Harmadik beadandó feladat dokumentácó

Készítette:

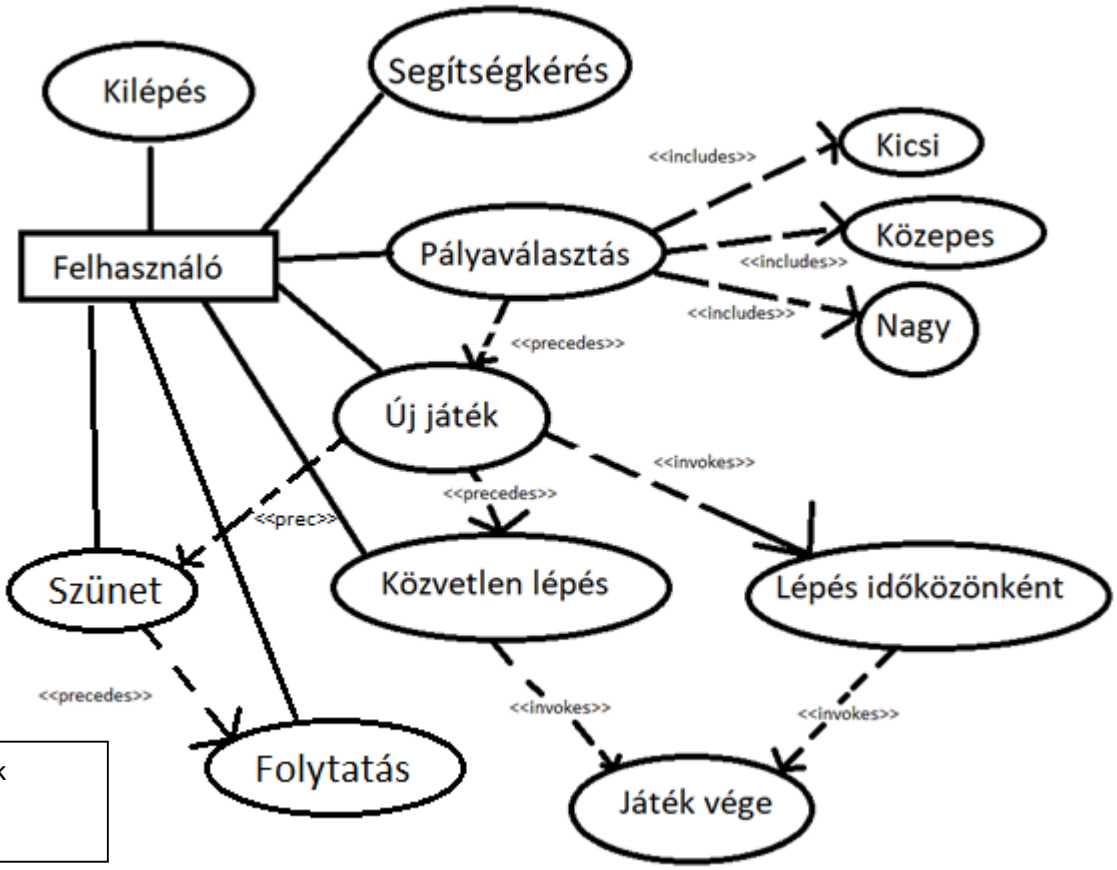
Név: Hertendi Áron Levente
E-mail: gjqd64@inf.elte.hu
Neptun-kód: GJQD64

Feladat: 11, Snake

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 játéknak három különböző méretű és elrendezésű pályára lesz szüksége. Erre alkalmas lesz 3 statikusan előkészített fájl, mely tartalmazza a pályák elrendezését, ez maradhat a kódban.
- A feladatot Xamarin alkalmazásként, Android platformon grafikus felülettel valósítom meg.
- Elhelyezek az ablakban egy gombrácsot a játék állásának vizuális reprezentációjának céljából, illetve alá egy vezérlőpanelt, ahol a felhasználó kiválaszthatja a 3 pálya egyikét, megtekintheti a Help menüt, illetve új játékot kezdhet. A felhasználó számára nem elérhető ez a vezérlőpanel játék közben.
- A grafikus felület statikus méretű, a rajzon dinamikusan változtatjuk a méretarányokat a pálya méretétől függően.
- A játék végét Alert jelzi az elért eredmény megjelenítésével.
- A kígyó rögzített időközönként mozogni fog a beállított irányba, de a fordulásra, illetve erőltetett lépésre azonnal reagál.



