2. Beadandó feladat dokumentáció

Készítette:
Kunhalmi Attila
E-mail: bw7of5@inf.elte.hu

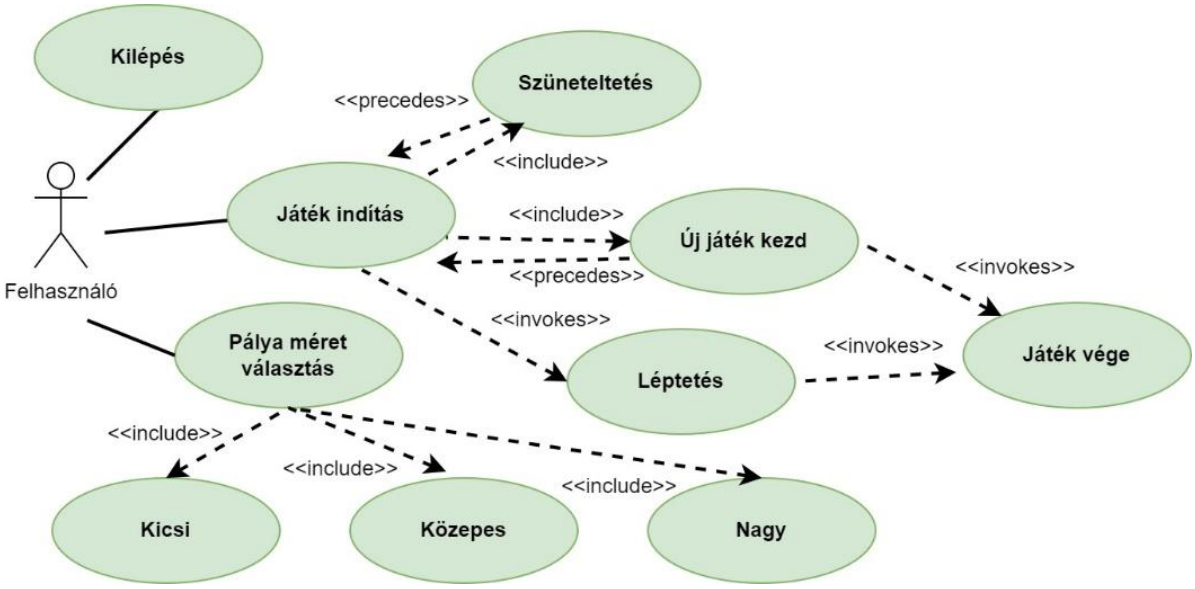
Feladat:

Készítsük 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 pályaméreten játszhatjuk: ***Small*** (ez 3db legenerált akadályt tartalmaz), ***Medium*** (ez 8db legenerált akadályt tartalmaz), ***Large*** (ez 2db legenerált akadályt tartalmaz). A program indításkor Large pályaméret kerül alapértelmezett beállításra.
- A feladatot egyablakos asztali alkalmazásként WPF grafikus felülettel valósítjuk meg.
- Az ablakban elhelyezünk egy menüt a következő menüpontokkal: ***Size setting*** (Small, Medium, Large). Az ablak alján megjelenítünk két nyomógombot, amivel a játék elindítását, szüneteltetését, illetve újraindítását végezhetjük. Továbbá egy címke mutatja az összes megszerezett pontot és az eddig maximális elért pontját a játékosnak. Új játék indításakor az aktuálisan szerzett pontok nullázódnak.
- A játéktáblán a navigációs nyomógombok (w-s-a-d) leütésének hatására a kígyó megváltoztatja a pozícióját. A táblán előre véletlenszerű helyeken generált tojások jelennek meg egyesével és ezek pozícióját nem engedjük megváltoztatni, amíg a kígyó meg nem ette őket. Ha a kígyó az akadályok valamelyikébe vagy a játéktábla falába ütközik a játék véget ér.
- A játék automatikusan feldob egy dialógusablakot, amikor vége a játéknak (akadályba ütköztünk vagy saját magunkba haraptunk a kígyóval). Szintén dialógusablakokkal jelezzük a pályaváltoztatás tényét.

