

Feladat:

8. Aszteroidák

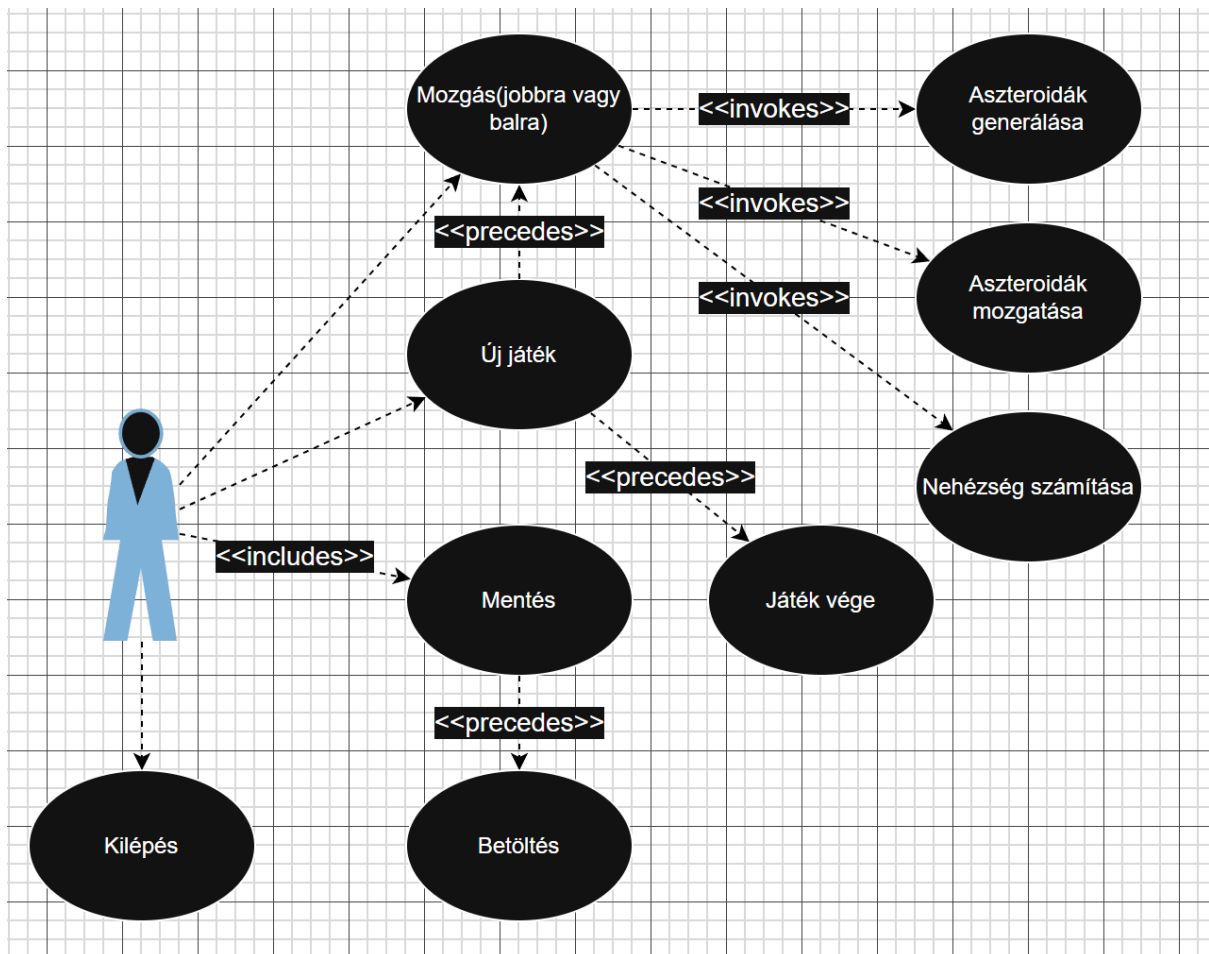
Készítsünk programot, amellyel az aszteroidák játékot játszhatjuk. A feladatunk az, hogy egy űrhajó segítségével átnavigáljuk egy aszteroidamezőt. Az űrhajóval a képernyő alsó sorában tudunk balra, illetve jobbra navigálni. A képernyő felső sorában meghatározott időközönként véletlenszerű pozícióban jelennek meg az aszteroidák, amelyek folyamatosan közelednek állandó sebességgel a képernyő alja felé. Az idő múlásával egyre több aszteroida jelenik meg egyszerre, így idővel elkerülhetlenné 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 játék követelményei
 - Lehetőség az űrhajó irányítására jobbra és balra a képernyő alján
 - Aszteroidák generálása véletlenszerű pozícióban, az idővel telésével egyre gyakrabban
 - A játéknak detektálnia kell, ha az űrhajó ütközik egy aszteroidával
 - A játék állapotainak változtatása (játék, szüneteltetés, mentés, betöltés)
 - Pontozási rendszer (a különböző méretű és sebességű aszteroidák hány pontot adnak)
 - Nehézségi szintek a játék közben
- Megvalósítás
 - Egyablakos asztali alkalmazás Windows Forms alkalmazásával
 - Az ablakban elhelyezésre kerül egy menüsor a következő menüpontokkal: Game(New Game(Új játék), Pause/Play(a játék szüneteltetése és folytatása), Save Game(A játék mentése), Load Game(játékállás betöltése), Reset High Score(A jelenlegi legmagasabb pontszám értékét visszaállítja 0-ra), Exit(A játékból való kilépés)) valamint Help(Controls(Az