Az ábrán a felhasználói esetek láthatóak.

Tervezés:

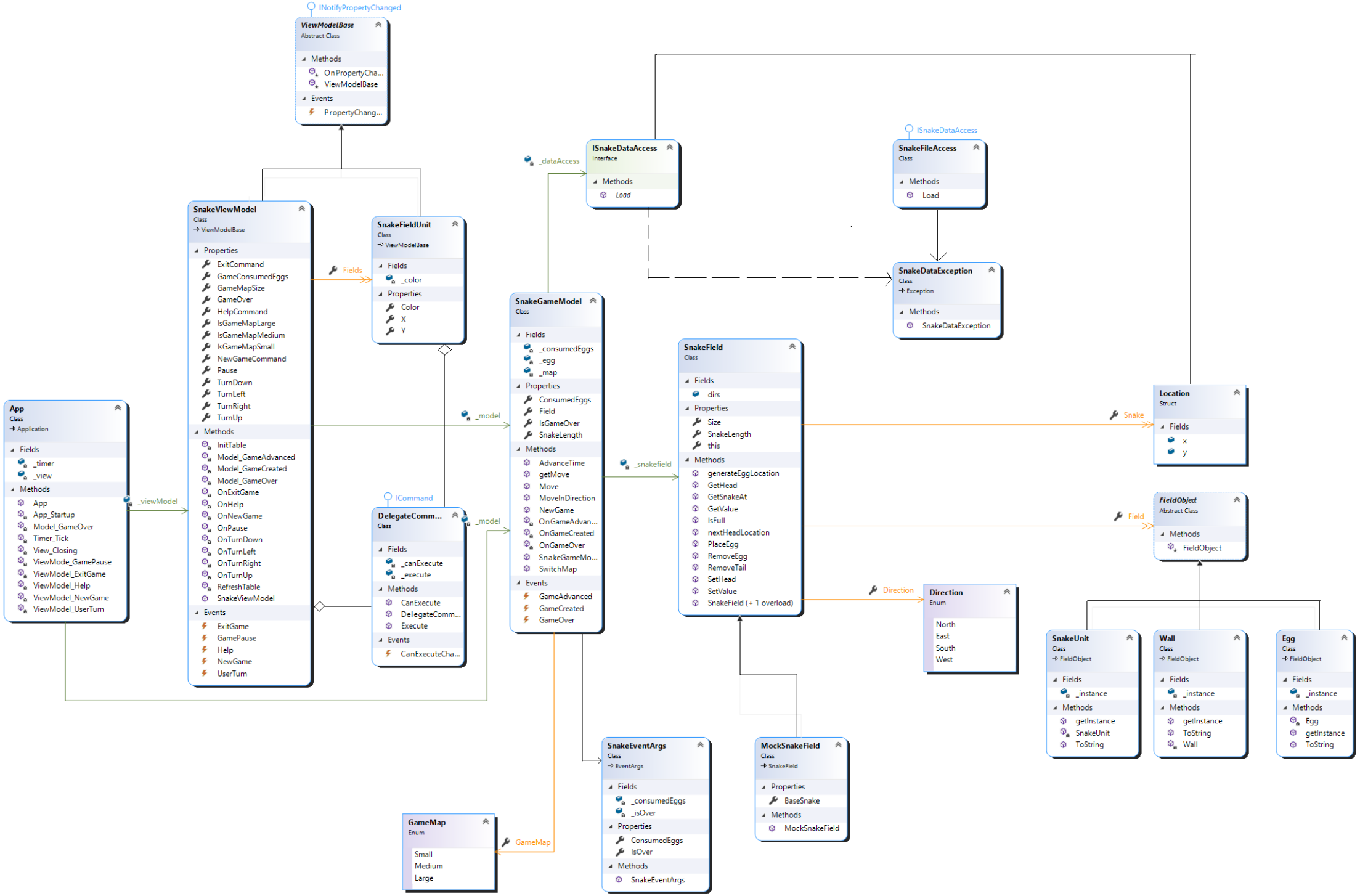
- Programszerkezet:
  - A programot MVVM architektúrában valósítjuk meg, ennek megfelelően View, Model, ViewModel és Persistence névtereket valósítunk meg az alkalmazáson belül. A program környezetét az alkalmazás osztály (App) végzi, amely példányosítja a modellt, a nézetmodell és a nézetet, biztosítja a kommunikációt, valamint felügyeli az adatkezelést.
- Perzisztencia:
  - Az adatkezelés feladata a pálya méretének és a falak helyzetének meghatározása
  - A FieldObject absztrakt osztály három gyermekével (Wall, Egg, SnakeUnit) definiálja a játék egyes elemeit reprezántáló objektumokat.
  - Ezt használva a SnakeFileAccess könnyedén át tud adni a modellnek egy, a falak helyzetét reprezentáló mátrixot egy megfelelő formátumú fájlból a Load metódus definiálásával, megvalósítva az ISnakeDataAccess interfészt.
  - Helytelen formátum, vagy nem található/foglalt fájl esetén SnakeDataException hibát dob a Load metódus.
  - A helyes formátumú fájl első sora a tábla méretét tartlamazza (>=10), második sora a falak számát (>=0), majd ezután soronként egy fal x és y koordinátáját szóközzel elválasztva.
  - A SnakeFileAccess az Android csomagban kerül definiálásra, függőségként.
- Modell:
