Szoftver labor 4. házi feladat

27 – such_team

Konzulens:
Szabó Ádám Imre

Csapattagok:
Nusser Ádám P7PY3B nusee0726@hotmail.com
Szabó Antal R75PCE szabo.antal.92@gmail.com
Tallér Bátor UGA4OQ tallerbator@gmail.com
Török Attila FULGDF torokati44@gmail.com

2014. május 16.

Tartalomjegyzék

Ábrák jegyzéke

2. Követelmény, projekt, funkcionalitás

2.1. Bevezetés

2.1.1. Cél

A projekt követelményeinek, alapvető felépítésének és funkcionalitásának ismertetése; ezek segítségével körbehatárolni a fejlesztés menetét és a végleges program felépítését, működését. Fejlesztés közben ezeket folyamatosan figyelembe kell majd venni, eltérni tőlük nem szabad.

2.1.2. Szakterület

Az elkészítendő szoftver egy számítógépes játék, így nem egy kifejezett szakterület részére készül, hanem általános felhasználásra, szórakoztatásra. A játék az úgynevezett „tower defense" kategóriába sorolható, ezért azoknak ajánlott, akik szeretik az ilyen típusú játékokat.

2.1.3. Definíciók, rövidítések

architekturális kép A szoftver belső felépítését szemléltető ábra.

archívum Olyan számítógépes fájl, amely több másik fájlt foglal magában.

BME Budapest Műszaki Egyetem rövidítése.

Eclipse Fejlesztőkörnyezet főként a Java programozási nyelvhez.

Enterprise Architect Egy UML diagramok készítését lehetővé tevő szoftver.

Facebook Internetes közösségi oldal

fejlesztőkörnyezet Olyan számítógépes program vagy programok összessége, ami lehetővé teszi vagy leegyszerűsíti egy fejlesztési munka végrehajtását.

funkció A program működésének egy külön megfogalmazható része.

Git Egy verziókezelő rendszer.

GitHub A Git verziókezelő rendszerre épülő internetes szolgáltatás.

HSZK Hallgatói Számítógép Központ rövidítése.

háttértár A számítógépnek egy olyan tárhelye, ami a számítógép kikapcsolása után is megőrzi az adatokat.

IntelliJ IDEA Fejlesztőkörnyezet főként a Java programozási nyelvhez.

jar A Java programozási nyelv által használt fájltípus; egy archívum, amiben egy adott program vagy programmodul futtatásához szükséges fájlok vannak.

JRE6 Java Runtime Environment 6 rövidítése. Ez egy olyan program, ami szükséges a Java programozási nyelvben írt programok használatához.

PC Personal Computer rövidítése, jelentése személyi számítógép.

proto/prototípus A program olyan állapota, amikor minden belső működés meg van valósítva és működik, de grafikus felület még nincsen hozzá.

Skype Egy interneten keresztüli telefonálásra használható program.

szkeleton A program olyan állapota, amikor a program belső felépítése készen van, de nem csinál semmit.

szoftver Számítógépen futtatható program.

TeX Dokumentum formázó és betűszedő rendszer, segítségével magas színvonalú szöveges dokumentumok hozhatók létre (pl. ez a dokumentáció).

TeXworks Fejlesztőkörnyezet TeX-hez.

UML Unified Modeling Language rövidítése, egy rendszer modelező eszköz.

use-case Egy felhasználó és egy rendszer közötti, adott célt elérő interakció leírása.

verziókezelő Olyan számítógépes program, aminek segítségével eltárolható fájloknak régebbi verziói, így később vissza lehet térni egy adott verzióhoz, illetve nyomon lehet követni egy fájl változását.

wiki Egy olyan weboldal, aminek tartalmát a felhasználók szerkeszthetik, így mindenki hozzáadhatja a saját tudását egy adott témához.

2.1.4. Hivatkozások

Szoftver labor 4 - `https://www.iit.bme.hu/~szoftlab4/`

2.1.5. Összefoglalás

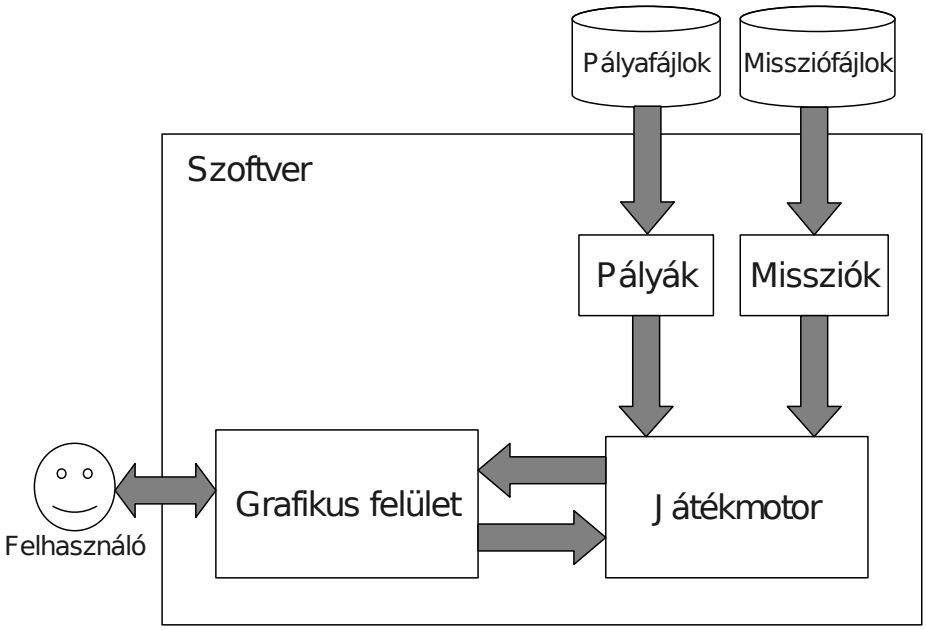
A dokumentum további részeiben található:

- Áttekintés, ami nagyvonalakban bemutatja a szoftvert
- A szoftverrel kapcsolatos követelmények leírása, később ezek figyelembevételével kell a programot fejleszteni
- Lényegesebb use-case-ek felsorolása
- Szótár, ami a szoftverrel kapcsolatos nem hétköznapi, illetve nem hétköznapi értelmében használt szavak definícióit tartalmazza
- A projekt kivitelezésének terve
- Projekt napló

2.2. Áttekintés

2.2.1. Általános áttekintés

A szoftver legmagasabb szintű architekturális képe:



2.1. ábra. Architekturális kep

A szoftver legfontosabb alrendszere a Játékmotor. Ebben zajlik a tulajdonképpeni játék, ez futtatja a logikát, ami által az egységek belépnek, és haladnak végig az utakon a Végzet Hegye felé, ez tartja számon a tornyokat, azok fejlesztéseit, ez gondoskodik arról, hogy minden torony a megfelelő egységet sebezze, a megfelelő módon és mértékben. A pályát (az utak elrendezését) a Pályák alrendszertől kéri le, és a forgatókönyvet, ami szerint belépnek az egységek, pedig a Missziók alrendszertől. A pillanatnyi állapotot, amely a kirajzoláshoz szükséges, szolgáltatja a Grafikus felület felé, valamint végrehajtandó parancsokat, vezérlést fogad attól.

A Pályák alrendszer egyszerűen betölti a Pályafájlokból a pályákat, nyilván tartja, és a Játékmotor rendelkezésére bocsátja őket.

A Missziók alrendszer a Pályák-hoz nagyon hasonlóan a Missziófájlokból betölti a missziókat, nyilván tartja, és a Játékmotor rendelkezésére bocsátja őket.

A Grafikus felület egyrészt a játék megjelenítését szolgálja a Felhasználó felé, másrészt ezen keresztül irányítható a Játékmotor.

Hálózatot a szoftver egyáltalán nem használ. Háttértáron pedig a program saját fájljai - .jar archivum(ok), a kirajzoláshoz szükséges képek, és az egyes pályákat, valamint missziókat tároló fájlok - számára kell helyet

biztosítani. Ezeken kívül nincs további futási idejű tárhelyigénye.

2.2.2. Funkciók

A szoftver egy úgynevezett tower defense játék. Az ilyen játékok lényege általában az, hogy a pályán lévő, kijelölt úton különböző ellenséges egységek indulnak el a belépési pontból a céljuk felé. A játékos feladata pedig az, hogy ezt megakadályozza azzal, hogy az út mentén valamilyen erőforrásokat felhasználva tornyokat épít, amelyek a hozzájuk elég közel kerülő egységeket bizonyos mértékben és időközönként sebzik, ezzel életerejüket folyamatosan csökkentik. Ha egy egység életereje elfogy, megáll, elpusztul, eltűnik, és további fenyegetést nem jelent.

A játék akkor ér véget, ha minden ellenséget elpusztítottunk, ezzel megvédtük a célt, vagy ha legalább egy ellenség célba ér. Előbbi esetben a játékos nyer, utóbbi esetben a játékos veszít, az ellenség pedig fényes győzelmet arat és örökké uralja a világot.

Ebben a konkrét esetben a játéktér a Középföldén elfekvő Mordorban található, az ellenségek az ádáz Gyűrű Szövetségének tagjai, akik lehetnek emberek, tündék, törpök vagy hobbitok, és céljuk a Végzet Hegye, amelyben az Egy Gyűrűt szándékoznak elpusztítani. Minden ellenségtípus eltérő tulajdonságokkal (sebesség, életerő, stb.) rendelkezik. Például egy törp lassabb, mint egy tünde, de több az életereje. Egy ember pedig gyorsabb egy törpnél, de szívósabb, mint egy tünde. A hobbitok pedig általában népes csoportokban érkeznek, így lassúságuk és kevés életerejük ellenére komoly kihívást jelenthetnek.

A játékos, aki a jóságos Szarumánt személyesíti meg, ezt nem hagyhatja, hiszen így felettese, Szauron is odaveszne. A védekező tornyok felépítéséhez varázserejét tudja felhasználni, amely minden ellenséges egység elpusztításakor bizonyos mértékben megnő.

Ezen kívül lehetősége van - szintén varázserejét elköltve - a tornyait különböző mágikus kövekkel ellátni, így azok bizonyos típusú ellenség esetén nagyobbat sebeznek. Más kövek növelik a tornyok tüzelési gyakoriságát vagy hatótávolságát.

Az utakra építhet akadályokat, amelyekre érve az egységek lelassulnak, így esetleg tovább tartózkodnak a tornyok hatókörében, azoknak több esélyük van megsebezni őket.

A tornyokon kívül a játékosnak lehetősége van akadályokat is felruházni varázskövekkel. A varázskövekkel felruházott akadályok jobban lelassítják a beléjük érkező ellenségeket. A lassítás mértéke függ az ellenség típusától.

Egyes pályákon több belépési pont és több, a célba vezető kijárat is szerepelhet, és azokat összetett úthálózat is összekötheti, amelyen az egységek véletlenszerű útvonalakat választanak ki. A támadás több hullámban történik, minden hullámban egy bizonyos időben minden típusú ellenségből meghatározott létszámú csoport jön, adott késleltetéssel egymáshoz képest. A hullámok leírását minden pályához missziók tartalmazzák, amelyek így lehetnek eltérő nehézségűek vagy hangulatúak.

2.2.3. Felhasználók

A szoftver felhasználói számára nincs szükség különösebb előképzettségre, a használata rövid időn belül elsajátítható, és a játék élvezhető bárki számára, aki a szótárban felsorolt definiciókat képes értelmezni.

2.2.4. Korlátozások

A szoftverre vonatkozó előírás és korlátozás is egyben egyedül az, hogy a standard Java könyvtárkészletet kell használnia, és semmilyen más csomagot nem vehet igénybe.

2.2.5. Feltételezések, kapcsolatok

Szoftver labor 4 feladatkiírás - `https://www.iit.bme.hu/~szoftlab4/feladat.shtml`

2.3. Követelmények

2.3.1. Funkcionális követelmények

Azonosító	Leírás	Ellenőrzés	Prioritás	Forrás	Use-case	Komment
1.01	Tornyokat lehet építeni	bemutatás	alapvető	megrendelő	Torony építése	
1.02	Az ellenségek igyekszenek eljutni a célig	bemutatás	alapvető	megrendelő		
1.03	A tornyok képesek lőni	bemutatás	alapvető	megrendelő		
1.04	Az ellenségek sebződnek a tornyok lövedékétől	kiértékelés	alapvető	megrendelő		
1.05	Az ellenségek bizonyos mértékű sebződés után meghalnak	bemutatás	alapvető	megrendelő		
1.06	Az elpusztított ellenségek után varázserőt kap a játékos	bemutatás	alapvető	megrendelő		
1.07	A különböző ellenségek, különböző tulajdonságokkal rendelkezeknek	bemutatás	opcionális	csapat		Különböző életerő, sebesség
1.08	Vannak utak	bemutatás	alapvető	megrendelő		
1.09	Az ellenségek az utakról nem térnek le	bemutatás	alapvető	megrendelő		
1.10	Tornyot nem lehet útra építeni	bemutatás	fontos	megrendelő	Torony építése	
1.11	Akadályokat lehet építeni	bemutatás	alapvető	megrendelő	Akadály építése	
1.12	Akadályt csak útra lehet rakni	bemutatás	fontos	megrendelő	Akadály építése	
1.13	Az akadályok lassítják az ellenségeket	bemutatás	fontos	megrendelő		
1.14	A tornyoknak van tüzelési gyakorisága, hatótávolsága	bemutatás	alapvető	megrendelő		
1.15	Tornyokat és akadályokat varázskövekkel lehet ellátni	bemutatás	alapvető	megrendelő	Torony/ akadály echantolás	
1.16	Különböző varázskövek vannak	bemutatás	opcionális	megrendelő		

1.17	Idővel egyre gyakrabban érkeznek ellenségek, egyre nagyobb csoportban	bemutatás	fontos	megrendelő		
1.18	A játékot el lehet indítani	bemutatás	alapvető	megrendelő	Új játék indítása	
1.19	A játékot fel lehet adni	bemutatás	opcionális	csapat	Feladás	
1.20	Több pálya van	bemutatás	opcionális	csapat	Pálya kiválasztása	
1.21	A játékból ki lehet lépni	bemutatás	alapvető	csapat	Kilépés	
1.22	Ha az összes ellenség meghalt, a játékos nyer	bemutatás	alapvető	csapat		
1.23	Több útvonal vezet a célig	bemutatás	alapvető	csapat		
1.24	Ha egy ellenség eljut a célig, a játékos veszít	bemutatás	alapvető	csapat		

2.3.2. Erőforrásokkal kapcsolatos követelmények

Azonosító	Leírás	Ellenőrzés	Prioritás	Forrás	Komment
2.01	Git	nincs	alapvető	csapat	Elosztott verziókezelő
2.02	Github account	nincs	alapvető	csapat	Git tárhely
2.03	JRE6	bemutatás	alapvető	megrendelő	
2.04	Eclipse	nincs	opcionális	csapat	Java IDE
2.05	IntelliJ IDEA	nincs	opcionális	csapat	Java IDE
2.06	HSZK-ban találhatókkal azonos vagy jobb teljesítményű PC	bemutatás	alapvető	megrendelő	
2.07	Monitor	nincs	alapvető	csapat	
2.08	Egér	nincs	alapvető	csapat	
2.09	Enterprise Architect	nincs	opcionális	csapat	UML modellező
2.10	TeXworks	nincs	opcionális	csapat	LaTeX editor

2.3.3. Átadással kapcsolatos követelmények

Azonosító	Leírás	Ellenőrzés	Prioritás	Forrás	Komment
3.01	Szkeleton átadás	bemutatás	alapvető	megrendelő	márc. 26.
3.02	Proto átadás	bemutatás	alapvető	megrendelő	ápr. 23.
3.03	Teljes program átadása	bemutatás	alapvető	megrendelő	máj. 14.
3.04	Útmutató alapján telepíthető, külső segítség nélkül	bemutatás	fontos	megrendelő	
3.05	A programnak működnie kell a BME HSZK számítógépein	bemutatás	alapvető	megrendelő	

2.3.4. Egyéb nem funkcionális követelmények

Nincs egyéb nem funkcionális követelmény.

2.4. Lényeges use-case-ek

2.4.1. Use-case leírások

Use-case neve	**Új játék indítása**
Rövid leírás	Új játék indul el
Aktorok	Játékos
Forgatókönyv	1. A játékos kiválasztja a menüben az „Új játék” feliratú gombot, és megnyomja. 2. Ezután kiválaszthatja a nehézséget és a pályát. 3. Elindul a játék a kiválasztott pályával és nehézséggel.

Use-case neve	**Pálya kiválasztása**
Rövid leírás	Ki kell választani a pályát amelyen játszani kíván.
Aktorok	Játékos
Forgatókönyv	Új játék indítása után egy listából ki kell választani a pályát amelyen játszani kíván.

Use-case neve	**Nehézségi szint kiválasztása**
Rövid leírás	Ki kell választani, hogy milyen nehézségen kíván játszani.
Aktorok	Játékos
Forgatókönyv	Új játék indítása, és a pálya kiválasztása után, ki kell választani három opció közül, hogy mennyire legyen a játék nehéz.

Use-case neve	**Torony építése**
Rövid leírás	Új tornyot épít
Aktorok	Játékos
Forgatókönyv	Olyan mezőre kattintva, ahol nincsen út vagy torony, ott új torony épül, ha van elég varázserő hozzá.

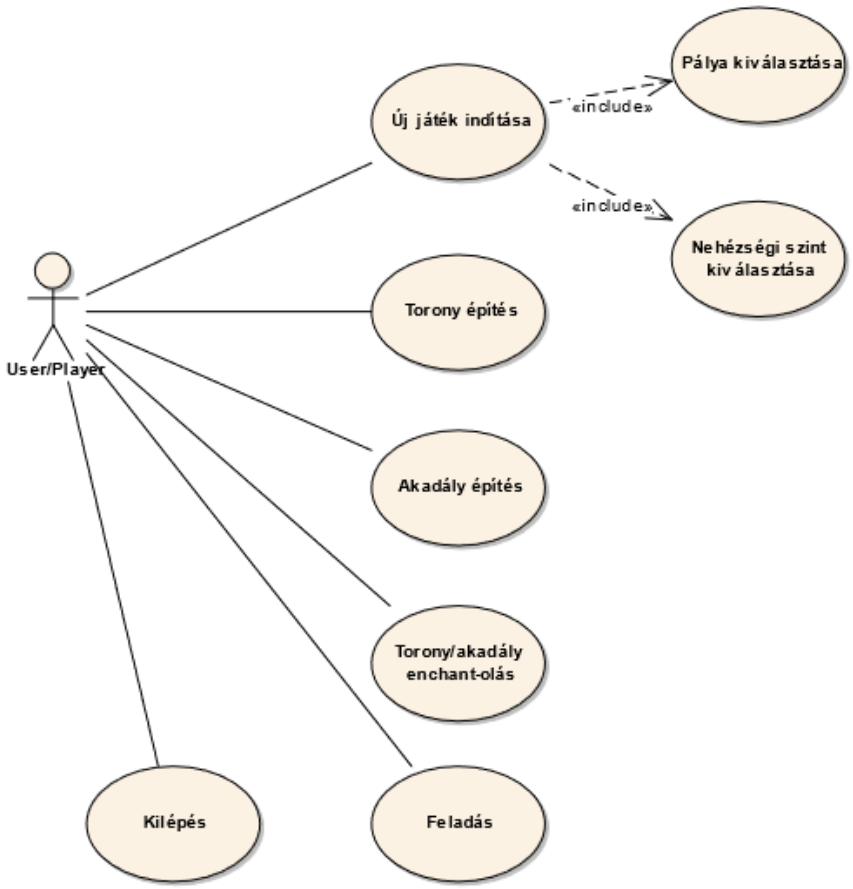
Use-case neve	**Akadály építése**
Rövid leírás	Új akadályt épít
Aktorok	Játékos
Forgatókönyv	Olyan mezőre kattintva ahol út van, és még nincsen akadály, ott akadály épül, ha van elég varázserő hozzá.

Use-case neve	**Torony/akadály enchant-olás**
Rövid leírás	Már létező tornyot vagy akadályt enchant-olunk
Aktorok	Játékos
Forgatókönyv	Toronyra vagy akadályra kattintva ki kell választani, hogy milyen drágakővel szeretné enchant-olni, és ha van elég varázserő hozzá akkor megtörténik az enchant.

Use-case neve	**Feladás**
Rövid leírás	Feladjuk a jelenlegi játékot
Aktorok	Játékos
Forgatókönyv	A feladás gombra kattintva a játék végetér, és megnyílik a főmenü.

Use-case neve	**Kilépés**
Rövid leírás	Kilép az alkalmazásból
Aktorok	Játékos
Forgatókönyv	Az ablak jobb felső sarkában, a bezárásra kattintva az alkalmazás kilép.

2.4.2. Use-case diagram



2.2. ábra. Use-case diagram

2.5. Szótár

akadály A játékos készítheti, az utakra lehet építeni. A rajta áthaladó ellenségeket lelassítja. Lehet enchantolni.

cél A pályán kijelölt hely; ha ide eljutnak az ellenségek, akkor vége a játéknak és a játékos vesztett.

életerő Egy szám, ami minden ellenséghez rendelve van. Ha ez a szám eléri a 0-t vagy kisebb lesz, az ellenség meghal.

ellenség A pályán az utakon haladnak, van egy életerő tulajdonságuk, ami lövés hatására csökken, majd ha eléri a 0-t, akkor az ellenség meghal.

enchantolás Torony vagy akadály varázskővel való felruházása.

építés A játékos által egy torony vagy akadály elhelyezése a játéktérre.

hatótávolság A tornyok tulajdonsága, azt határozza meg, hogy a torony milyen messzire tud lőni.

játék elvesztése A játék egy lehetséges végződése, akkor következik be, ha egy ellenség elér a célhoz.

játék megnyerése A játék egy lehetséges végződése, akkor következik be, az összes ellenség meghal anélkül, hogy elérnének a célhoz.

lövés Az az esemény, amikor a torony egy a hatótávolságán belül lévő ellenség életerejét csökkenti.

meghal Az az esemény, amikor egy ellenség életereje eléri a 0-t.

nehézség A játék különböző tulajdonságainak egy adott beállítása, ezzel változtatható, hogy milyen nehéz teljesíteni a játékot.

út A pályán kijelölt rész, amin az ellenségek eljuthatnak a célig. Az ellenségek csak ezen haladhatnak, az akadályokat csak erre szabad építeni, a tornyokat viszont az utakra nem lehet építeni.

pálya Az a tér, amin a játék zajlik; ezen vannak az utak, és erre lehet építeni a tornyokat és akadályokat.

sebződés Ha egy ellenség sebződést kap, akkor lemegy az életereje.

torony A játékos által készíthető dolgok közül a fontosabb; ezzel lehet megakadályozni az ellenségek célhoz jutását. Lőni tud, és lehet enchantolni.

tower defense Egy játékműfaj, amelyben a játék célja, hogy a pályán áthaladni próbáló ellenségeket megakadályozzuk abban, hogy elérjék a célt. Ezt tornyok és akadályok segítségével érhetjük el.

tüzelési gyakoriság A tornyok tulajdonsága, azt határozza meg, hogy egy adott idő alatt mennyi lövést tud leadni egy torony.

varázserő A játékos tulajdonsága; egy mennyiség, az ellenségek elpusztulásakor növekszik, torony és akadály építésekor és enchantoláskor pedig csökken. A játékos csak akkor tud építeni és enchantolni, ha a jelenlegi varázserő több, mint amennyi az építéshez vagy enchantoláshoz szükséges.

varázskő Olyan játékelem, amit hozzá lehet rendelni tornyokhoz és akadályokhoz (enchantolás), ami ettől valamilyen szempontból jobb lesz, pl. tüzelési gyakoriság növekedése. Készítéséhez varázserő szükséges, ha nincsen elég, akkor nem készíthető.

2.6. Projekt terv

2.6.1. Csapat

A csapat 4 főből áll. A feladatokat úgy próbáljuk kiosztani, hogy lehetőleg mindenki azonos nehézségű feladatot kapjon, miközben a személyes preferenciáit is betartjuk.

Név	Felelősségek
Nusser Ádám	UML, dokumentáció
Szabó Antal	Kód, dokumentáció
Tallér Bátor(csapatvezető)	Menedzsment, kód
Török Attila	Dokumentáció, kód

2.6.2. Kommunikáció

Verziókezelés Mivel többen dolgozunk a projekten, ezért fontos, hogy mindig mindenkinek elérhető legyen a legfrissebb forráskód, illetve dokumentáció. Ezért valamilyen módon meg kell osztanunk egymással az elkészített dokumentumokat, programkódokat. Erre a Git elosztott verziókezelő szoftvert választottuk. Ehhez a központi tárhelyet a GitHub biztosítja.
Azért esett erre a megoldásra a választás, mert így könnyen tudjuk követni ki mit csinált, egyszerűen tudunk visszaállni egy korábbi verzióra és közösen meg tudjuk vitatni, hogy mely változások kerüljenek be a végleges projektbe. Mind a programkódot, mind a dokumentációt verziózzuk, az utóbbi azért tehető meg, mert TeX-et használunk a dokumentáció készítéséhez, aminek a forrásfájlai sima szöveges dokumentumok.

Wiki A GitHubnak van egy wiki szolgáltatása, ahová feladatkiosztásokat és egyéb fontos tudnivalókat rakunk fel.

Facebook A csapatnak létrehoztunk egy privát Facebook csoportot, ahová a csapat bármely tagja írhat bejegyzéseket, illetve hozzászólhat bejegyzésekhez.

Megbeszélések Továbbá hetente (ha szükség van rá) tartunk egy konferenciabeszélgetést, ahol megbeszéljük ki hol tart a munkájában, milyen problémák és kérdések merültek fel. Ehhez a Skype nevű programot használjuk. Ezeken kívül minden héten találkozunk a kijelölt konzultációs időpontban, szerdánként 8:15-kor.

2.6.3. Használt programok

Verziókezelés A verziókezeléshez a feljebb bővebben kifejtett Git-et használjuk.

Dokumentáció A dokumentációhoz a TeXworks szoftvert használjuk. Ezzel könnyen lehet .tex fájlokat szerkeszteni és ezekből dokumentumot generálni.
A dokumentációban található UML diagrammok készítéséhez az Enterprise Architect nevű szoftvert használjuk.

Fejlesztőkörnyezet A forráskódot Eclipse, illetve IntelliJ IDEA nevű fejlesztőkörnyezetekben készítjük.