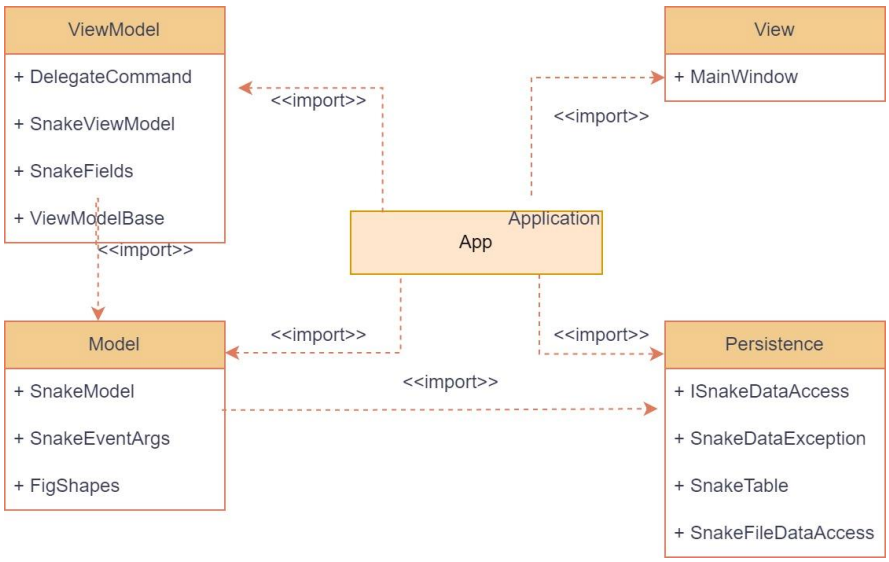
A felhasználói esetek az 1. ábrán láthatóak:



1. ábra: Felhasználói esetdiagram

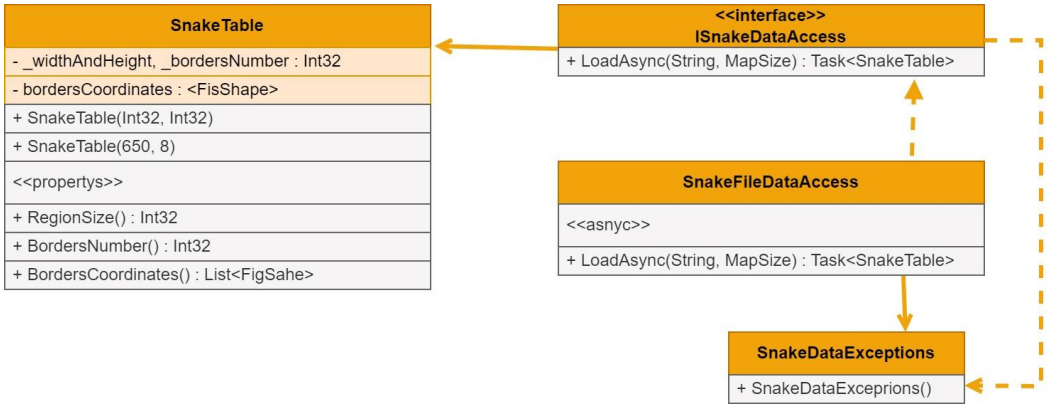
Tervezés:

- Programszerkezet:
  - A programot MVVM architektúrában valósíjuk 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 ViewModel és View csomagok a WPF függő projektjében kapnak helyet.