  - Mielőtt a program lényegi funkcionalitását definiáljuk, meghatározunk egy segéd típust, ami egy kígyó és a hozzá tartozó pályának az uniója. Ez a típus tartalmazza egy dinamikus hosszúságú listában a kígyó koordinátáit, sorban, a fejétől hátrafelé, illetve a hozzá tartozó pályában ezzel a listával szinkronban reprezentáljuk a kígyó helyzetét, csakúgy a tojását és a falakét. Ennek a pályának szolgál alapul a korábban beolvasott, falakat tartalmazó mátrix, melyet könnyű lesz a nézetnek elkérnie kirajzolásra. Ez a típus (SnakeField) tartalmazza még a játékból adódó funkciókat és lekérdezéseket, melyeket gyakran használnánk. Például a kígyó elejének lekérése (SetHead), a végének eltávolítása (RemoveTail), vagy segédfunkciók a következő pozíció kiszámításához (nextHeadLocation), véletlen tojásgenerálás, stb… A kígyó aktuális irányát a Direction enumeráció segítségével tartjuk számon.
  - A játék magja, a SnakeGameModel, csak eseményekkel kommunikál a nézettel, két esemény van jelen, az egyik egy sikeres lépést (GameAdvanced), a másik pedig a játék végét jelenti (GameOver). A pályaválasztáskor a (SwitchMap) metódus kerül meghívásra a megfelelő GameMap típusú elemmel, és a felhasználói eseteken is látható főbb esémenyek NewGame(Start gomb), AdvanceGame(Időzített lépés), TurnInDirection(Közvetlen lépés) is itt kerülnek definiálásra.
  - Az a SnakeEventArgs-ok magukkal hordozzák az elfogyasztott tojások számát, illetve a játék végének állapotát.