irányítással és a pontozással kapcsolatos adatok), About(A játékkal kapcsolatos adatok))

- A játékteret a főablak üres területe reprezentálja, ahol kirajzolásra kerülnek az aszteroidák és az űrhajó is, amelyet a jobbra és balra nyilakkal lehet irányítani
- A játéknak automatikusan felismeri, ha vége a játéknak (az aszteroida és az űrhajó ütközik) és egy üzenet jelenik meg. Szintén külön ablakokkal működik a mentés és a betöltés is, ahol a fájlneveket a felhasználó adja meg



Tervezés

Programszerkezet:

- **View:**
 - A nézetet a WPF keretrendszer valósítja meg, **MVVM** architektúrát követve.
 - A központi nézetet a MainWindow.xaml. A nézet logikáját és állapotát a GameViewModel osztály(a GameViewModel.cs fájlban) biztosítja.
 - A GameViewModel osztály tárolja a GameModel referenciáját, valamint a perzisztencia(IGamePersistance, IHighScoreManager) és a dialógusablakok (IDialogService) szolgáltatásait
 - Az alkalmazás indításakor (App.xaml.cs) hozza létre a GameViewModelet és az ezt megjelenítő MainWindow-t A nézetet a DataContext tulajdonságán keresztül kötjük össze a GameViewModelmel
- **Megjelenítés és frissítés**
 - A játékteret nem egy imperatív OnPaint metódus valósítja meg (mint a Windows Formsban), hanem a WPF adatkötési mechanizmusa
 - A frissítés eseményvezérelt
 1. A GameModel belső időzítője frissíti a játékállapotot
 2. A GameModel eseményeket vált ki(pl.: ScoreChanged, GameTimeChanged)
 3. A GameViewModel feliratkozik ezekre az eseményekre, és az OnModelUpdate metódusban frissíti a saját, nézethez kötött tulajdonságait(pl.: Score, GameTime, List<Asteroids>)
 4. Az OnPropertyChanged() hívása jelzi a WPF-nek, hogy a felületet frissítenie kell
- **Vezérlők és Interakciók**
 - A menüpontok(MenuItem) nem eseménykezelőket, hanem az ICommand interfészen alapuló parancsokat használnak, melyek a GameViewModel-ben Vannak definiálva
 - A dialógusablakokat(mentés, betöltés, Game over, stb...) egy absztrakt IDialogService interfész kezeli, melyet a WpfDialogService implementálja

