7. Prototípus koncepciója

10 – itee_team

Konzulens:
Budai Péter

Csapattagok:

Elekes Tamás Csaba	E30C8Z	elekestamas22@gmail.com
Seres Márk Dániel	EUQ8V5	seres.dani@gmail.com
Rédey Bálint Attila	DAVRIZ	botvinnik09@gmail.com
Nagy András	VWBG06	nagyandrasgall@gmail.com
Fuksz Domonkos	GIT0NQ	fukszdomonkos@gmail.com

2014. március 31.

7. Prototípus koncepciója

7.0. Specifikáció változás

7.0.1. Módosítás

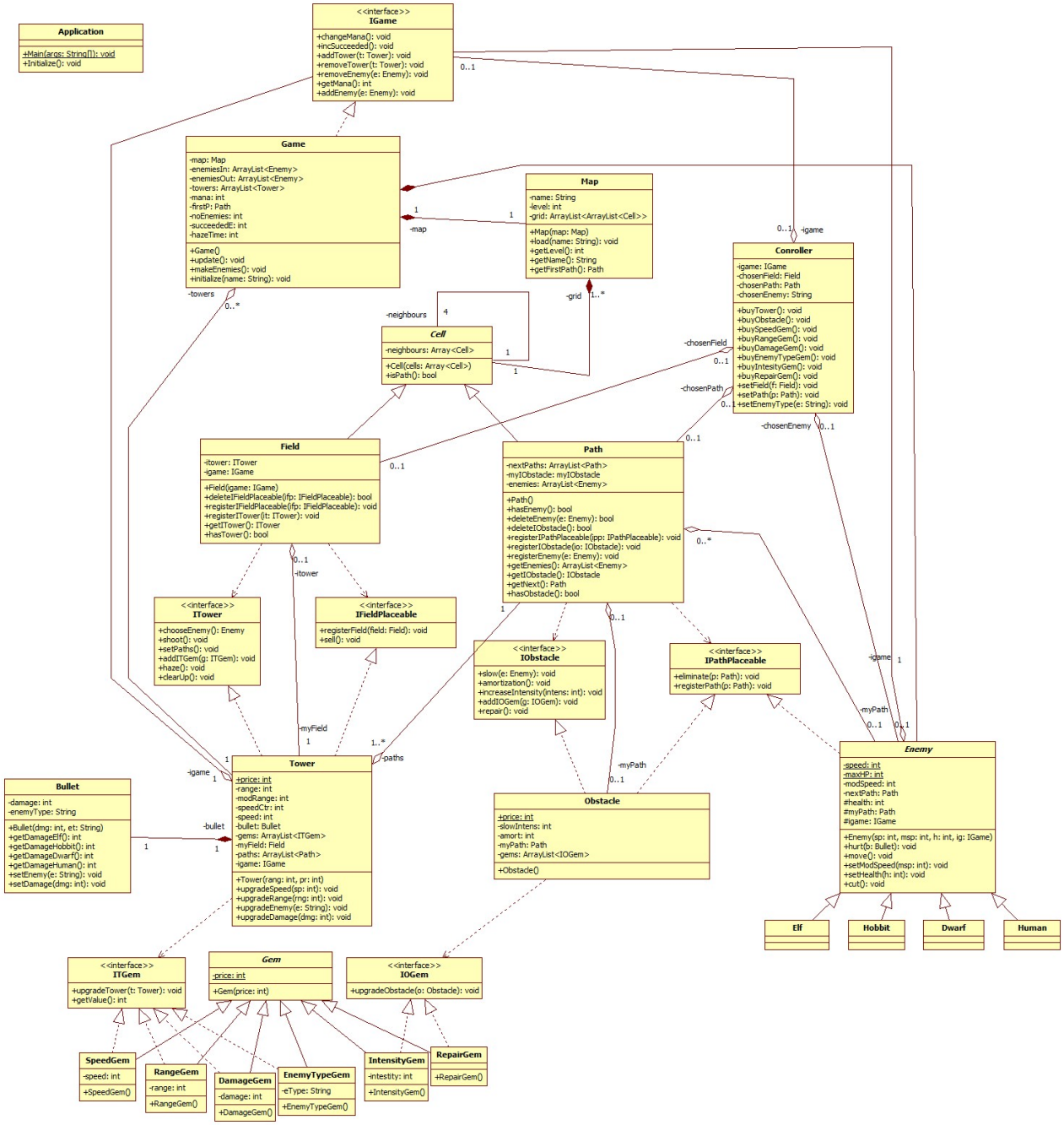
A tornyokra időnként köd ereszkedik, aminek következtében a látás erősen lecsökken. Ez hatással van a lövésre. A játékosok által járható útvonalon lehetnek elágazások és becsatlakozások. Az elágazásokon az egyes játékosok véletlenszerűen mennek a különböző irányokba. A tornyokban elvétve lehetnek olyan lövedékek, amelyek az eltalált játékost kettőbe vágják. A két játékos egymástól függetlenül él tovább, csökkentett életerővel.
A fentiek fényében a következő módosításokat hajtottuk végre:

- A köd torony hatósugarán változtat, így amíg a köd tart, addig a torony rövidebbre lát el, mint addig. Köd alatt is lehet fejleszteni a tornyot, ekkor ugyanúgy érvényre jut, de a köd által módosított hatókör fog nőni, majd a köd elmúltával olyan érték áll be, mintha nem lett volna köd és fejlesztették volna a tornyot. Köd véletlenszerűen keletkezik és minden tornyot érint. A köd időtartamát Game osztály hazeTime attribútuma szabja meg. Ez mindig ugyanannyi idő, az attribútum csak arról gondoskodik, hogy ha letelt, akkor visszaálljon minden (számol). A torony interfészébe bekerültek a haze() és clearUp() metódusok, előbbi ködbe borít, utóbbi kitisztít. A torony kapott egy modRange attribútumot, aminek segítségével kényelmesen kezelhető a ködbe borulás.

- Az elágazások és becsatlakozások tekintetében nem változtattunk, a modellen, mert az eddigiek alapján ezt már eddig is tudta. Vagyis amikor az Enemy továbblép, meghívódik a getNext() metódus, ami a következő Path címével tér vissza annak a Path-nak a nextPaths listájából, amelyikről lépni akar. Ha a listában több Path van, az pont az elágazásnak felel meg, ekkor véletlenszerűen adunk egy címet.

- Amikor egy Enemy-n meghívjuk a hurt metódust, átadunk egy Bulletet, amin az adott típusú Enemy meghívja a rá jellemző getDamage metódust, hogy megtudja a sebzést. Időnként 0-t kap majd, aminek hatására a meghívódik az Enemy-ben felvett cut() metódus, ami létrehoz egy ugyanilyen típusú Enemy-t (felüldefiniálás révén), aminek feleakkora életerőt ad, mint amennyi neki van, valamint a sajátját is felezi, továbbá minden egyéb attribútum értékét átmásolja az „ikertestvérébe”. Ezen felül, mivel az „ikertestvér”nek is be kell kerülnie a pályán lévő enemy-k közé, hogy vezérelhessük, az IGame interfészbe tettünk egy addEnemy fv-t, hogy az Enemyi betehesse az új Enemy-t.