2.6.4. Mérföldkövek, határidők

Dátum	Leírás	Ellenőrzés
2014.02.24.	Követelmény, projekt, funkcionalitás	beadás
2014.03.03.	Analízis modell kidolgozása 1.	beadás
2014.03.10.	Analízis modell kidolgozása 2.	beadás
2014.03.17.	Szkeleton tervezése	beadás
2014.03.24.	Szkeleton	beadás
2014.03.26.	**Szkeleton**	bemutatás
2014.03.31.	Prototípus koncepciója	beadás
2014.04.07.	Részletes tervek	beadás
2014.04.22.	Prototípus	beadás
2014.04.23.	**Prototípus**	bemutatás
2014.05.12.	Grafikus változat	beadás
2014.05.14.	**Grafikus bemutató**	bemutatás
2014.05.16.	Összefoglalás	beadás

A feladatot 3 lépcsőben oldjuk meg, a lépcsőket vastagon szedve jelöltük. Ezek bővebben kifejtve:

A **szkeleton** változat célja annak bizonyítása, hogy az objektum és dinamikus modellek a definiált feladat egy modelljét alkotják. A szkeleton egy program, amelyben már valamennyi, a végső rendszerben is szereplő business objektum szerepel. Az objektumoknak csak az interfésze definiált. Valamennyi metódus az indulás pillanatában az ernyőre szöveges változatban kiírja a saját nevét, majd meghívja azon metódusokat, amelyeket a szolgáltatás végrehajtása érdekében meg kell hívnia. Amennyiben a metódusból valamely feltétel fennállása esetén hívunk meg más metódusokat, akkor a feltételre vonatkozó kérdést interaktívan az ernyőn fel kell tenni és a kapott válasz alapján kell a továbbiakban eljárni. A szkeletonnak alkalmasnak kell lenni arra, hogy a különböző forgatókönyvek és szekvencia diagramok ellenőrizhetők legyenek. Csak karakteres ernyőkezelés fogadható el, mert ez biztosítja a rendszer egyszerűségét.

A **prototípus** program célja annak demonstrálása, hogy a program elkészült, helyesen működik, valamennyi feladatát teljesíti. A prototípus változat egy elkészült program kivéve a kifejlett grafikus interfészt. A változat tervezési szempontból elkészült, az ütemezés, az aktív objektumok kezelése megoldott. A business objektumok - a megjelenítésre vonatkozó részeket kivéve - valamennyi metódusa a végleges algoritmusokat tartalmazza. A megjelenítés és működtetés egy alfanumerikus ernyőn követhető, ugyanakkor a megjelenítés fájlban is logolható, ezzel megteremtve a rendszer tesztelésének lehetőségét. Különös figyelmet kell fordítani az interfész logikájára, felépítésére, valamint arra, hogy az mennyiben tükrözi és teszi láthatóvá a program működését, a beavatkozások hatásait.

A **grafikus** (teljes) változat a prototípustól elvileg csak a kezelői felület minőségében különbözhet. Ennek változatnak az értékelésekor a hangsúlyt sokkal inkább a megvalósítás belső szerkezetére, semmint a külalakra kell helyezni.

2.7. Napló

Kezdet	Időtartam	Résztvevők	Leírás
2014.02.19. 08:15	1,5 óra	**Nusser** **Szabó** **Tallér** **Török**	Kezdeti megbeszélés, use-case-ek átbeszélése, feladatok kiosztása
2014.02.21. 18:00	1 óra	**Szabó**	Dokumentáció elkezdése
2014.02.22. 13:30	3 óra	**Nusser**	Use-case -ek leírása, és diagram készítése.
2014.02.22. 18:00	2 óra	**Tallér**	Követelmények befejezése
2014.02.22. 19:00	1 óra	**Nusser**	Use-case-ek befejezése
2014.02.22. 21:30	1,5 óra	**Nusser** **Szabó** **Tallér** **Török**	Közös megbeszélés
2014.02.23. 17:00	2,5 óra	**Tallér**	Projekt terv elkészítése
2014.02.23. 18:00	0,5 óra	**Szabó**	Bevezetés definíciók nélkül
2014.02.23. 21:30	1,5 óra	**Szabó**	Definíciók elkészítése
2014.02.23. 21:00	1,5 óra	**Tallér**	Dokumentum összeszerkesztése, ellenőrzése, segítés a többi csapattagnak
2014.02.23. 22:30	3 óra	**Török**	Áttekintés elkészítése
2014.02.24. 01:00	1 óra	**Szabó**	Szótár elkészítése
2014.02.24. 12:00	1 óra	**Nusser** **Szabó** **Tallér** **Török**	Utolsó simítások

3. Analízis modell kidolgozása 1

3.1. Objektum katalógus

3.1.1. Enemy

Az **Enemy** osztály példányai egy-egy ellenséget tárolnak. Rendelkeznek típussal (fajjal), pillanatnyi pozícióval, hátralévő életerővel. Az idő haladtával próbálnak végighaladni egy véletlenszerűen választott lehetséges útvonalon.

3.1.2. EnemyType

Leírja egy ellenségtípus (jelen esetben a négy faj: ember, tünde, törp vagy hobbit) tulajdonságait, mint például az alapsebessége és a kezdeti életereje. Minden ellenségnek van egy referenciája egy példányára, amely meghatározza a viselkedését.

3.1.3. Game

Ez a játék logikáját magába foglaló osztály. Számon tartja a pályán lévő ellenfeleket, akadályokat és tornyokat, valamint végrehajtja a kiválasztott misszió által leírt forgatókönyvet, ami alapján az ellenségek érkeznek.

3.1.4. Gem

Egy épületekre rakható varázskő tulajdonságait tárolja. A **Game** osztály fog példányokat létrehozni belőle, különböző tulajdonságokkal. Ha egy épületre rá van rakva egy fajta varázskő, akkor az épület egy referenciát tárol a példányára.

3.1.5. Map

A **Map** a pályát képviseli, melyben NxM darab cella van. A pálya egy külső fájlból betölthető, ami új játék indulásakor fog megtörténni, miután kiválasztja a játékos a pályát. A pálya különböző tulajdonságai is lekérdezhetőek, amelyek majd a kirajzoláshoz kellenek.

3.1.6. Mission

Egy **Mission** objektumban lesz eltárolva, hogy milyen ellenségek, mikor és honnan jönnek be a pályára. Ez a három adat majd egy segéd osztályban lesz összefogva, és egy tárolóban lesz tárolva. A játék főciklusa le tudja majd kérdezni a következő ellenséget, melyet az objektum visszaad, ha már elég idő eltelt az előző egység óta (ha nem akkor nem ad vissza egységet).

3.1.7. Obstacle

Egy akadályt valósít meg. Ha egy ellenség belelép egy olyan mezőbe, amin van egy **Obstacle**, akkor azt egy bizonyos mértékben lelassítja, az ellenség fajától függően.

3.1.8. Projectile

Ez az osztály egy lövedék leírása. Amikor egy torony lő, akkor példányosítja ezt az osztályt a cél ellenség átadásával, így egy lövedék mindig a torony által meghatározott ellenség után megy, amit ha megfelelően megközelít (eltalálja az ellenséget), akkor levesz az életerejéből és megsemmisíti magát.

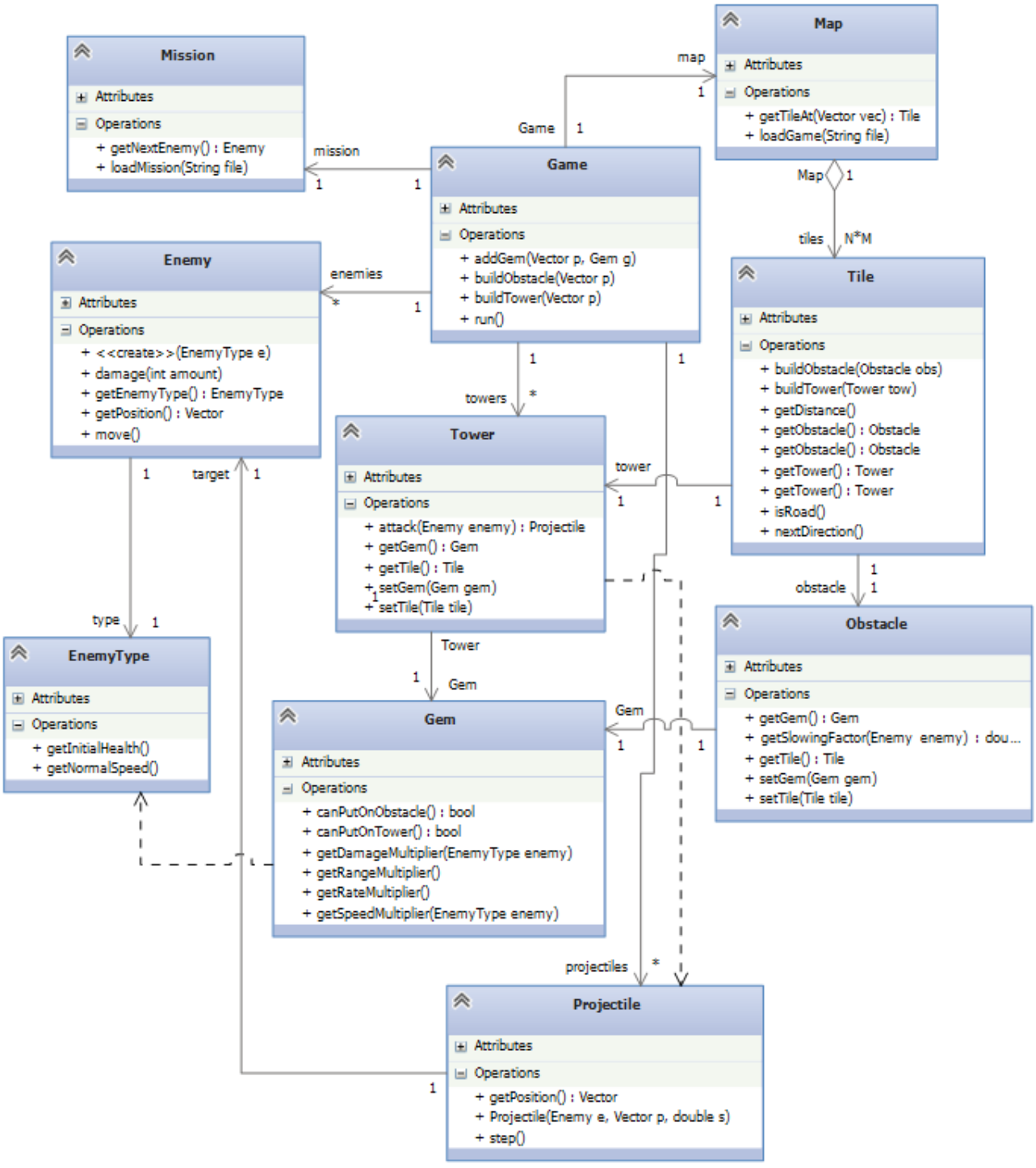
3.1.9. Tile

A **Tile** objektumból egy NxM-es tömböt fog tárolni a **Map** objektum. Ezek a pálya egyes celláit fogják képviselni. Egy cellán lehet maximum egy darab épület, amely vagy egy torony, vagy egy akadály, attól függően, hogy van-e a cellán út, vagy nincsen. A cellán levő épület referenciája lekérhető, vagy ha még nincsen rajta, és a cellára kattintás történik, akkor épül egy megfelelő épület.
Ha egy cellán van út, akkor lekérdezhető, hogy az a cella milyen távol van a céltól (az utat követve, cellákban mérve). Továbbá az is tárolva van, és lekérdezhető, hogy az út merre folytatódik (és az ellenségeknek merre kell majd menniük). Ha elágazás van, akkor véletlenszerűen választ irányt és arra küldi az ellenséget (ez minden egyes ellenségre véletlenszerű).

3.1.10. Tower

Egy tornyot valósít meg. Ha egy ellenség a hatókörébe ér, akkor elindít egy **Projectile**-t az irányába, a tüzelési gyakoriságának megfelelő időközönként. Ha több ellenfél van a hatókörében, akkor arra lő, aki a legközelebb van a célhoz.

3.2. Statikus struktúra diagramok



3.1. ábra. Osztálydiagram

3.3. Osztályok leírása

3.3.1. Enemy

- Felelősség
  Az ellenségek pozíciójának, sebességének és életerejének nyilvántartása.
- Attribútumok
  - **EnemyType** type: Az egység típusa.
  - double health: Az ellenség fennmaradó életereje.
  - **Vector** position: Pillanatnyi pozíció a pályán.
  - Direction dir: Megadja milyen irányban kell mozognia.
- Metódusok
  - boolean move(double dt): Az egységet annyival mozgatja az útján tovább, amennyit dt idő alatt halad. Ha az ellenség életereje 0 vagy kisebb, akkor true-val tér visza, egyébként false-al.
  - **Vector** getPosition(): A position attribútum értékével tér vissza.
  - boolean damage(double amount): Csökkenti az ellenség életerejét amount-al, igazzal tér vissza ha az ellenség meghalt.
  - **EnemyType** getEnemyType(): Visszaadja az ellenség típusát.

3.3.2. EnemyType

- Felelősség
  Leírja egy bizonyos típusú (fajú) ellenség alapvető tulajdonságait. Egy-egy példányára hivatkozik az összes ellenség, amelyeknek ezáltal meghatározza a viselkedését.
- Attribútumok
  - int cost: Az ilyen fajtájú ellenségek ára varázserőben.
  - double initialHealth: Az ilyen fajtájú ellenségek kezdeti életereje.
  - double normalSpeed: Az ilyen fajtájú ellenségek akadályoztatás nélküli haladási sebessége.
- Metódusok
  - double getInitialHealth(): Visszaadja az initialHealth attribútum értékét.
  - double getNormalSpeed(): Visszaadja a normalSpeed attribútum értékét.

3.3.3. Game

- Felelősség
  A többi osztály nyilván tartása és összekötése, a játékbeli események vezérlése. A felhasználói felülettől érkező parancsok végrehajtása, és a játék állapotának rendelkezésre bocsájtása a kijelzéshez.
- Attribútumok
  - **ArrayList<Projectile>** projectiles: Eltárolja a jelenleg játékban lévő lövedékeket.
  - **Map** map: Referencia a kiválasztott pályára, amin a játék folyik.
  - **Mission** mission: Referencia a kiválasztott misszióra, amely alapján zajlik a játék.
  - **ArrayList<Building>** buildings: Eltárolja a játékos megépített tornyait.

  - **ArrayList<Gem>** gems: Az összes lehetséges, toronyra illetve akadályra rakható varázskő nyilvántartása.
  - **ArrayList<EnemyType>** types: Ez a lehetséges ellenségtípusok listája.
  - **ArrayList<Enemy>** enemies: Az összes éppen látható ellenség található meg benne.
  - int magic: A játékos jelenlegi varázsereje.

- Metódusok
  - void run(): Ez a metódus futtatja a főciklust, amelyben maga a játék működik.
  - void buildTower(**Vector** position): Épít egy tornyot a paraméterül kapott helyen évő mezőre.
  - void buildObstacle(**Vector** position): Épít egy akadályt a paraméterül kapott helyen lévő mezőre.
  - void addGem(**Vector** position, **Gem** gem): A paraméterként kapott helyen lévő toronyra vagy akadályra rárakja a paraméterként kapott varázskövet.

3.3.4. Gem

- Felelősség
  Egy fajta varázskő tulajdonságainak tárolása.

- Attribútumok
  - int cost: A varázskő ára varázserőben.
  - double rangeMultiplier: Megadja, hogy a varázskővel ellátott toronynak hányszorosára nő a hatótávolsága.
  - double rateMultiplier: Megadja, hogy a varázskővel ellátott toronynak hányszorosára nő a tüzelési sebessége.
  - **HashMap<EnemyType**, **Double>** damageMultiplier: Megadja, hogy a varázskővel ellátott toronynak hányszorosára nő a sebzése egy adott típusú ellenséggel szemben.
  - **HashMap<EnemyType**, **Double>** speedMultiplier: Megadja, hogy a varázskővel elátott akadályon áthaladó adott típusú ellenség sebessége hányadára csökken.

- Metódusok
  - **Gem**(double rangeMultiplier, double rateMultiplier, **HashMap<EnemyType**, Double> damageMultiplier, **HashMap<EnemyType**, **Double>** speedMultiplier): Létrehoz egy **Gem** objektumot a paraméterként kapott tulajdonságokkal.
  - double getRangeMultiplier(): Visszaadja a varázskő hatótávolság szorzóját.
  - double getRateMultiplier(): Visszaadja a varázskő tüzelési sebesség szorzójáz.
  - double getDamageMultiplier(**EnemyType** enemyType): Visszaadja varázskő sebzés szorzóját egy adott típusú ellenséghez.
  - double getSpeedMultiplier(**EnemyType** enemyType): Visszaadja varázskő sebesség szorzóját egy adott típusú ellenséghez.
  - boolean canPutOnTower(): Ha a varázskő használható tornyokra, igazat ad vissza, különben hamisat.
  - boolean canPutOnObstacle() Ha a varázskő használható akadályokra, igazat ad vissza, különben hamisat.

3.3.5. Map

- Felelősség
  Tartlamaz az összes cellára referenciát, felelős a kapott fájlból beolvasni a pályát, és felépíteni azt.
- Attribútumok
  - **Tile**[][] tiles: A pályán található cellák tömbje, a pálya kirajzolásához használható.
- Metódusok
  - **Tile** getTileAt(**Vector** position): Visszaadja a position helyen talalható cellát.
  - loadGame(**String** file): Betölti a kapott elérési útvonalon a fájlt, és felépíti az alapján a pályát.

3.3.6. Mission

- Felelősség
  Felelőssége, felépíteni a listát, amely a beérkező ellenségeket és időzítésüket tartalmazza. Majd a megfelelő időközönként ki kell adja ezeket az egységeket a **Game** osztálynak.
- Attribútumok
  - **List<Spawn>** spawnList: A Spawn segédosztályban tárolt ellenség-idő-belépési pont, adatokat tárolja.
  - double lastSpawn: Tárolja az előző spawn idejét.
- Metódusok
  - **Enemy** getNextEnemy(double time): A spawnList listaból a következő elemet megvizsgaálja, és ha spawnolni kell a következő ellenséget, akkor visszaadja azt.
  - loadMission(**String** file): Betölti a kapott elérési útvonalon a fájlt, és felépíti az alapján a listát (spawnList).

3.3.7. Obstacle

- Felelősség
  Felelős **Projectile**-ok létrehozásához, azok megfelelő felparaméterezésével. Továbbá felelős azért, hogy **Projectile**-okat csak a megadott időközönként lőjjön ki.
- Attribútumok
- int cost: Az akadály ára varázserőben.
- **Gem** gem: Eltárol egy referenciát egy **Gem** típusú objektumra, ami meghatározza, hogy az adott épület milyen echant alatt áll.
- **Map<EnemyType**, double> speedMultiplier: Megadja mekkora az adott típusú ellenfélre kifejtett hatása az akadálynak.
- **Tile** tile: Egy referencia arra a mezőre, ahol az épület található.
- Metódusok
  - double getSlowingFactor(**Enemy** enemy): Visszatér azzal az értékkel, amivel az adott ellenfelet lassítja.
  - **Gem** getGem(): Visszaadja az épületen található varázskövet.

  - void setGem(**Gem** gem): Beállítja az épületen lévő varázskövet. Ha már volt az épületen varázskő, akkor az előző megszűnik.
  - void setTile(**Tile** tile): Beállítja, hogy az akadály melyik **Tile**-on van.
  - **Tile** getTile(): Visszaadja, hogy az akadály melyik **Tile**-on van.

3.3.8. Projectile

- Felelősség
  Követni a cél ellenséget, majd sebezni ha eléri.
- Attribútumok
  - double damage: A lövedék sebzése, ennyivel csökkenti a cél ellenség életerejét amikor eltalálja.
  - **Vector** position: A lövedék pozíciója.
  - double speed: A lövedék sebessége.
  - **Enemy** enemy: A lövedék cél **Enemy**-je.
- Metódusok
  - Projectile(**Enemy** enemy, **Vector** position, double speed): Konstruktor, átveszi a cél **Enemy**-t, a kezdő pozíciót és sebességet.
  - boolean move(double dt): dt * speed-el mozgatja a lövedéket az ellenség irányába. Ha eltalálta az ellenséget vagy az ellenség már meghalt, akkor true-t ad vissza, különben false-t.
  - **Vector** getPosition(): Visszaadja a lövedék pozícióját.

3.3.9. Tile

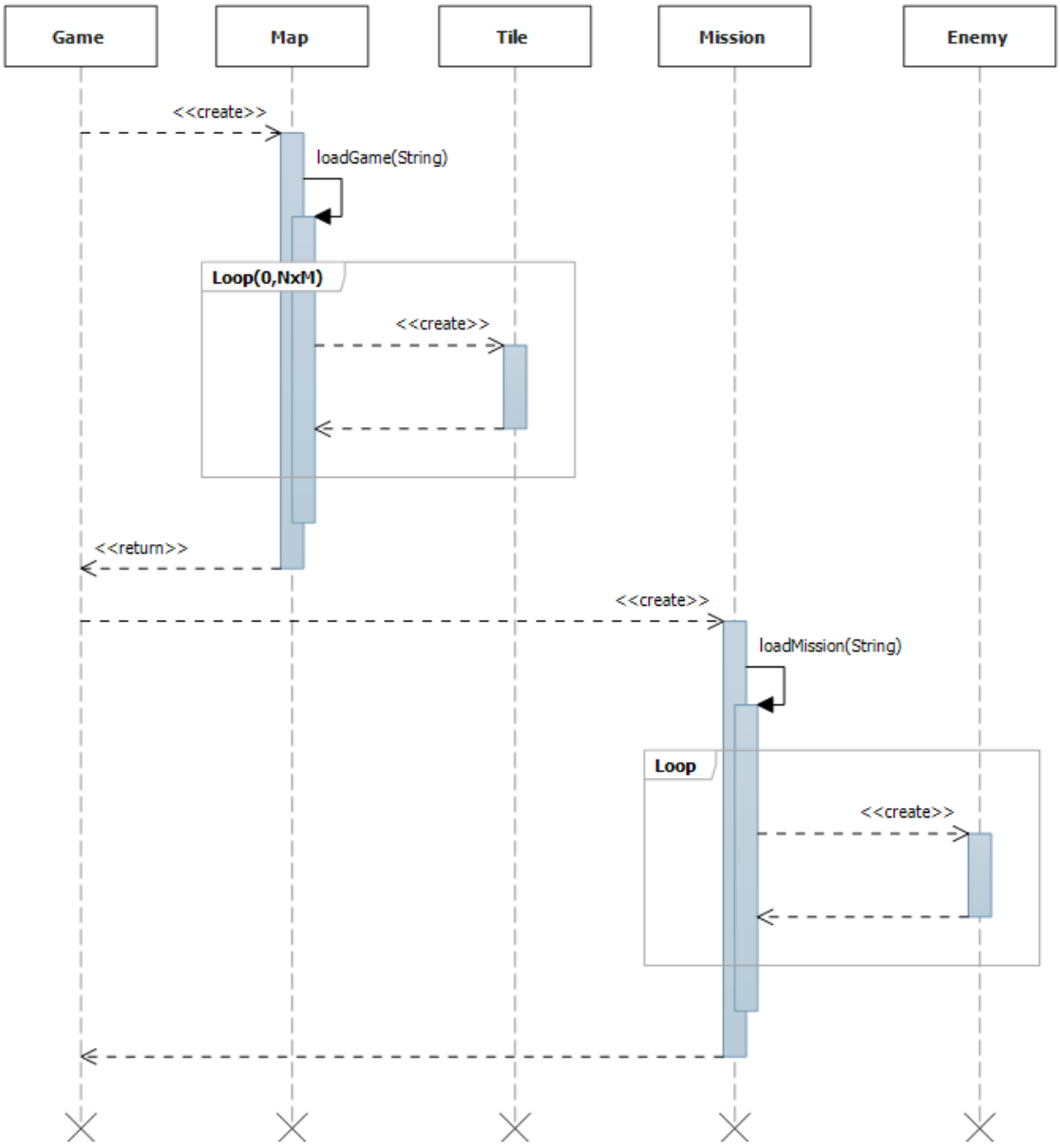
- Felelősség
  Egy cellán lehet maximum egy darab épület, amely vagy egy torony, vagy egy akadály, attól függően, hogy van-e a cellán út, vagy nincsen. A cellán levő épület referenciája lekérhető, vagy ha még nincsen rajta, és a cellára kattintás történik, akkor épül egy megfelelő épület. Felelős még az egyes ellenségek következő irányának a meghatározására, és vissza kell tudnia adni, hogy van-e rajta út, valamint ha van akkor milyen távol van a céltól.
- Attribútumok
  - **Building** building: A cellában levő épület.
  - int distance: Hányadik cella az utat követve a céltól.
  - boolean road: Van-e út a cellában.
  - double up: Annak a valószinűsége, hogy a cellából felfelé fog menni az ellenség.
  - double down: Annak a valószinűsége, hogy a cellából lefelé fog menni az ellenség.
  - double left: Annak a valószinűsége, hogy a cellából balra fog menni az ellenség.
  - double right: Annak a valószinűsége, hogy a cellából jobbra fog menni az ellenség.
- Metódusok
  - void buildTower(): A cellán egy tornyot épít.
  - void buildObstacle(): A cellán egy akadályt épít.
  - **Tower** getTower(): Visszaadja a cellán lévő tornyot.
  - **Obstacle** getObstacle(): Visszaadja a cellán lévő akadályt.

- int getDistance(): Visszaadja a distance értékét.
- boolean isRoad(): Visszaadja a road értékét.
- **Direction** nextDirection(): Visszad egy irány enum-ot, amely megadja az ellenség irányát.

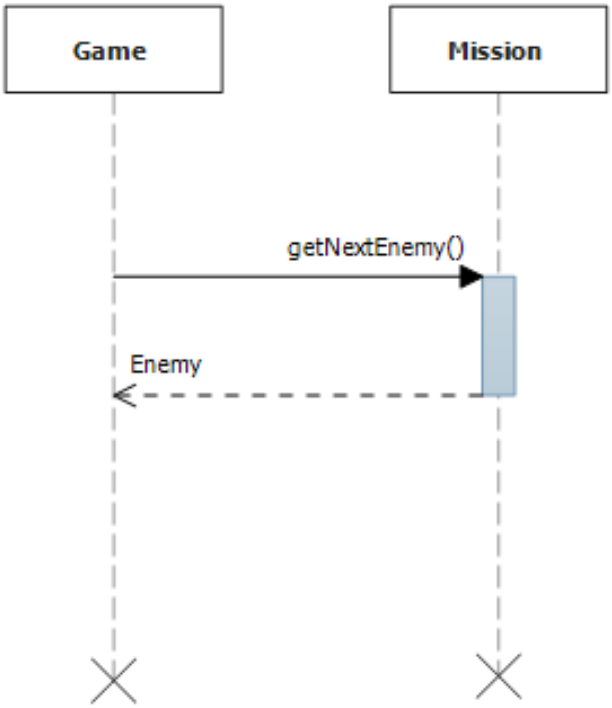
3.3.10. Tower

- Felelősség
  Felelős **Projectile**-ok létrehozásához, azok megfelelő felparaméterezésével. Továbbá felelős azért, hogy **Projectile**-okat csak a megadott időközönként lőjjön ki.
- Attribútumok
  - int cost: A torony ára varázserőben.
  - **Gem** gem: Eltárol egy referenciát egy **Gem** típusú objektumra, ami meghatározza, hogy az adott épület milyen echant alatt áll.
  - **HashMap<EnemyType**, double> damage: Megadja mekkora az adott típusú ellenfélre kifejtett hatása a toronynak.
  - **Tile** tile: Egy referencia arra a mezőre, ahol az épület található.
- Metódusok
  - **Projectile** attack(**Enemy** enemy): Kilő egy **Projectile**-t a paraméterként megkapott ellenségre, majd a visszatérési értékében visszaadja azt. A **Projectile**-t felparaméterezi az ellenséghez megfelelő sebzési adatokkal.
  - **Gem** getGem(): Visszaadja az épületen található varázskövet.
  - void setGem(**Gem** gem): Beállítja az épületen lévő varázskövet. Ha már volt az épületen varázskő, akkor az előző megszűnik.
  - void setTile(**Tile** tile): Beállítja, hogy a torony melyik **Tile**-on van.
  - **Tile** getTile(): Visszaadja, hogy a torony melyik **Tile**-on van.

