1. **Beadandó feladat dokumentáció**

**Készítette:**

Varga Lili Anna

E-mail: [vargalili012@gmail.com](mailto:vargalili012@gmail.com)

**Feladat:**

Készítsünk programot, amellyel az aszteroidák játékot játszhatjuk. A feladatunk az, hogy egy űrhajó segítségével átnavigáljuk egy aszteroidamezőn. Az űrhajóval a képernyő alsó sorában tudunk balra, illetve jobbra navigálni. A képernyő felső sorában meghatározott időközönként véletlenszerű pozícióban jelennek meg az aszteroidák, amelyek folyamatosan közelednek állandó sebességgel a képernyő alja felé. Az idő múlásával egyre több aszteroida jelenik meg egyszerre, így idővel elkerülhetetlenné válik az ütközés. A játék célja az, hogy az űrhajó minél tovább elkerülje az ütközést. 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.

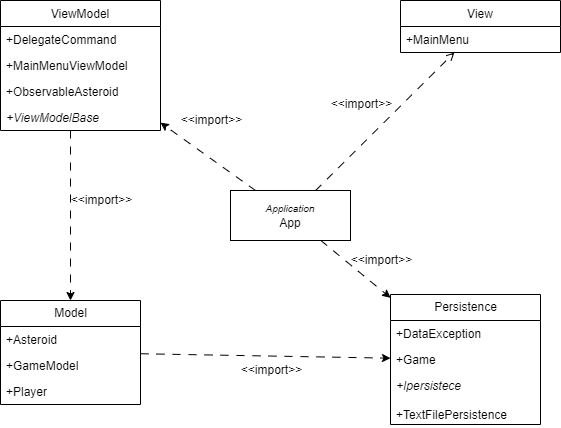
**Elemzés:**

* A program indításakor egy menü jelenik meg, melyben új játékot indíthatunk, vagy kiléphetünk a programkor.
* A játék automatikusan feldob egy dialógusablakot, amikor vége a játéknak (ha ütköztünk egy aszteroidával).
* Egy új ablakban elhelyezünk egy menüt a következő menüpontokkal: Játék betöltése, Játék mentése.
* A képen képernyőkép, kör, szöveg, tervezés látható

  Automatikusan generált leírásA játéktábla egy új ablak, melyen idővel folyamatosan növekvő aszteroidák jelennek meg. A játékos fel, le, balra és jobbra tud mozogni, hogy elkerülje az aszteroidákat, a pálya határain belül.

. ábra Felhasználói esetek

**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 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 csomagok a WPF függő projektjében kapnak helyet.

2. ábra Csomagdiagram

* **Perzisztencia**
  + Az adatkezelés feladata a játék állapotával kapcsolatos információk tárolása, valamint a betöltés/mentés biztosítása.
  + A Game osztály a játék aktuális állapotának reprezentációját valósítja meg, amely tartalmazza a játékos (Player) helyzetét, a játékidőt (GameTime), valamint az aszteroidák (Asteroid) listáját. A játékállapot lehetőséget biztosít a játékelemek módosítására (AddAsteroid).
  + A hosszú távú adattárolás lehetőségeit az IPersistence interfész adja meg, amely lehetőséget ad az állapot betöltésére (Load), valamint mentésére (Save). Az interfész megvalósítását többféle perzisztencia-megoldás (pl. szöveges fájl alapú) szolgáltathatja.
  + Az interfész szöveges fájl alapú adatkezelésre a TextFilePersistence osztály valósítja meg. A fájlkezelés során fellépő hibákat a DataException kivétel jelzi.
  + A program az adatokat szöveges fájlként tudja eltárolni, melyek az ".txt" kiterjesztést kapják. Ezeket az adatokat a programban bármikor be lehet tölteni, illetve ki lehet menteni az aktuális állást.
  + A fájl első sora a játékos helyzetét tartalmazza (X és Y koordináták), majd a játékidőt (GameTime). Ezután az aszteroidák koordinátái következnek. A szövegfájl számokat tartalmaz, amelyeket szóköz választ el egymástól.