7.0.2. Módosított osztálydiagram

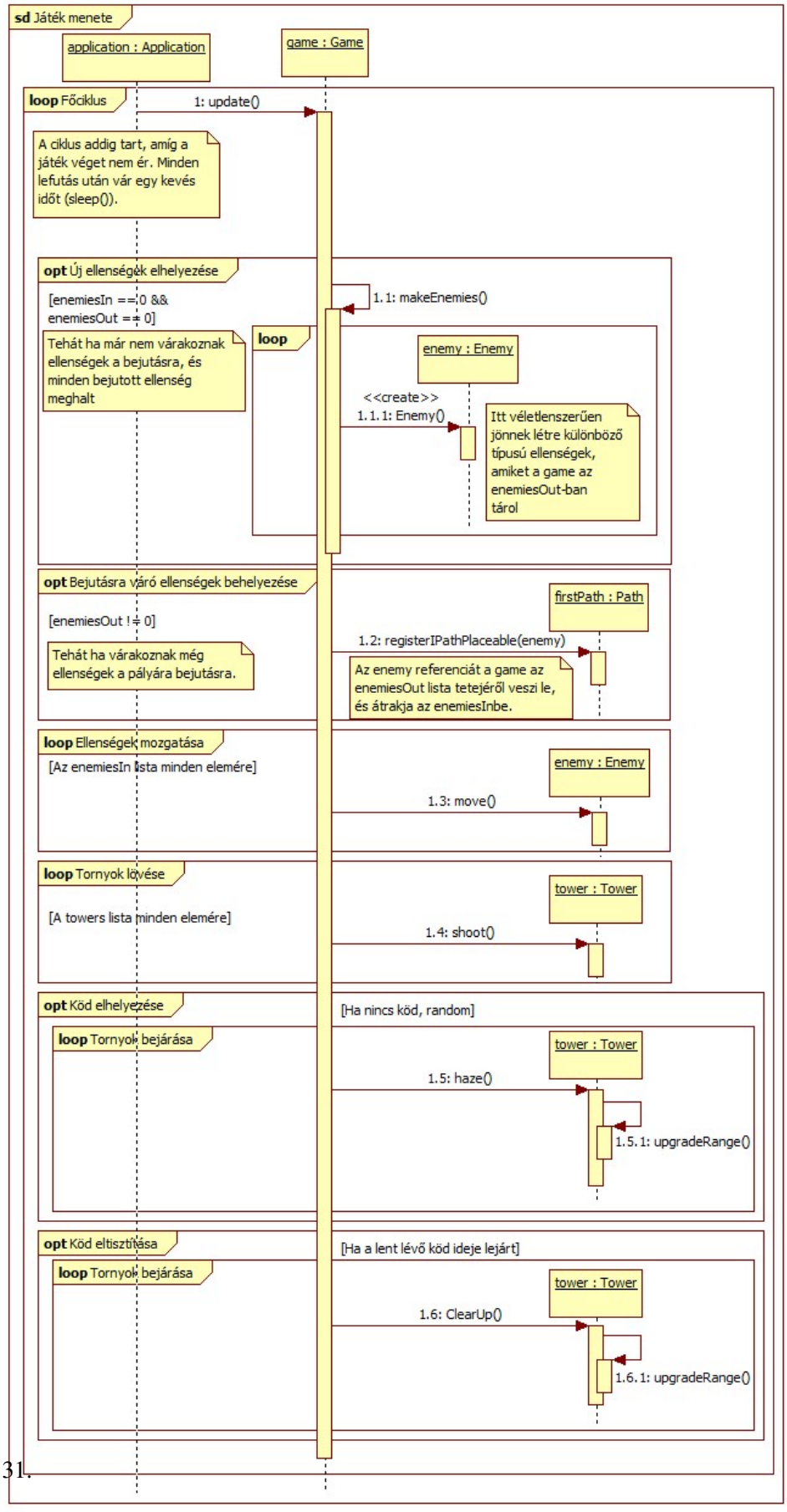


7.1. ábra. Módosított osztálydiagram

7.0.3. Módosított szekvenciadiagramok

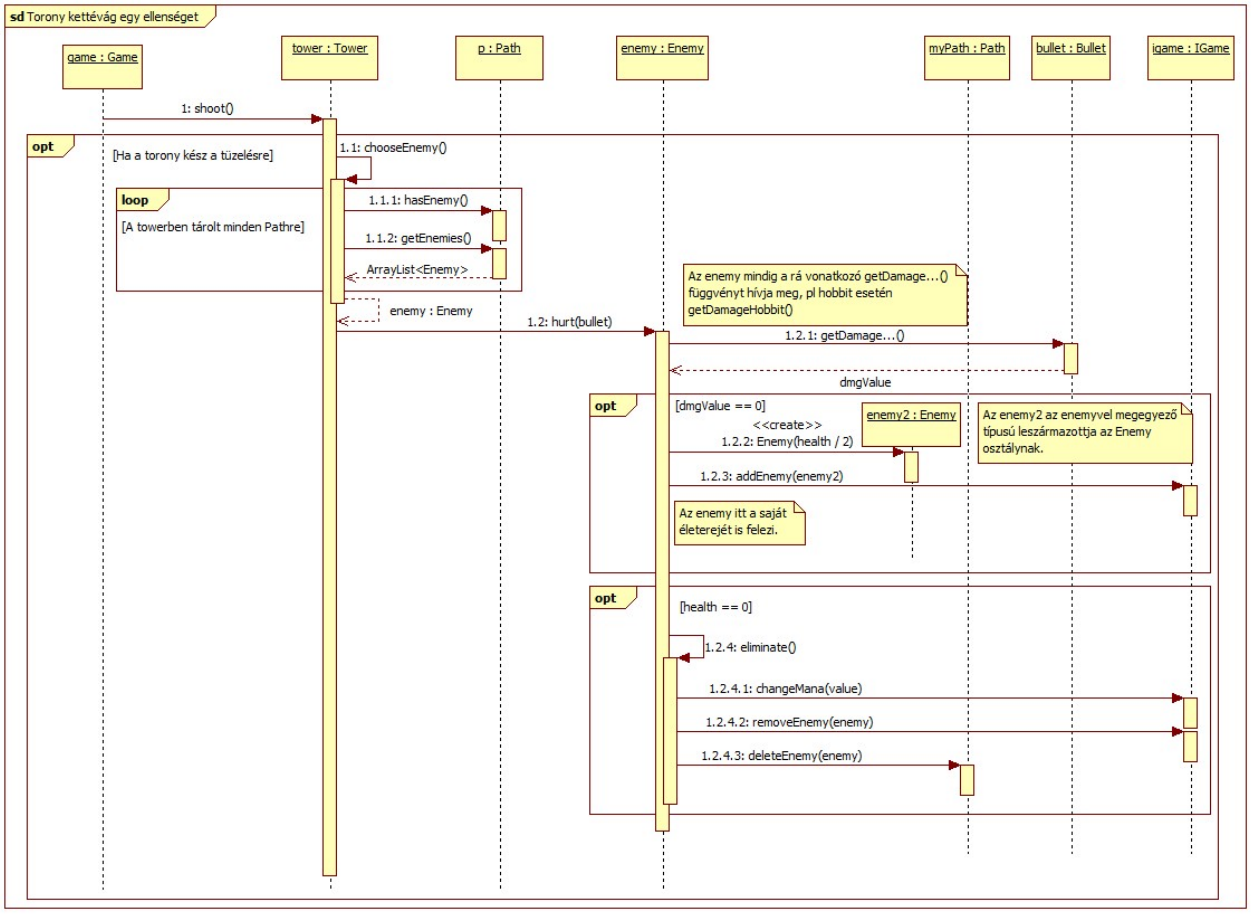
A módosított diagramok alapjai az analízis modell szekvenciadiagramjai.

Játék menete



7.2. ábra. Játék menete szekvenciadiagram

Torony kettévág egy ellenséget



7.3. ábra. Torony kettévág egy ellenséget szekvenciadiagram

7.1. Prototípus interface-definíciója

7.1.1. Az interfész általános leírása

Az interfész teljesen karakteres alapú, a szabványos ki- és bemenetet használja. Indítás után a 7.1.2 pontban leírt parancsokat lehet megadni, és a 7.1.3 pontban leírt módon kapjuk a kimenetet.
Lehetséges a ki- és bemenet átirányítása is, így például előre összeállított teszteseteket elmenthetünk fájlban, és átirányítással megadhatjuk a programnak, továbbá a kimenetét is fájlba irányíthatjuk a könnyebb elemzés érdekében.
Az átirányítás a hagyományos módon történik, például a

```
java „osztálynév” < bemenet.txt > kimenet.txt
```

parancs hatására a program megkapja soronként a bemenet.txt tartalmát, és a kimenet.txt fájlba írja a kimenetét.

7.1.2. Bemeneti nyelv

A bemeneti nyelv azoknak az utasításoknak a halmazát jelenti, amikkel a Prototípust vezérelni, és ezáltal tesztelni lehet.

Egy sorban 1 utasítás megadására van lehetőség. Az üres sorokat a parser figyelmen kívül hagyja. Minden parancs esetén a paramétereket szóközzel elválasztva kell felsorolni. Az utasítás végét nem kell pontosvesszővel lezárni, mert minden utasítás új sorba kerül.