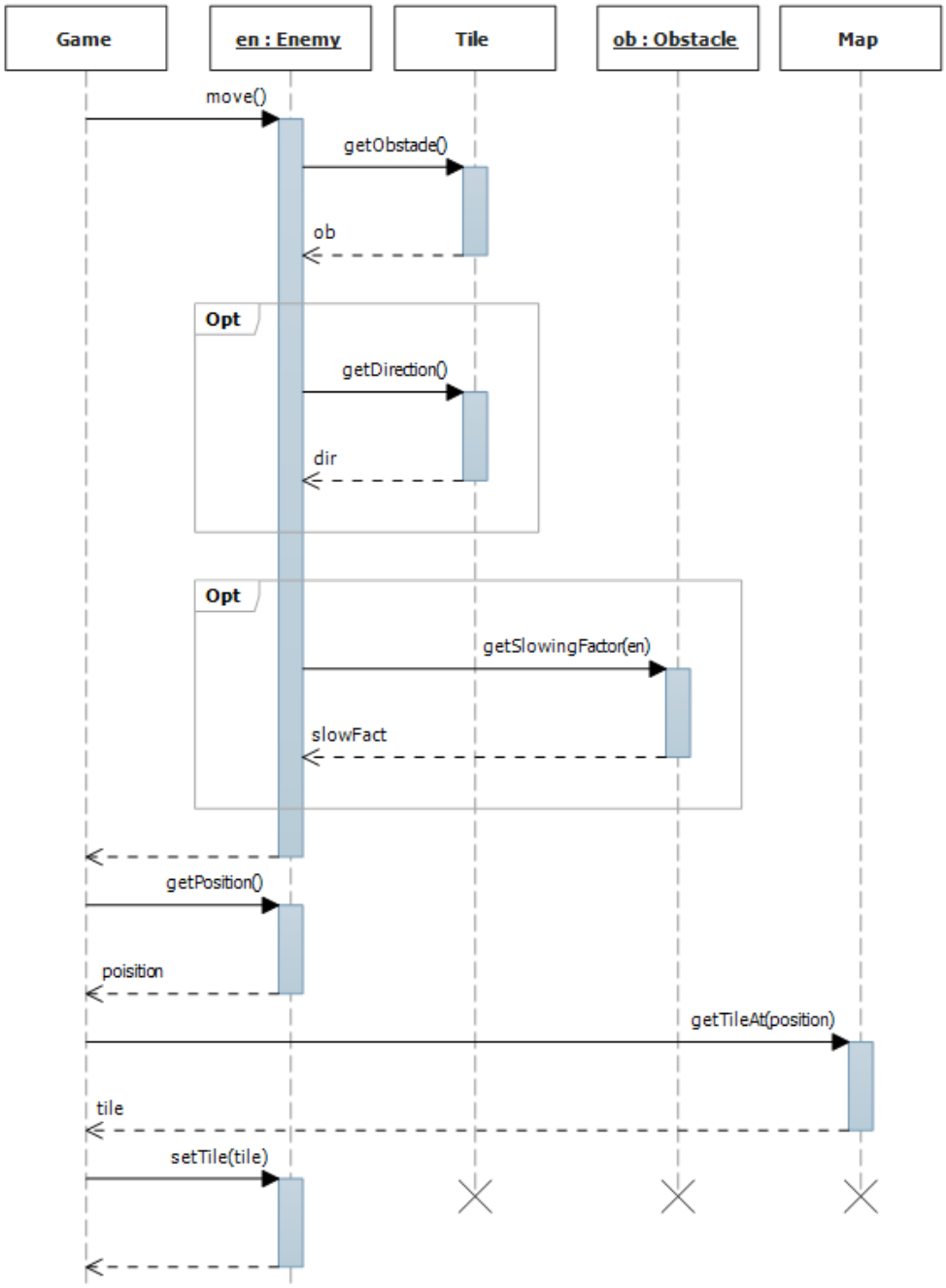
3.4. Szekvencia diagramok



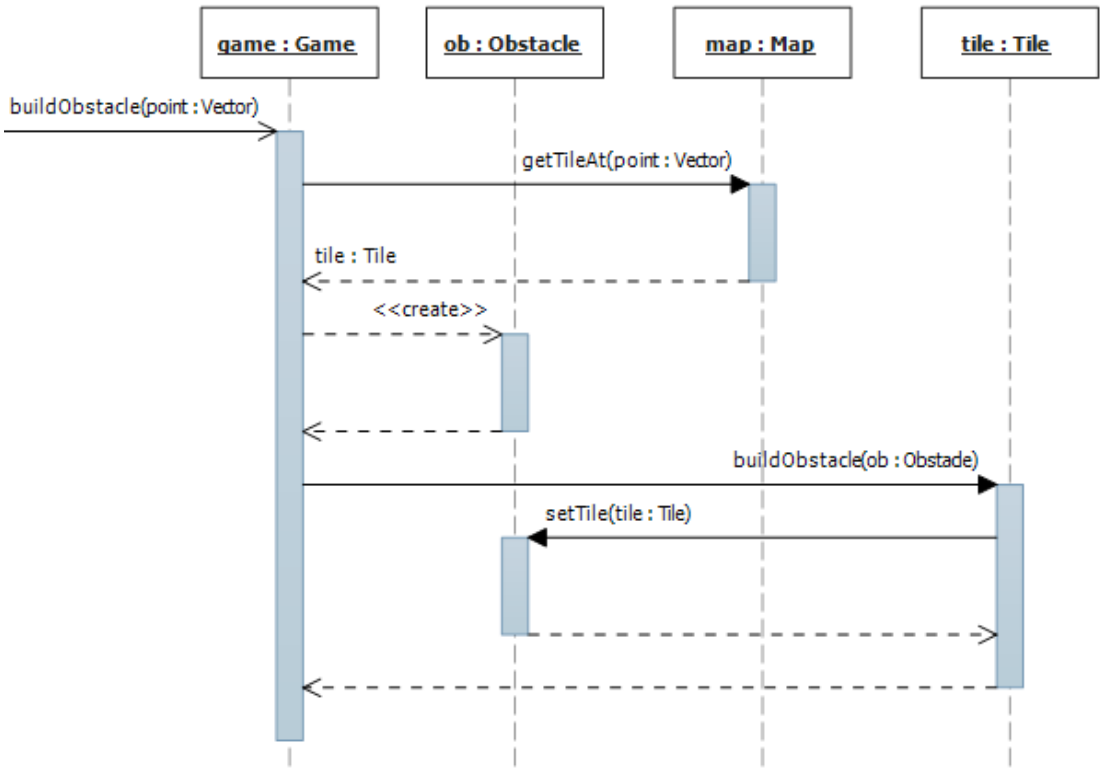
3.2. ábra. Játék indítása



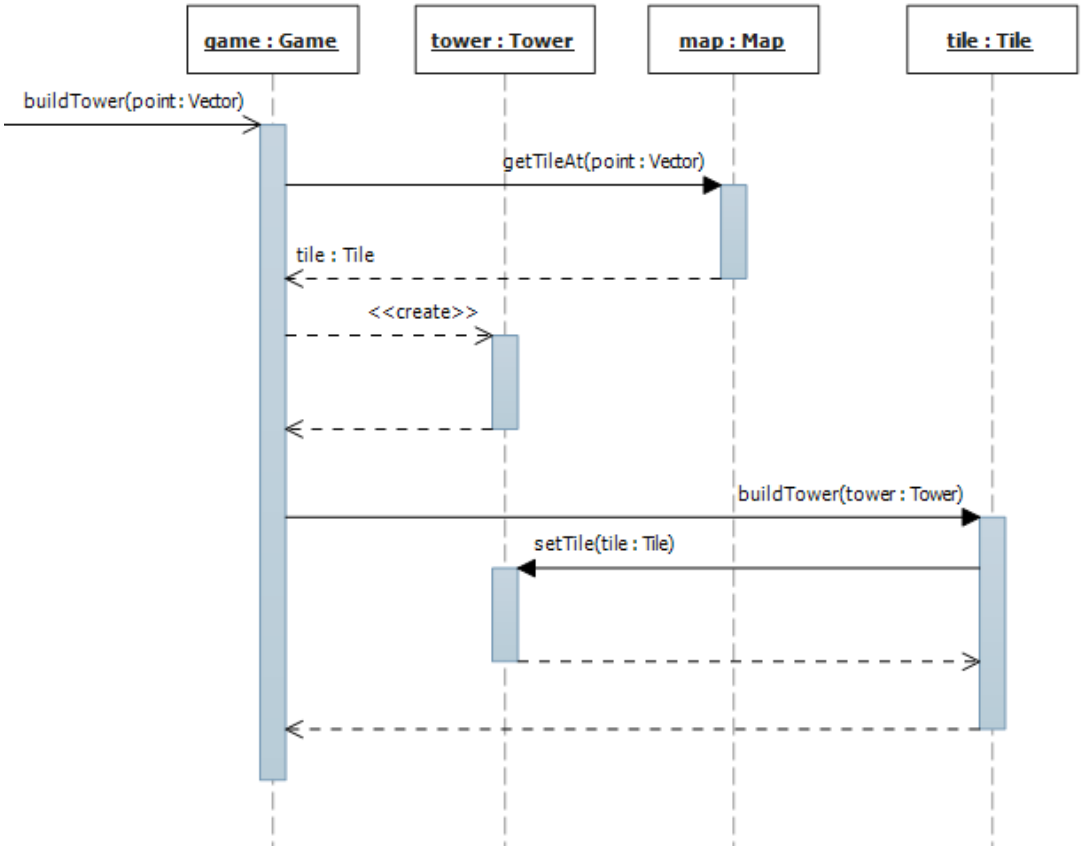
3.3. ábra. Ellenségek ütemezése



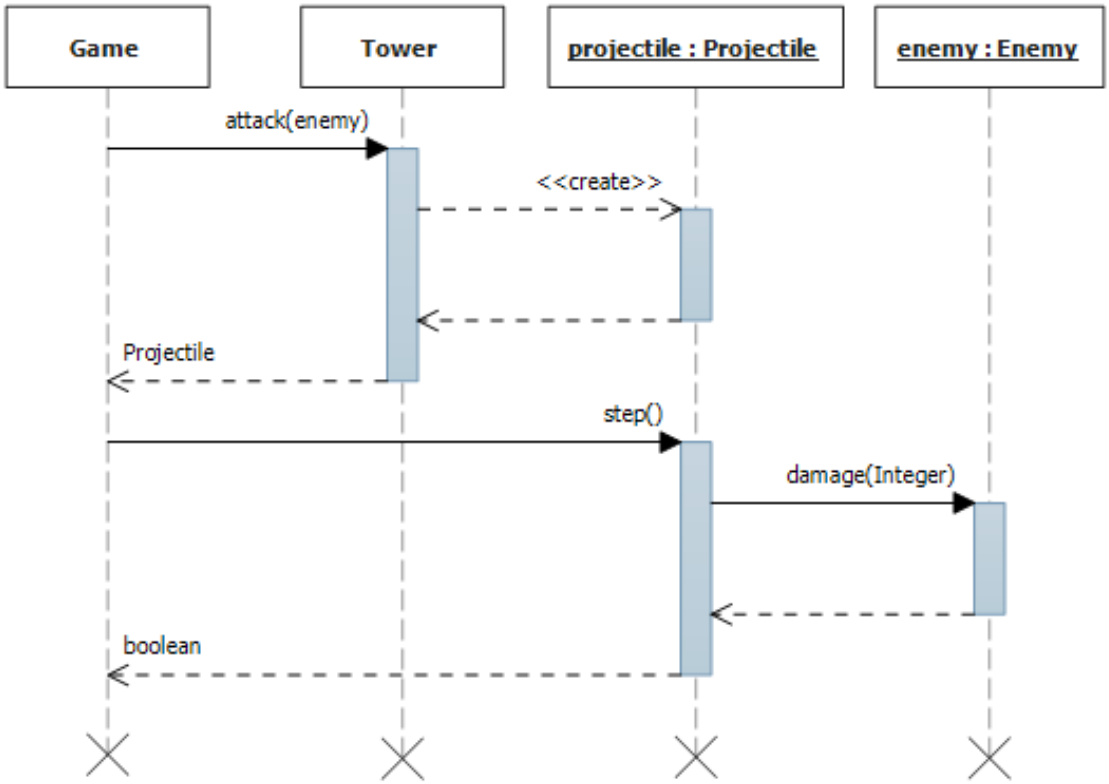
3.4. ábra. Ellenség mozgatása



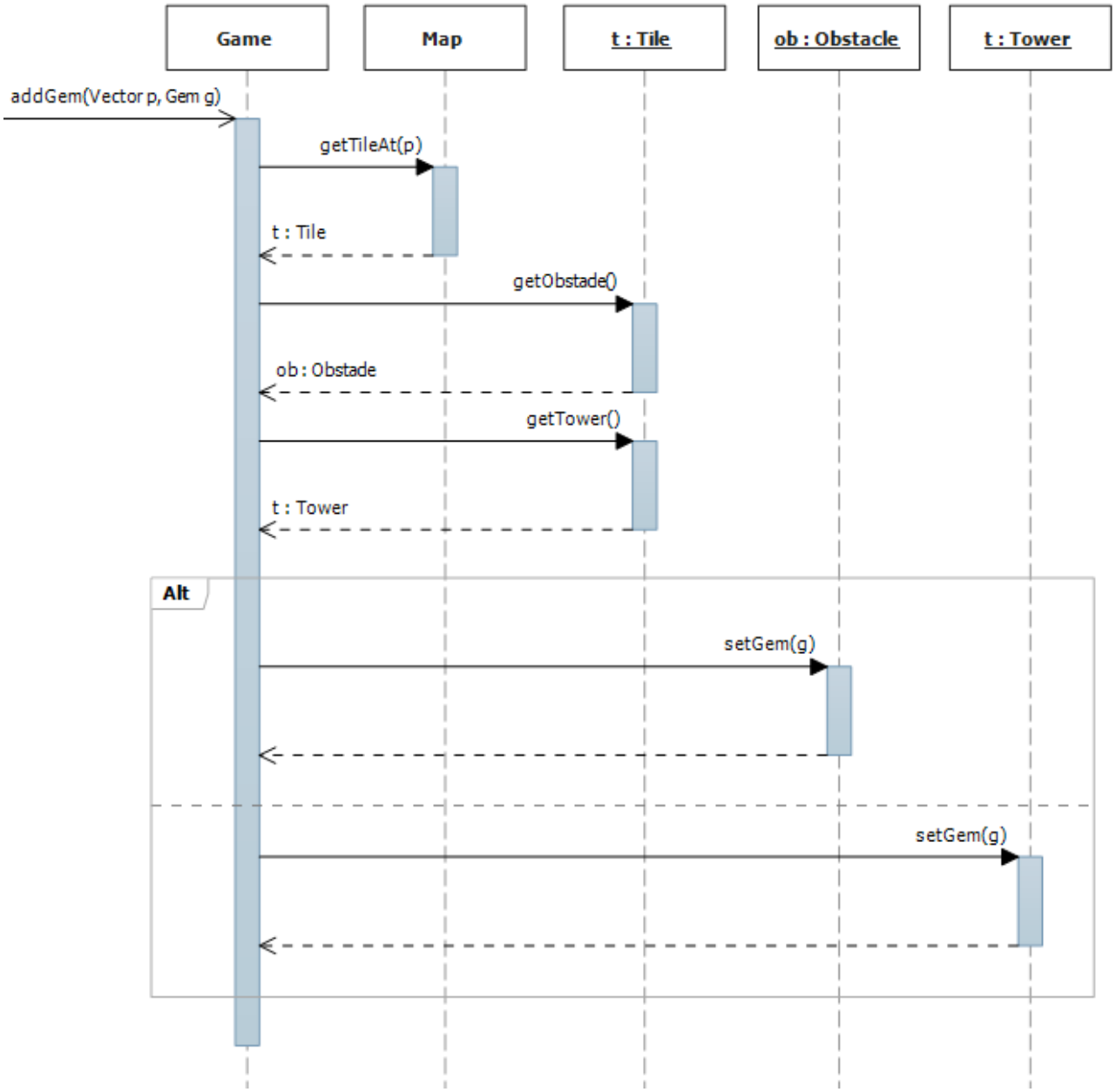
3.5. ábra. Akadály építése



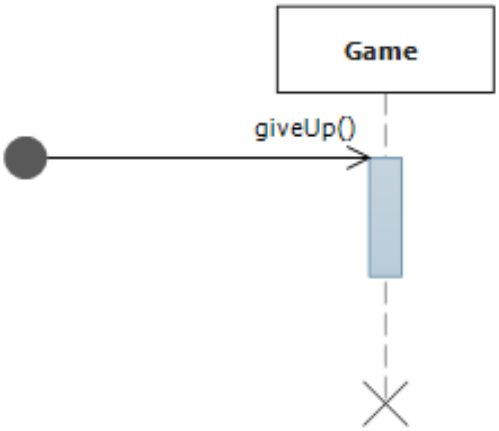
3.6. ábra. Torony építése



3.7. ábra. Torony tüzelése

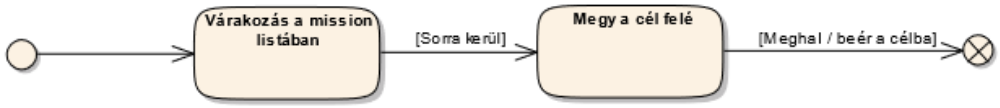


3.8. ábra. Varázskő felvétele

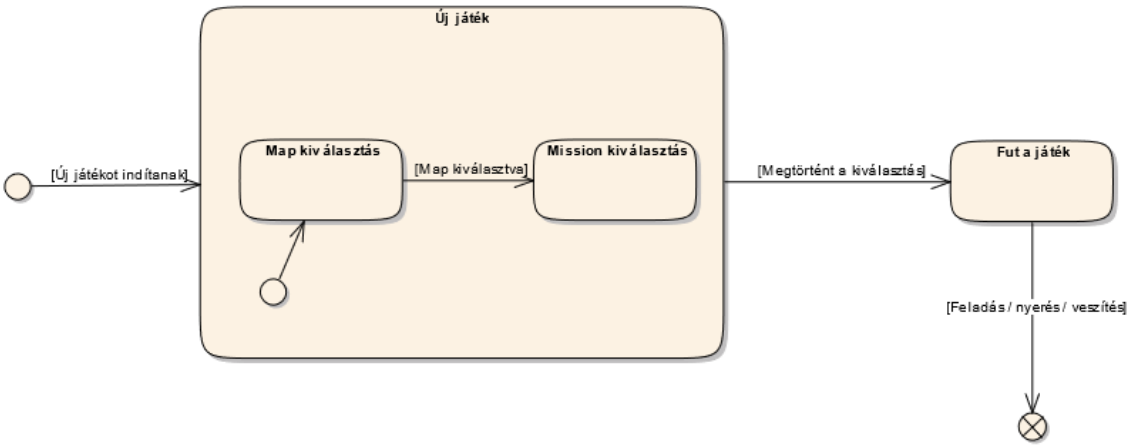


3.9. ábra. Feladás

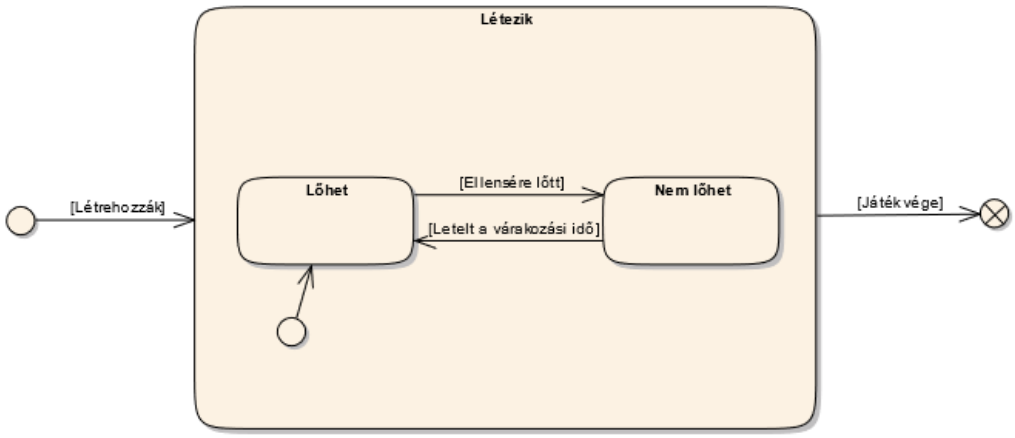
3.5. State-chartok



3.10. ábra. Egy ellenség állapotdiagramja



3.11. ábra. A játék állapotdiagramja



3.12. ábra. Egy torony állapotdiagramja

3.6. Napló

Kezdet	Időtartam	Résztvevők	Leírás
2014.02.26. 8:15	1,5 óra	**Nusser** **Szabó** **Tallér** **Török**	Elkezdtük tárgyalni a feladatot. Megvitattuk a program alapvető működését, a főbb osztályokat, metódusokat, szekvenciákat.
2014.02.27. 14:15	1 óra	**Nusser** **Szabó** **Tallér** **Török**	Folytattuk a szerdai megbeszélést. Kiosztottuk a feladatokat. Felelősségek: Ádám: Map, Tile, Mission Antal: Gem, Projectile Bátor: Building, Obstacle, Tower Török: Enemy, EnemyType, Game osztályok
2014.02.28. 16:30	2,5 óra	**Nusser**	Map, Tile és Mission osztályok leírása.
2014.02.28. 18:30	1,5 óra	**Tallér**	Tower, Obstacle osztályok leírása.
2014.02.28. 19:00	1 óra	**Szabó**	Projectile osztály leírása.
2014.02.28. 20:00	1 óra	**Török**	Game, EnemyType, Enemy osztályok leírása.
2014.03.01. 18:30	1.5 óra	**Nusser**	Szekvencia diagramok készítése.
2014.03.02. 19:30	4,5 óra	**Szabó**	Gem leírás elkészítése, szekvenciadiagramok készítése, dokumentáció egészének átnézése, hibák javítása, formázás.
2014.03.02. 20:00	2 óra	**Nusser**	Szekvenciadiagramok, állapotdiagramok készítése.
2014.03.02. 20:00	4 óra	**Tallér**	Szekvenciadiagramok, osztálydiagram készítése.
2014.03.02. 20:00	4 óra	**Szabó**	Szekvenciadiagramok, osztálydiagram készítése. Apróbb módosítások, hibajavítások, konzisztencia ellenőrzése.
2014.03.02. 20:00	2 óra	**Nusser**	State chart diagramok készítése.
2014.03.03. 00:00	3 óra	**Török**	Diagramok beszúrása, a dokumentum rendezése, formázása, apróbb hibák javítása.
2014.03.02. 20:00	0,5 óra	**Tallér**	Ellenőrzés, hibajavítás.

4. Analízis modell kidolgozása 2

4.1. Objektum katalógus

4.1.1. Enemy

Az **Enemy** osztály példányai egy-egy ellenséget tárolnak. Rendelkeznek típussal (fajjal), pillanatnyi pozícióval, hátralévő életerővel. Az idő haladtával próbálnak végighaladni egy véletlenszerűen választott lehetséges útvonalon.

4.1.2. EnemyType

Leírja egy ellenségtípus (jelen esetben a négy faj: ember, tünde, törp vagy hobbit) tulajdonságait, mint például az alapsebessége és a kezdeti életereje. Minden ellenségnek van egy referenciája egy példányára, amely meghatározza a viselkedését.

4.1.3. Game

Ez a játék logikáját magába foglaló osztály. Számon tartja a pályán lévő ellenfeleket, akadályokat és tornyokat, valamint végrehajtja a kiválasztott misszió által leírt forgatókönyvet, ami alapján az ellenségek érkeznek.

4.1.4. Map

A **Map** a pályát képviseli, melyben N darab **Waypoint** van. A pálya egy külső fájlból betölthető, ami új játék indulásakor fog megtörténni, miután kiválasztja a játékos a pályát. A pálya különböző tulajdonságai is lekérdezhetőek, amelyek ahhoz kellenek, hogy meg lehessen allapítani, hogy lehet-e oda új tornyot/akadályt építeni.

4.1.5. Mission

Egy **Mission** objektumban lesz eltárolva, hogy milyen ellenségek, mikor és honnan jönnek be a pályára. Ez a három adat majd egy segéd osztályban lesz összefogva, és egy tárolóban lesz tárolva. A játék főciklusa le tudja majd kérdezni a következő ellenséget, melyet az objektum visszaad, ha már elég idő eltelt az előző egység óta (ha nem akkor nem ad vissza egységet).

4.1.6. Obstacle

Egy akadályt valósít meg. Ha egy ellenség belelép egy olyan mezőbe, amin van egy **Obstacle**, akkor azt egy bizonyos mértékben lelassítja, az ellenség fajától függően.

4.1.7. ObstacleGem

Egy akadályokra rakható varázskő tulajdonságait tárolja, a **Gem** leszármazottja. Képes az akadályok alaptulajdonságait módosítani.

4.1.8. Projectile

Ez az osztály egy lövedék leírása. Amikor egy torony lő, akkor példányosítja ezt az osztályt a cél ellenség átadásával, így egy lövedék mindig a torony által meghatározott ellenség után megy, amit ha megfelelően megközelít (eltalálja az ellenséget), akkor levesz az életerejéből és megsemmisíti magát.

4.1.9. Waypoint

A **Waypoint**-ok kitűznek egy útvonalat, amelyen az ellenségek menni fognak. El van tárolva bennük a pozíciójuk, a távolságuk a céltól, valamint a következő **Waypoint** ahova az út folytatódik. Ezek mind le is kérdezhetőek.