A képen szöveg, képernyőkép, Betűtípus, sor látható

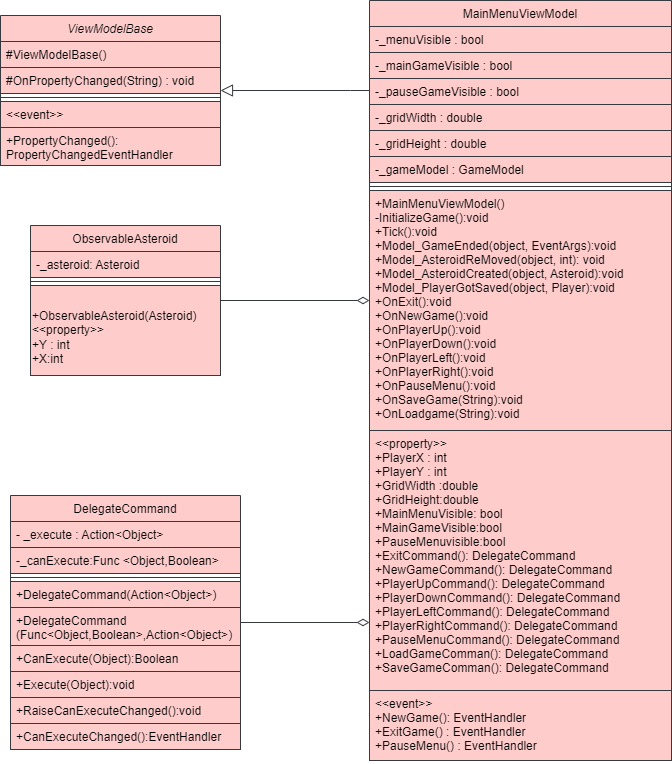
Automatikusan generált leírás

. ábra Persistence osztálydiagram

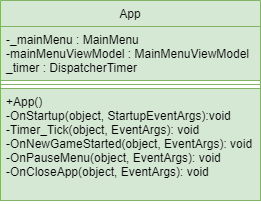
* **Model**
  + A modell lényegi részét a GameModel osztály valósítja meg, amely szabályozza a játék pálya tevékenységeit, valamint egyéb paramétereit, mint például az idő (GameTime) és a játékos (Player) pozíciója, illetve az aszteroidák kezelése (Asteroids). Az osztály lehetőséget biztosít a játék előrehaladására (AdvanceGame), az aszteroidák mozgásának kezelésére (MoveAsteroid), és új aszteroidák létrehozására (CreateAsteroid).
  + A játékos pozícióját mozgási metódusok (MovePlayerLeft, MovePlayerRight, MovePlayerUp, MovePlayerDown) szabályozzák, amely lehetőséget biztosít a játékos különböző irányokba történő mozgatására a panel határain belül.
  + A játék állapotváltozásairól az AsteroidCreated, AsteroidMoved, AsteroidReMoved események tájékoztatják a felhasználót. Az események argumentumai információkat tartalmaznak az újonnan létrehozott aszteroidákról, illetve a már meglévők pozíciójának változásairól. A játék végéről a GameEnded esemény tájékoztat, amikor egy ütközés történik.
  + A Player osztály tárolja a játékos aktuális pozícióját (X és Y koordináta), és lehetőséget biztosít a játékos mozgatására a képernyő bal, jobb, felső, illetve alsó határain belül. A játékos alapértelmezett mozgási sebessége (speed) 10 egység, ami minden irányban módosítja a játékos pozícióját.
  + Az ütközéseket a CollidesWith metódus kezeli, amely figyeli a játékos és az aszteroidák közötti átfedést. Ha az X és Y koordináták alapján a játékos és egy aszteroida ütköznek, a metódus igaz értéket ad vissza, amely a GameEnded esemény kiváltásához vezet, ezzel véget vetve a játéknak.
  + A játékállapot mentését és betöltését a LoadGame és SaveGame metódusok kezelik, amelyeket az adatkezelés felülete, az IPersistence interfészen keresztül valósít meg. Az IPersistence interfész lehetőséget ad a játékadatok betöltésére (Load) és mentésére (Save). A LoadGame a játékos adatait, az aszteroidák listáját és a játékidőt tölti be a megadott elérési útvonalról, míg a SaveGame elmenti ezeket az adatokat.



