Programozás III. beadandó

Dokumentáció

Tóth Bianka Patrícia

ÓE NIK

2014/15 I. félév

Tartalomjegyzék

[A játék ismertetése 3](#_Toc404620715)

[Megvalósítás 4](#_Toc404620716)

[Osztályok és a fontosabb függvények részletes ismertetése 4](#_Toc404620717)

[Egyszerű osztályok 4](#_Toc404620718)

[Ablak osztályok (.xaml) 6](#_Toc404620719)

[Felhasználói segédlet 10](#_Toc404620720)

# A játék ismertetése

A *Frogger in the sea* a klasszikus, 1981-ben bemutatott *Frogger* mintájára íródott. A cél mindkét játékban az, hogy az idő lejárta előtt a béka átjusson a pálya túloldalára úgy, hogy az áthaladó játékelemekkel ne ütközzön.

Ebben a változatban a béka halak és tengeri állatok (ráják, rákok, vízisíklók) között kell hogy megtalálja az utat a tenger mélyéről a vízfelszín felé. A halak nem jelentenek veszélyt, a tengeri állatokkal való ütközés viszont életvesztést okoz. A két kategóriát azzal jeleztem, hogy az élőlények különböző sebességgel mozognak: a halak lassan, a tenger állatkák gyorsabban.

A békának 30 másodperce van, hogy feljusson a vízfelszínre. Kezdetben három élettel rendelkezik, azonban egy pálya sikeres teljesítése után az életek száma megnövekszik eggyel. Öt pálya létezik; az egyikből másikba való átjutáskor pontok számolódnak a meglévő életek és a megmaradt idő alapján.

A játék véget ér, ha

a) a béka az összes életét elvesztette,

b) a béka teljesítette az ötödik pályát is.

A játék véget érésekor a ponthoz hozzáadódik még egy pontszám, ami az aktuális pálya szintjéből generálódik.

Az életvesztés szintén kétféleképp történhet:

a) lejár az idő

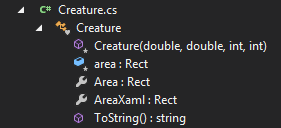
b) a béka ütközik egy tengeri állattal.

# Megvalósítás

## Osztályok és a fontosabb függvények részletes ismertetése

### Egyszerű osztályok

#### Creature

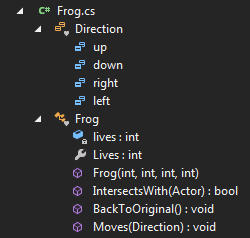
**Rect area, Rect Area**: az élőlény kiterjedését szimbolizáló változó. X, Y tulajdonsága az élőlény pozícióját, Width, Height tulajdonsága pedig a szélességet, magasságot jelentik.

**Rect AreaXaml**: mivel a kódban oszlopokkal és sorokkal dolgozok, készítettem egy plusz tulajdonságot az area-hoz, ami elvégzi egy mező (egy sorban lévő oszlop) szélességével és magasságával való felszorzást és leosztást, mikor adatkötésre kerül sor.

**Creature(double, double, int, int)**: konstruktor, amiben értéket kap az area (terület).

**ToString()**: felülírt metódus.

#### Frog

**int lives, int Lives**: életek száma.

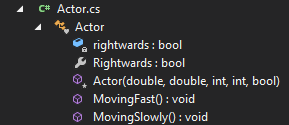
**Frog(int, int, int, int)**: konstruktor, ami bekéri az Area Rect paramétereit és továbbítja azt az ős konstruktora felé.

**bool IntersectsWith(Actor)**: azt vizsgálja, hogy egy másik állattal ütközik-e a béka. Akkor ütköznek, ha a két állat egy sorban van (Area.Y) és oszlopukat (Area.Y) tekintve nincs köztük 0,8 oszlopnál nagyobb eltérés.

**void Moves(Direction)**: a béka mozgását biztosítja. Paraméterként egy enum Direction típusú változót kér, amit előtte deklaráltam: lehet right, left, down és up.

**void BackToOriginal()**: visszahelyezi a békát a kiindulópontra.

#### Actor

**bool rightwards, bool Rightwards**: igaz, ha az állat jobbra megy, hamis, ha balra.

**Actor(double, double, int, int, bool):** konstruktor, amelyben az area (terület) elemein kívül szerepel a rightwards értéke is.