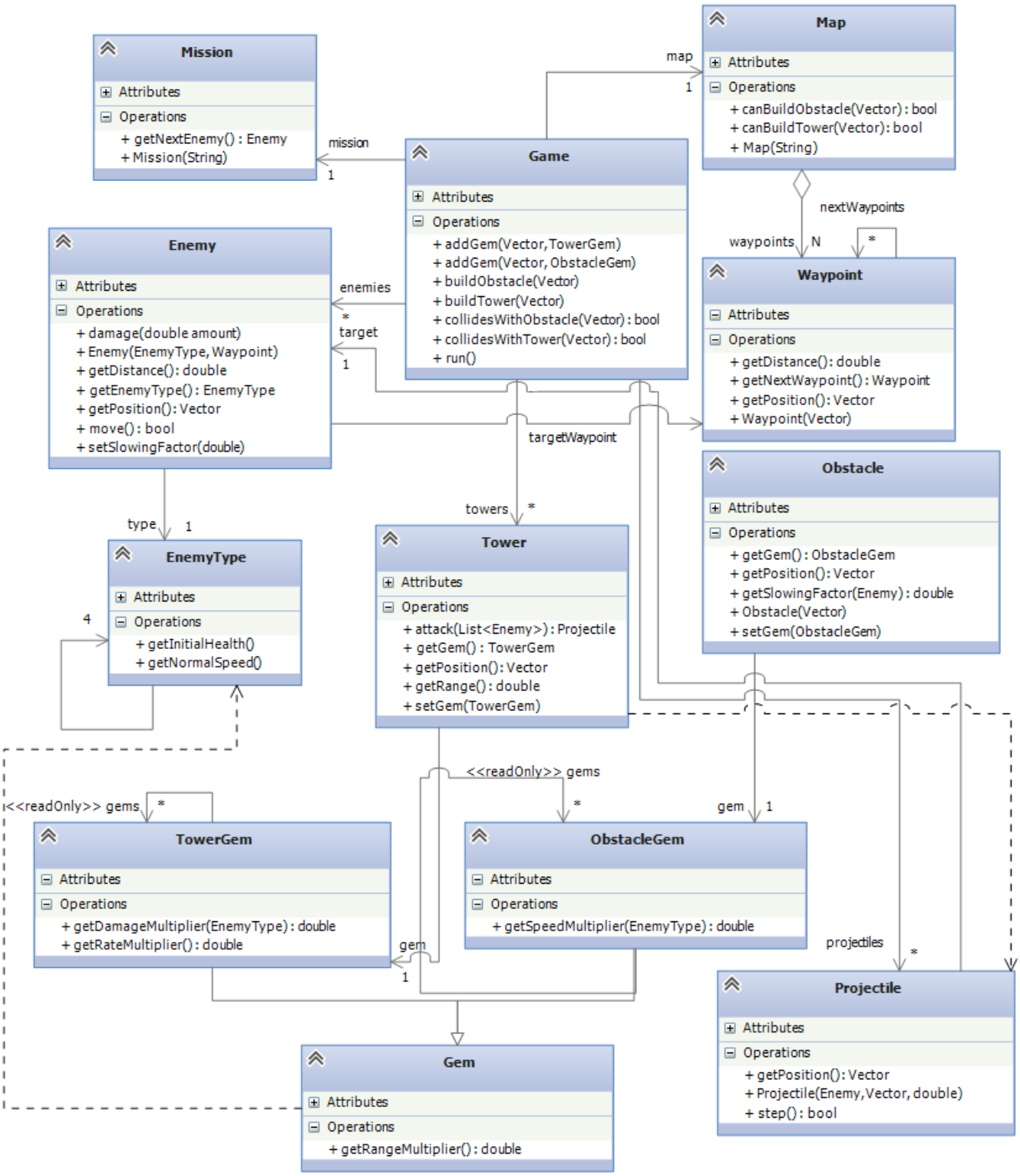
4.1.10. Tower

Egy tornyot valósít meg. Ha egy ellenség a hatókörébe ér, akkor elindít egy **Projectile**-t az irányába, a tüzelési gyakoriságának megfelelő időközönként. Ha több ellenfél van a hatókörében, akkor arra lő, aki a legközelebb van a célhoz.

4.1.11. TowerGem

Egy tornyokra rakható varázskő tulajdonságait tárolja, a **Gem** leszármazottja. Képes módosítani a tornyok alaptulajdonságait.

4.2. Statikus struktúra diagramok



4.1. ábra. Osztálydiagram

4.3. Osztályok leírása

4.3.1. Enemy

- Felelősség
  Az ellenségek pozíciójának, sebességének és életerejének nyilvántartása.
- Attribútumok
  - **EnemyType** type: Az egység típusa.
  - double health: Az ellenség fennmaradó életereje.
  - Vector position: Pillanatnyi pozíció a pályán.
  - **Waypoint** targetWaypoint: Az a **Waypoint**, ami felé az ellenség jelenleg tart.
  - double slowingFactor: Ha az ellenség belelép egy akadályba, akkor beállítja ennek az értékét. Az **EnemyType** sebességét ezzel kell beszorozni, hogy megkapjuk az ellenség tényleges jelenlegi sebességét.
- Metódusok
  - Enemy(EnemyType et, Waypoint start): Létrehoz egy **EnemyType** típusú **Enemy**-t a start **Waypoint**-helyén.
  - bool move(): Az ellenséget a sebességének megfelelő mértékben mozgatja a célja irányába. Ha az ellenség életereje 0 vagy kisebb, akkor igazzal tér visza, egyébként hamissal.
  - bool damage(double amount): Csökkenti az ellenség életerejét amount-al, igazzal tér vissza ha az ellenség meghalt.
  - Vector getPosition(): A position attribútum értékével tér vissza.
  - **EnemyType** getEnemyType(): Visszaadja az ellenség típusát.
  - double getDistance(): Visszaadja az ellenség céltól való távolságát a legrövidebb úton haladva.
  - void setSlowingFactor(double sf): Beállítja a slowingFactor-t sf-re.

4.3.2. EnemyType

- Felelősség
  Leírja egy bizonyos típusú (fajú) ellenség alapvető tulajdonságait. Egy-egy példányára hivatkozik az összes ellenség, amelyeknek ezáltal meghatározza a viselkedését. Az osztályból nem lehet példányosítani, csakis a statikus tagként szereplő objektumokat lehet felhasználni.
- Attribútumok
  - int cost: Az ilyen fajtájú ellenségek ára varázserőben.
  - double initialHealth: Az ilyen fajtájú ellenségek kezdeti életereje.
  - double normalSpeed: Az ilyen fajtájú ellenségek akadályoztatás nélküli haladási sebessége.
  - EnemyType dwarf: törp típus.
  - EnemyType elf: tünde típus.
  - EnemyType hobbit: hobbit típus.
  - EnemyType human: ember típus.
- Metódusok
  - double getInitialHealth(): Visszaadja az initialHealth attribútum értékét.

  - double getNormalSpeed(): Visszaadja a normalSpeed attribútum értékét.

- Megjegyzés
  Tervezésnél úgy láttuk, hogy leszármazás felesleges a különböző ellenségtípusok megkülönböztetésére, mert minden ellenségtípus ugyanúgy viselkedik, egyedül a paramétereik különböznek. Ezért hoztunk létre egy **EnemyType** osztályt. Minden **Enemy**-nek van pontosan egy hozzá tartozó **EnemyType**-ja. Az osztály statikus tagként tartalmazza a lehetséges ellenségtípusokat. Minden más osztály, ami az ellenségekkel foglalkozik vagy lekéri az **EnemyType**-on keresztül a szükséges adatokat, vagy egy HashMap-ben tárolja az adatokat, ahol a kulcsok a lehetséges **EnemyType**-ok. Így egy újabb ellenség bevétele sem okoz különösebb nehézségeket.

4.3.3. Game

- Felelősség
  A többi osztály nyilván tartása és összekötése, a játékbeli események vezérlése. A felhasználói felülettől érkező parancsok végrehajtása, és a játék állapotának rendelkezésre bocsájtása a kijelzéshez.

- Attribútumok
  - **Map** map: Referencia a kiválasztott pályára, amin a játék folyik.
  - **Mission** mission: Referencia a kiválasztott misszióra, amely alapján zajlik a játék.
  - List<**Enemy**> enemies: Az összes jelenleg élő ellenség található meg benne.
  - List<**Projectile**> projectiles: Eltárolja a jelenleg játékban lévő lövedékeket.
  - List<**Tower**> towers: Eltárolja a játékos megépített tornyait.
  - int magic: A játékos jelenlegi varázsereje.

- Metódusok
  - boolean run(): Ez a metódus futtatja a főciklust, amelyben maga a játék működik.
  - void buildTower(**Vector** position): Épít egy tornyot a paraméterül kapott helyen évő mezőre.
  - void buildObstacle(**Vector** position): Épít egy akadályt a paraméterül kapott helyen lévő mezőre.
  - void addGem(**Vector** position, **TowerGem** gem): A paraméterként kapott helyen lévő toronyra rárakja a paraméterként kapott varázskövet.
  - void addGem(**Vector** position, **ObstacleGem** gem): A paraméterként kapott helyen lévő akadályra rárakja a paraméterként kapott varázskövet.
  - boolean collidesWithTower(Vector p): megadja, hogy ha p helyre építenénk tornyot, az belelógna-e egy már megépített toronyba.
  - boolean collidesWithObstacle(Vector p): megadja, hogy ha p helyre építenénk akadályt, az belelógna-e egy már megépített akadályba.

4.3.4. Gem

- Felelősség
  Egy általános varázskő tulajdonságainak tárolása. Absztrakt osztály.

- Attribútumok
  - int cost: A varázskő ára varázserőben.
  - double rangeMultiplier: Megadja, hogy a varázskővel ellátott toronynak hányszorosára nő a hatótávolsága.

- Metódusok
  - double getRangeMultiplier(): Visszaadja a varázskő hatótávolság szorzóját.

4.3.5. Map

- Felelősség
  Tartlamaz az összes **Waypoint** referenciát, felelős a kapott fájlból beolvasni a pályát, és felépíteni azt, valamint meg kell mondja, hogy van-e út a megadott Vector-nál.
- Attribútumok
  - List <**Waypoint**> waypoints: A pályán található **Waypoint**-ok listája.
- Metódusok
  - Map(String file): Betölti a kapott elérési útvonalon a fájlt, és felépíti az alapján a pályát.
  - bool canBuildObstacle(**Vector**): A **Waypoint**-ok alapján visszaadja, hogy lehet-e akadályt építeni a megadott helyen.
  - bool canBuildTower(**Vector**): A **Waypoint**-ok alapján visszaadja, hogy lehet-e tornyot építeni a megadott helyen.

4.3.6. Mission

- Felelősség
  Felelőssége, felépíteni a listát, amely a beérkező ellenségeket és időzítésüket tartalmazza. Majd a megfelelő időközönként ki kell adja ezeket az egységeket a **Game** osztálynak.
- Attribútumok
  - List<Spawn> spawnList: A Spawn segédosztályban tárolt ellenség-idő-belépési pont, adatokat tárolja.
  - double lastSpawn: Tárolja az előző spawn idejét.
- Metódusok
  - Mission(String file): Betölti a kapott elérési útvonalon a fájlt, és felépíti az alapján a listát (spawnList).
  - **Enemy** getNextEnemy(): A spawnList listaból a következő elemet megvizsgaálja, és ha spawnolni kell a következő ellenséget, akkor visszaadja azt.

4.3.7. Obstacle

- Felelősség
  Felelős, az ellenfelek lassításáért, úgy, hogy meg kell tudnia mondani a pozícióját, valamint, hogy az adott ellenséget mennyire lassítja.
- Attribútumok
  - int cost: Az akadály ára varázserőben.
  - **Gem** gem: Eltárol egy referenciát egy **Gem** típusú objektumra, ami meghatározza, hogy az adott épület milyen echant alatt áll.
  - **Map<EnemyType**, double> slowingFactor: Megadja mekkora az adott típusú ellenfélre kifejtett hatása az akadálynak.
  - **Vector** position: Az akadály koordinátáit tárolja.

- Metódusok
  - Obstacle(Vector pos): Létrehoz egy **Obstacle** objektumot, a pozícióját pos-ra állítva.
  - int getCost(): Visszatér a torony árával.
  - double getSlowingFactor(**Enemy** enemy): Visszatér azzal az értékkel, amivel az adott ellenfelet lassítja.
  - **Gem** getGem(): Visszaadja az épületen található varázskövet.
  - void setGem(**Gem** gem): Beállítja az épületen lévő varázskövet. Ha már volt az épületen varázskő, akkor az előző megszűnik.
  - Vector getPosition(): Visszaadja a position attribútumot.
  - double getRange(): Visszaadja az akadály hatótávolságát.

4.3.8. ObstacleGem

- Felelősség
  Egy akadályra rakható varázskő tulajdonságainak tárolása.
- Ősosztályok
  - Gem
- Attribútumok
  - HashMap<**EnemyType**, Double> speedMultiplier: Megadja, hogy a varázskővel elátott akadályon áthaladó adott típusú ellenség sebessége hányadára csökken.
- Metódusok
  - double getSpeedMultiplier(**EnemyType** enemyType): Visszaadja varázskő sebesség szorzóját egy adott típusú ellenséghez.

4.3.9. Projectile

- Felelősség
  Követni a cél ellenséget, majd sebezni ha eléri.
- Attribútumok
  - double damage: A lövedék sebzése, ennyivel csökkenti a cél ellenség életerejét amikor eltalálja.
  - **Vector** position: A lövedék pozíciója.
  - double speed: A lövedék sebessége.
  - **Enemy** enemy: A lövedék cél **Enemy**-je.
- Metódusok
  - Projectile(**Enemy** enemy, Vector position, double speed): Konstruktor, átveszi a cél **Enemy**-t, a kezdő pozíciót és sebességet.
  - boolean step(): speed-el mozgatja a lövedéket az ellenség irányába. Ha eltalálta az ellenséget vagy az ellenség már meghalt, akkor true-t ad vissza, különben false-t.
  - Vector getPosition(): Visszaadja a lövedék pozícióját.

4.3.10. Tower

- Felelősség
  Felelős **Projectile**-ok létrehozásához, azok megfelelő felparaméterezésével. Továbbá felelős azért, hogy **Projectile**-okat csak a megadott időközönként lőjjön ki.
- Attribútumok
  - int cost: A torony ára varázserőben.
  - **Gem** gem: Eltárol egy referenciát egy **Gem** típusú objektumra, ami meghatározza, hogy az adott épület milyen echant alatt áll.
  - **HashMap<EnemyType**, double> damage: Megadja mekkora az adott típusú ellenfélre kifejtett hatása a toronynak.
  - Vector position: Visszatér az épület koordinátáival.
- Metódusok
  - **Projectile** attack(List <**Enemy**>): Először megnézi, hogy lőhet-e, ha nem akkor semmivel se tér vissza. Ha igen akkor végignézi a kapott listában az ellenségeket, és amelyik a hatótávolságán belül van, és a legközelebb a célhoz, arra kilő egy **Projectile**-t, majd a visszatérési értékében visszaadja azt. A **Projectile**-t felparaméterezi az ellenséghez megfelelő sebzési adatokkal.
  - int getCost(): Visszatér a cost attribútummal.
  - **Gem** getGem(): Visszaadja az épületen található varázskövet.
  - void setGem(**TowerGem** gem): Beállítja az epületen lévő varázskövet. Ha már volt az épületen varázskő, akkor az előző megszűnik.
  - Vector getPosition(): Visszaadja a position attribútumot.

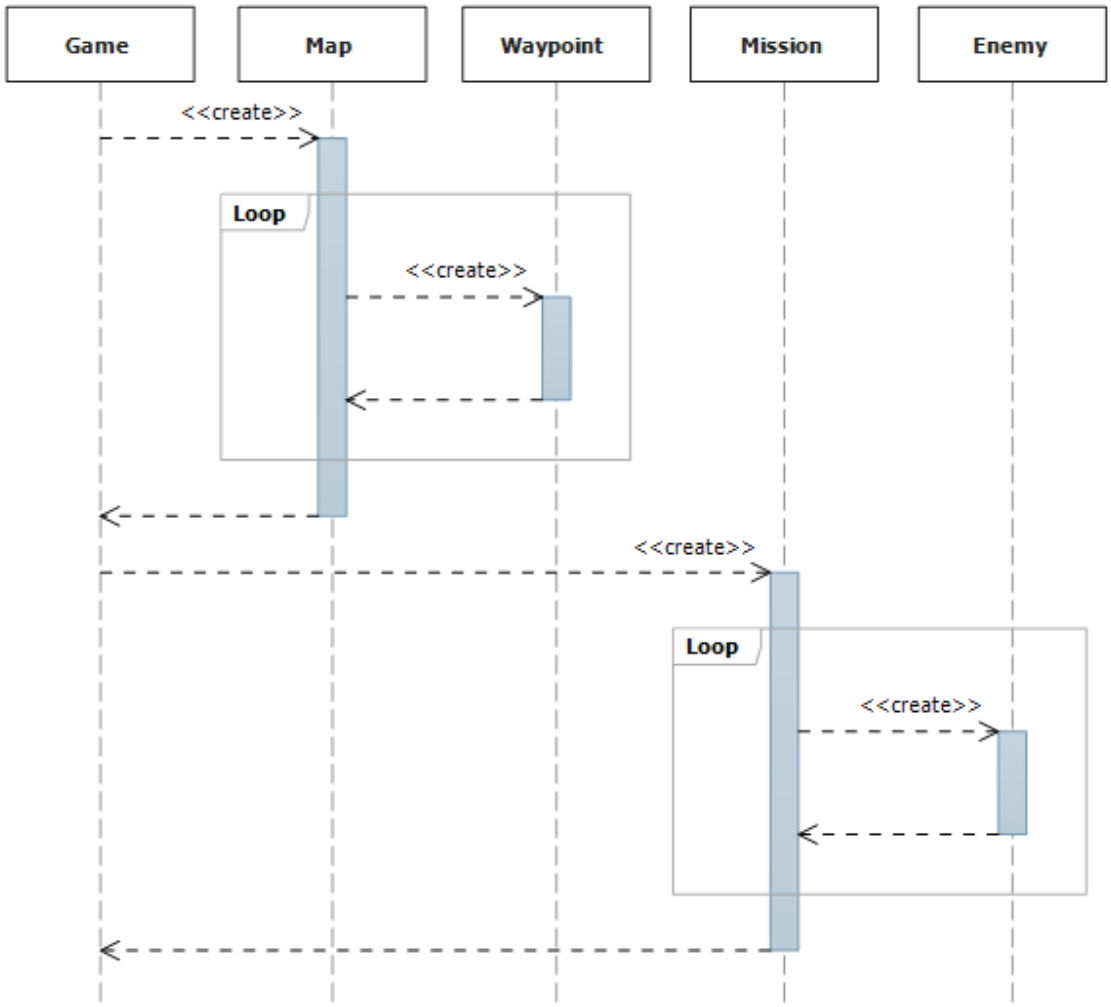
4.3.11. TowerGem

- Felelősség
  Egy toronyra rakható varázskő tulajdonságainak tárolása.
- Ősosztályok
  - Gem
- Attribútumok
  - double rateMultiplier: Megadja, hogy a varázskővel ellátott toronynak hányszorosára nő a tüzelési sebessége.
  - HashMap<**EnemyType**, double> damageMultiplier: Megadja, hogy a varázskővel ellátott toronynak hányszorosára nő a sebzése egy adott típusú ellenséggel szemben.
- Metódusok
  - double getRateMultiplier(): Visszaadja a varázskő tüzelési sebesség szorzóját.
  - double getDamageMultiplier(**EnemyType** enemyType): Visszaadja varázskő sebzés szorzóját egy adott típusú ellenséghez.

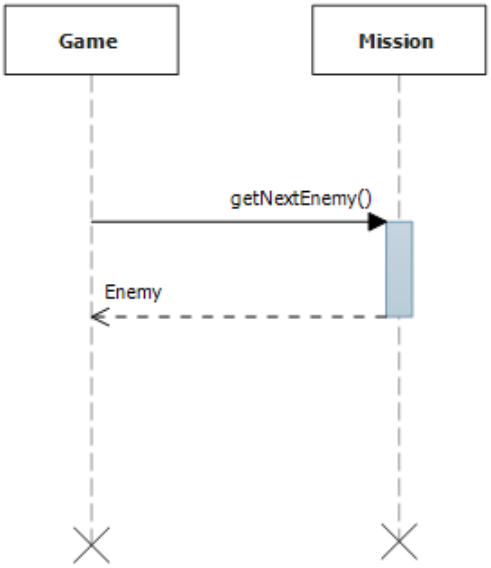
4.3.12. Waypoint

- Felelősség
  Útvonalat kijelölni az ellenségeknek, úgy, hogy megadja a pozícióját, amely felé az ellenségek mehetnek, valamint a következő **Waypoint**-ot ami felé menniük kell, ha egyszer elérték ezt a **Waypoint**-ot.
- Attribútumok
  - double distance: A **Waypoint** távolságát a céltól tárolja.
  - List <Waypoint, double> nextWaypoints: A következő **Waypoint**-okat és a hozzájuk tartozó valószínűségeket tárolja el.
  - Vector position: A **Waypoint** helyét tárolja.
- Metódusok
  - double getDistance(): Visszaadja a distance attribútumot.
  - **Waypoint** getNextWaypoint(): Visszatér a nextWaypoints listából véletlenszerűen kiválasztott **Waypoint**-al
  - Vector getPosition(): visszatér a position attribútummal.
  - **Waypoint**(Vector v): Konstruktor, beállítja a position attribútumot.

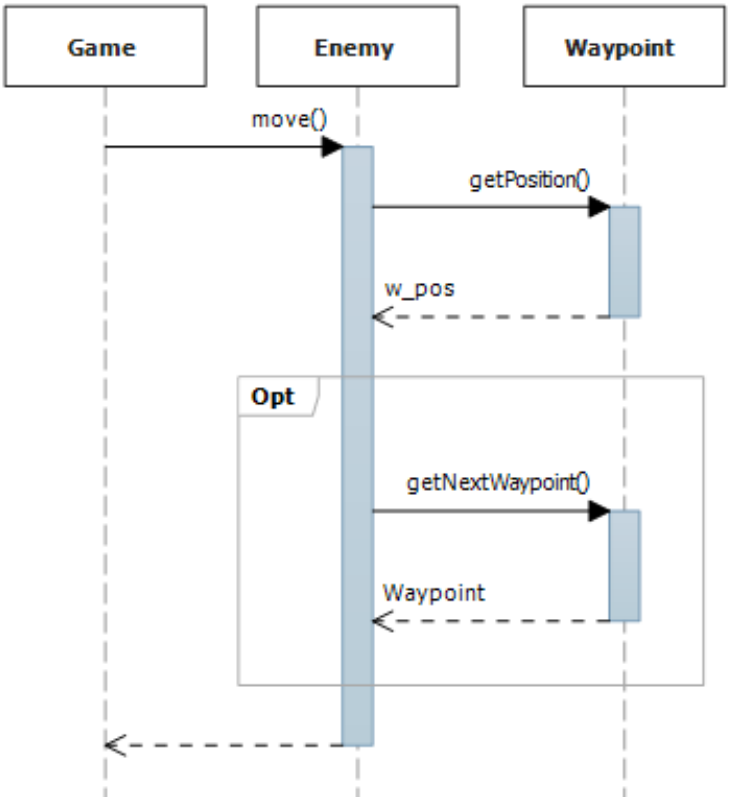
4.4. Szekvencia diagramok



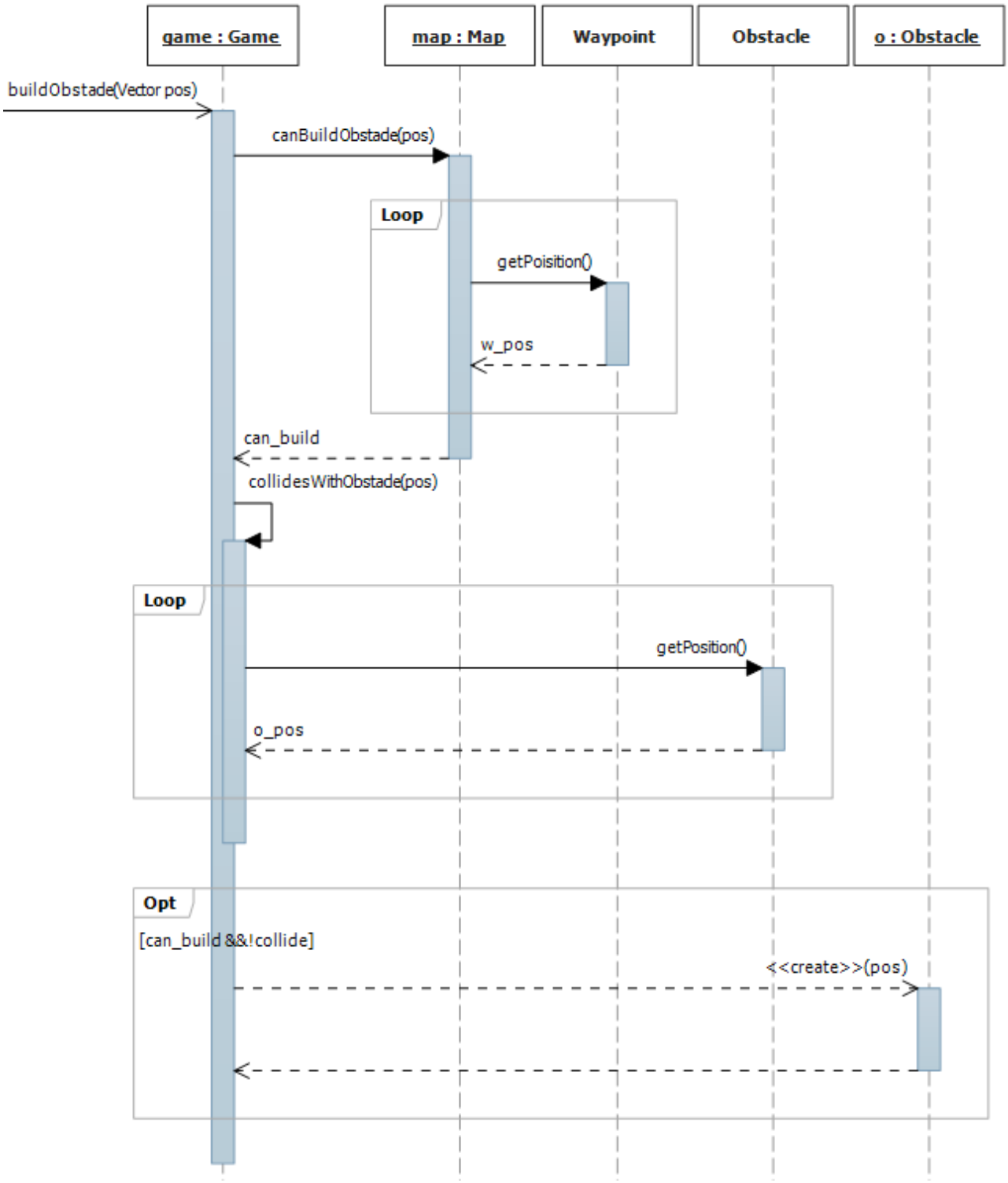
4.2. ábra. Inicializálás. Betölti a mapet és a missiont.