- loadmap

  Leírás: Betölti a pályát.

  Opciók: #1 → filename

  Példa: loadmap tesztpalya.txt

- update

  Leírás: Az idő múlását vezérli. Egy egész szám a paramétere, hogy hány tick fusson le.

  Opciók: #1 → [0-9]+

  Példa: update 10

- draw

  Leírás: Kirajzolja a pályát, és a fontosabb adatokat:

  - mennyi manája van a játékosnak
  - egy torony mennyibe kerül
  - egy akadály mennyibe kerül
  - egy kristály mennyibe kerül

  Opciók: –

  Példa: draw

- buy

  Leírás: Objektumok (torony, akadály) elhelyezése a pályán, és fejlesztésük Gemekkel.

  Opciók: #1 → [tower | obstacle | gem] (mit veszünk)

  #2 → x-y (hova vesszük)

  #3 → [speed | range | damage | enemy | inensity | repair] (milyen gem)

  #4 → [elf | hobbit | dwarf | human] (milyen ellenségre)

  3. paraméter csak Gem esetén létezik, míg 4. paraméter csak Enemy-Gem esetén.

  Példa:
  - buy tower 5-3
  - buy obstacle 10-12
  - buy gem 5-3 enemy hobbit

- random

  Leírás: A véletlenszerűséget tudja ki és bekapcsolni.

  Opciók: #1 → [true | false]

  Példa: random true

- exit

  Leírás: Kilép a programból.

  Opciók: –

  Példa: exit

A pálya formátuma

A pályát karakterekkel rajzoljuk ki. Szabályok: pálya széle mindenhol "field", kivéve "entry point" és "exit point"

x field

en entry point

r right

l left

u up

d down

ru right, up

rl right, left

rd right, down

ul up, left

ud up, down

dr down, right

rul right, up, left

rud right, up, down

rdl right, down, left

dul down, up, left

ex exit point

7.1.3. Kimeneti nyelv

Egy bemeneti parancs után ha a programnak lényeges függvénye hívódik meg, kiírja azt.
Ha helyénvaló, az objektum katalógus beli nevét is kiírja.
Ha több függvényhíváson keresztül ír ki, minden függvény mélyén egy tabbal beljebb ír ki.

Bemenet: loadmap
Kimenet: nincs

Bemenet: update <n>
Lehetséges kimenetek:

- update — minden frissítés elején kiírja
- makeEnemies — ha a pályán kívül várakozó és bent lévő ellenségekről tárolt tömb is üres.
- *katalógus név* created — egyes ellenségek konstruktorában kiírja a katalógus nevét
- katalógus név move — ha egy ellenség lép, ezt írja ki

- katalógus név crossroad — ha ellenségelágazáshoz ért
- katalógus név shoot katalógus név target — ha egy torony ellenségre lő, ezt írja ki
- haze — amikor köd ereszkedik a tornyokra

Példa:

```
update 2
update
        makeEnemies
                Hobbit0 created
                Hobbit1 created
                Dwarf0 created
update
        Hobbit0 move
        Hobbit0 crossroad
```

Bemenet: draw
Kimenet:

- Fentebb definiált módon, táblázatosan kirajzolja a pályát. A pálya egy celláját egy számpár határozza meg, az oszlop és sor koordinátája.
- Your mana: n – a játékos manája
- Tower price: n – egy torony ára
- Obstacle price: n – egy akadály ára
- Gem price: n – egy kristály ára

Példa:

```
draw
        0       1       2       3       4       5
0       x       x       x       x       x       x
1       en      r       rd      r       d       x
2       x       x       d       x       d       x
3       x       x       d       x       d       x
4       x       x       r       r       r       ex
5       x       x       x       x       x       x
Your mana: 100
Tower price: 20
Obstacle price: 10
Gem price: 50
```

Bemenet: buy <tpye><coord><gem><enemy>
Lehetséges kimenetek:

- katalógus név created – a létrejövő objektum kiírja a katalógusban tárolt nevét
- cell occupied – ha a kapott koordinátán nem lehet elhelyezni az objektumot

