

2.Beadandó feladat

Készítette:

Csóka Máté

E-mail: gt9bq9@inf.elte.hu

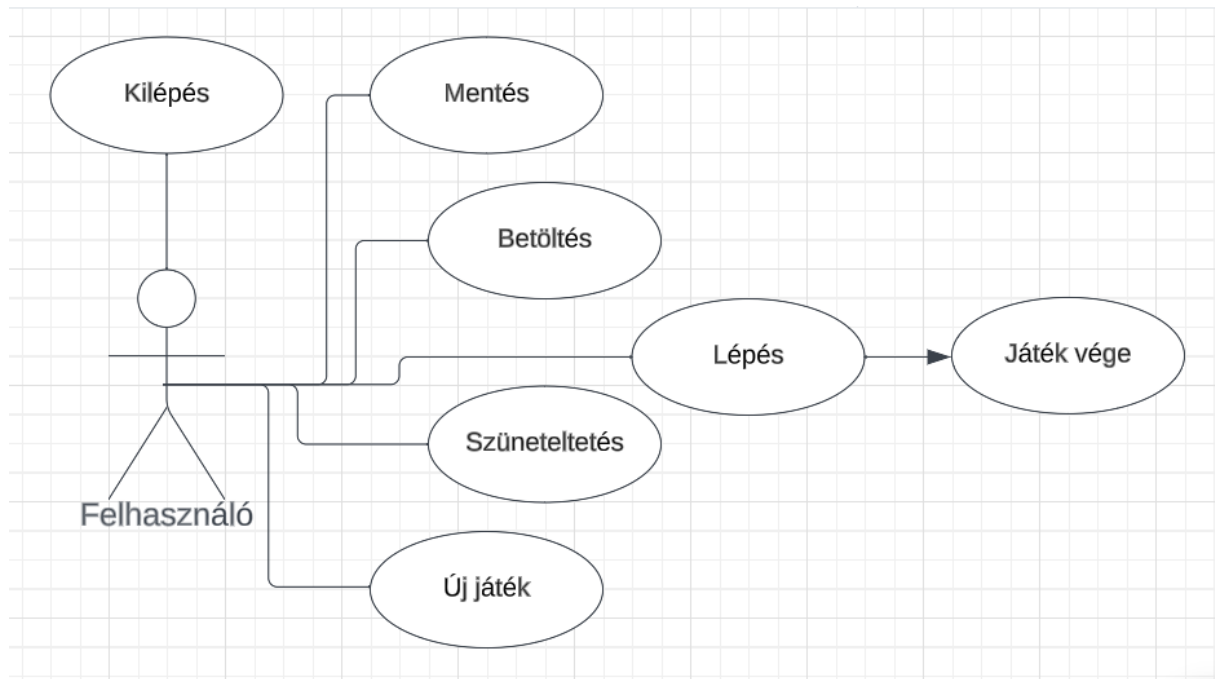
Feladat:

Készítsünk programot, amellyel az alábbi motoros játékot játszhatjuk. A feladatunk, hogy egy gyorsuló motorral minél tovább tudjunk haladni. A gyorsuláshoz a motor üzemanyagot fogyaszt, egyre többet. Adott egy kezdeti mennyiség, amelyet a játék során üzemanyagcellák felvételével tudunk növelni. A motorral a képernyő alsó sorában tudunk balra, illetve jobbra navigálni. A képernyő felső sorában meghatározott időközönként véletlenszerű pozícióban jelennek meg üzemanyagcellák, amelyek folyamatosan közelednek a képernyő alja felé. Mivel a motor gyorsul, ezért a cellák egyre gyorsabban fognak közeledni, és mivel a motor oldalazó sebessége nem változik, idővel egyre nehezebb lesz felvenni őket, így egyszer biztosan kifogyunk üzemanyagból. A játék célja az, hogy a kifogyás minél később következzen be. A program biztosítson lehetőséget új játék kezdésére, valamint játék szüneteltetésére (ekkor nem telik az idő, és nem mozog semmi a játékban). Ismerje fel, ha vége a játéknak, és jelenítse meg, mennyi volt a játékidő. Ezen felül szüneteltetés alatt legyen lehetőség a játék elmentésére, valamint betöltésére.