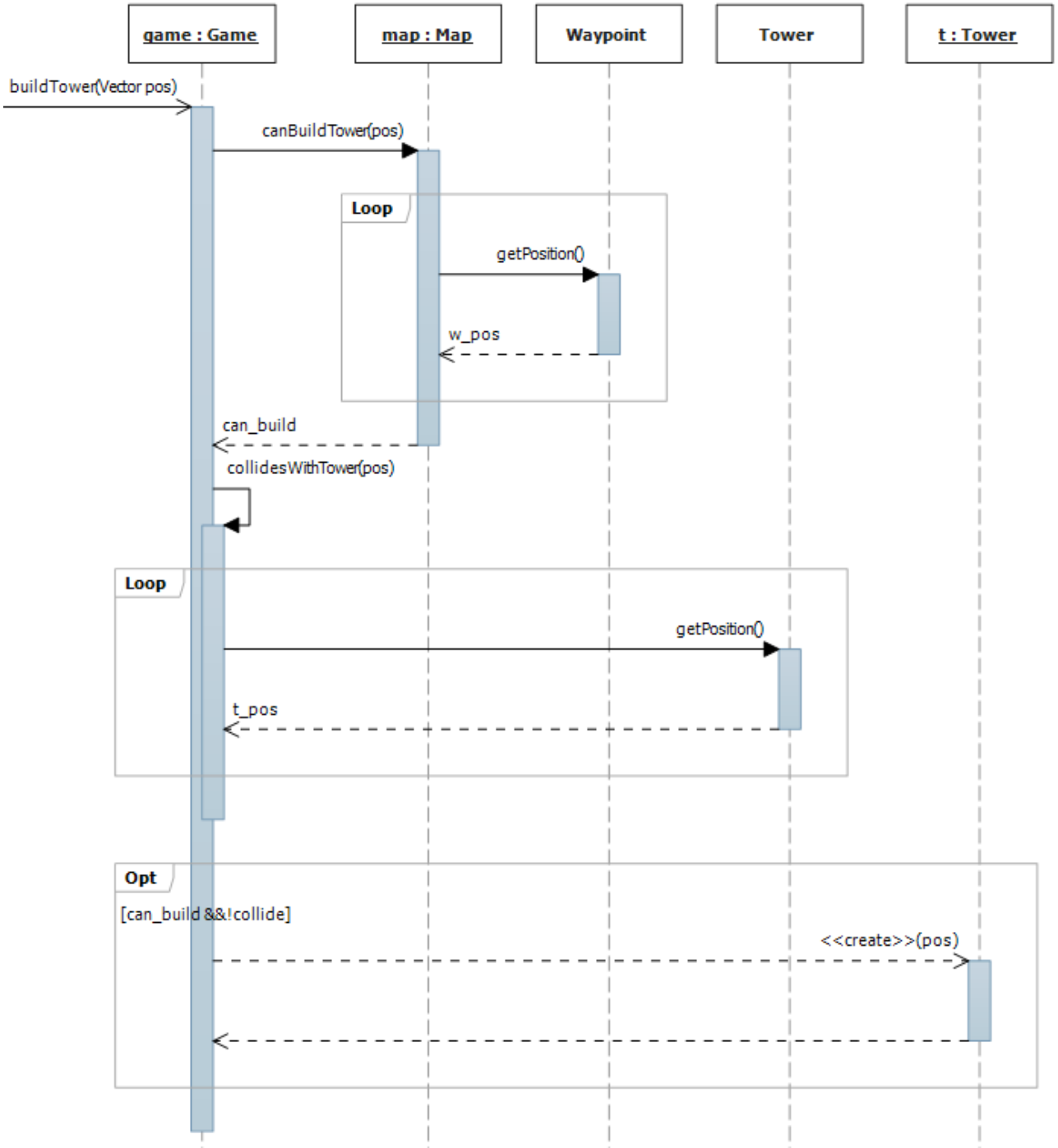
4.3. ábra. Ellenségek ütemezése.



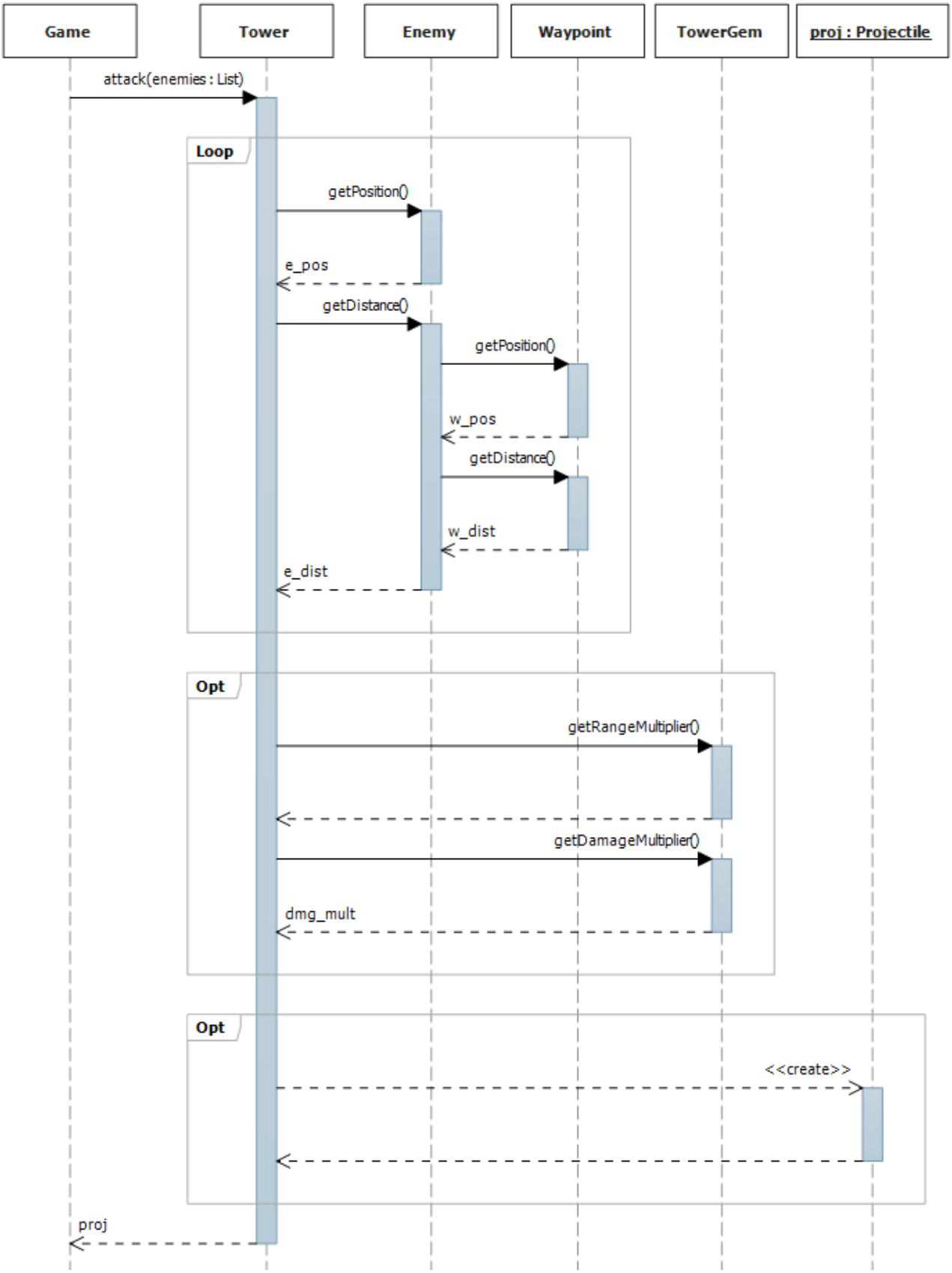
4.4. ábra. Ellenség mozgatása. Ha elérte a waypointot lekéri a következőt.



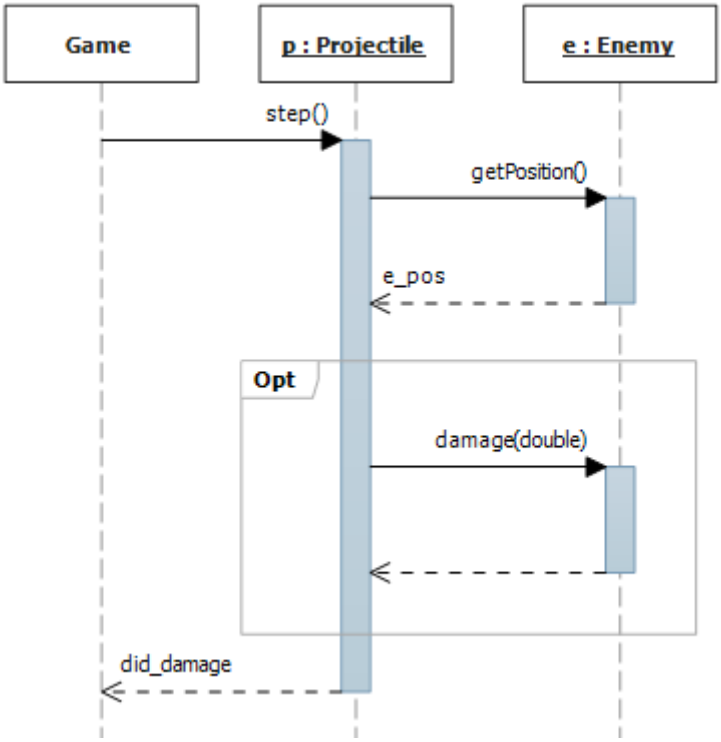
4.5. ábra. Akadály építése. Megnézi, hogy szabad-e a helyre építeni, ha szabad épít.



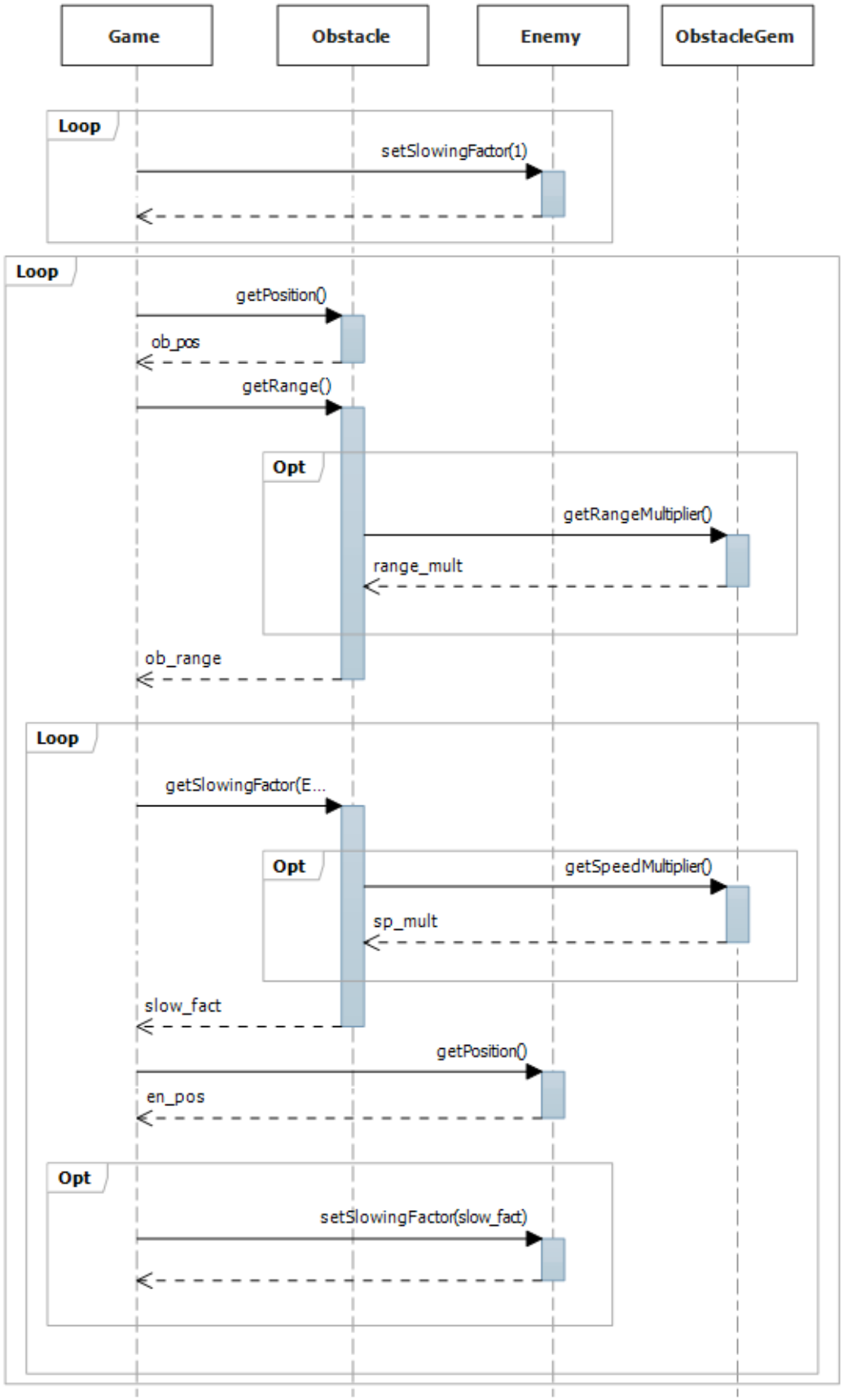
4.6. ábra. Torony építése. Megnézi, hogy szabad-e a helyre építeni, ha szabad épít.



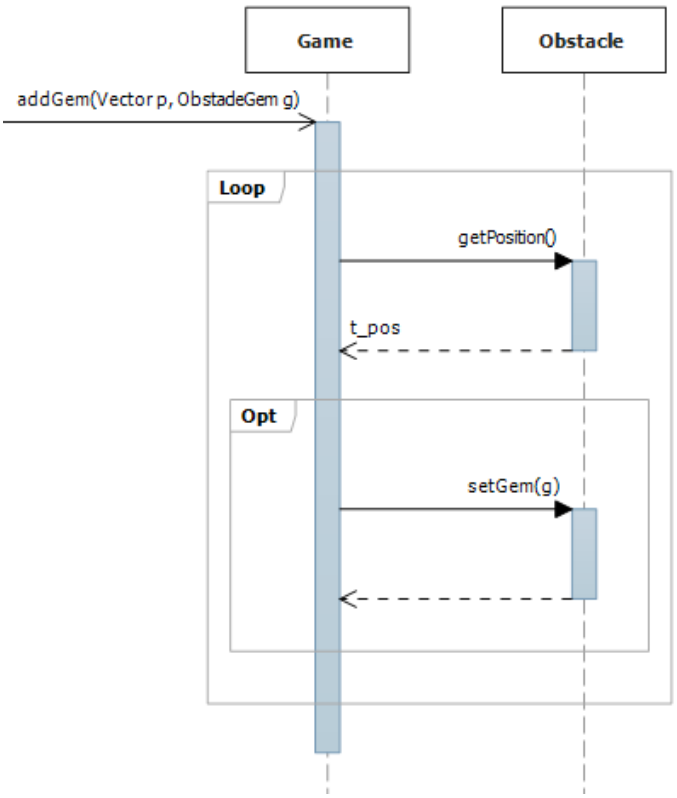
4.7. ábra. Ellenségek támadása. Minden ellenségre megnézi mennyire van távol a céltól és a legelsőre lő.



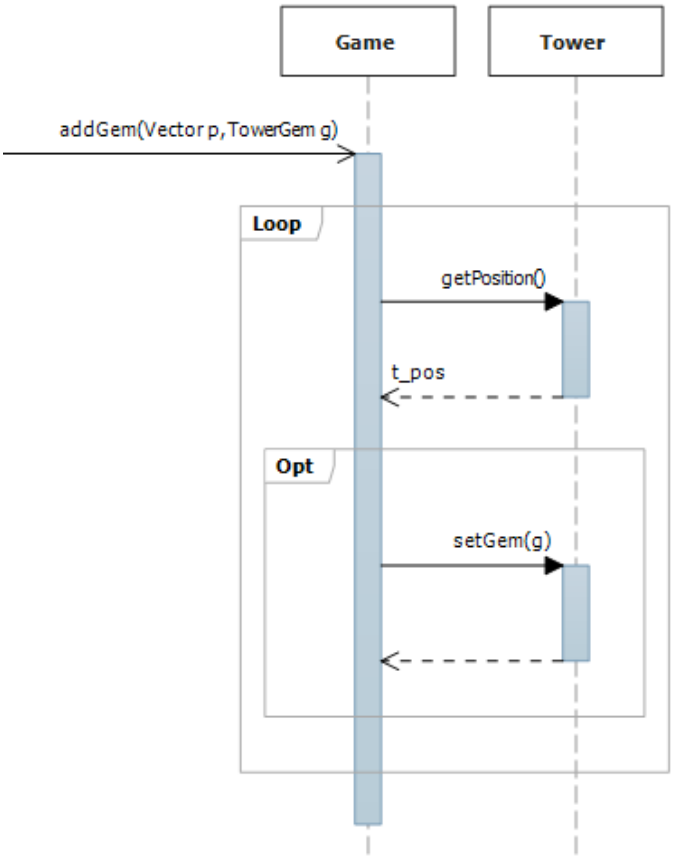
4.8. ábra. Lövedék léptetése.



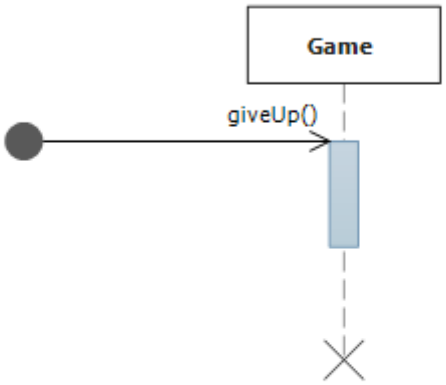
4.9. ábra. Minden akadály megnézi minden ellenségre, hogy a hatókörében van-e, ha igen lelassítja őket.



4.10. ábra. Varázskő feltétele akadályra.

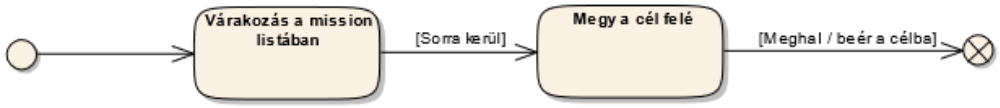


4.11. ábra. Varázskő feltétele toronyra.

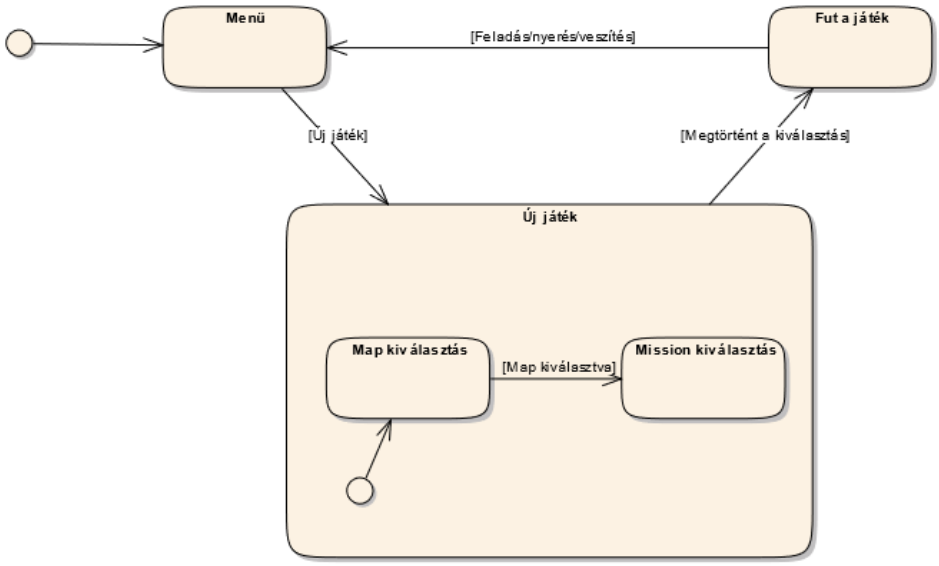


4.12. ábra. Feladás.

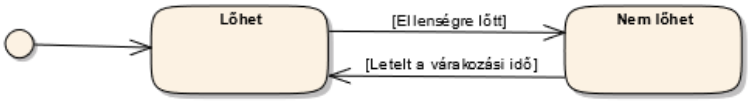
4.5. State-chartok



4.13. ábra. Egy ellenség állapotdiagramja



4.14. ábra. A játék állapotdiagramja



4.15. ábra. Egy torony állapotdiagramja

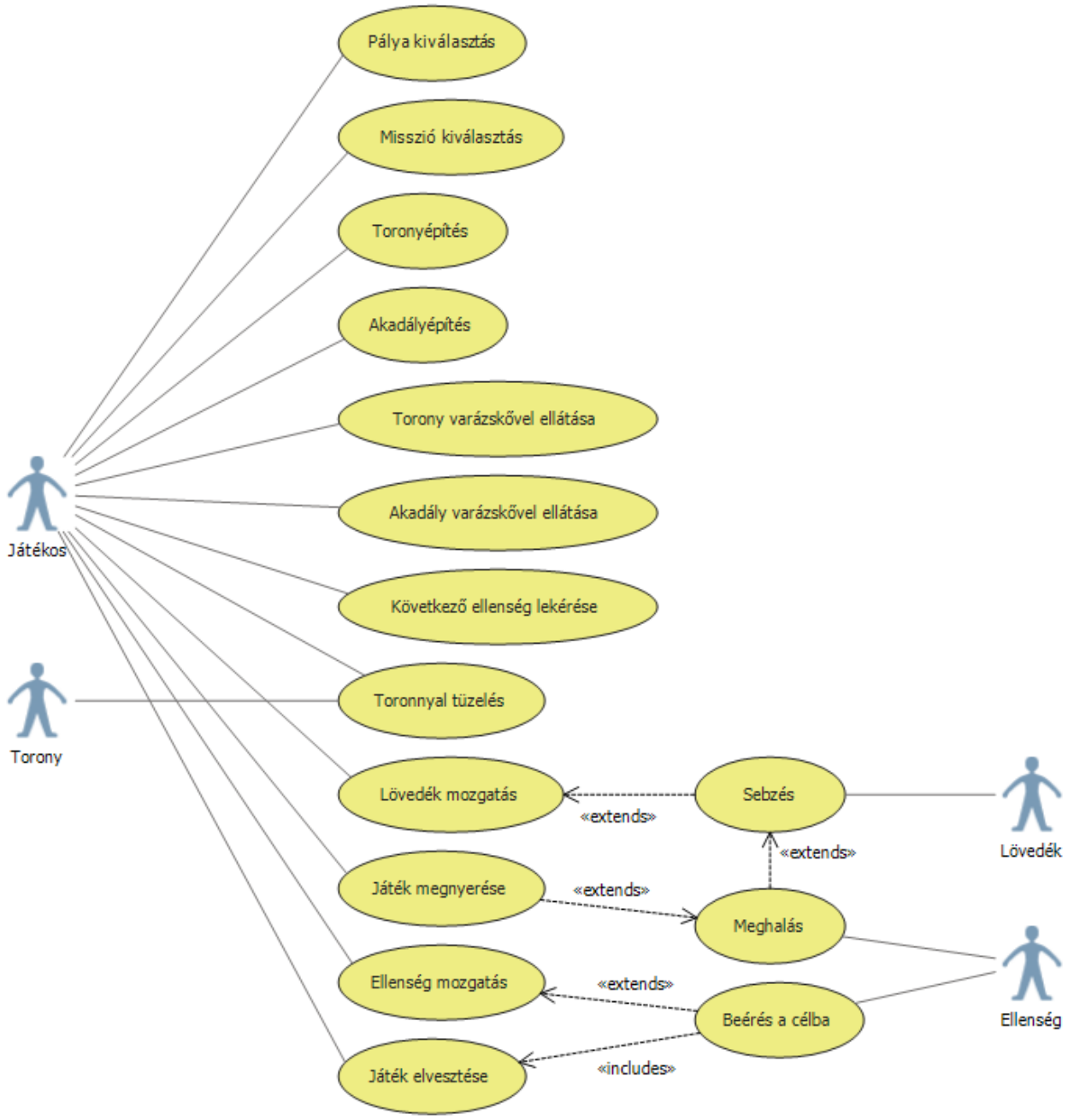
4.6. Napló

Kezdet	Időtartam	Résztvevők	Leírás
2014.03.05. 8:00	1 óra	**Nusser** **Szabó** **Tallér** **Török**	Kezdeti megbeszélés, hibák áttekintése.
2014.03.09. 10:30	4 óra	**Tallér**	Szekvencia- és osztálydiagramok elkészítése.
2014.03.09. 17:00	2 óra	**Nusser**	Map és Waypoint osztályok leírása, state chartok kijavítása.
2014.03.09. 22:00	2,5 óra	**Tallér**	Osztályok leírásának frissítése, ellenőrzése. Dokumentum összeszerkesztése.
2010.03.09. 22:30	2 óra	**Nusser**	Tower, Obstacle és Projectile osztályok átírása.
2010.03.09. 22:30	2 óra	**Szabó**	Enemy, Gem, Game osztályok frissítése.
2010.03.10. 8:30	0,5 óra	**Tallér**	Ellenőrzés.

5. Szkeleton tervezése

5.1. A szkeleton modell valóságos use-case-ei

5.1.1. Use-case diagram



5.1. ábra. Use case diagram

5.1.2. Use-case leírások

Use-case neve	**Pálya kiválasztás**
Rövid leírás	A felhasználó kiválaszt egy pályát.
Aktorok	Felhasználó
Forgatókönyv	A program felkínál egy listát az elérhető pályák neveiről, amelyből a felhasználó kiválasztja azt, amelyiken játszani szeretne.

Use-case neve	**Misszió kiválasztás**
Rövid leírás	A felhasználó kiválaszt egy missziót.
Aktorok	Felhasználó
Forgatókönyv	A program felkínál egy listát az előzőleg kiválasztott pályán elérhető missziók neveiről, amelyből a felhasználó kiválasztja azt, amelyiken játszani szeretne.

Use-case neve	**Toronyépítés**
Rövid leírás	A felhasználó felépít egy tornyot.
Aktorok	Felhasználó
Forgatókönyv	A felhasználó megad egy pozíciót, ahova felépül egy torony.

Use-case neve	**Akadályépítés**
Rövid leírás	A felhasználó felépít egy akadályt.
Aktorok	Felhasználó
Forgatókönyv	A felhasználó megad egy pozíciót, ahova felépül egy akadály.