**MovingFast()**: gyors mozgás (tengeri állatok). Az area.X értékét módosítja a rightwards értékének megfelelően.

**MovingSlowly()**: lassú mozgás (halak). Az area.X értékét módosítja a rightwards értékének megfelelően.

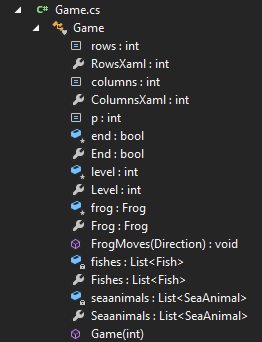
#### Fish

**Fish(int, int, int, int, bool)**: konstruktor. Az area X és Y értékének itt elég int-et megadni, hiszen úgyis sor és oszlop formátumban adom meg őket. A halakkal nem lehet ütközni.

#### SeaAnimal

**SeaAnimal(int, int, int, int, bool)**: konstruktor. Az area X és Y értékének itt elég int-et megadni, hiszen úgyis sor és oszlop formátumban adom meg őket. A tegneri állatokkal nem lehet ütközni.

#### Game

**const int rows**: a pálya magasságának (sorok száma). Mivel ez az érték nem fog változni és több osztályban is felhasználom, konstans értéket adtam neki.

**int RowsXaml**: a SeaWindow ezt az értéket olvassa be: a sorok száma fel van szorozva a szintén pixel értékkel.

**const int columns**: a pálya szélessége (oszlopok száma). Mivel ez az érték nem fog változni és több osztályban is felhasználom, konstans értéket adtam neki.

**int ColumnsXaml**: a SeaWindow ezt az értéket olvassa be: a sorok száma fel van szorozva a szintén pixel értékkel.

**int const p**: egy mező (egy sorban lévő oszlop) szélessége és magassága pixelben.

**int level, int Level**: a játék szintjét adja meg.

**Frog frog, Frog Frog**: a béka.

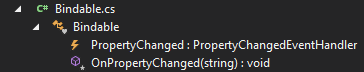
**FrogMoves(Direction)**: ha nincs vége a játéknak, a béka a megadott irányba mozog egy mezőnyi értéket. A lehetséges irányok: fel, le, balra, jobbra.

**List<Fish> fishes, List<Fish> Fishes**: a halakat tartalmazó lista.

**List<SeaAnimal> seaanimals, List<SeaAnimal> Seaanimals**: a tengeri állatokat tartalmazó lista.

**Game(int)**: konstruktor. A békát a kiindulópontra helyezem, majd a szintnek megfelelően feltöltöm a pályát a halakkal és tengeri állatokkal, megadva a pozíciójukat és az irányukat (a saját konstruktorukban).

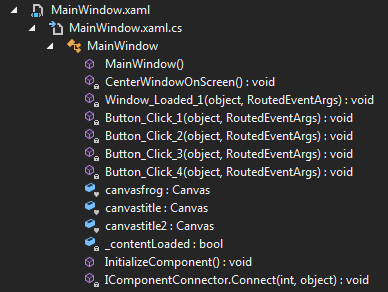
#### Bindable

Segédosztály az adatkötés megvalósításához.

### Ablak osztályok (.xaml)

Az struktúrákban megjelenített xaml grafikus elemekről külön nem írok.

#### MainWindow

A kezdőképernyő, azaz a menü.

**MainWindow**: konstruktor.

**void CenterWindowOnScreen()**: elhelyezi az ablakot a képernyő közepére.

**void WindowLoaded(…)**: képek elhelyezése a felületen.

**void Button\_Click\_1(…)**: új játékot nyit meg, level = 1.

**void Button\_Click\_2(…)**: előhívja a help (segítség) menüt.

**void Button\_Click\_3(…)**: exit – kilép a játékból.

**void Button\_Click\_4(…)**: X gomb – kilép a játékból.

#### HelpWindow

A menüből elérhető felhasználói segédlet.

**HelpWindow()**: konstruktor.

**void CenterWindowOnScreen()**: elhelyezi az ablakot a képernyő közepére.

**void WindowLoaded(…)**: képek és szöveg elhelyezése a felületen.

**void Button\_Click\_1(…)**: OK gomb, visszatér a menübe.

#### DeadFrogWindow

Ha a béka életet veszt, akkor jelenik meg ez az ablak 1 mp-ig (vagy a bezárásáig).

**DeadFrogWindow()**: konstruktor.

**void CenterWindowOnScreen()**: elhelyezi az ablakot a képernyő közepére.

**void WindowLoaded(…)**: képek és szöveg elhelyezése a felületen.

**void Button\_Click\_1(…)**: X (exit) gomb, új játékot kezd.

**DispatcherTimer timer, timer\_Tick**: az ablak elegendő, ha csak 1 másodpercig jelenik meg. Ehhez van szükség az időzítőre és az eseménykezelőre.