- **Rajzolósi elemek(xaml)**

- A korábbi Draw... metódusok helyett deklaratív XAML elemek Vannak
 1. **Aszteroidák:** Egy ItemsControl van adatkötéssek a GameViewModel.Asteroids (ObservableCollection) kollekciójához kötve. Az ItemsControl.ItemTemplate határozza meg egy aszteroida kinézetét(Elipse elemek). Az aszteroidák pozíciója a Canvas.Left és a Canvas.Top tulajdonságok adatkötéssel Vannak beállítva
 2. **Aszteroida részletesség:** A színek és a kráterek láthatóságát/méretét adatkötéshez kapcsolt konverterek(AsteroidFillConverter, CraterSizeConverter) dinamikusan határozzák meg az aszteroida mérete alapján
 3. **Úrhajó:** Egy <Grid> elem, amely <Rectangle> és <Polygon> elemekből épül fel. A pozíciója a GameViewModel.SpaceShipPosition(Thickness) tulajdonsághoz van kötve
 4. **Felhasználói felület:** A pontszám, idő és a nehézség megjelenítésére a StatusBar-ban lévő TextBlock elemekkel történik, melyek közvetlenül a GameViewModel(pl.: Score, GameTime, Difficulty) tulajdonságaihoz vannak kötve
 5. **Háttér:** A DrawStarts() funkció nincs implementálva a WPF alkalmazásban, ezért a játéktér egy egyszerű fekete <Grid>(Background="Black") adja

- **Billentyűzet Eseménykezelői**

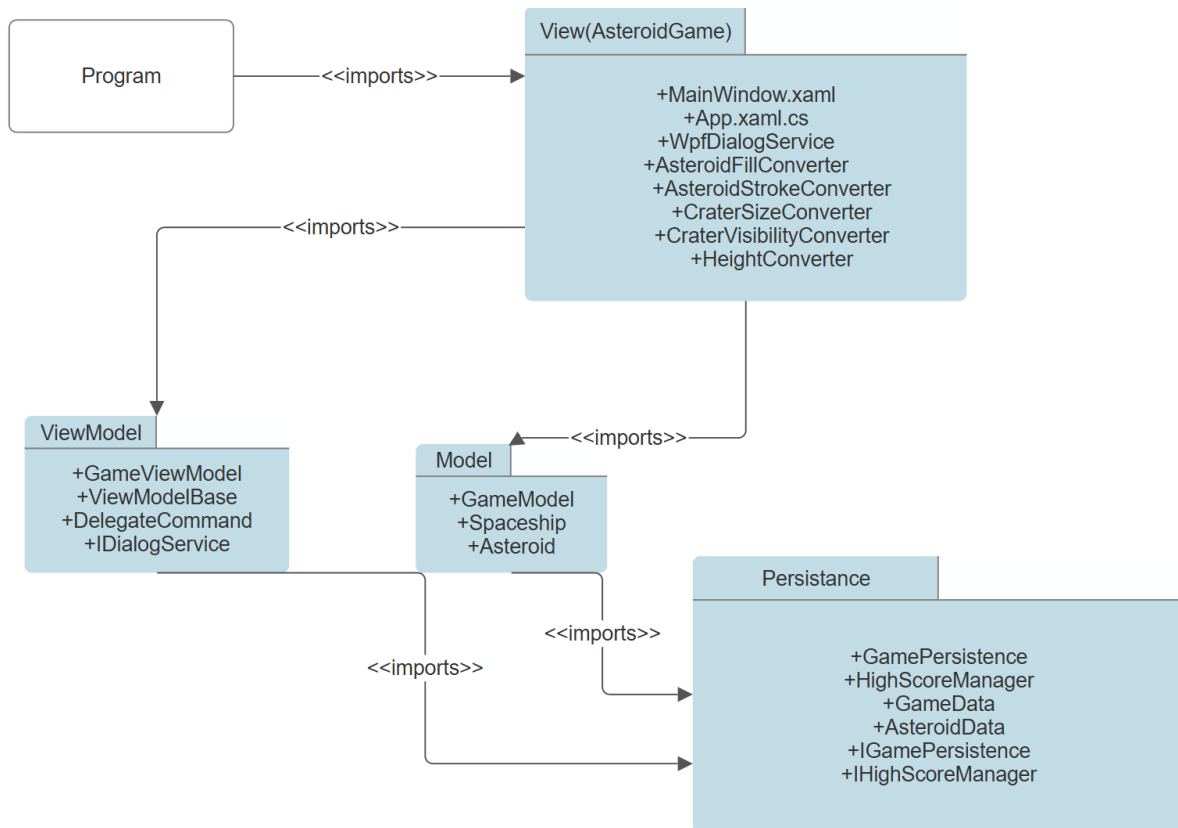
- A billentyűleütéseket (OnWindowKeyDown) és a felengedéseket (OnWindowKeyUp) az App.xaml.cs osztály kezeli
- Ezek az eseménykezelők nem közvetlenül a modellt, hanem a GameViewModel megfelelő metódusait hívják(pl.: _viewModel.SetMovingLeft(true), _viewModel.TogglePause()), tisztán tartva a felelősségi köröket