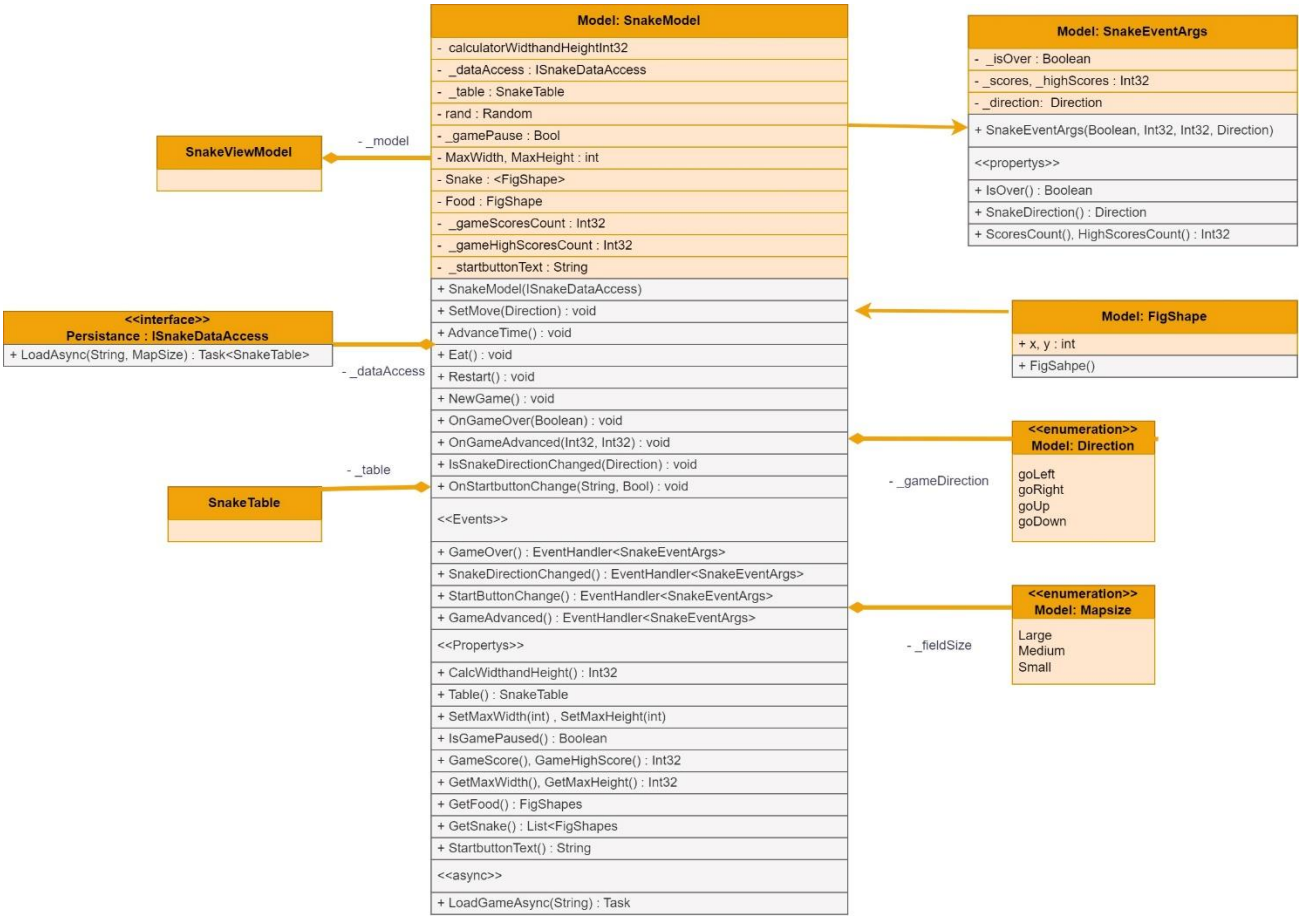
2. ábra: Alkalmazás csomagdiagramja

- Perzisztencia:
  - Az adatkezelés feladata a SnakeGame táblával kapcsolatos információk tárolása, valamint a játék betöltés biztosítása.
  - A SnakeTable osztály egy érvényes Snake játéktáblát biztosít, ami ellenőrzi a megadható maximális és minimális pályaméret nagyságát, valamint az érvényes koordinátákat az akadályok számára. A játéktábla alapértelmezés szerint n × n-es, de a játék indításakor mindig a legnagyobb megjeleníthető pályaméret fogad. A tábla lehetőséget az állapotok lekérdezésére (RegionSize, BordersNumber, BoordersCoordinate).
  - A fájlkezelés során fellépő hibákat a SnakeDataException kivétel jelzi. A program az adatokat szöveges fájlként tudja eltárolni, melyek az txt kiterjesztést kapják. Ezeket az adatokat a programban bármikor be lehet tölteni.



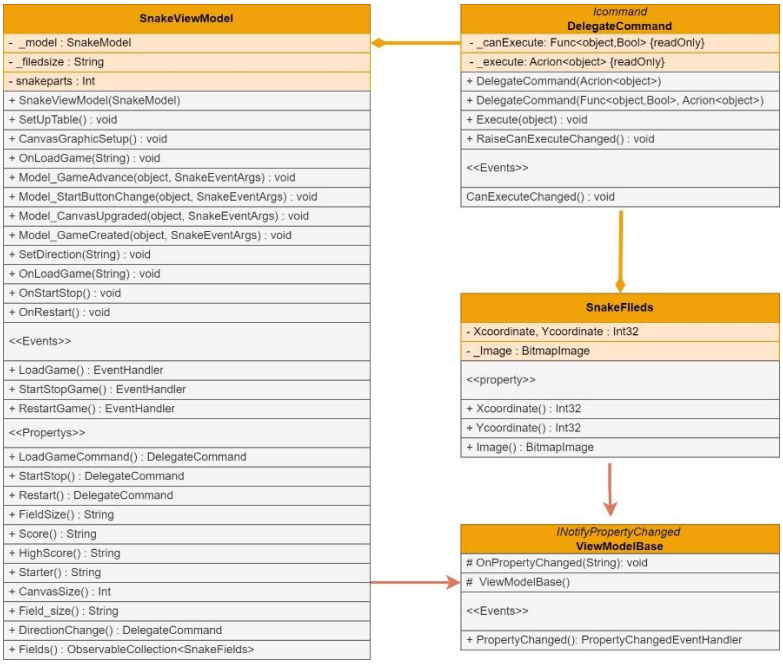
3. ábra: Perzisztencia csomag osztálydiagramja