**Window\_PrewiewKeyDown(…)**: enter, space vagy escape gombra tudjunk kilépni a játékból.

#### EndWindow

Megjelenik, ha a béka az összes életét elvesztette.

**EndWindow(bool, bool)**: konstruktor, ami a következő két boolean értéket várja paraméterként:

**bool drowned**: igaz, ha a béka az idő lejárta miatt vesztette el utolsó életét. Ennek értékétől függ egy felirat megjelenése az ablakban.

**bool newhighscore**: igaz, ha új legmagasabb pontszám született. Ennek értékétől függ egy felirat megjelenése az ablakban.

**void CenterWindowOnScreen**: elhelyezi az ablakot a képernyő közepére.

**void WindowLoaded(…)**: képek és szöveg elhelyezése a felületen.

**void Button\_Click\_1(…)**: új játékot kezd.

**void Button\_Click\_2(…)**: exit gomb, kilép a játékból.

**void Button\_Click\_3(…)**: X (exit) gomb, kilép a játékból.

**Window\_PrewiewKeyDown(…)**: enterre új játékot kezd, escape gomb lenyomására kilép.

#### YouWinWindow

Megjelenik, ha a játékos teljesítette az 5. pályát is.

**YouWinWindow(bool, bool)**: konstruktor, ami a következő két értéket várja paraméterként:

**int score**: az elért pontszám

**bool newhighscore**: igaz, ha az elért pontszám új high score.

**void CenterWindowOnScreen**: elhelyezi az ablakot a képernyő közepére.

**void WindowLoaded(…)**: képek és szöveg elhelyezése a felületen.

**void Button\_Click\_1(…)**: exit gomb, kilép a játékból.

**void Button\_Click\_2(…)**: új játékot kezd.

**void Button\_Click\_3(…)**: X (exit) gomb, kilép a játékból.

#### SeaWindow (a főablak)

**game Game**: a játék osztály egy példánya.

**DispatcherTimer timer**: időzítő, ami a halak és tengeri állatok mozgásának kivitelezéséért felel.

**DispatcherTimer timer2**: időzítő, ami a pálya megoldására kiadott idő letelésért felel.

**int timetosolve**: a pálya megoldására kiadott idő, 30 mp.

**int remainingtime**: a pálya megoldására megmaradt idő, ami folyamatosan fogy, amikor a SeaWindow ablak fut.

**int score**: a játékos pontszáma (minden szintlépéskor változik).

**int highscore**: új legnagyobb eddig elért pontszám. Tárolása egy txt fájlban történik (highscore.txt).

**Brush charactercolour**: a SeaWindow ablakban elhelyezett feliratok színe.

**SeaWindow()**: konstruktor, ami akkor hívódik meg, ha a játék új score-ral, az első szinttől indul, 3 élettel.

**SeaWindow(int, int, int)**: konstruktor, ami akkor hívódik meg, mikor szintlépés történik. Paraméterként átadódik a következő szint (level) értéke, a béka életei (még a növelés előtt) és a játékos pontszáma (ez átadás előtt növelve van).

**void CenterWindowOnScreen()**: elhelyezi az ablakot a képernyő közepére.

**void WindowLoaded(…)**: képek és feliratok elhelyezése az ablakban. Meghívja azokat a függvényeket, amik az állatokat adják hozzá a pályához és beállítja a megfelelő értékekre az időzítőket.

**void timer\_Tick(…)**: ez az eseménykezelő felel a halak és a tengeri állatok mozgásáért, illetve minden Tick-re leellenőrzi, hogy ne ütközik-e a béka valamelyik tengeri állattal. Ha igen, de a békának még van élete, akkor a DeadFrogWindow töltődik be; ha nincs több élete, akkor az EndWindow (ezzel együtt pedig vizsgálva van a highscore is). Az eseménykezelő még azt is vizsgálja, hogy a béka nem érte-e el az utolsó sort (alulról nézve): ha igen, akkor betöltődik az új szintű pálya, ha pedig már a nemlétező 6. pálya jönne, akkor a játékos nyert, így betöltődik a YouWinWindow.

