1. Beadandó feladat dokumentáció

**Készítette:**

Palotai Richárd

E-mail: [g4jd80@inf.elte.hu](mailto:g4jd80@inf.elte.hu)

**Feladat:**

Készítsünk programot, amellyel a következő két személyes játékot lehet játszani.

Adott egy 𝑛 × 𝑛 mezőből álló tábla, amelyen két játékos űrhajói helyezkednek el,

középen pedig egy fekete lyuk. A játékos 𝑛 − 1 űrhajóval rendelkezik, amelyek

átlóban helyezkednek el a táblán (az azonos színűek egymás mellett, ugyanazon

az oldalon).

A játékosok felváltva léphetnek. Az űrhajók vízszintesen, illetve függőlegesen

mozoghatnak a táblán, de a fekete lyuk megzavarja a navigációjukat, így nem egy

mezőt lépnek, hanem egészen addig haladnak a megadott irányba, amíg a tábla

széle, a fekete lyuk, vagy egy másik, előtte lévő űrhajó meg nem állítja őket (tehát

másik űrhajót átlépni nem lehet). Az a játékos győz, akinek sikerül űrhajóinak

felét eljuttatnia a fekete lyukba.

A program biztosítson lehetőséget új játék kezdésére a pályaméret megadásával

(5 × 5, 7 × 7, 9 × 9), valamint játék mentésére és betöltésére. Ismerje fel, ha vége

a játéknak, és jelenítse meg, melyik játékos győzött.

**Elemzés:**

* A játékot ketten lehet játszani egy kék és egy piros játékossal. A játék idő mindig a pálya méretének tízszerese percben, azaz n\*10 perc. A program elinduláskor automatikusan egy 3x3-as pályát generál.
* A feladatot egyablakos asztali alkalmazásként Windows Presentation Foundation grafikus felülettel valósítjuk meg.
* Az ablakban elhelyezünk egy menüt a következő menüpontokkal: File (New Game, Save Game, Load Game Quit Game), Settings (Set Map Size 🡪 Text Field). Új pályát két féle képpen is létre lehet hozni: (1. File 🡪 New Game ; 2. Settings 🡪 Set Map Size 🡪 Text Field + Enter). Az ablak alján megjelenítünk egy státuszsort, amely a lépések számát, illetve a hátralévő időt jelzi.
* A játéktáblát egy n x n nyomógomból álló rács reprezentálja. A nyomógomb egérkattintás hatására kijelölődik, közvetlen ezután W, A, S vagy D gombok megnyomására léptetődnek a mezőn álló hajók. Csak olyan hajót lehet kiemelni és léptetni ami az aktuális játékos színével van jelölve, mást nem is enged a program. A feketejuk mezőt nem lehet kijelölni se léptetni, ugyanez érvényes az üres mezőkre.
* A játék automatikusan feldob egy dialógusablakot, amikor vége a játéknak (nyert az egyik játékos, vagy letelt az idő). Szintén dialógusablakkal végezzük el a mentést, illetve betöltést, a fájlneveket a felhasználó adja meg.
* A felhasználó esetek az 1. ábrán láthatók.

A diagram of a diagram

Description automatically generated

1.ábra: Felhasználói esetek diagramja

**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 modellt, 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 **View** csomagok a WPF

függő projektjében kapnak helyet.

A diagram of a application

Description automatically generated

2. ábra: Az alkalmazás csomagdiagramja

• Perzisztencia(3. ábra):

• Az adatkezelés feladata a Fekete lyuk táblával kapcsolatos információk

tárolása, valamint a betöltés/mentés biztosítása.

• A BlackHoleMap osztály egy érvényes Black Hole táblát biztosít, ahol minden mezőre ismert az értéke

(\_fieldValues). A tábla alapértelmezés

szerint 3 × 3-as, de ez a konstruktorban

paraméterezhető. A tábla lehetőséget az állapotok lekérdezésére

(IsFieldShip, IsFieldEmpty, IsFieldBlackHole, GetFieldValue), valamint szabályos

léptetésre (StepFieldValue), illetve direkt beállítás (SetFieldValue)

elvégzésére.