4. ábra Model osztálydiagram

* **NézetModell**
  + A nézetmodell megvalósításához felhasználunk egy általános utasítás (DelegateCommand), valamint egy ős változásjelző (ViewModelBase) osztályt.
  + 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, amely gombokból áll.
  + A nézetmodell feladatait a MainMenuViewModel 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 (\_gameModel), de csupán információkat kér le tőle, illetve a játék menetét szabályozza. Direkt nem avatkozik a játék futtatásába.
  + A játékmező számára egy külön mezőt biztosítunk (ObservableAsteroid), amely eltárolja a pozíciót, valamint a gombok megjelenítését a képernyőn. A mezőket egy felügyelt gyűjteménybe helyezzük a nézetmodellben (Asteroids).

5. ábra NézetModel osztálydiagram

* **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.
  + A fájlnév bekérését betöltéskor és mentéskor, valamint a figyelmeztető üzenetek megjelenését beépített dialógusablakok segítségével végezzük.
  + A MainWindow.xaml fájl tartalmazza a vizuális felület definícióját. A felület 3 fő részből áll:
  + Menu: A főmenü gombokat tartalmazza, mint a „New Game” és „Exit Game”.
  + MainGame: A játéktér, amely dinamikusan frissülő aszteroidákat (ItemsControl) és játékos pozícióját (összekötve a Canvas objektummal) jeleníti meg.
  + PauseMenu: Megállított játék esetén előhozható menü, amely tartalmazza a „Load Game” és „Save Game” lehetőségeket.
  + Az adatkötés segítségével a vezérlők automatikusan frissülnek a modell változásaira, továbbá billentyűparancsok is támogatottak („WASD” a játékos mozgatásához, „Esc” a szünet menü megnyitásához).
  + A vizuális elemek (gombok, szövegek, és egyéb UI komponensek) kialakításánál fontos volt az egyszerű kezelhetőség és az intuitív megjelenés biztosítása. Az adatkötés és az MVVM architektúra támogatja a tiszta kódbázist, és lehetővé teszi a felület és a logika elválasztását.
* **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.

6. ábra Vezérlés osztálydiagram

* **Tesztelés**
  + A játékmodell funkcionalitása egységtesztekkel került ellenőrzésre a GameModelTests osztályban, ahol különböző játékfunkciók helyes működése és a perzisztencia tesztelése történt meg.
  + Az alábbi tesztesetek lettek megvalósítva:
    - AdvanceGame\_CreatesAsteroids: A teszt során a AdvanceGame metódust ismételten meghívva biztosítottuk, hogy aszteroidák jönnek létre a játék során, és azok száma nő.
    - MoveAsteroid: A teszt célja annak ellenőrzése, hogy az MoveAsteroid metódus helyesen mozgatja az aszteroidákat. Amennyiben van aszteroida, akkor a metódus az y-koordinátájukat növeli, miközben az x-koordinátát változatlanul hagyja.
    - GameEndsWhenCollided: Ez a teszteset igazolja, hogy a játék véget ér, ha a játékos és egy aszteroida ütközik. Az aszteroidát a játékos pozíciójába helyezve teszteljük, hogy az GameEnded esemény helyesen kiváltódik.
    - Boundaries: A teszt a játékos mozgási korlátait ellenőrzi. A játékos nem hagyhatja el a képernyő bal, jobb, felső vagy alsó széleit. A teszt során a játékos pozícióját minden irányba addig mozgatjuk, amíg el nem éri a játékterület határait.
    - PersistenceSave: A perzisztencia tesztelése érdekében ellenőrizzük, hogy a SaveGame metódus meghívásra kerül-e a perzisztencia rétegen keresztül a megadott mentési útvonallal.
    - PersistenceLoad: A játék betöltésének tesztje, amely mockolt perzisztencia réteggel történik. A LoadGame meghívása után ellenőrizzük, hogy a játékos, az aszteroidák és a játékidő helyesen kerültek-e visszatöltésre, valamint, hogy a betöltésért felelős perzisztencia metódus meghívódik-e.