2. beadandó feladat

Készítette:

Mikus Márk

NK-kód: CM6TSV

email: [kyussfia@gmail.com](mailto:kyussfia@gmail.com)

# Feladat:

## Fénymotor párbaj

Készítsük programot, amellyel a Tronból ismert fénymotor párbajt játszhatjuk. Adott egy 𝑛 × 𝑛 elemből álló játékpálya. A két játékos a bal, illetve jobb oldal közepén indul egy-egy fénymotorral, amely egyenesen halad (rögzített időközönként) a legutoljára beállított irányba (függőlegesen, vagy vízszintesen). A motorokkal lehetőség van balra, illetve jobbra fordulni. A fénymotor mozgás közben fénycsíkot húz, ami a játék végéig ott marad. Az a játékos veszít, aki előbb nekiütközik a másik játékos motorjának, bármelyikük fénycsíkjának vagy a pálya szélének. A program biztosítson lehetőséget új játék kezdésére a pályaméret megadásával (12 × 12, 24 × 24, 36 × 36), valamint játék szüneteltetésére (ekkor nem telik az idő, és nem mozognak a motorok). Ismerje fel, ha vége a játéknak, és jelenítse meg, melyik játékos győzött.

# Elemzés:

* A játékot három pályamérettel játszhatjuk: kicsi (12 x 12), közepes (24 x 24), nagy (36 x 36).
* A feladatot egyablakos asztali alkalmazásként WPF grafikus felülettel valósítjuk meg.
* Az ablakban elhelyezünk egy Új játék menüt a következő menüpontokkal: Kis, Közepes, Nagy. Ahol a felhasználó a pályaméretet választhatja ki. A felületen helyet kap még egy óra, amelyen az eltelt időt lehet követni, továbbá egy szüneteltetésre való gomb, amelynek kezdetben nincs funkciója, csak a játék közben.
* A játéktáblát egy a n x x-es gombrács fogja reprezentálni, a gomboknak kattintás eseménye nincsen.
* Egy implementált időzítő eseményeire reagál majd a nézet, s lépkednek a játékosok.
* A játékosok irányítása legyen:
  + Kék játékos balra: A
  + Kék játékos jobbra: D
  + Piros játékos balra: Bal nyíl
  + Piros játékos jobbra: Jobb nyíl
* A játék automatikusan feldob egy dialógusablakot, amikor vége a játéknak (Döntetlen vagy Győzelem esetén), s megáll az időmérő.

# 

# Tervezés:

### Programszerkezet

A programot háromrétegű MVVM architektúrában valósítjuk meg: a megjelenítési elemekért a View, a megjelenítési logikáért a ViewModel, míg a játéklogikáért a Model komponens lesz felelős. Az **App** alkalmazás osztálypéldányosítja, majd szinkronizálja a komponenseket. Itt lettek elhelyezve a játék végét jelző üzenet dobozok is, illetve a Nézet komponens **KeyUp** eseményeire történő feliratkozás is.

### Modell

A modell lényegi részét a **LightDuelModel** osztály adja. A könnyebb karbantartás végett egyéb segédtípusok kerültek bevezetésre. A **Players** felsoroló a mezők lehetséges értékeit (No, Blue, Red), míg a Player típus magát a játékos objektumot reprezentálja. A Player típusnak 4 tagja van: egy oszlop és egy sorszám, egy irány és egy típus, hogy a játékos melyik színt képviseli. Az irány a mozgása irányát a sorszámok pedig a pozícióját határozzák meg.

A mezők tárolására a **List <List <Players>>** lista szolgál. A modell két alapvető eseménnyel rendelkezik: **ticked** és a **gameOver**. A **ticked** eseményt a nézettmodell időzítője váltja ki adott időközönként, amennyiben új játékot kezdünk a nézeten, a **newGame(size)** interfész metódus közvetítésével. A **gameOver** esemény akkor váltódik ki amikor vége a játéknak, valamilyen okból.

További két segédosztály kerül implementálása:

1. **PlayerMoveEventArgs**: A játéktábla frissítését segítő esemény argumentumok.
2. **GamoOverEventArgs**: A játék végeredményét közlő esemény argumentumok.