- **Persistence**

- Feladata a játékállapot tárolása valamint a high score tárolása
- A GamePersistence osztály csinálja a játékállapot fájlba történő mentését valamint betöltését
- A HighScore osztály végzi a high score tárolását és betöltését szöveges fájlba/fájlból
- A játékállapot egy .save kiterjesztésű fájlban kerül lementésre amely tartalma:
 1. Pontszám
 2. Játékidő
 3. Képernyő méretei

4. Űrhajó X pozíciója

5. Aszteroidák adatai(X, Y, BaseSize, Speed)



• Model

- A GameModel osztály valósítja meg a játék lényegi részeit, szabályozza a játékmenetet valamint a játék paramétereit
- A SpaceShip osztály az űrhajót reprezentálja, kezeli a mozgást a képernyő határainak figyelembe vételével. Tulajdonságai: X, Y, Width, Height, metódusai: MoveLeft(), MoveRight()
- Az Asteroid osztály tartalmazza az aszteroidát, kezeli a mozgást és a méret- és sebességváltozásokat a különböző instanceok esetében. Tulajdonságai: X, Y, Width, Height, Speed, metódusai: Move()
- A játék időbeli kezelését egy külső időzítő oldja meg és hívja meg az update metódust. A játékidő az Update hívások között eltelt idő összegéből számolódik
- Az állapotváltozásról események tájékoztatnak (GameOver, ScoreChanged, GameTimeChanged, HighScoreChanged)
- A játék a nehézséget dinamikusan számolja a model az idő és a pontszám alapján:

```

1. double timeFactor = Math.Min(_gameTime.TotalSeconds / 60.0, 2.0);
2. double scoreFactor = Math.Min(_score / 1000.0, 2.0);
3. double difficulty = 1.0 + (timeFactor + scoreFactor) / 2.0;
    
```

- Az aszteroidák generálása valószínűségi alapon történik, a nehézségtől függően:
 - Kis aszteroida(40% esély): 15-25 pixel, gyors
 - Közepes aszteroida(30% esély): 25-40 pixel, közepes sebesség
 - Nagy aszteroida(20 % esély): 40-60 pixel, lassú
 - Óriási aszteroida(10% esély): 60-80 pixel, nagyon lassú
 - DrawHighScore(): A legmagasabb pontszám, az aktuális pontszám és a nehézségi szint megjelenítése