Példa1:

```
buy tower 5-3
Tower0 created
```

Példa2:

```
buy gem 5-3 enemy hobbit
HobbitGem0 created
```

Bemenet: random
Kimenet: nincs

Bemenet: exit
Kimenet: nincs

7.2. Összes részletes use-case

Use-case neve	**Játék inicializálása**
Rövid leírás	A játék indulása után, inicializálja a változókat.
Aktorok	Application, Game
Forgatókönyv	

Use-case neve	**Ellenség lépése**
Rövid leírás	Egy ellenség az egyik celláról a másikra lép.
Aktorok	Game
Forgatókönyv	Az enemy elkéri a cellájától a következő cellát, törölteti magát az aktuális celláról, és beregisztráltatja magát a következőre.

Use-case neve	**Ellenség meghal**
Rövid leírás	Egy ellenség meghal, tehát teljesen eltűnik.
Aktorok	Game
Forgatókönyv	Az enemy törölteti magát az összes enemy listájából (Game), és az aktuális cellájából is.

Use-case neve	**Ellenség bejut a végzet hegyéhez**
Rövid leírás	Egy ellenség bejut a végzet hegyéhez. A játék vége.
Aktorok	Game
Forgatókönyv	Annyiban tér el a sima mozgástól, hogy a Game ellenőrzi, hogy Szarumán varázslata elfogyott-e már, és ha igen, akkor véget ér a játék.

Use-case neve	**Ellenség elágazáshoz ér**
Rövid leírás	Egy ellenség egy út-celláról legalább 2 másik cellára léphet tovább.
Aktorok	Game
Forgatókönyv	Az ellenség mozgásának egy speciális esete. Amikor az enemy elkéri a Path-tól a következő cellát, akkor beállítástól függően vagy determinisztikusan a listában szereplő első cellát kapja meg, vagy veletlenszerűen a listából egyet.

Use-case neve	**Torony megvétele**
Rövid leírás	Üres mezőre tornyot vesz a felhasználó.
Aktorok	Controller, Tower
Forgatókönyv	Először a Controller setField metódusával kiválaszt a felhasználó egy mezőt amire lerak egy tornyot.

Use-case neve	**Toronyra kristály vétele**
Rövid leírás	Torony fejlesztése kristállyal.
Aktorok	Controller, Tower
Forgatókönyv	A felhasználó kiválaszt egy mezőt, és ha van rajta torony fejleszti egy kiválasztott kristállyal.

Use-case neve	**Torony eladása**
Rövid leírás	Torony eladása.
Aktorok	Controller, Tower
Forgatókönyv	A felhasználó kiválaszt egy mezőt, és a rajta lévő tornyot eladja.

Use-case neve	**Toronyra köd ereszkedik**
Rövid leírás	Toronyra köd ereszkedik, ami a hatósugarát lecsökkenti.
Aktorok	Tower, Game
Forgatókönyv	A Game kiválaszt egy tornyot, amire ködöt rak. Lecsökken a hatósugara a toronynak, ezért frissíti az út cellák listáját. (setPath)

Use-case neve	**Torony lő**
Rövid leírás	Egy, a torony a hatósugarában lévő ellenségre tüzel az ellenség.
Aktorok	Tower, Enemy
Forgatókönyv	A torony lekér a hatósugarában lévő mezők közül egy ellenséget, amire kilövi a bullet-jét.

Use-case neve	**Akadály megvétele**
Rövid leírás	Egy üres útra akadályt vesz a játékos.
Aktorok	Controller, Obstacle
Forgatókönyv	Először a Controller setPath metódusával kiválaszt a felhasználó egy mezőt amire lerak egy tornyot.

Use-case neve	**Akadály lassít**
Rövid leírás	A lerakott akadály lassítja a rajta áthaladó ellenséget.
Aktorok	Path
Forgatókönyv	Amikor egy ellenség egy akadállyal terhelt cellára lép, akkor a Path beállítja az enemy modSpeed-jét.

Use-case neve	**Akadály elromlik**

Rövid leírás	Akadály elromlik
Aktorok	Path
Forgatókönyv	A path törli az akadályt.