Use-case neve	**Torony varázskővel ellátása**
Rövid leírás	A felhasználó ellát egy tornyot egy varászkővel.
Aktorok	Felhasználó
Forgatókönyv	A felhasználó kiválaszt egy tornyot, majd egy varázskövet, amivel a toronyot erősíti.

Use-case neve	**Akadály varázskővel ellátása**
Rövid leírás	A felhasználó ellát egy akadályt egy varázskővel.
Aktorok	Felhasználó
Forgatókönyv	A felhasználó kiválaszt egy akadályt, majd egy varázskövet, amivel az akadályt erősíti.

Use-case neve	**Következő ellenség lekérése**
Rövid leírás	A misszió következő ellensége elindul.
Aktorok	Felhasználó
Forgatókönyv	A misszió soron következő ellensége létrejön, és elindul a pályán a célja felé.

Use-case neve	**Toronnyal tüzelés**
Rövid leírás	Egy torony lő egyet.
Aktorok	Felhasználó, Torony
Forgatókönyv	Egy kiválasztott torony kibocsát egy lövedéket egy megadott ellenség felé.

Use-case neve	**Ellenség mozgatás**
Rövid leírás	Egy ellenség halad.
Aktorok	Ellenség
Forgatókönyv	A kiválasztott ellenség megtesz egy lépést a célja felé vezető úton.

Use-case neve	**Beérés a célba**
Rövid leírás	Egy ellenség elérte a célt.
Aktorok	Ellenség
Forgatókönyv	Egy ellenség mozgása után elérte a célt, így a játéknak vége.

Use-case neve	**Játék elvesztése**
Rövid leírás	A játék véget ér.
Aktorok	Játékos
Forgatókönyv	A játék befejeződik, a játékos vesztett

Use-case neve	**Lövedék mozgatás**
Rövid leírás	Egy lövedék halad.
Aktorok	Lövedék
Forgatókönyv	A kiválasztott lövedék megtesz egy lépést a célpontja felé.

Use-case neve	**Sebzés**
Rövid leírás	Egy lövedék célt ér.
Aktorok	Lövedék
Forgatókönyv	Egy lövedék mozgása közben elérte a célpontját, azt megsebezte.

Use-case neve	**Meghalás**
Rövid leírás	Egy ellenség életereje elfogy.
Aktorok	Ellenség
Forgatókönyv	Ha egy lövedék célba érésekor az ellenségnek kevesebb életereje van, mint amennyit a lövedék sebez, az ellenség meghal, eltűnik.

Use-case neve	**Játék megnyerése**
Rövid leírás	Az utolsó ellenség is meghal.
Aktorok	Játékos
Forgatókönyv	Egy ellenség halálakor a játékos megnyeri a játékot, ha nincs több ellenség életben.

5.2. A szkeleton kezelői felületének terve, dialógusok

A szkeleton menüvezérelt módon fog működni. A felhasználónak meg kell adnia a kívánt parancsnak a kódját, majd a program lefuttatja azt. A program indítása után ki kell választani a pályát utána pedig a missziót. A menü felépítése itt látható:

```
1.  Torony építés
       *1.1 Érvényes helyre akarunk építeni?  I/N
       *1.2 Ütközik másik toronnyal?  I/N
2.  Akadály építés
       *2.1 Érvényes helyre akarunk építeni?  I/N
       *2.2 Ütközik másik akadállyal?  I/N
3.  Következő ellenség lekérése
       *3.1 Van következő ellenség?  I/N
4.  Ellenség mozgatás
       *4.1 Elérte az ellenség a Waypointot?  I/N
       *4.2 Hatósugarában van egy akadálynak?  I/N
              *4.2.1 Az akadályon van varázskő?  I/N
5.  Torony tüzelés
       *5.1 Van a tornyon varázskő?  I/N
       *5.2 Van a torony hatósugarán belül ellenség?  I/N
6.  Lövedék mozgatása
       *6.1 Elérte az ellenséget?  I/N
7.  Varázskő felrakása
       *7.1 Toronyra vagy akadályra?  T/A
       *7.2 Érvényes helyet adtunk meg?  I/N
8.  Ellenségek lassítása
       *8.1 Van varázskő az akadályon I/N?
       *8.2 Van ellenség az akadály hatókörében I/N?
9.  Kilépés
       *8.1 Nyerés, vesztés, feladás vagy kilépés a programból?  N/V/F/K
```

A *-gal jelölt menüpontokat nem lehet kiválasztani, ezt a program automatikusan megteszi, ha a felhasználó a parancs szülőjét meghívta. Ezek (1 kivétellel) igen/nem típusú kérdések. A kérdések végén láthatóak a lehetséges válaszok. A 8. pontnál az N, V, F parancsok visszaléptetik a felhasználót a program elejére, tehát újra megkérdezi, hogy melyik pályát és missziót akarjuk kiválasztani.
Az alábbi példa egy olyan interakciót mutat be, ahol a felhasználó építeni akar egy tornyot, ami érvényes helyen van, de ütközne egy másik toronnyal:

```
?  Adja meg a parancs kódját:  1
- 1.  Torony építése
>       ->[:Game].buildTower(pos):
>              ->[:Map].canBuildTower(pos):
>                     ->[:Waypoint].getPosition():
?                     2.1.  Érvényes a hely?  I/N: I
<                     <-[:Waypoint].getPosition()
<              <-[:Map].canBuildTower():
>              ->[:Game].collidesWithTower(pos):
>                     ->[:Tower].getPosition():
?                     2.2.  Ütközik másik toronnyal?  I/N: I
<                     <-[:Tower].getPosition()
>              <-[:Game].collidesWithTower(pos)
>       <-[:Game].buildTower()
```

```
?  Adja meg a parancs kódját:
```

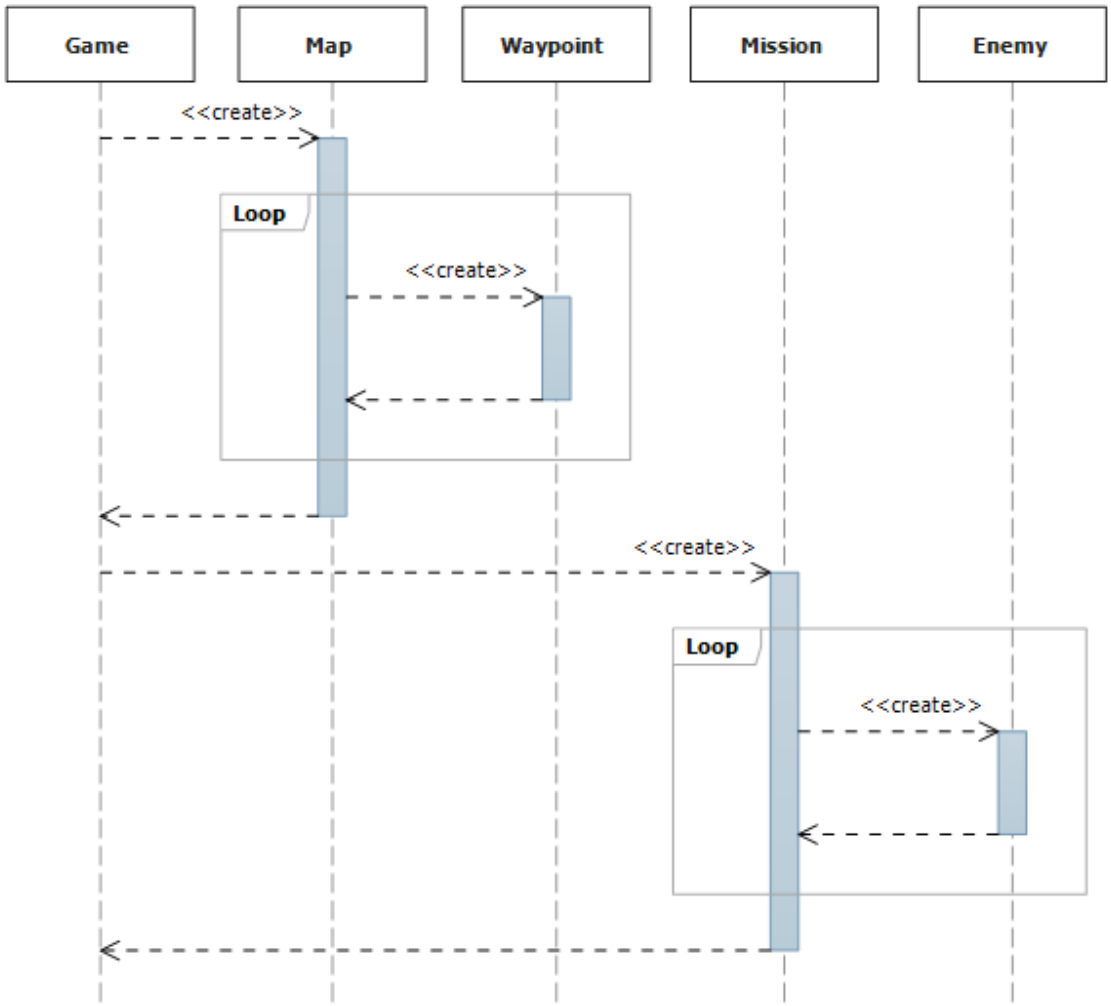
Minden sor elején van 1 karakter, ami a sor típusát jelöli.

- ’?’: kérdés, felhasználói interakcióra vár
- ’-’: megjegyzés
- ’>’: metódusba lépés
- ’<’: metódusból visszatérés

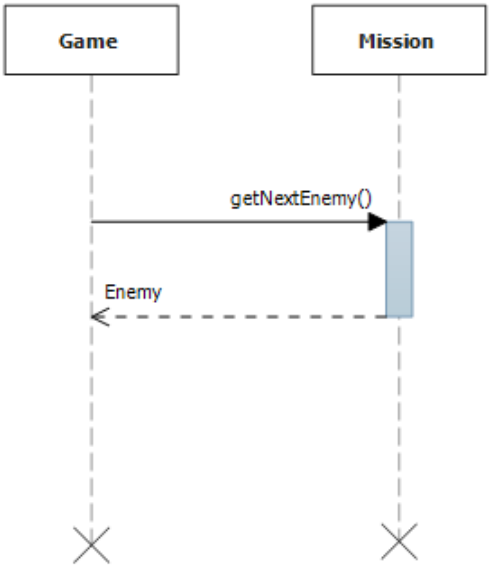
A metódusba lépéskor és visszatérésnél, egy nyíl jelöli, hogy hívásról vagy visszatérésről van-e szó majd szögletes zárójelek között található az objektum típusa, ezután pedig a metódus neve. Például: ->[:Osztály].metódus():

5.3. Szekvencia diagramok a belső működésre

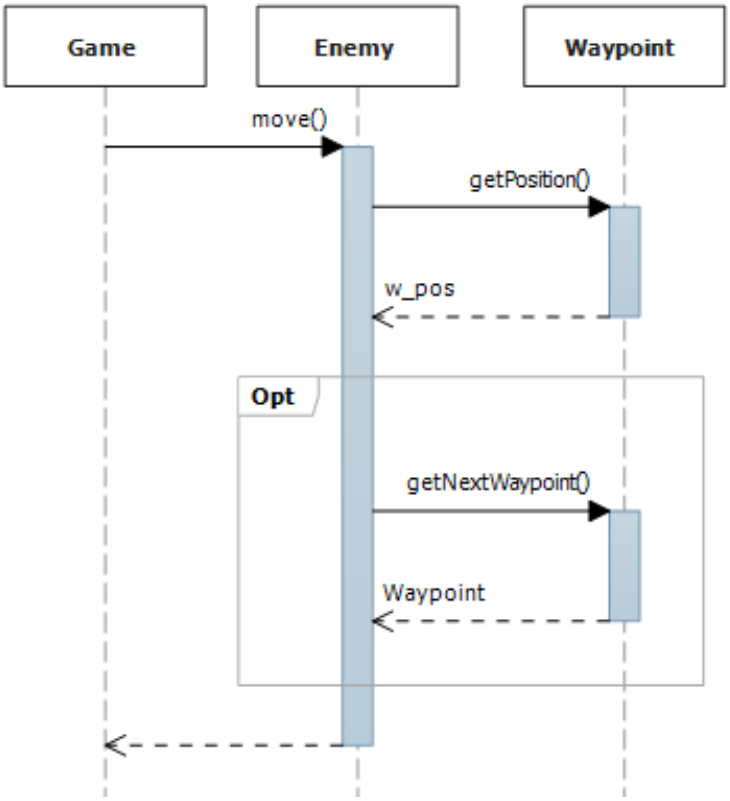
Az előző fejezetben lévő szekvencia diagramok továbbra is érvényesek.



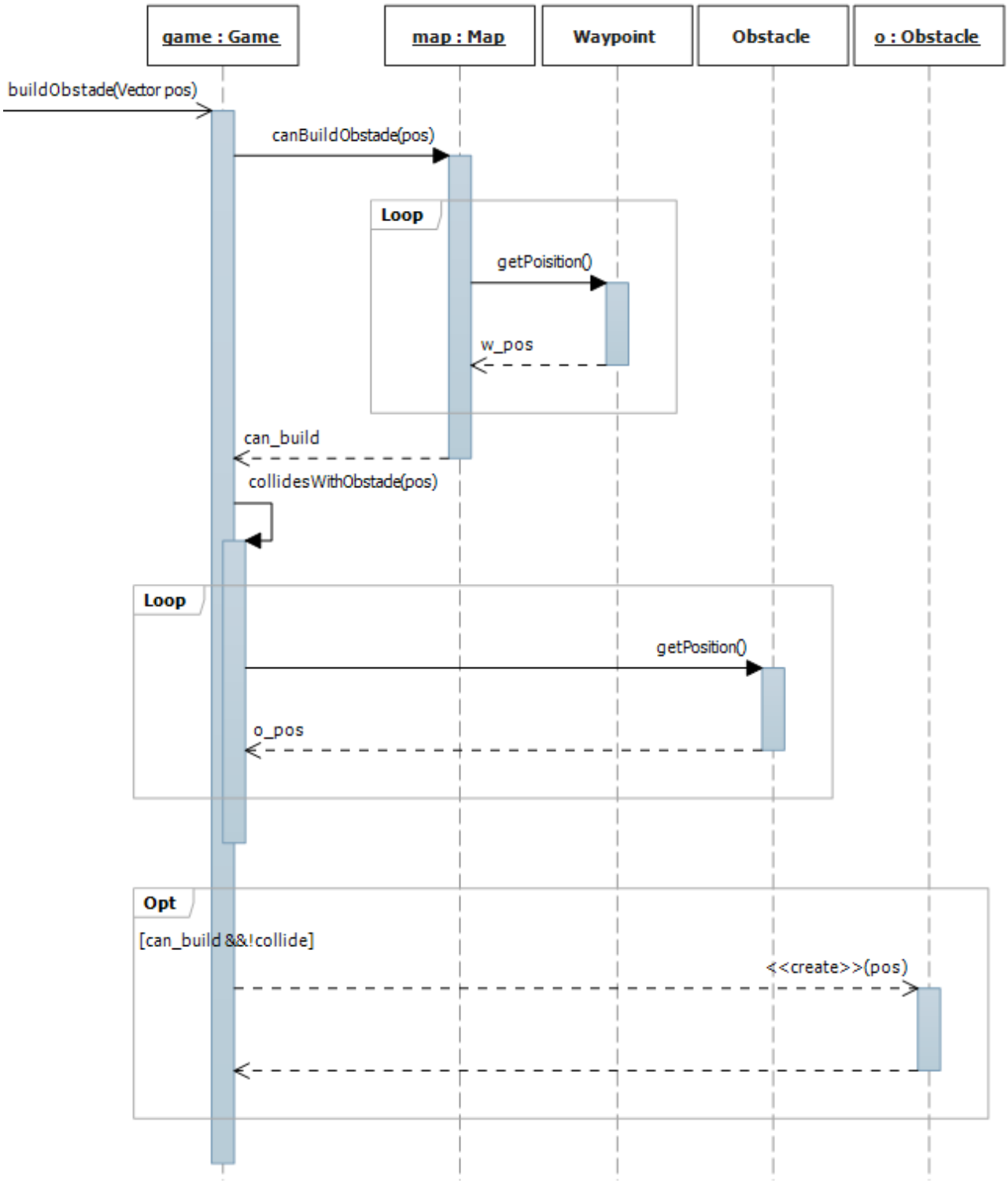
5.2. ábra. Inicializálás. Betölti a mapet és a missiont.



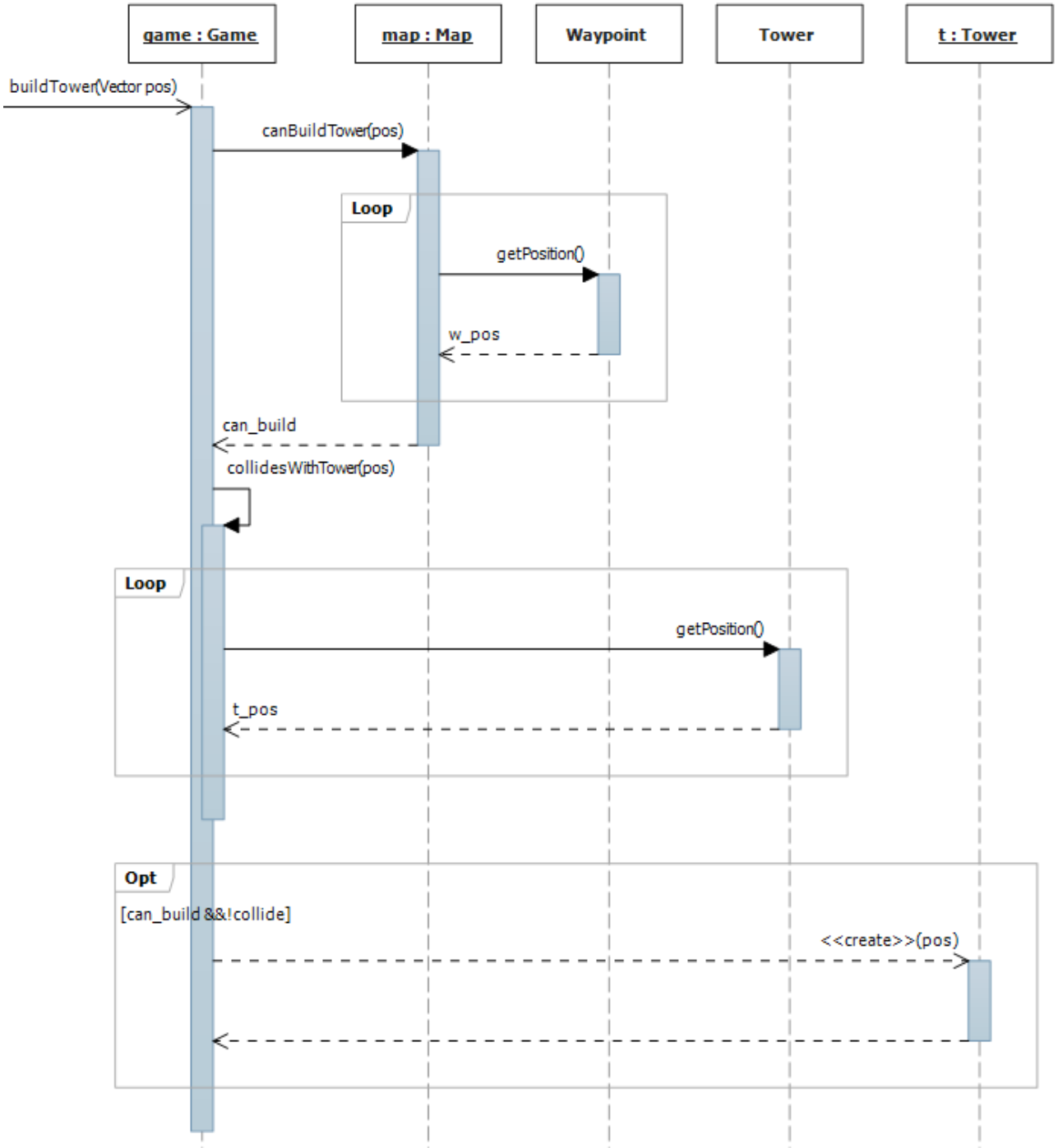
5.3. ábra. Ellenségek ütemezése.



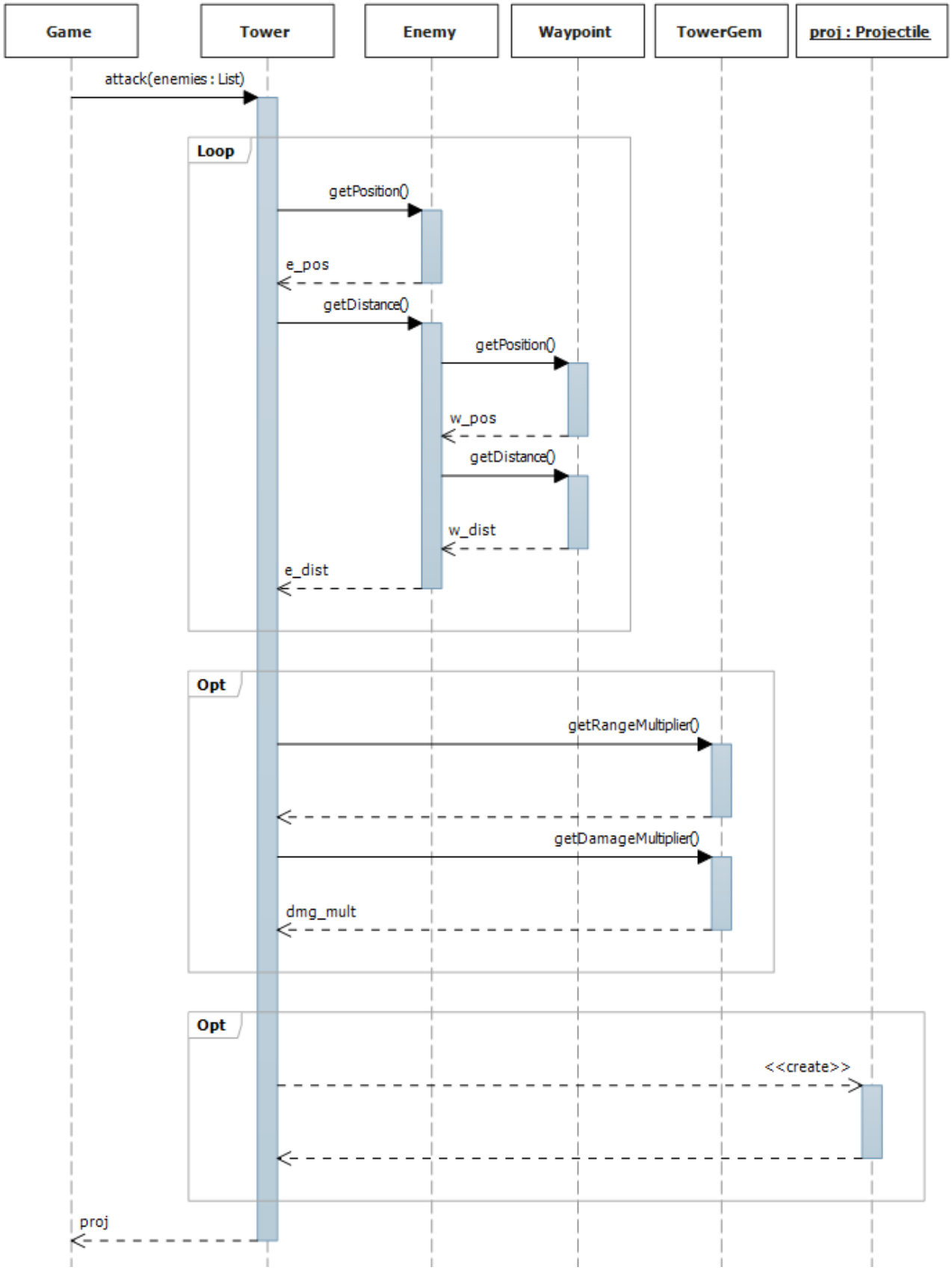
5.4. ábra. Ellenség mozgatása. Ha elérte a waypointot lekéri a következőt.



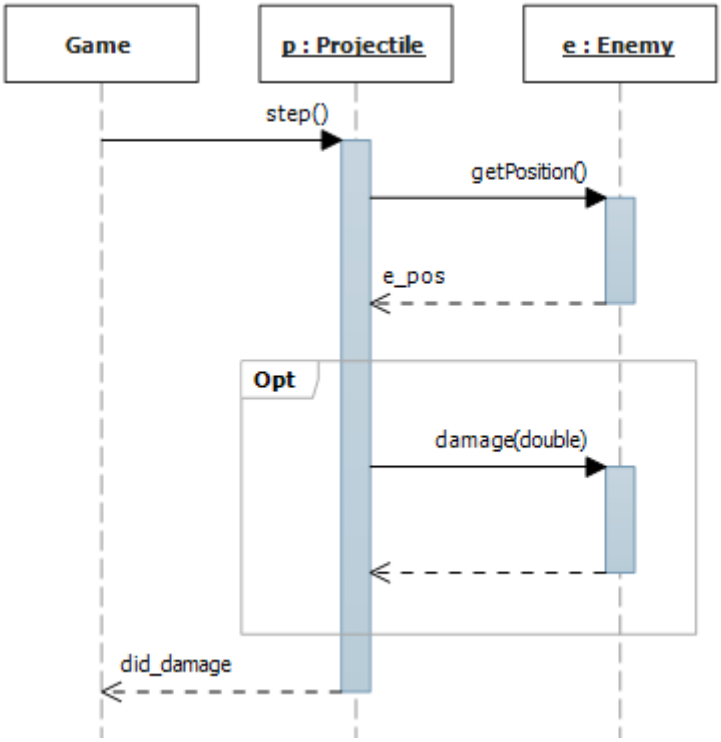
5.5. ábra. Akadály építése. Megnézi, hogy szabad-e a helyre építeni, ha szabad épít.