- Nézet-Modell:
    - 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, 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 (SnakeFieldUnit), amely eltárolja a pozíciót, illetve a színt (A nézetmodell feladata meghatározni az aktuális színt az objektum alapján, de a FieldObject toString metódusával ez lekérdezhető.). 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, és egy külön DockPanel vezérlőben a technikai funkciókat. 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, 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.

  A teljes osztálydiagram a következő oldalon, illetve a csatolt fájlban tekinthető meg, továbbá a kód projektjében lehet mélyebb interaktív betekintést szerezni.

Tervezés:

A Modellben található SnakeField illetve SnakeGameModel osztályok funkcionalitása teljeskörűen, a privát event invoke metódusok kivételével minden publikus eljárás és a játék közbeni típusinvariánsok egységtesztek segítségével vannak ellenőrizve a SnakeTests csomagban (SnakeFieldTests, SnakeModelTests).


INotifyPropertyChanged
ViewModelBase
Abstract Class
Methods
OnPropertyCha...
ViewModelBase
Events
PropertyChang...
ISnakeDataAccess
SnakeFileAccess
Class
Methods
Load
_dataAccess
ISnakeDataAccess
Interface
Methods
Load
SnakeDataException
Class
Exception
Methods
SnakeDataException
SnakeViewModel
Class
ViewModelBase
Properties
ExitCommand
GameConsumedEggs
GameMapSize
GameOver
HelpCommand
IsGameMapLarge
IsGameMapMedium
IsGameMapSmall
NewGameCommand
Pause
TurnDown
TurnLeft
TurnRight
TurnUp
Methods
InitTable
Model_GameAdvanced
Model_GameCreated
Model_GameOver
OnExitGame
OnHelp
OnNewGame
OnPause
OnTurnDown
OnTurnLeft
OnTurnRight
OnTurnUp
RefreshTable
SnakeViewModel
Events
ExitGame
GamePause
Help
NewGame
UserTurn
Fields
SnakeFieldUnit
Class
ViewModelBase
Fields
_color
Properties
Color
X
Y
SnakeGameModel
Class
Fields
_consumedEggs
_egg
_map
Properties
ConsumedEggs
Field
IsGameOver
SnakeLength
Methods
AdvanceTime
getMove
Move
MoveInDirection
NewGame
OnGameAdvan...
OnGameCreated
OnGameOver
SnakeGameMo...
SwitchMap
Events
GameAdvanced
GameCreated
GameOver
SnakeField
Class
Fields
dirs
Properties
Size
SnakeLength
this
Methods
generateEggLocation
GetHead
GetSnakeAt
GetValue
IsFull
nextHeadLocation
PlaceEgg
RemoveEgg
RemoveTail
SetHead
SetValue
SnakeField (+ 1 overload)
Location
Struct
Fields
x
y
Snake
FieldObject
Abstract Class
Methods
FieldObject
Field
Direction
Enum
North
East
South
West
Direction
App
Class
Application
Fields
_timer
_view
Methods
App
App_Startup
Model_GameOver
Timer_Tick
View_Closing
ViewMode_GamePause
ViewModel_ExitGame
ViewModel_Help
ViewModel_NewGame
ViewModel_UserTurn
_viewModel
ICommand
DelegateComm...
Class
Fields
_canExecute
_execute
Methods
CanExecute
DelegateComm...
Execute
Events
CanExecuteCha...
_model
_model
_snakefield
SnakeUnit
Class
FieldObject
Fields
_instance
Methods
getInstance
SnakeUnit
ToString
Wall
Class
FieldObject
Fields
_instance
Methods
getInstance
ToString
Wall
Egg
Class
FieldObject
Fields
_instance
Methods
Egg
getInstance
ToString
SnakeEventArgs
Class
EventArgs
Fields
_consumedEggs
_isOver
Properties
ConsumedEggs
IsOver
Methods
SnakeEventArgs
MockSnakeField
Class
SnakeField
Properties
BaseSnake
Methods
MockSnakeField
GameMap
Enum
Small
Medium
Large
GameMap