Ezen esemény argumentumok segítségével üzen a Model, a Nézetmodellnek, az **App** alkalmazás osztályon belül összekötött vezérlőn keresztül.

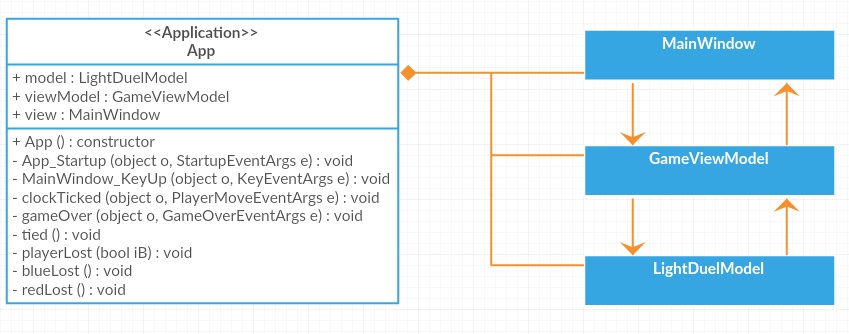
### NézetModell

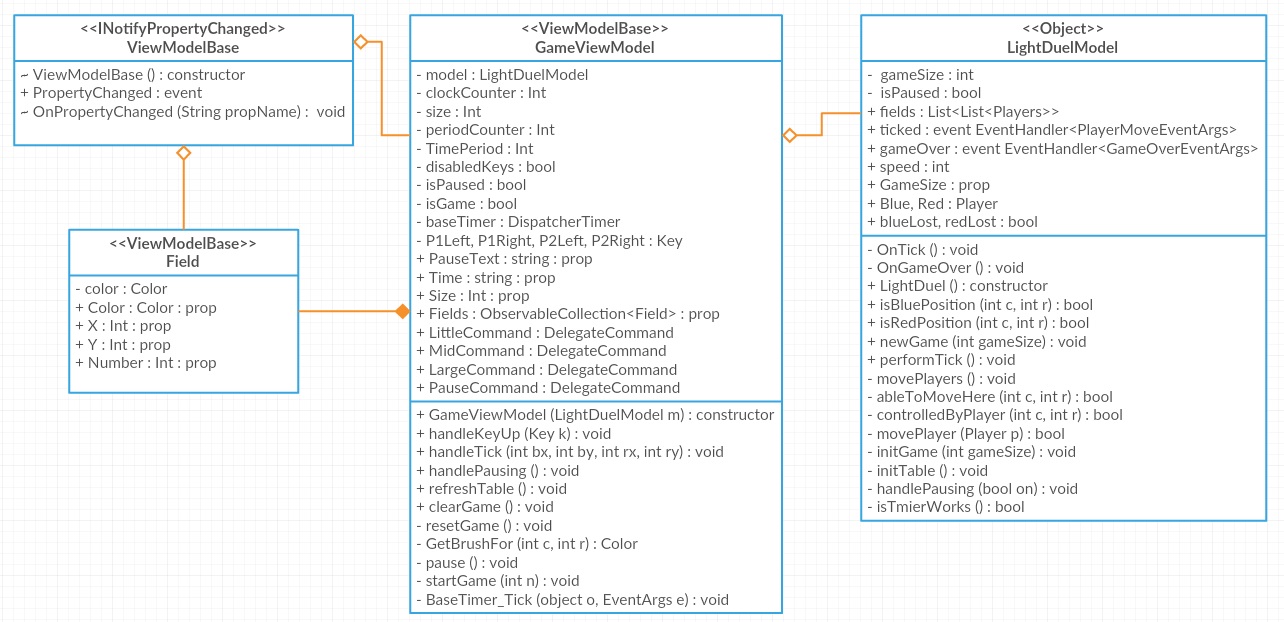
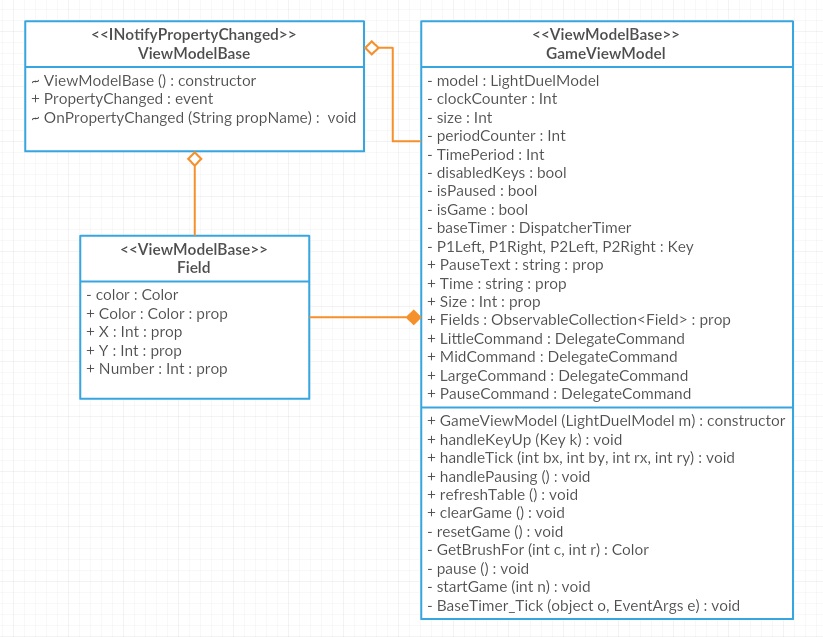
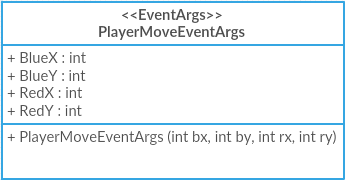
A nézetmodell komponens, a GameViewModel, az **INotifyPropertyChanged** osztály leszármazottja, amelynek az **OnPropertyChanged** metódusával, fogjuk szinkronizálni a nézettel, a modellt. A nézettmodell feladata, hogy frissen tartsa a nézetet, miközben a modellel kommunikál. A **DelegateCommand**-okon keresztül a nézetről beérkező kéréseket a nézetmodell fogadja, illetve a megfelelő logikát hajtja végre. A mezőgyűjteményt az **ObrservableCollection** adja, amely a **Field** mezőtípusra lett példányosítva. A **Field** is egy **INotifyPropertyChanged** leszármazott.