Osztálydiagram

Mermaid

```
1. classDiagram
2.     direction TB
3.
4.     %% --- INTERFACES ---
5.     class IValueConverter {
6.         <<Interface>>
7.         +Convert(value, targetType, parameter, culture)
8.         +ConvertBack(value, targetType, parameter, culture)
9.     }
10.    class INotifyPropertyChanged {
11.        <<Interface>>
12.        +PropertyChanged : event
13.    }
14.    class ICommand {
15.        <<Interface>>
16.        +CanExecute(parameter) : bool
17.        +Execute(parameter)
18.        +CanExecuteChanged : event
19.    }
20.    class IDialogService {
21.        <<Interface>>
22.        +ShowMessage(message, title)
23.        +ShowConfirmation(message, title) : bool
24.        +ShowSaveDialog(filter, defaultExt, initialDirectory) : string
25.        +ShowOpenDialog(filter, defaultExt, initialDirectory) : string
26.    }
27.    class IGamePersistence {
28.        <<Interface>>
29.        +SaveGame(filePath, gameModel)
30.        +LoadGame(filePath) : GameData
31.    }
32.    class IHighScoreManager {
33.        <<Interface>>
34.        +LoadHighScore() : int
35.        +SaveHighScore(score)
36.        +GetSaveDirectory() : string
37.    }
38.
39.    %% --- VIEW (AsteroidGame & AsteroidGame.View) ---
40.    class App {
41.        -MainWindow _window
42.        -GameViewModel _viewModel
43.        +App()
44.        -Application_Startup(sender, e)
45.        -OnGameOver(sender, e)
46.        -OnGameLoaded(sender, e)
47.    }
48.    App --o MainWindow : _window
49.    App --o GameViewModel : _viewModel
50.
51.    class MainWindow {
52.        +MainWindow()
```

```

53.         -MainWindow_Loaded(sender, e)
54.     }
55.
56.     class WpfDialogService {
57.         +ShowMessage(message, title)
58.         +ShowConfirmation(message, title) : bool
59.         +ShowSaveDialog(filter, defaultExt, initialDirectory) : string
60.         +ShowOpenDialog(filter, defaultExt, initialDirectory) : string
61.     }
62.     WpfDialogService ..|> IDialogService : implements
63.
64.     class HeightConverter {
65.         +Offset : double
66.         +Convert(value, targetType, parameter, culture)
67.         +ConvertBack(value, targetType, parameter, culture)
68.     }
69.     HeightConverter ..|> IValueConverter : implements
70.
71.     class AsteroidFillConverter {
72.         +Convert(value, targetType, parameter, culture)
73.         +ConvertBack(value, targetType, parameter, culture)
74.     }
75.     AsteroidFillConverter ..|> IValueConverter : implements
76.     AsteroidFillConverter ..> Asteroid : uses
77.
78.     class AsteroidStrokeConverter {
79.         +Convert(value, targetType, parameter, culture)
80.         +ConvertBack(value, targetType, parameter, culture)
81.     }
82.     AsteroidStrokeConverter ..|> IValueConverter : implements
83.     AsteroidStrokeConverter ..> Asteroid : uses
84.
85.     class CraterSizeConverter {
86.         +Convert(value, targetType, parameter, culture)
87.         +ConvertBack(value, targetType, parameter, culture)
88.     }
89.     CraterSizeConverter ..|> IValueConverter : implements
90.
91.     class CraterVisibilityConverter {
92.         +Convert(value, targetType, parameter, culture)
93.         +ConvertBack(value, targetType, parameter, culture)
94.     }
95.     CraterVisibilityConverter ..|> IValueConverter : implements
96.     CraterVisibilityConverter ..> Asteroid : uses
97.
98.     %% --- VIEWMODEL (AsteroidGame.ViewModel) ---
99.     class ViewModelBase {
100.         +PropertyChanged : event
101.         +ViewModelBase()
102.         #OnPropertyChanged(propertyName)
103.     }
104.     ViewModelBase ..|> INotifyPropertyChanged : implements
105.
106.     class DelegateCommand {
107.         -readonly _execute : Action
108.         -readonly _canExecute : Predicate
109.         +CanExecuteChanged : event
110.         +DelegateCommand(execute, canExecute)
111.         +CanExecute(parameter) : bool
112.         +Execute(parameter)
113.         +RaiseCanExecuteChanged()
114.     }
115.     DelegateCommand ..|> ICommand : implements
116.
117.     class GameViewModel {
118.         -GameModel _gameModel
119.         -readonly IGamePersistence _persistence
120.         -readonly IHighScoreManager _highScoreManager
121.         -readonly IDialogService _dialogService
122.         -Thickness _spaceshipPosition