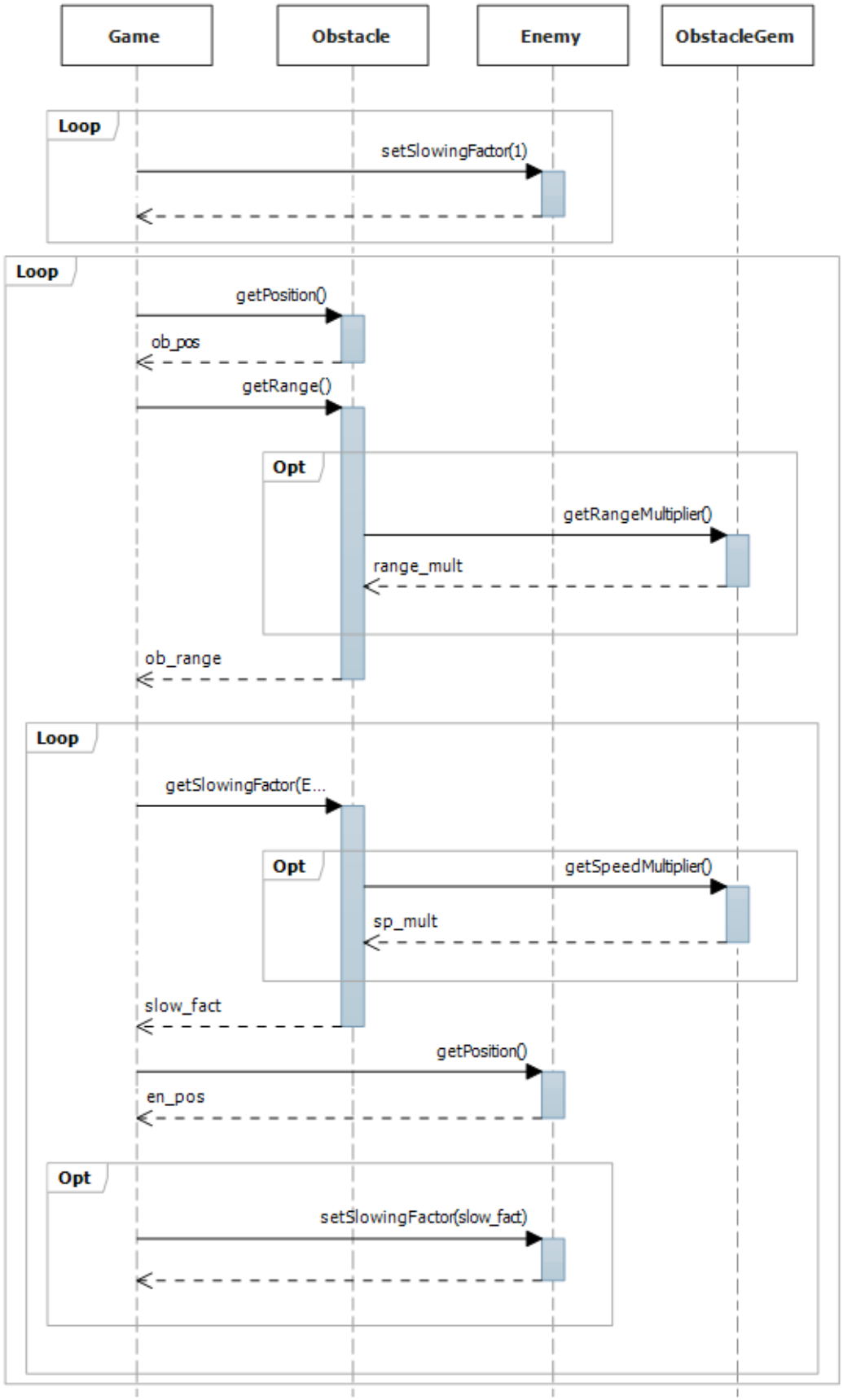
5.6. ábra. Torony építése. Megnézi, hogy szabad-e a helyre építeni, ha szabad épít.



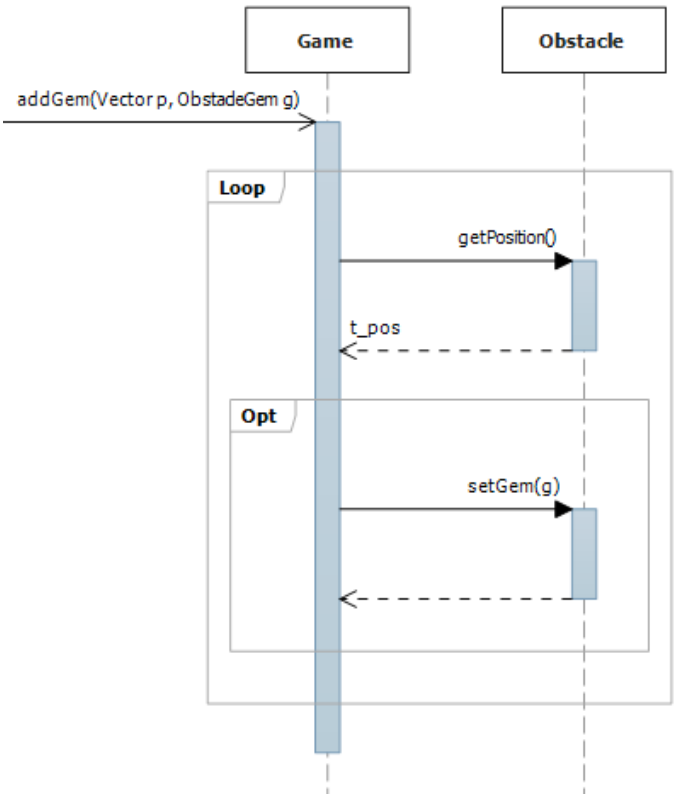
5.7. ábra. Ellenségek támadása. Minden ellenségre megnézi mennyire van távol a céltól és a legelsőre lő.



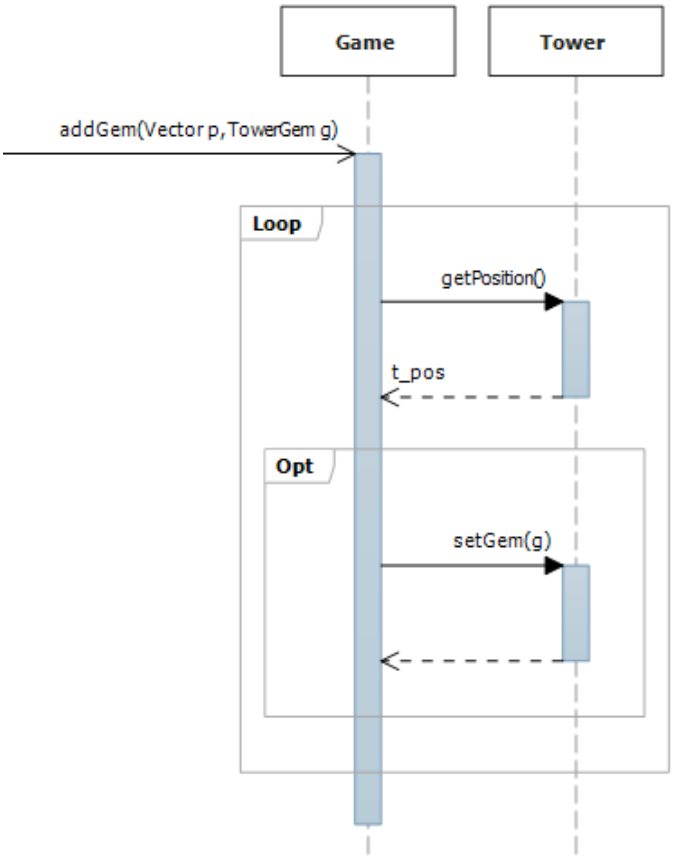
5.8. ábra. Lövedék léptetése.



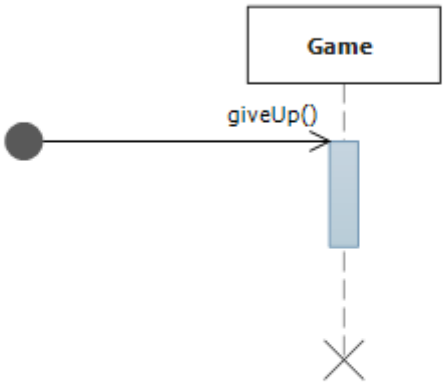
5.9. ábra. Minden akadály megnézi minden ellenségre, hogy a hatókörében van-e, ha igen lelassítja őket.



5.10. ábra. Varázskő feltétele akadályra.

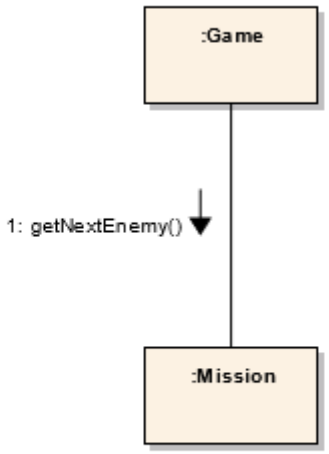


5.11. ábra. Varázskő feltétele toronyra.

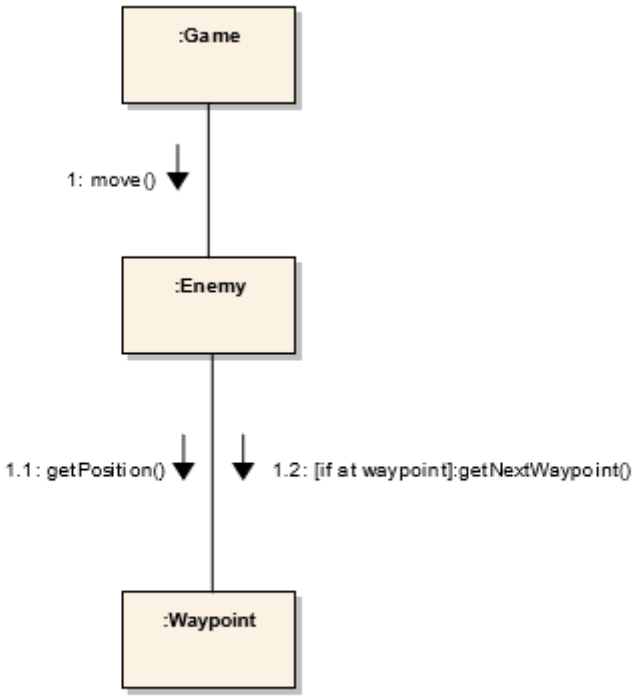


5.12. ábra. Feladás.

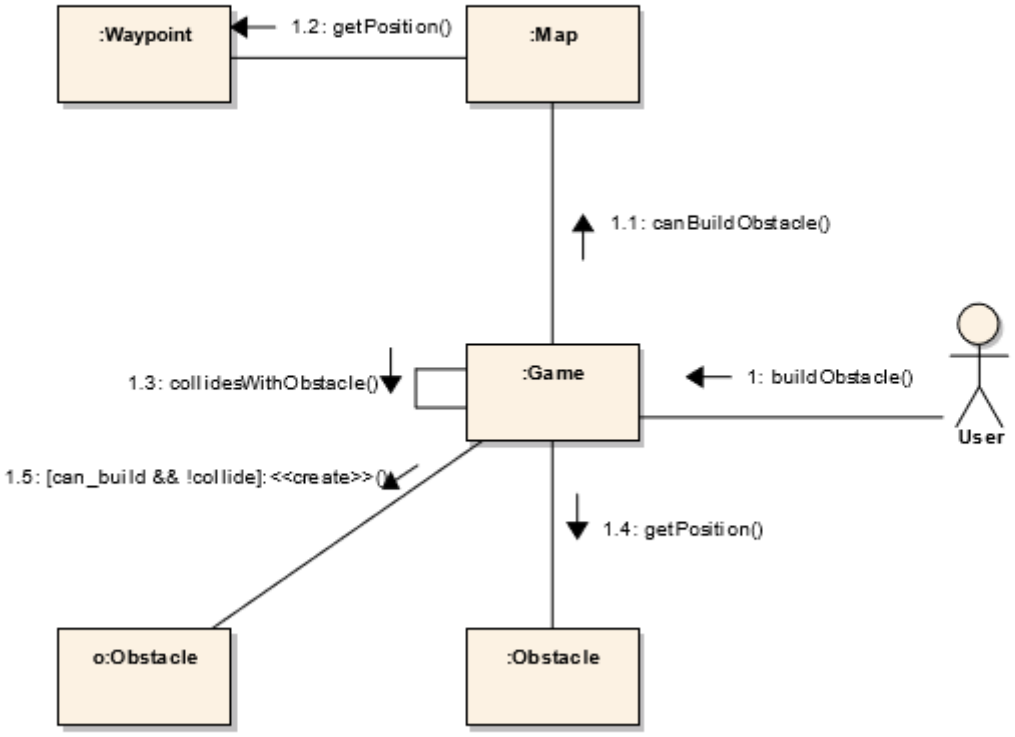
5.4. Kommunikációs diagramok



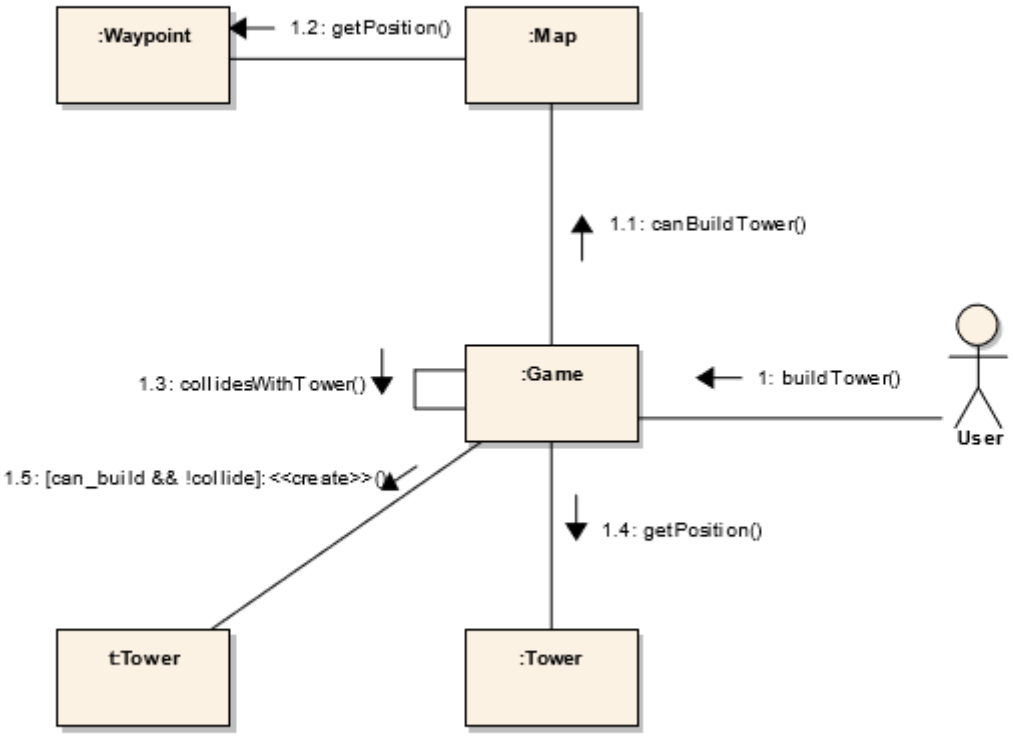
5.13. ábra. Ellenségek ütemezése.



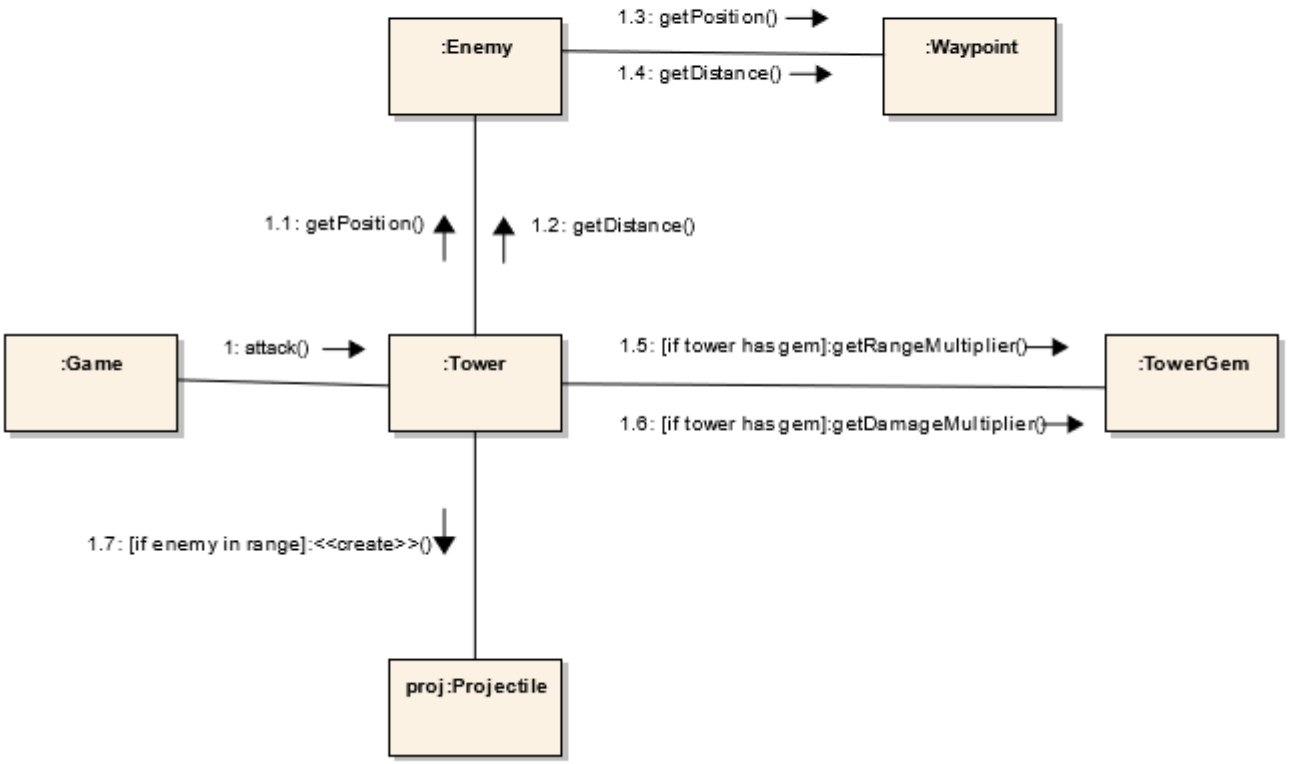
5.14. ábra. Ellenség mozgatása.



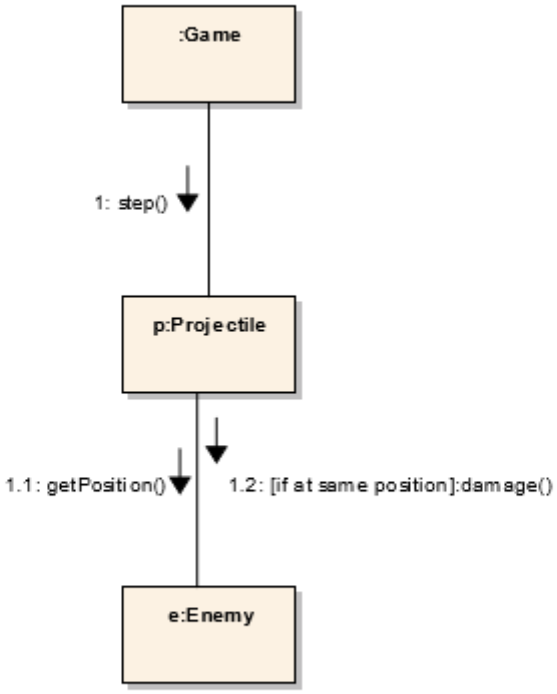
5.15. ábra. Akadály építése.



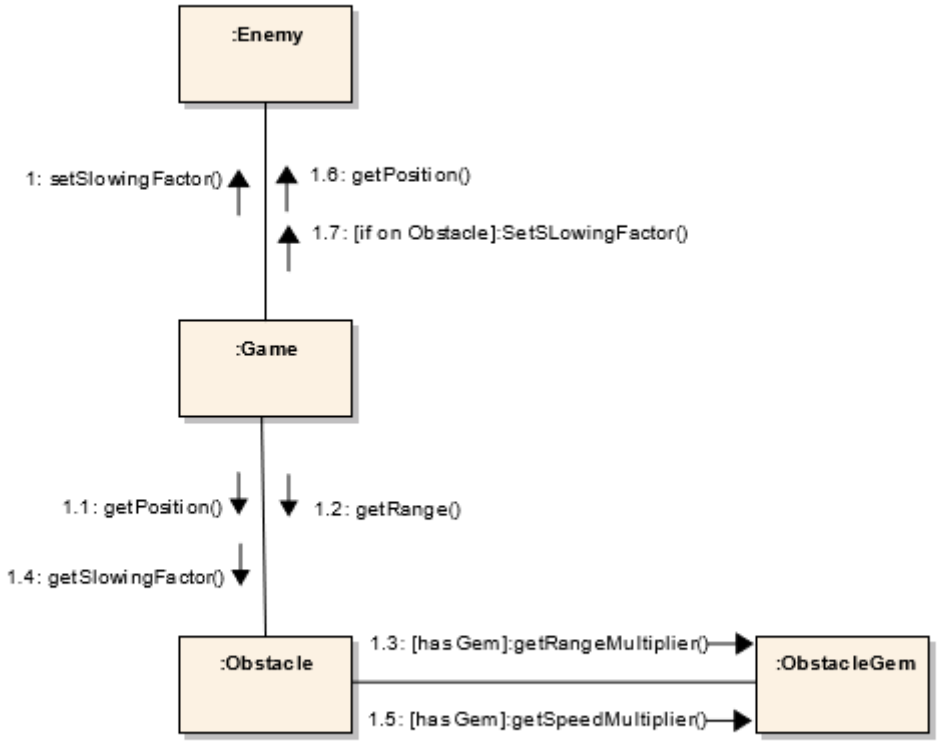
5.16. ábra. Torony építése.



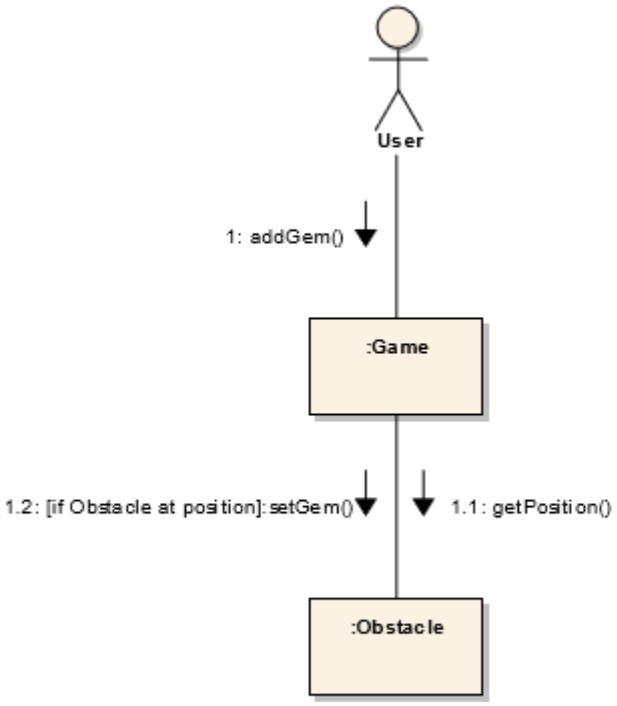
5.17. ábra. Torony tüzelése egy ellenségre.



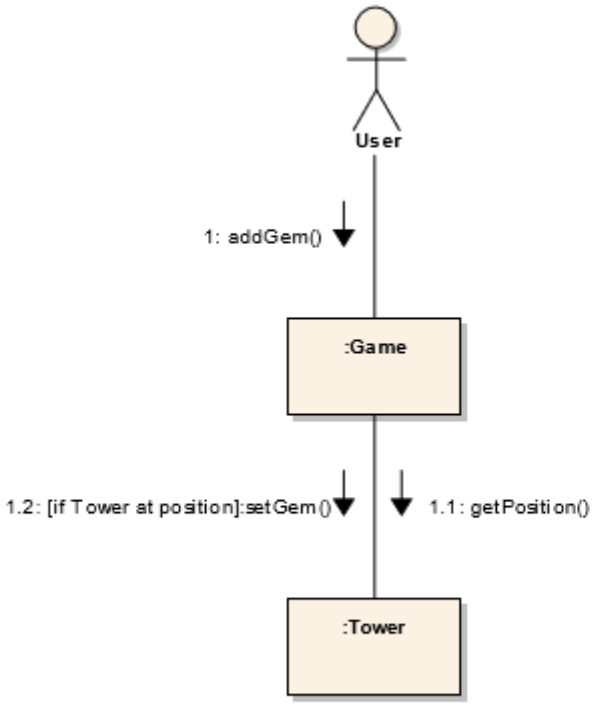
5.18. ábra. Lövedék mozgatása.



5.19. ábra. Akadályon áthaladó ellenség lassítása.



5.20. ábra. Varázskő feltétele akadályra.



5.21. ábra. Varázskő feltétele toronyra.

5.5. Napló

Kezdet	Időtartam	Résztvevők	Leírás
2014.03.16. 19:00	1 óra	**Nusser** **Szabó** **Tallér** **Török**	Értekezlet. Felhasználói kezelőfelület, use-case-ek megvitatatása. Döntés: Nusser elkészíti a kommunikációs diagramokat. Szabó elkészíti a use case diagramot. Tallér a kezelőfelület leírásokat. Török elkészíti a use case-ek szöveges leírását.
2014.03.16. 20:00	3,5 óra	**Nusser**	Tevékenység: Kommunikációs diagramok készítése.
2014.03.16. 20:00	2 óra	**Török**	Tevékenység: Török leírja a szkeleton use case-eit.
2014.03.16. 21:00	2 óra	**Szabó**	Tevékenység: Szabó elkészíti a Use-Case diagramot.
2014.03.16. 22:00	1,5 óra	**Tallér**	Tevékenység: Kezelőfelület dokumentációjának elkészítése.
2014.03.17. 08:15	0,5 óra	**Tallér**	Tevékenység: Dokumentum összeszerkesztése.

6. Szkeleton beadás

6.1. Fordítási és futtatási útmutató

6.1.1. Fájllista

Fájl neve	Méret	Keletkezés ideje	Tartalom
Enemy.java	1 967 byte	2014.03.20 10:51	Enemy osztály.
EnemyType.java	530 byte	2014.03.21 22:53	EnemyType osztály.
Game.java	9 606 byte	2014.03.20 9:25	Game osztály. Innen indul a program.
Gem.java	403 byte	2014.03.21 22:53	Gem osztály.
Map.java	926 byte	2014.03.20 10:51	Map osztály.
Mission.java	771 byte	2014.03.20 10:51	Mission osztály.
Obstacle.java	1 693 byte	2014.03.21 22:53	Obstacle osztály.
ObstacleGem.java	599 byte	2014.03.20 10:32	ObstacleGem osztály.
Projectile.java	1 165 byte	2014.03.20 10:51	Projectile osztály.
Tower.java	2 085 byte	2014.03.20 10:51	Tower osztály.
TowerGem.java	575 byte	2014.03.20 10:32	TowerGem osztály.
Vector.java	50 byte	2014.03.20 10:31	Vector segédosztály.
Waypoint.java	853 byte	2014.03.21 22:53	Waypoint osztály.

6.1.2. Fordítás

```
javac -d bin src/szoftlab4/*.java
```

6.1.3. Futtatás

```
cd bin
java szoftlab4.Game
```

6.2. Értékelés

Tag	Munka százalékban	Aláírás
Nusser	24 %	
Szabó	27 %	
Tallér	34 %	
Török	16 %	

6.3. Módosítások

6.3.1. Menü

A 7. Varázskő felrakása menüpontba raktunk egy *7.2 Érvényes helyet adtunk meg? kérdést.
Egy új menüpontot raktunk a programba
8. Ellenségek lassítása
8.1 Van varázskő az akadályon (I/N)?
8.2 Van ellenség az akadály hatókörében (I/N)?
Ezzel együtt a 8. Kilépés menüpont, 9. Kilépésre változott.