- Modell:
  - A modell lényegi részét a SnakeModel osztály valósítja meg, amely szabályozza a tábla tevékenységeit, valamint a játék egyéb paramétereit, úgymint az kígyó mozgásának irány (_direction). A típus lehetőséget ad új játék kezdésére (NewGame), ekkor csak Score pontjaink nullázódnak, valamint a játék korábbi szerzett HighScore pontjainak törlésével egy teljesen újbóli játékkezdésre (ResetGame). Új játéknál megadható a kiinduló játéktábla mérete is, különben automatikusan generálódik a legnagyobb pályaméretre. Az idő előre léptetését időbeli lépések végzéséve (AdvanceTime) tehetjük meg.
  - A kígyó mezőn való állapotváltozásáról a (SetMove()) metódus tájékoztat.
  - A kígyó tojásevését az Eat() metódus valósítja meg esemény, míg a játék végéről a GameOver változó tájékoztat.



4. ábra: Modell csomag osztálydiagramja

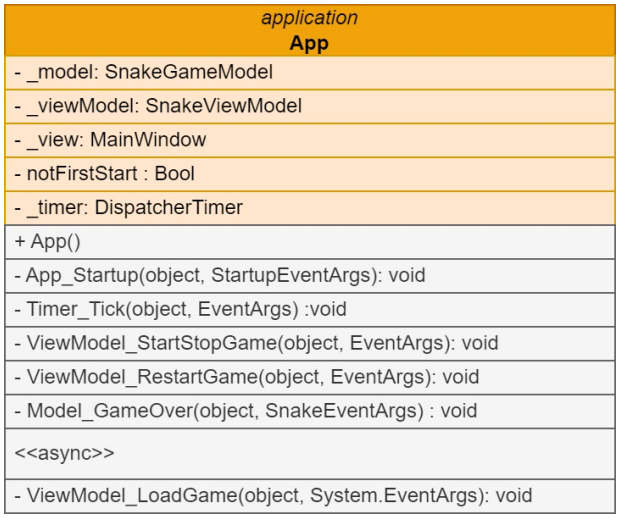
- Nézetmodell:
  - A nézetmodell megvalósításához felhasználunk egy általános utasítás (DelegateCommand), valamint egy ős változásjelző (ViewModelBase) osztályt.
  - A nézetmodell feladatait a SnakeViewModel osztály látja el, amely parancsokat biztosít az új játék kezdéséhez, játék betöltéséhez, valamint a játék újrakezdéséhez.
  - 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 pályaméretet szabályozza. Direkt nem avatkozik a játék futtatásába.



5. ábra: Nézetmodell osztálydiagramja

- Nézet:
    - A nézet csak egy képernyőt tartalmaz, a MainWindow osztályt. A nézet egy canvas felületen jeleníti meg a játékmezőt, a grafikai elemekkel együtt. Minden adatot adatkötéssel kapcsolunk a felülethez, továbbá azon keresztül szabályozzuk a gombok funkcióit.
    - A játék újrakezdésekor és méretváltoztatás során, 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.



6. ábra: Vezérlés osztálydiagramja

Tesztelés:

- A modell funkcionalitása egységtesztek segítségével lett ellenőrizve a SnakeGameTest osztályban.
- Az alábbi tesztesetek kerültek megvalósításra:
    - `SnakeModelNewGameTest`: Pályaméret beállításainak érvényesítése.
    - `SnakeEatTest`: A kígyó etetését szimulálja, hogyan változik a kígyó mérete és pontok száma.
    - `SnakeGameOver`: A játék végét imitálja, mikor a kígyó feje adott akadályba ütközik a pályán.