```

```

123.         +Asteroids : ObservableCollection~Asteroid~
124.         +SpaceshipPosition : Thickness
125.         +Spaceship : Spaceship
126.         +Score : int
127.         +GameTime : TimeSpan
128.         +HighScore : int
129.         +IsGameOver : bool
130.         +IsPaused : bool
131.         +NewGameCommand : ICommand
132.         +TogglePauseCommand : ICommand
133.         +SaveGameCommand : ICommand
134.         +LoadGameCommand : ICommand
135.         +ResetHighScoreCommand : ICommand
136.         +ExitCommand : ICommand
137.         +ShowControlsCommand : ICommand
138.         +ShowAboutCommand : ICommand
139.         +GameOver : event
140.         +GameLoaded : event
141.         +GameViewModel(persistence, highScoreManager, dialogService)
142.         +SetSize(width, height)
143.         +StartNewGame()
144.         +TogglePause()
145.         +SetMovingLeft(isMoving)
146.         +SetMovingRight(isMoving)
147.         +StopGame()
148.         -UpdateSpaceshipPosition()
149.         -OnModelUpdate(sender, e)
150.         -OnGameOver(sender, e)
151.         -ExecuteNewGame()
152.         -ExecuteTogglePause()
153.         -ExecuteSaveGame()
154.         -ExecuteLoadGame()
155.         -ExecuteResetHighScore()
156.     }
157.     GameViewModel --|> ViewModelBase
158.     GameViewModel --o GameModel : _gameModel
159.     GameViewModel ..> IGamePersistence : uses
160.     GameViewModel ..> IHighScoreManager : uses
161.     GameViewModel ..> IDialogService : uses
162.     GameViewModel ..> DelegateCommand : uses
163.     GameViewModel ..> Asteroid : uses
164.     GameViewModel ..> Spaceship : uses
165.
166.     %% --- MODEL (AsteroidGameMechanic.Model) ---
167.     class GameModel {
168.         -readonly IHighScoreManager _highScoreManager
169.         -readonly IGamePersistence _gamePersistence
170.         -readonly Random _random
171.         -int _score
172.         -bool _isGameOver
173.         -bool _isPaused
174.         -TimeSpan _gameTime
175.         -System.Timers.Timer _timer
176.         -bool _isMovingLeft
177.         -bool _isMovingRight
178.         +Spaceship Spaceship
179.         +List~Asteroid~ Asteroids
180.         +Score : int
181.         +IsGameOver : bool
182.         +IsPaused : bool
183.         +GameTime : TimeSpan
184.         +HighScore : int
185.         +GameOver : event
186.         +ScoreChanged : event
187.         +GameTimeChanged : event
188.         +HighScoreChanged : event
189.         +GameModel(screenWidth, screenHeight, highScoreManager, gamePersistence)
190.         +PerformGameTick(elapsedTime)
191.         +SetHighScore(highScore)
192.         +SetMovingLeft(isMoving)