Elemzés:

- A program indításkor automatikusan új játékot indít, szüneteltetett állapotban.
- A feladatot egyablakos asztali alkalmazásként Windows Forms grafikus felülettel valósítjuk meg.
- Az ablakban elhelyezünk egy menüt a következő menüpontokkal: New Game, Start, Pause, Save, Load. Az ablak alján megjelenítünk egy státuszsort, amely az üzemanyagszintet, illetve a hátralévő időt jelzi.
- A játéktáblát egy 13x13 gombokból álló rács reprezentálja. A rács alsó sorában van a motor (feketével jelzett mező), ezt irányíthatja a játékos a nyilak megnyomásával. A rács felső sorában véletlenszerűen jelennek meg üzemanyagcellák (pirossal jelzett mező), egyszerre csak egy és erre is csak 30% esély van. Az üzemanyagcellák egyre jobban gyorsulva esnek lefelé. Az üzemanyagszint folyamatosan csökken, az idő múlásával egyre gyorsabban. Ha az üzemanyagcella, érintkezik a motor cellával, akkor az üzemanyagszinthez, +5 egység adódik. Maximum üzemanyagszint 50 egység lehet.
- A játék automatikusan kidob egy dialógusablakot, amikor vége a játéknak (elfogyott az üzemanyag). Szintén dialógusablakkal végezzük el a mentés, betöltést.

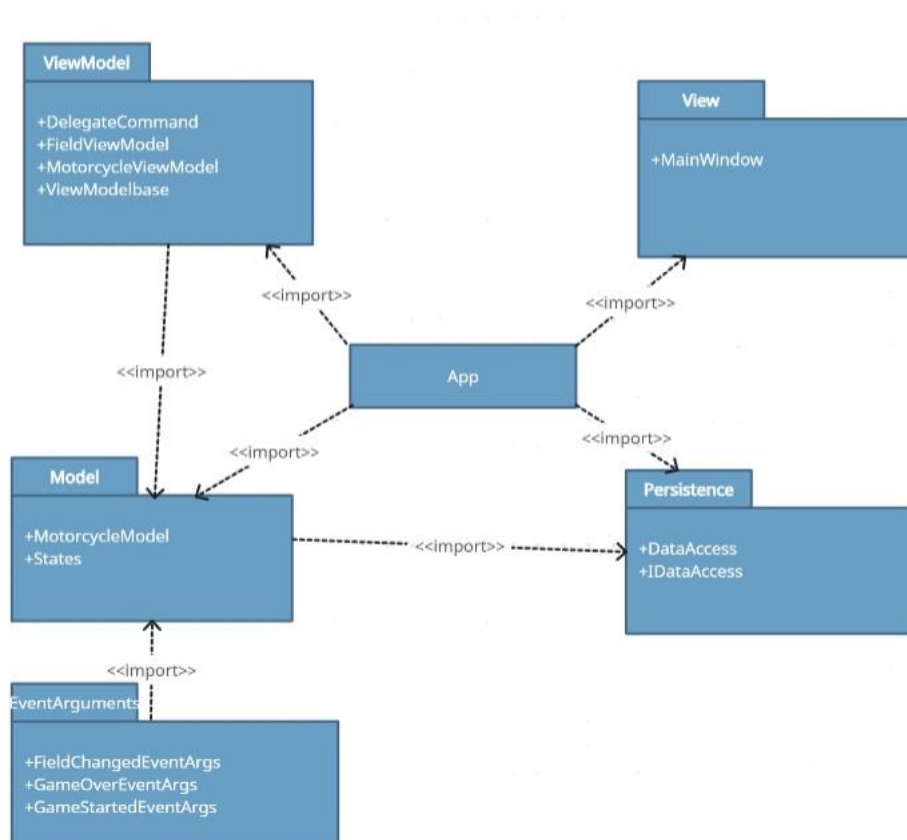
- A felhasználói esetek az 1. ábrán láthatóak.



