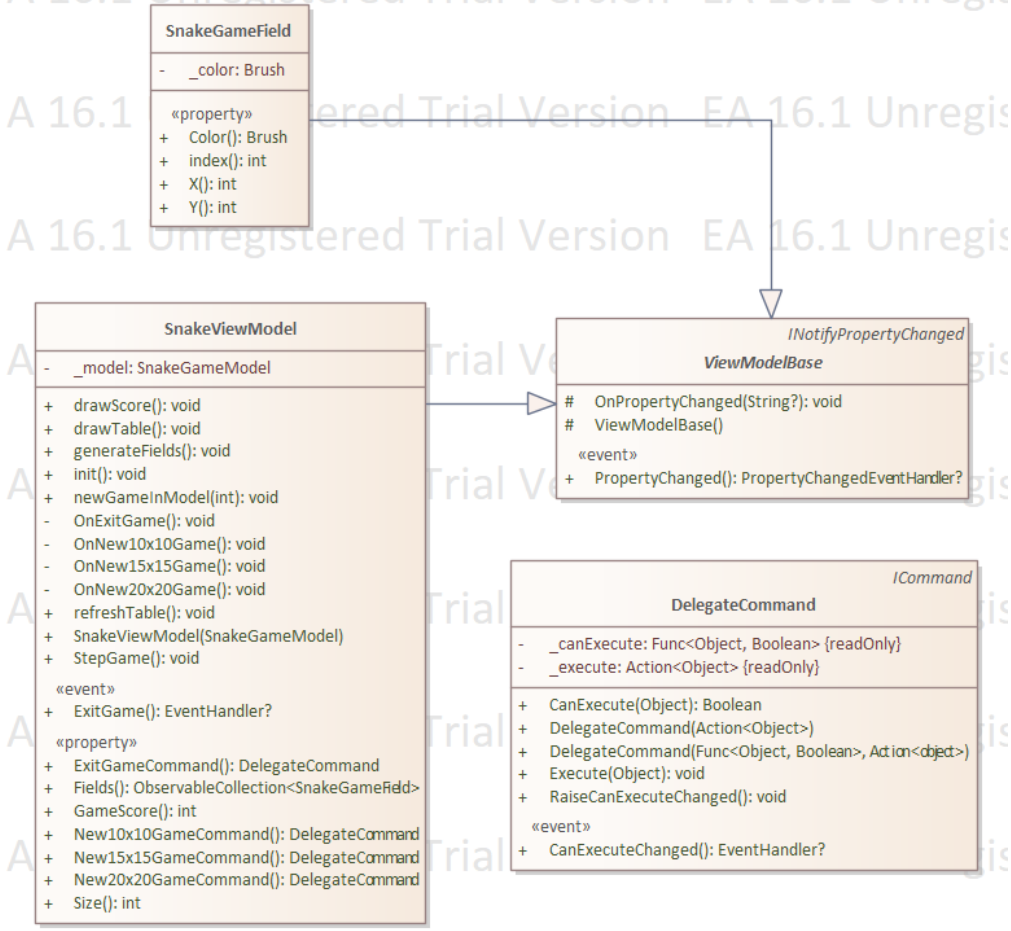
- A játék ezen része a megadott pályák beolvasásával foglalkozik, melyet a modell megkap példányosításkor, hogy beállítsa a tábla értékeit. Az interfészt szöveges fájlkezelésre a ***DataAccess*** osztály valósítja meg, ez az ***IDataAccess***-ből szárzmazik. Lehetőséget ad egy paraméteren keresztül megadott útvonalon egy fájlt beolvasni + adatait eltárolni.
- A programhoz három darab előre elkészített, nem változtatható szöveges fájl kapcsolódik, melyek a 3 különböző pálya adatait tartalmazzák.
- A fájl n sorból, n oszlopból áll, ahol n a tábla mérete. Csupán 0,1,2,3,4-eket tartalmaz, a játékmező típusától függően:
  0- üres mező
  1- a kígyó teste
  2- a kígyó feje
  3- fal
  4- tojás



4. ábra: A Persistence csomag osztálydiagramja

Nézetmodell:

- A nézetmodell megvalósításához felhasználunk egy általános utasítás (***DelegateCommand***), valamint egy ős változásjelző (***ViewModelBase***) osztályt.
- A nézetmodell feladatait a ***SnakeViewModel*** osztály látja el, amely parancsokat biztosít az új játék kezdéséhez (ezáltal a nehézség kiválasztásához), továbbá 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 játék léptetését hívja meg a modellen.
- A játékmező számára egy külön mezőt biztosítunk (***SnakeGameField***), amely eltárolja a koordinátákat (***x, y***), a lineáris pozícióját(***index***), továbbá a mező színét(***Color***). A mezőket egy felügyelt gyűjteménybe helyezzük a nézetmodellbe (***Fields***).



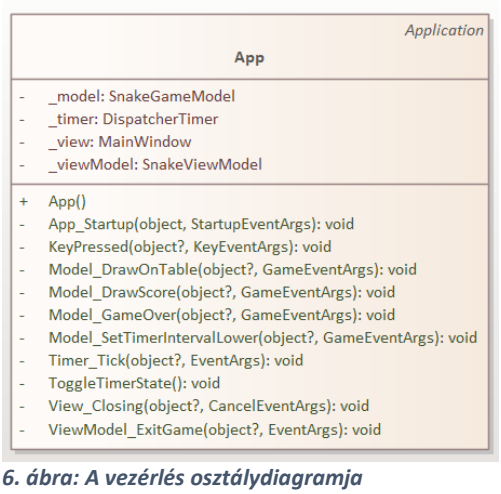
5. ábra: A nézetmodell osztálydiagramja

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:

- Az ***App*** osztály feladata az egyes rétegek példányosítása (App_Startup), összekötése, a nézetmodell, valamint a modell eseményeinek lekezelése, 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

Tesztelés

A modell funkcionalitása egységtesztek segítségével lett leellenőrizve a *SnakeGameTest* osztályban, az alábbi tesztesetekkel:

- `TenBoardTestMethod, FifteenBoardTestMethod, TwentyBoardTestMethod` ellenőrzik a 10x10-es, 15x15ös, 20x20-as pályákon a tábla megfelelő mezőinek típusát.
- `EatTest`:
  azt vizsgálja, hogy a kígyó elfogyassza-e a tojást → ezáltal növekszik a pontszáma, és a testhossza
- `WallCollisionTest`:
  azt teszteli, hogy valóban vége lesz-e a játéknak, ha a kígyót hagyjuk falnak menni
- `BorderCollisionTest`:
  azt teszteli, hogy valóban vége lesz-e a játéknak, ha a kígyót hagyjuk a pálya szélének menni
- `SnakeCollisionTest`:
  azt teszteli, hogy valóban vége lesz-e a játéknak, ha a kígyót hagyjuk önmagának nekimenni