Use-case neve	**Akadály eladás**
Rövid leírás	A játékos elad egy akadályt.
Aktorok	Controller
Forgatókönyv	A felhasználó kiválaszt egy utat, és a rajta lévő akadályt eladja.

7.3. Tesztelési terv

Teszt-eset neve	**Ellenség hullám indítása**
Rövid leírás	A Game létrehoz 3 ellenséget, a játékos vesz 2 tornyot, és mindkettőt fejleszti 1-1 kristállyal. Ez után utukra indítja az ellenségeket. Végén elad a játékos egy tornyot.
Teszt célja	Ez a teszteset az ellenség hullám létrehozását, tornyok vételét, fejlesztését, tüzelését és eladását teszteli. Ebben a tesztesetben fogunk látni ellenség halálát is. A játékos nyer.

Teszt-eset neve	**Akadály és ellenség bejut a végzet hegyéhez**
Rövid leírás	A pályán elhelyez a játékos egy akadályt, amit fejleszt egy intenzitás növelő kristállyal, majd bejut az ellenség a végzet hegyéhez. Amikor az utolsó ellenség is áthaladt rajta elhasználódik.
Teszt célja	A teszteset akadály vételt, fejlesztést, működést és elhasználódást, teszteli, illetve a végzet hegyéhez való bejutást. A játékos veszít.

Teszt-eset neve	**Köd és ellenség kettészelés**
Rövid leírás	A pályára feltesz a Game 3 ellenséget, a játékos vesz 3 tornyot, amikor beér az első ellenség a torony hatósugarába, köd száll a tornyokra, és kettőbe vágó lövedékeket lőnek onnantól. Miközben végigérnek az ellenségek a pályán, meghalnak.
Teszt célja	Ez a teszteset a tornyokra ereszkedő ködöt teszteli és a kettészelő lövedékeket.

Teszt-eset neve	**Ellenség elágazáshoz ér**
Rövid leírás	A pályára feltesz a Game 3 ellenséget. A pálya kialakításából adódóan 1 lépés után elágazáshoz érnek.
Teszt célja	A teszteset azt teszteli elágazásoknál merre mennek tovább az ellenségek. Ha a véletlenszerűség be van kapcsolva a programban, akkor egy véletlen utat választ, ha nem akkor a 0. elemét adja vissza az utakat tároló tömbnek.

7.4. Tesztelést támogató segéd- és fordítóprogramok specifikálása

A prototípustól adott bemenet esetén elvárt, illetve a kapott eredmények összehasonlítását saját programmal végezzük. Az összehasonlító program parancssorból futtatható, két paramétert vár: a várt és a kapott eredményeket tartalmazó fájlok neveit. A program lefutása után megmondja, hogy a két fájl megegyezik-e, és ha nem, akkor kiírja az eltérő sorok sorszámát.
A program futtatása a következő parancs kiadásával történik, a lefordított .class file mellett:

```
java TxtComparer <elso_file_neve> <masodik_file_neve>
```

7.5. Napló

Kezdet	Időtartam	Résztvevők	Leírás
2014.03.26. 08:15	1,5 óra	**Elekes Fuksz Seres Nagy Rédey**	Konzultáció
2014.03.27. 15:00	0,5 óra	**Elekes**	Torony use-casek
2014.03.27. 17:00	1 óra	**Rédey**	7.0.1 pont dokumentálása, 7.0.2 osztálydiagram módosítása
2014.03.28 18:00	1 óra	**Seres**	Módosításhoz tartozó szekvencia diagramok elkészítése
2014.03.29. 14:00	2 óra	**Elekes Fuksz Seres Nagy Rédey**	Skype konferencia bemeneti/kimeneti nyelvekről, pálya térképéről
2014.03.30 17:00	1 óra	**Elekes**	Kimeneti nyelv specifikálása, teszt-esetek megírása.
2014.03.28 19:00	1 óra	**Nagy**	Enemy és Obstacle use-case-ek megírása
2014.03.30 15:30	1,5 óra	**Nagy**	Bemeneti nyelv specifikálása
2014.03.30 22:00	10 perc	**Seres**	7.4-es pont kitöltése
2014.03.31. 00:30	2,5 óra	**Fuksz**	Dokumentáció véglegesítése