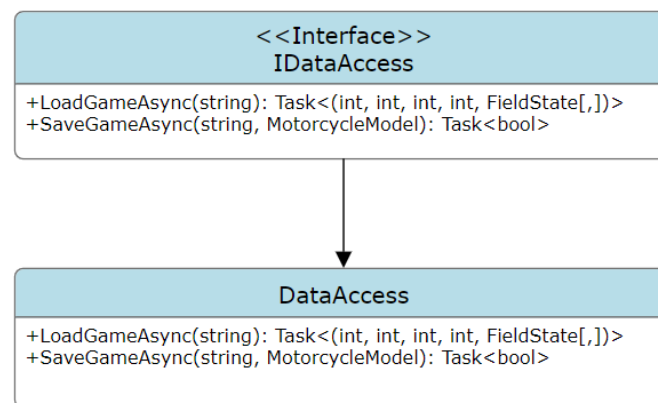
1.ábra: Felhasználói esetek diagramja

Tervezés:

- Programszerkezet:
 - A programot MVVM architektúrában valósítjuk meg, ennek megfelelően **View**, **Model**, **ViewModel** és **Persistence** névterekét valósítunk meg az alkalmazáson belül. A program környezetét az alkalmazás osztály (**App**) végzi, amely példányosítja a modellt, a nézetmodellt és a nézetet, biztosítja a kommunikációt, valamint felügyeli az adatkezelést. A program csomagszerkezete a 2. ábrán látható.
 - A **Persistence** és **Model** csomagok felületfüggetlenek, míg a **ViewModel** és **View** csomagok WPF függőek. Egy projekten belül valósítottam meg őket.
- Perzisztencia (3. ábra):
 - Az adatkezelés feladata a Motorcycle táblával kapcsolatos információk tárolása, valamint a betöltés/mentés biztosítása.
 - A hosszú távú adattárolás lehetőségeit az **IDataAccess** interfész adja meg, amely lehetőséget ad a tábla betöltésére (**LoadGame**), valamint mentésére (**SaveGame**).
 - Az interfész szöveges fájl alapú adatkezelésre a DataAccess osztály valósítja meg.
 - A program az adatokat szöveges fájlként tudja eltárolni. Ezeket az adatokat a programban bármikor be lehet tölteni, illetve ki lehet menteni az aktuális állást.
 - A fájl első sora megadja a tábla méretét. A második sor az időt, majd sorban következik: az üzemenyagszint, a sebesség, majd a mátrix értékei.