**void timer2\_Tick(…)**: eseménykezelő, ami visszaszámol a 30 mp-ből. Ha 10 mp alatt jár az idő, akkor ennek a feliratát pirossal jeleníti meg (majd később visszaállítja azt). Ha lejárt az idő, de a békának még van élete, akkor a DeadFrogWindow töltődik be; ha nincs több élete, akkor az EndWindow (ezzel együtt pedig vizsgálva van a highscore is).

**void FrogLives()**: a béka életeinek megfelelően jeleníti meg a békaéleteket ábrázoló canvasokat. Ezek maximális száma 7 lehet, mert 3 élettel kezd a játékos és minden szintlépéskor kap egy új életet.

**int AddScore()**: minden szintlépéskor számolódó pontszám, ami egy-egy szorzással képződik a béka életeiből és a fennmaradó időből.

**int AddLevelScore()**: a béka végső halálakor a játékos aktuális pályájának értékéből számol egy pontszámot, amit majd a score-hoz kell adni.

**void ReadingHighScore()**: beolvassa a highscore.txt-ből az aktuális legmagasabb pontszámot és értékül adja ezt a highscore változónak.

**void ModifyHighScore()**: mivel a függvény meghívása előtt a highscore változó értéke már változtatva lett, ezért egyszerűen csak törli a txt fájlt és létrehoz egy újat az új high score-ral.

**Random rand**: randomszám-generátor.

**void AddFrog()**: adatkötésekkel együtt létrehozza a békát.

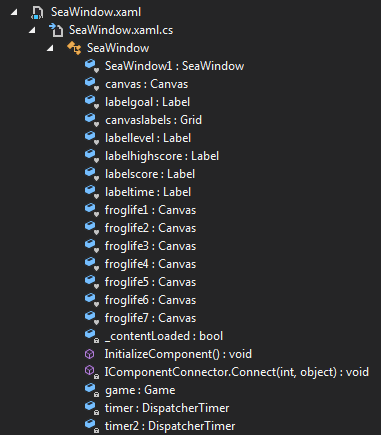
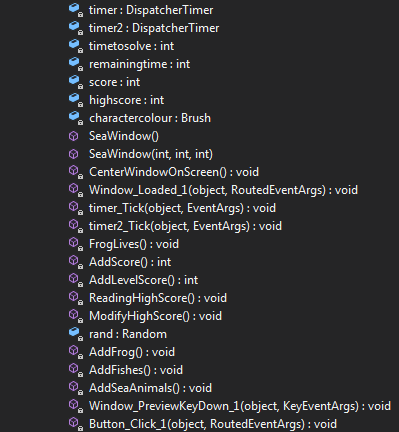
**void AddFishes()**:a Fishes listán végigmegy és ez alapján létrehoz mindegyik állatnak egy Rectangle-t. A randomszám-generátor és az állatok haladási irányának segítségével tölt be a szereplőknek egy-egy képet. Ez a függvény felel az adatkötésekért is.

**void AddSeaAnimals()**: a SeaAnimal listán végigmegy és ez alapján létrehoz mindegyik állatnak egy Rectangle-t. A randomszám-generátor és az állatok haladási irányának segítségével tölt be a szereplőknek egy-egy képet. Ez a függvény felel az adatkötésekért is.

**Window\_PrewiewKeyDown(…)**: ha még nincs vége a játéknak, akkor a lenyomott iránybillentyűtől függően továbbítja a békának a megfelelő irányt.

**void Button\_Click\_1(…)**: X (exit) gomb, új játékot kezd.

**A SeaWindow struktúrája:**

****

# Felhasználói segédlet

A **játék célja**, hogy a béka az **öt pályán** végigugráljon úgy, hogy ezt minél kevesebb idő alatt és minél több életet megtartva tudja megtenni. Ezekből minden játék végén **pontszám (score)** generálódik, amiből (az aktuális gépen a legnagyobb) nyilván van tartva a **highscore**.txt nevű fájlban. Ez az érték látszódik a SeaWindow fejlécen is.

A játék közben (SeaWindow ablak) **négy billentyű aktív**: a négy irány billentyű, amivel a béka mozgatható. Más állatot nem lehet mozgatni.

A halakkal való ütközés nem jár veszéllyel, a tengeri állatokkal való ütközés igen: a béka életet veszít. A **béka életei**t az alsó fejlécben láthatjuk. Ha ezek elfogytak, a játéknak vége van.   
**30 mp** áll rendelkezésre egy pálya teljesítésére – ha nem sikerül végigjutni, akkor a béka megfullad, egy élettel kevesebbje lesz.

Kevésbé bő felhasználói segédlet található a menü (MainWindow) **help menüpontjában.**