• A hosszú távú adattárolás lehetőségeit az IBlackHoleDataAccess interfész

adja meg, amely lehetőséget ad a tábla betöltésére (LoadAsync), valamint

mentésére (SaveAsync). A műveleteket hatékonysági okokból aszinkron

módon valósítjuk meg.

• Az interfészt szöveges fájl alapú adatkezelésre a BlackHoleFileDataAccess

osztály valósítja meg. A fájlkezelés során fellépő hibákat a

BlackHoleDataException kivétel jelzi.

• A program az adatokat szöveges fájlként tudja eltárolni, melyek az .bgh

kiterjesztést kapják. 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 fájl többi sora pedig space-el elválasztva egy mátrixot ad meg ahol 1 a piros és 2 a kék játékos hajóit jelöli, valamint 3 a fekete lyukat, illetve 0 az üres mezőket.

A screenshot of a computer

Description automatically generated

3.ábra: A Persistence csomag osztálydiagramja

• Modell(4. ábra):

• A modell lényegi részét a BlackHoleGameModel osztály valósítja meg, amely

szabályozza a tábla tevékenységeit, valamint a játék egyéb paramétereit,

úgymint az idő (\_gameTime) és az aktuális játékos (\_currentPlayer). A típus

lehetőséget ad új játék kezdésére (NewGame), valamint lépésre

(Step). Új játéknál megadható a kiinduló játéktábla is, különben

automatikusan generálódnak kezdő mezők. Az idő előreléptetését időbeli

lépések végzéséve (AdvanceTime) tehetjük meg.

• A mezők állapotváltozásáról a FieldChanged esemény tájékoztat. Az

esemény argumentuma (BlackHoleFieldEventArgs) tárolja a megváltozott

mező pozícióját.

• A játékállapot megváltozásáról (lépések száma, hátra lévő idő) a

GameAdvanced esemény, míg a játék végéről a GameOver esemény

tájékoztat. Az események argumentuma (BlackHoleEventArgs) tárolja a

győzelem állapotát, a lépések számát, valamint a játékidőt.

• A modell példányosításkor megkapja az adatkezelés felületét, amelynek

segítségével lehetőséget ad betöltésre (LoadGame) és mentésre

(SaveGame)

A screenshot of a computer

Description automatically generated

4.ábra: A Model csomag osztálydiagramja

Nézetmodell (5. ábra):

• A nézetmodell megvalósításához felhasználunk egy általános utasítás

(**DelegateCommand**), valamint egy ős változásjelző (**ViewModelBase**)

osztályt.

5Eseményvezérelt alkalmazások 2024/2025 őszi félév

• A nézetmodell feladatait a **BlackHoleViewModel** 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 játékmező számára egy külön mezőt biztosítunk (**BlackHoleField**), amely

eltárolja a pozíciót, szöveget, engedélyezettséget, valamint a lépés

parancsát (**ClickCommand és a View\_KeyDown**). A mezőket egy felügyelt gyűjteménybe

helyezzük a nézetmodellbe (**Fields**).

A diagram of a computer

Description automatically generated

5.ábra: A nézetmodell osztálydiagramja

**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.

**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.

A screenshot of a computer program

Description automatically generated

*6.ábra: A vezérlés osztálydiagramja*

**Tesztelés:**

• A modell funkcionalitása egységtesztek segítségével lett ellenőrizve a

**BlackHoleGameModelTest** osztályban.

• Az alábbi tesztesetek kerültek megvalósításra:

• **BlackHoleGameModelNewGameTest**: Új játék indítása, a mezők

Kitöltése.

• **BlackHoleGameModelStepTest**: Játékbeli lépés hatásainak ellenőrzése,

játék megkezdése előtt, valamint után. Több lépés végrehajtása azonos

játékmezőn, esemény kiváltásának ellenőrzése.

• **BlackHoleGameModelAdcanceTimeTest**: A játékbeli idő kezelésének

ellenőrzése, beleértve a játék végét az idő lejártával.

• **BlackHoleGameModelLoadTest**: A játék modell betöltésének tesztelése

mockolt perzisztencia réteggel.