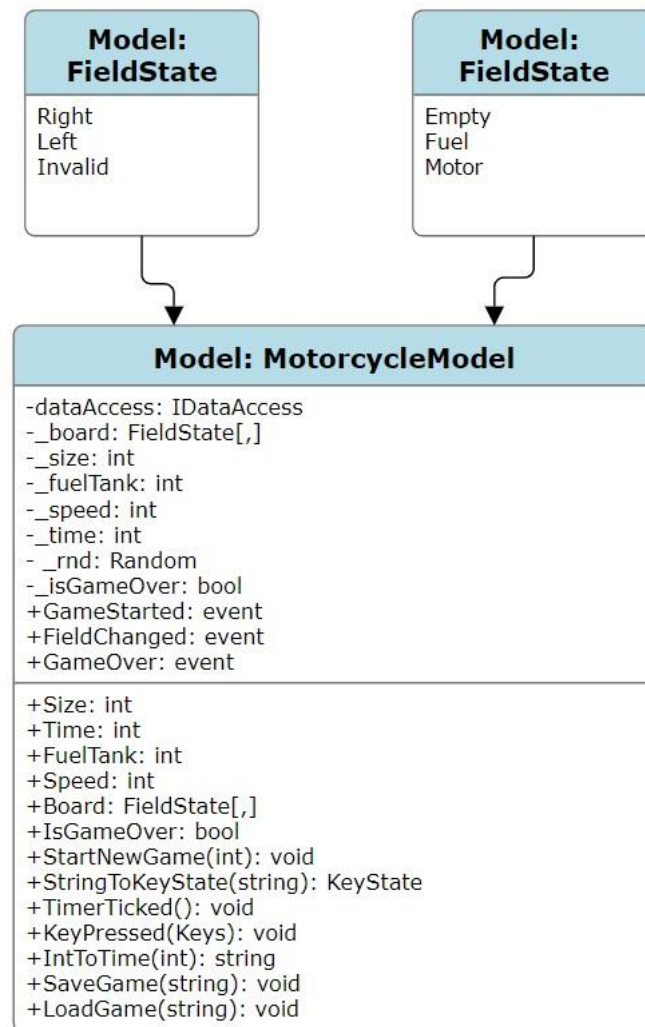


2.ábra: Az alkalmazás csomagdiagramja



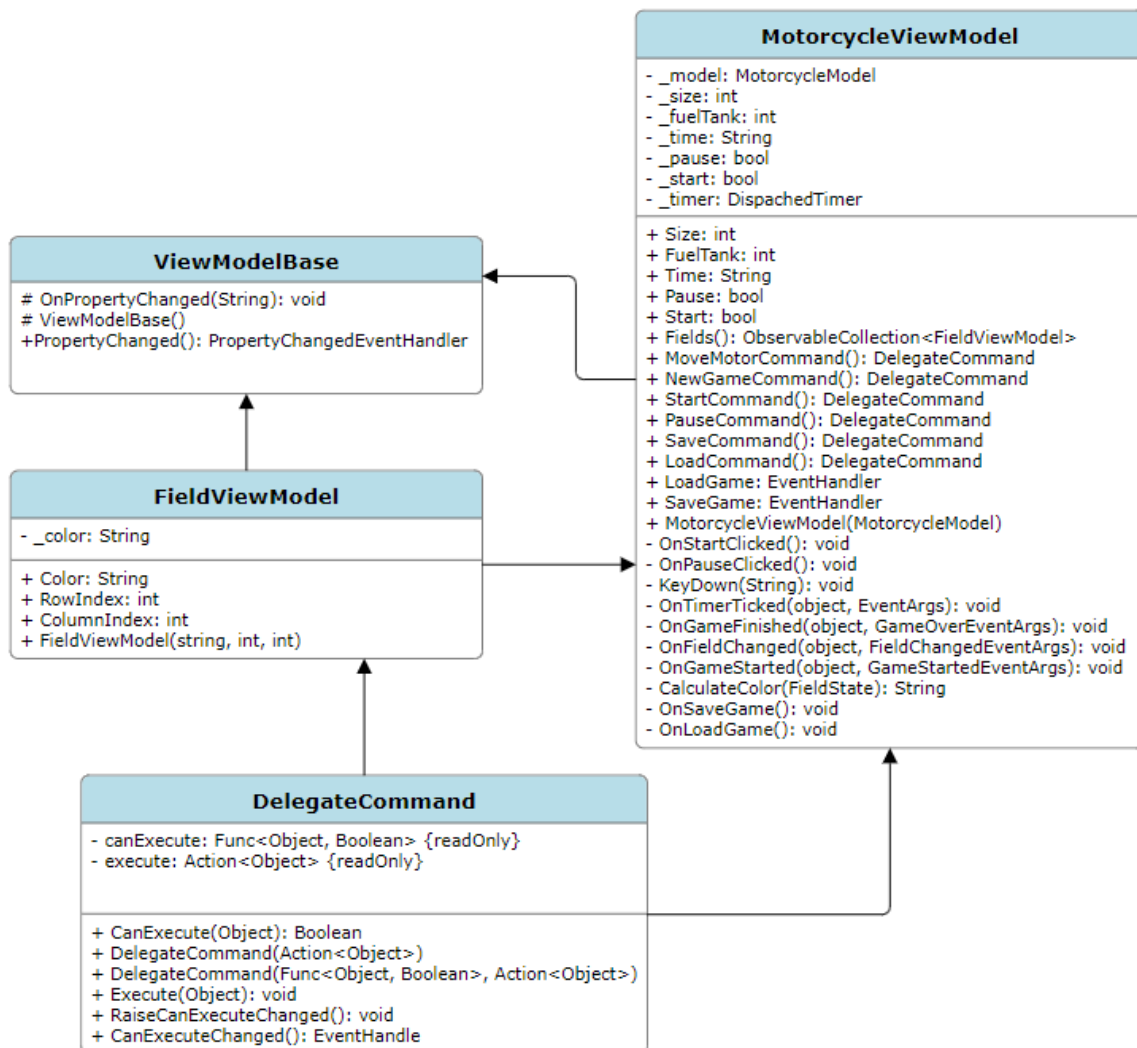
3.ábra: A Perzisztencia osztály csomagdiagramja