Itt foglal helyet a **DispatcherTimer** időzítő is, ami a játék alapját adja. Az időzítő üzen a modellnek, aki aztán visszaüzen az általa kiváltott **ticked** eseménnyel, amit miután fogad a nézetmodell, az **OnPropertyChanged** metódussal szinkronizálja az adatokat a Nézettel.

### Nézet

A nézet XAML formátumban íródott. A főablak, egy fix méretű, középre pozícionált **Window**, amelyben egy 3 soros **Grid** foglal helyet. Az első sorban a **Menu** található, a pályaméretekkel, alatta a második sorban a Szünet gomb (**Button**), illetve az óra (**Label)** található. A harmadik sor egy nem fixált magasságú **ItemControl**, amelyben az **ItemTemplate** egyes mezőleírójának a definíciója található. Itt állítjuk be azt, hogy a a megfelelő **Field** property **Color** tagjára bindelődjön a gomb háttérszíne.





# 

# 

# Tesztelés:

### A modell funkcionalitása egységtesztek segítségével lett ellenőrizve a LightDuelWpfTest névtér alatti UnitTest1 nevű osztályban.

### Az alábbi tesztesetek kerültek megvalósításra:

* + testModelNotNull
  + testModellInitialized
  + testStartGameLittle
  + testStartGameMid
  + testStartGameLarge
  + testModelMethodIsBluePos
  + testModelMethodIsRedPos
  + testStartGameMultipleTimes
  + testPauseGame
  + testControllingBluePlayer
  + testControllingRedPlayer
  + testTiedGame
  + testBlueWon
  + testRedWon