```



```

193.         +SetMovingRight(isMoving)
194.         +Stop()
195.         +TogglePause()
196.         +SetGameState(score, gameTime, spaceshipX, asteroids)
197.         +GetDifficultyDescription() : string
198.         +StartGame()
199.         -InitializeGame()
200.         -CheckCollision(spaceship, asteroid) : bool
201.     }
202.     GameModel --* Spaceship : Spaceship
203.     GameModel --* "0..*" Asteroid : Asteroids
204.     GameModel ..> IHighScoreManager : uses
205.     GameModel ..> IGamePersistence : uses
206.
207.     class Spaceship {
208.         -readonly int _screenWidth
209.         -readonly int _speed
210.         +X : int
211.         +Y : int
212.         +Width : int
213.         +Height : int
214.         +Spaceship(x, y, screenWidth)
215.         +MoveLeft()
216.         +MoveRight()
217.     }
218.
219.     class Asteroid {
220.         +X : int
221.         +Y : int
222.         +Width : int
223.         +Height : int
224.         +BaseSize : int
225.         +Speed : int
226.         +Asteroid(x, y, screenHeight, baseSize, speed)
227.         +Move()
228.     }
229.
230.     %% --- PERSISTENCE (AsteroidGameMechanic.Persistence) ---
231.     class GamePersistence {
232.         +SaveGame(filePath, gameModel)
233.         +LoadGame(filePath) : GameData
234.     }
235.     GamePersistence ..|> IGamePersistence : implements
236.     GamePersistence ..> GameModel : uses
237.     GamePersistence ..> GameData : uses
238.     GamePersistence ..> AsteroidData : uses
239.
240.     class HighScoreManager {
241.         -readonly string _saveDirectory
242.         -readonly string _highScoreFile
243.         +HighScoreManager(gameRootPath)
244.         +LoadHighScore() : int
245.         +SaveHighScore(score)
246.         +GetSaveDirectory() : string
247.     }
248.     HighScoreManager ..|> IHighScoreManager : implements
249.
250.     class GameData {
251.         +Score : int
252.         +GameTime : TimeSpan
253.         +SpaceshipX : int
254.         +List~AsteroidData~ Asteroids
255.         +ScreenWidth : int
256.         +ScreenHeight : int
257. 1.     GameData --o "0..*" AsteroidData : Asteroids
258. 2.
259. 3.     class AsteroidData {
260. 4.         +X : int
261. 5.         +Y : int
262. 6.         +BaseSize : int

```

```
263. 7.      +Speed : int
264. 8.      }
265. 9.
266.
```

- **Tesztelés**

- A modell funkcionalitását MSTest egységtesztek végzik az AsteroidGameTest osztályban
- A teszteléshez Moq keretrendszert használtunk mock objektumok létrehozására
- Az alábbi tesztek kerültek megvalósításra:
 - Test_GameInitialization: Játék indítása, kezdeti állapot ellenőrzése
 - Test_SpaceshipMovement_Left/Right: Űrhajó mozgásának és határainak tesztelése
 - Test_AsteroidSpawning_OverTime: Aszteroidák generálásának időbeli változásának tesztelése
 - Test_AsteroidSpawning_Positions: Az aszteroidák generálási helyeinek tesztelése
 - TestCollisionDetection_GameOver: Ütközés detektálása és játék vége
 - TestCollisionDetection_NoCollision: Nincsen fp ütközés felvéve
 - Test_ScoringSystem_AsteroidPassing: Pontozási rendszer tesztelése
 - Test_PauseFunctionality_...: Szünetelés tesztelése
 - Test_HighScoreTracking_...: a legmagasabb pontszám követése és frissítésének tesztelése
 - Test_DifficultyProgression_OverTime: Nehézségi szint megfelfő változásának tesztelése
 - Test_GameStatePersistence_SetState: Játékállapot beállítása
 - Test_AsteroidSizeVariation_Types: Aszteroidák méretválltozásának tesztelése