- Modell (4. ábra):
 - A modell lényegi részét a **MotorcycleModel** osztály valósítja meg, amely szabályozza a tábla tevékenységeit, valamint a játék egyéb paramétereit, úgymint az idő (**TimerTicked**). A típus lehetőséget ad új játék kezdésére (**startNewGame**), valamint lépésre (KeyPressed).
 - A játékállapot változásáról a **FieldChangedEventArgs** esemény, míg a játék végéről a **GameOverEventArgs** esemény tájékoztat.
 - A modell példányosításkor megkapja az adatkezelés felületét, amelynek segítségével lehetőséget ad betöltésre (**LoadGame**) és mentésre (**SaveGame**)



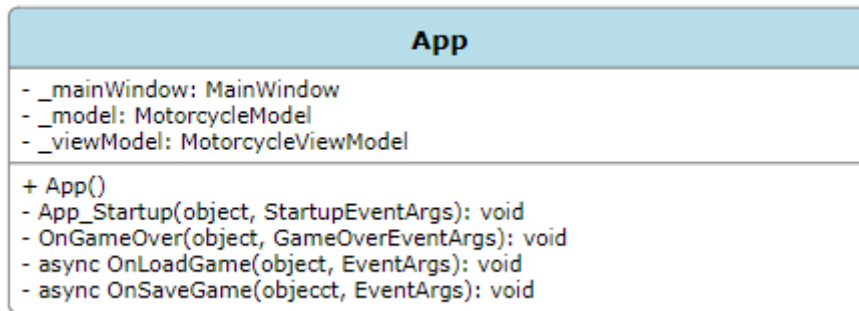
4.ábra: A Modell csomag osztálydiagramja

- Nézetmodell (5. ábra):
 - A nézetmodell megvalósításához felhasználunk egy általános utasítás (**DelegateCommand**), valamint egy ős változásjelző (**ViewModelBase**) osztályt.
 - A nézetmodell feladatait a **MotorcycleViewModel** 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 játékos számára egy külön mezőt biztosítunk (**FieldViewModel**), amely eltárolja a pozíciót és a cellák színét. 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 játéklemezőt, a menüt és a státuszsort. A játéklemező 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 fájlnevet bekérését betöltéskor és mentéskor, valamint a figyelmeztető üzenetek megjelenését beépített dialógusablakok segítségével végezzük.
- 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.



6. ábra: A vezérlés osztálydiagramja

Tesztelés:

- A modell funkcionalitása egységtesztek segítségével lett ellenőrizve a **MotorcycleGameTest** osztályban.
- Az alábbi tesztek kerültek megvalósításra:
 - **MotorcycleModelNewGameTest:** Új játék indítása, a mezők kitöltése, valamint a gyorsaság, üzemanyagmennyiség és az eltelt idő ellenőrzése.
 - **MotorcycleModelStepRightTest,**
MotorcycleModelStepLeftTest: Játékbeli lépések hatásainak ellenőrzése.
 - **MotorcycleModelTickedTest:** Játékbeli időtelés hatásainak ellenőrzése. (Üzemanyagcella lejjebb esése, gyorsulás, idő megnövelése)
 - **MotorcycleModelCaughtFuelCellTest:** A motor által elkapott üzemanyagcellák hatásainak ellenőrzése. (Üzemanyagtank megfelelő növelése)
 - **MotorcycleModelLoadTest:** A játék modell betöltésének tesztelése mockolt perzisztencia réteggel.