6.3.2. Game osztály

A Game.run metódus visszatérési értékét boolean-ra változtattuk. Igazzal tér vissza ha a játékos nyert/vesztett/-feladata, hamissal tér vissza ha kilép a játékból.

6.3.3. Obstacle

Az Obstacle-ből kimaradt a getRange() metódus, ami az osztálydiagramban és az osztály leírásánál nem szerepel, de a szekvencia diagramokban igen. A metódus visszaadja az akadály hatótávolságát.

6.4. Napló

Kezdet	Időtartam	Résztvevők	Leírás
2014.03.20. 9:00	2 óra	**Tallér**	Szekvencia kiíró metódusok megírása. Game osztály kódolása.
2014.03.21. 13:00	1 óra	**Szabó**	Segédmetódusok írása a Game osztályba.
2014.03.21. 21:00	2 óra	**Szabó**	Enemy, EnemyType és Projectile osztályok elkészítése.
2014.03.21. 22:00	1 óra	**Tallér**	Game osztály befejezése. Menü befejezése.
2014.03.21. 23:00	0,5 óra	**Tallér**	Kommentek hozzáadása, dokumentáció szerkesztése.
2014.03.22. 17:00	1 óra	**Török**	A Tower, Obstacle, Gem és ObstacleGem osztályok metódusainak létrehozása, dokumentálása.
2014.03.22. 19:00	1 óra	**Szabó**	Enemy, EnemyType és Projectile bővítése/javítása
2014.03.23. 10:00	2 óra	**Nusser**	Mission, Map, TowerGem és Waypoint osztályok elkészítése.
2014.03.23. 15:30	2 óra	**Szabó**	Az egész kód átnézése, komponensek illesztése, hibák javítása, hiányzó dolgok implementálása, formázás.
2014.03.23. 19:30	2 óra	**Tallér**	Hiányzó funkciók hozzáadása. Javítások. Dokumentáció elkészítése.

7. Prototípus koncepciója

7.0. Változások a specifikációban

7.0.1. Köd

A tornyokra időnként köd ereszkedik, aminek következtében a látás erősen lecsökken. Ez hatással van a lövésre.
Egy új Fog osztályt vezetünk be. Ennek van egy darab statikus getRangeMultiplier() metódusa, ami visszaadja mennyivel csökkenjen a látótáv, valamint van egy setFog(bool), amivel be lehet állítani, hogy van-e köd vagy nincs. Ha nincs köd beállítva getRangeMultiplier() 1-et ad vissza.

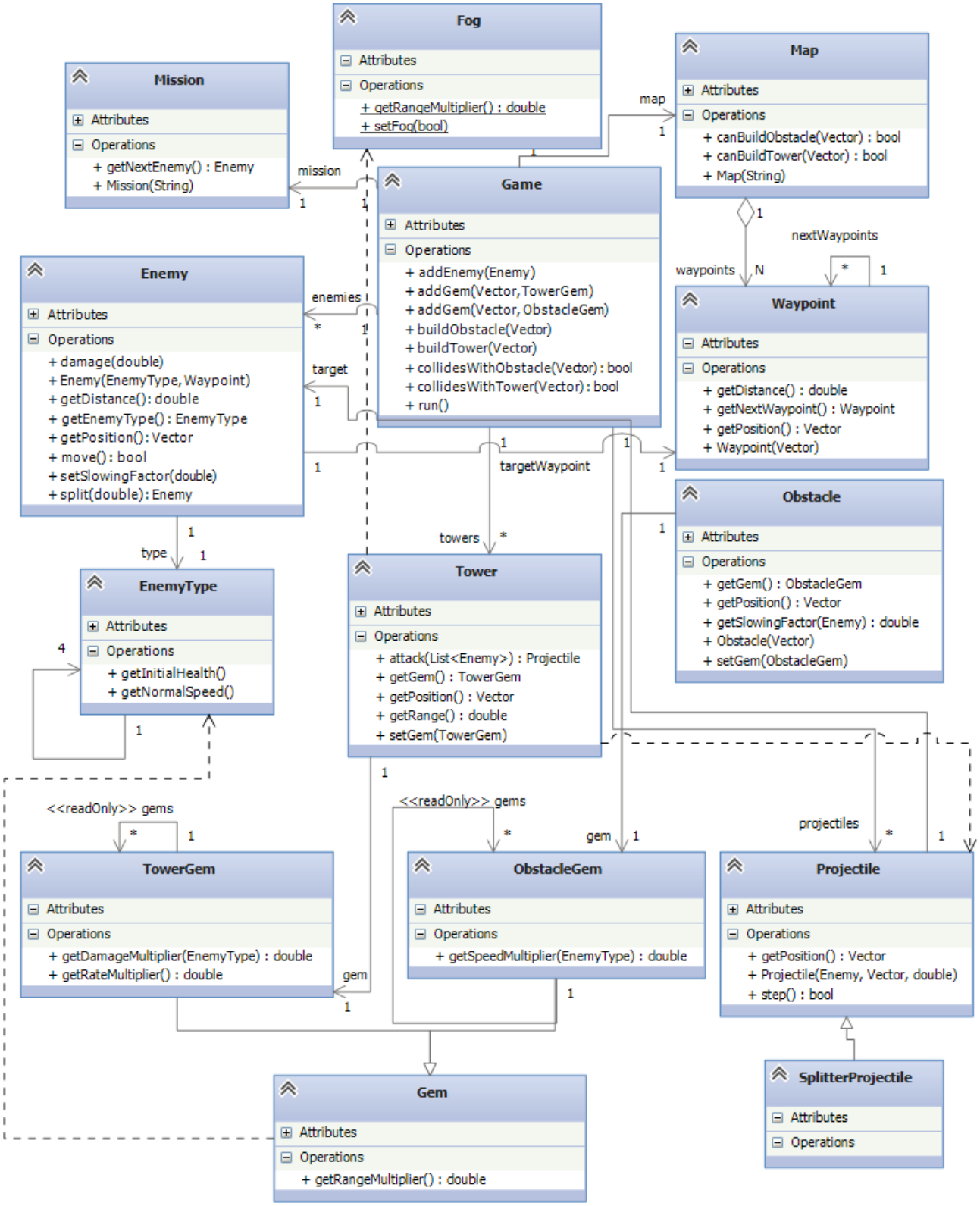
7.0.2. Elágazások

A játékosok által járható útvonalon lehetnek elágazások és becsatlakozások. Az elágazásokon az egyes játékosok véletlenszerűen mennek a különböző irányokba.
Mi a feladatot alapból így specifikáltuk, ezért változásra nincs szükség.

7.0.3. Új lövedék

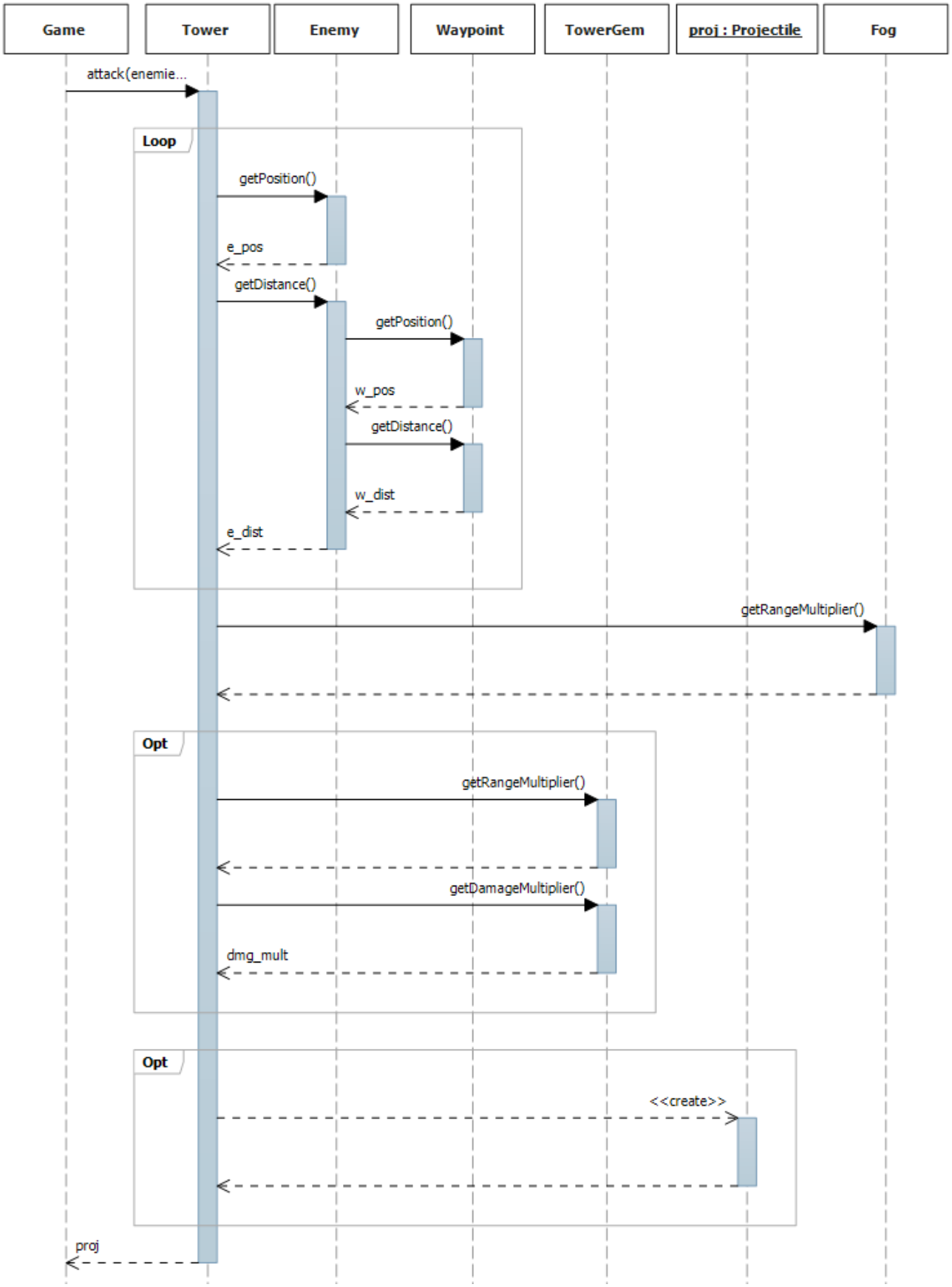
A tornyokban elvétve lehetnek olyan lövedékek, amelyek az eltalált játékost kettőbe vágják. A két játékos egymástól függetlenül él tovább, csökkentett életerővel.
Ehhez létrehoztunk egy új SplitterProjectile osztályt, ami a Projectile osztályból származik le. Valamint hozzáadtunk az Enemy osztályhoz egy split() metódust. A Game osztályhoz pedig egy addEnemy(Enemy) metódust adtunk, amit a SplitterProjectile hív meg egy callback-el, amikor kettévágta az ellenséget.

7.0.4. Módosult osztálydiagram

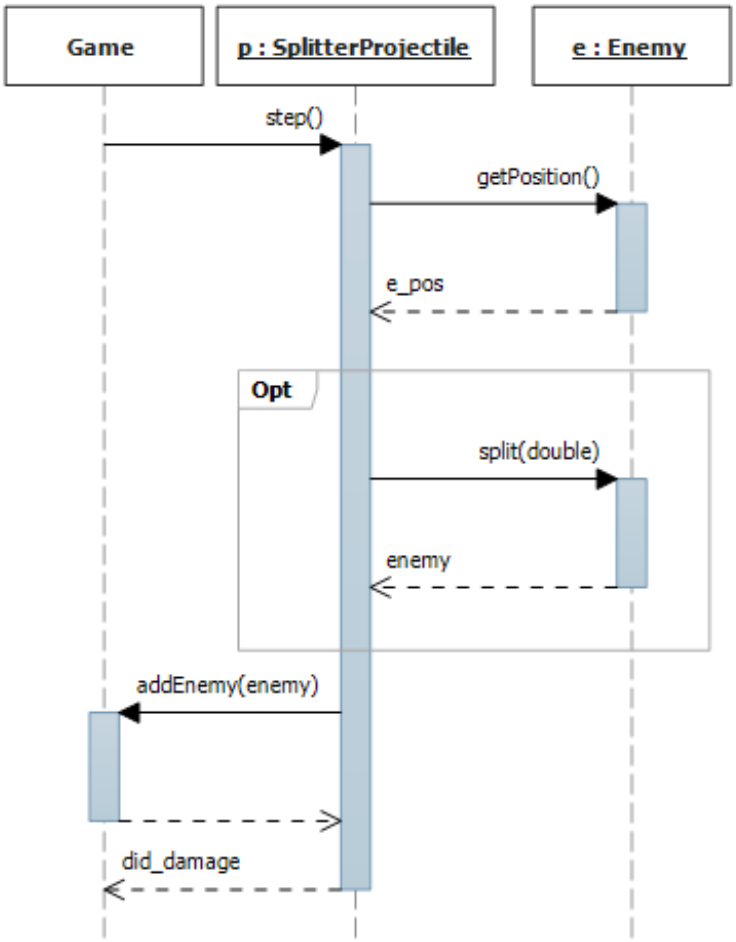


7.1. ábra. Osztály diagram

7.0.5. Módosult szekvencia diagramok



7.2. ábra. Ellenség támadása



7.3. ábra. Újfajta lövedék mozgatása

7.1. Prototípus interface-definíciója

7.1.1. Az interfész általános leírása

Az interfész csak a szabványos bemenetről fogad parancsokat, és a szabványos kimenetre írja ki az esetleges kimenetet. Ezáltal terminálból is használható, valamint az elkészítendő tesztelő segédprogram segítségével átirányítható a ki- és bemenet fájlokra, így van mód automatikus tesztelésre, előre elkészített teszteseteket felhasználva. A tesztesetek a prototípusnak adandó parancsok sorozatából, valamint az adott sorozatra adandó helyes kimenet található. A tesztek sikeresek, ha a valós, és a leírt elvárt kimenet megegyezik.

7.1.2. Bemeneti nyelv

A prototípus által elfogadott parancsok a következők:

- loadMap
  - Leírás: Egy pálya betöltése
  - Opciók: A betöltendő pálya neve
- loadMission
  - Leírás: Egy misszió betöltése
  - Opciók: A betöltendő misszió neve

- setFog
  - Leírás: A köd beállítása
  - Opciók: A köd kívánt állapota: 0 - nincs köd, 1 - van köd
- setCritical
  - Leírás: A kettévágó lövedék beállítása
  - Opciók: A kívánt hatás: 0 - nem kettévágó lövedék, 1 - kettévágó lövedék
- setWaypoint
  - Leírás: A csomópontokból való továbbhaladás irányának beállítása
  - Opciók: A következő csomópont azonosítója
- step
  - Leírás: A játéklogika léptetése
  - Opciók: A léptetni kívánt időegységek száma
- buildTower
  - Leírás: Toronyépítés
  - Opciók: Az építendő torony koordinátái
- buildObstacle
  - Leírás: Akadály építése
  - Opciók: Az építendő akadály koordinátái
- enchant
  - Leírás: Torony vagy akadály felszerelése drágakővel
  - Opciók: A drágakő típusa, illetve a felszerelendő épület koordinátái
- listEnemies
  - Leírás: A pályán lévő ellenségek kilistázása
  - Opciók: -
- listTowers
  - Leírás: A tornyok kilistázása
  - Opciók: -
- listObstacles
  - Leírás: Az akadályok kilistázása
  - Opciók: -
- listProjectiles
  - Leírás: A lövedékek kilistázása
  - Opciók: -

A pályákat és missziókat leíró fájlok XML formátumban lesznek, a következő DTD-k szerint:

```
<!DOCTYPE map
[
<!ELEMENT map (name, waypoint*, route)>
<!ELEMENT name (#PCDATA)
<!ELEMENT waypoint (id, coords)>
<!ELEMENT id (#PCDATA)
<!ELEMENT coords (x, y)>
<!ELEMENT x (#PCDATA)
<!ELEMENT y (#PCDATA)
<!ELEMENT route (a, b, chance)>
<!ELEMENT a (#PCDATA)
<!ELEMENT b (#PCDATA)
<!ELEMENT chance (#PCDATA)
]>

<!DOCTYPE mission
[
<!ELEMENT mission (name, enemy*)>
<!ELEMENT name (#PCDATA)
<!ELEMENT enemy (id, type)>
<!ELEMENT id (#PCDATA)
<!ELEMENT type (#PCDATA)
]>
```

7.1.3. Kimeneti nyelv

Kimenetet csak a következő 4 parancs ad, a következő formában:

listEnemies
Az összes élő ellenséget írja ki, soronként egyet:
<sorszám> <életerő> <koordináták>

listTowers
Az összes tornyot írja ki, soronként egyet:
<sorszám> <koordináták> <varázskő>

listObstacles
Az összes tornyot írja ki, soronként egyet:
<sorszám> <koordináták> <varázskő>

listProjectiles
Az összes tornyot írja ki, soronként egyet:
<sorszám> <koordináták> <célpont> <szétvágó-e>

7.2. Összes részletes use-case

Use-case neve	**buildObstacle**
Rövid leírás	A megadott koordinátákra épít egy akadályt.
Aktorok	Tesztelő
Forgatókönyv	A kapott koordinátákra megnézi a program, hogy épülhet-e akadály. Ha nem akkor nem épül meg és kiírja, hogy nem épült meg. Ha igen akkor bekerül az akadály listába egy új akadály a koordinátákkal.

Use-case neve	**buildTower**
Rövid leírás	A megadott koordinátákra épít egy tornyot.
Aktorok	Tesztelő
Forgatókönyv	A kapott koordinátákra megnézi a program, hogy épülhet-e torony. Ha nem akkor nem épül meg és kiírja, hogy nem épült meg. Ha igen akkor bekerül a torony listába egy új torony a koordinátákkal.

Use-case neve	**enchant**
Rövid leírás	A megadott koordinátán levő toronyra vagy akadályra egy megadott színű varázskövet tesz.
Aktorok	Tesztelő
Forgatókönyv	Ha a kapott koordinátán van egy akadály vagy torony akkor azt felruházza egy kapott színű varázskővel.

Use-case neve	**listEnemies**
Rövid leírás	Kilistázza az ellenségeket.
Aktorok	Tesztelő
Forgatókönyv	Kiírja a kimenetre a játékban levő ellenségeket.

Use-case neve	**listObstacles**
Rövid leírás	Kilistázza az akadályokat.
Aktorok	Tesztelő
Forgatókönyv	Kiírja a kimenetre a játékban levő akadályokat.

Use-case neve	**listProjectiles**
Rövid leírás	Kilistázza a lövedékeket.
Aktorok	Tesztelő
Forgatókönyv	Kiírja a kimenetre a játékban levő lövedékeket.

Use-case neve	**listTowers**
Rövid leírás	Kilistázza a tornyokat.
Aktorok	Tesztelő
Forgatókönyv	Kiírja a kimenetre a játékban levő tornyokat.

Use-case neve	**loadMap**
Rövid leírás	Betölti a kiválasztott pályát.
Aktorok	Tesztelő
Forgatókönyv	A bementeten megadott nevű pályát betölti, ha létezik olyan.

Use-case neve	**loadMission**
Rövid leírás	Betölti a kiválasztott küldetést.
Aktorok	Tesztelő
Forgatókönyv	A bementeten megadott nevű küldetést betölti, ha létezik olyan.

Use-case neve	**setCritical**
Rövid leírás	Beállítja, hogy kettévágó lövedékek keletkezzenek-e.
Aktorok	Tesztelő
Forgatókönyv	Kiveszi a véletlenszerűséget a kettévágó lövedékek keletkezéséből, és egy általunk megadott értékre állítja. (0-1)

Use-case neve	**setFog**
Rövid leírás	Beállítja, hogy legyen-e köd.
Aktorok	Tesztelő
Forgatókönyv	Kiveszi a véletlenszerűséget a köd keletkezéséből, és egy általunk megadott értékre állítja. (0-1)

Use-case neve	**setWaypoint**
Rövid leírás	Beállítja egy útelágazódásnak a következő célállomást.
Aktorok	Tesztelő
Forgatókönyv	Kiveszi a véletlenszerűséget a következő célállomás kiválasztásából, és egy általunk megadott célállomásra állítja egy ID alapján.

Use-case neve	**step**
Rövid leírás	Lépteti a programot.
Aktorok	Game osztály
Forgatókönyv	A tesztelhetőség miatt, egy időzített belső ciklus helyett ezzel lehet egyesével léptetni a program főciklusát, amibe beletartozik az ellenségek és lövedékek mozgatása, valamint a küldetés alapján, új ellenségek behozatala.

7.3. Tesztelési terv

Teszt-eset neve	**Ellenség mozgatása (nem érte el a következő Waypoint-ot)**
Rövid leírás	Ellenségek mozgásának tesztelése.
Teszt célja	Teszteli, hogy az ellenségek megfelelően mozognak-e a cél felé a pályán, ha nem kell Waypoint-ot váltaniuk.

Teszt-eset neve	**Ellenség mozgatása (elérte a következő Waypoint-ot)**
Rövid leírás	Ellenségek mozgásának tesztelése.
Teszt célja	Teszteli, hogy az ellenségek megfelelően mozognak-e a cél felé a pályán, ha Waypoint-ot kell váltaniuk.

Teszt-eset neve	**Lassítás**
Rövid leírás	Az ellenségek lassításának tesztelése.
Teszt célja	Teszteli, hogy az akadályok megfelelően lassítják-e a hatókörükben lévő ellenségeket, illetve hogy a hatókörükön kívülieket nem befolyásolják.

Teszt-eset neve	**Lassítás varázskővel ellátott akadályokkal**
Rövid leírás	Az ellenségek lassításának tesztelése.
Teszt célja	Teszteli, hogy az varázskővel ellátott akadályok helyesen működnek-e.

Teszt-eset neve	**Lövés**
Rövid leírás	A tornyok lövésének tesztelése.
Teszt célja	Teszteli, hogy a tornyok által kilőtt lövedék elér-e egy ellenséget, és megsebzi-e, illetve hogy a megfelelő időközönként lő-e a torony.

Teszt-eset neve	**Lövés ködben**
Rövid leírás	A tornyok lövésének tesztelése.
Teszt célja	Teszteli, hogy a tornyoknak kisebb lesz-e a hatótávolsága ködben.

Teszt-eset neve	**Lövés varázskővel ellátva**
Rövid leírás	A tornyok lövésének tesztelése.
Teszt célja	Teszteli, hogy a tornyoknak változnak-e a tulajdonságai varázskővel ellátva.

Teszt-eset neve	**Megfelelő célra lövés**
Rövid leírás	A tornyok célkiválasztását teszteli.
Teszt célja	Teszteli, hogy a tornyok a hatókörükben lévő ellenségek közül a célhoz legközelebbire lőnek-e.

Teszt-eset neve	**Lövedék által kettőbe vágás**
Rövid leírás	A kettévágó lövedék tesztelése.
Teszt célja	Teszteli, hogy az ellenségek két részre szakadnak-e a kettévágó lövedék hatására.

Teszt-eset neve	**Toronyépítés**
Rövid leírás	Toronyépítés tesztelése.
Teszt célja	Teszteli, hogy a torony megépül-e egy érvényes helyre és hogy nem lehet megépíteni egy érvénytelen helyre.

Teszt-eset neve	**Akadályépítés**
Rövid leírás	Akadályépítés tesztelése.
Teszt célja	Teszteli, hogy az akadály megépül-e egy érvényes helyre és hogy nem lehet megépíteni egy érvénytelen helyre.

Teszt-eset neve	**Torony varázskővel ellátása**
Rövid leírás	Torony varázskővel való ellátásának tesztelése.
Teszt célja	Teszteli, hogy a jó helyre rakott varázskő rákerül-e a toronyra, a rossz helyre rakott pedig nem.

Teszt-eset neve	**Akadály varázskővel ellátása**
Rövid leírás	Akadály varázskővel való ellátásának tesztelése.
Teszt célja	Teszteli, hogy a jó helyre rakott varázskő rákerül-e az akadályra, a rossz helyre rakott pedig nem.

7.4. Tesztelést támogató segéd- és fordítóprogramok specifikálása

A tesztelés emberi erővel hosszadalmas, és repetitív, a program működése viszont determinisztikus, így minden feltétel adott, hogy a teszt kiértékelését gép végezze el. A segédprogram teszt bemeneteket tárol, hozzájuk rendelve a várt kimenettel. A segédprogram lefuttatja a megadott tesztet (vagy többet), és összeveti a kimenetüket az elvárt kimenettel (az összehasonlítás szöveges alapon történik, leginkább a UNIX-ban található diff parancshoz lehet hasonlítani). Amennyiben a kimenet nem egyezik meg az elvárt kimenettel, úgy a teszt nem sikerült (fail), ellenkező esetben a teszt sikeres (pass). Hiba esetén a program kijelzi, hogy hol, és milyen eltérést talált.

7.5. Napló

Kezdet	**Időtartam**	**Résztvevők**	**Leírás**
2014.03.30. 20:30	2,5 óra	**Nusser**	Use-case leírások elkészítése.
2010.03.30. 22:00	3 óra	**Török**	Tevékenység: Török elkészíti a dokumentáció 7.1-es fejezetét.
2014.03.30. 22:00	2 óra	**Szabó**	Tesztesetek elkészítése.
2014.03.30. 22:00	1 óra	**Tallér**	Változások leírása. Változott diagramok elkészítése.

8. Részletes tervek

8.1. Osztályok és metódusok tervei

8.1.1. Enemy

- Felelősség
  Az ellenségek pozíciójának, sebességének és életerejének nyilvántartása.
- Attribútumok
  - - num: int: Az osszes ellenség számát tartja nyilván.
  - - type: **EnemyType**: Az egység típusa.
  - - health: Double: Az ellenség fennmaradó életereje.
  - - position: Vector: Pillanatnyi pozíció a pályán.
  - - targetWaypoint: **Waypoint**: Az a **Waypoint**, ami felé az ellenség jelenleg tart.
  - - nextWaypoint: **Waypoint**: A következő **Waypoint** ami felé tartani fog.
  - - slowingFactor: Double: Ha az ellenség belelép egy akadályba, akkor beállítja ennek az értékét. Az **EnemyType** sebességét ezzel kell beszorozni, hogy megkapjuk az ellenség tényleges jelenlegi sebességét.
  - ID: int: Identifikálja az egyes ellenségeket.
- Metódusok
  - + Enemy(et: EnemyType, start: Waypoint): Létrehoz egy **EnemyType** típusú **Enemy**-t a start **Waypoint**-helyén.
  - + Enemy(type: EnemyType, start: Waypoint, ID: int): Megadott ID-val hoz létre ellenséget.
  - + Enemy(en: Enemy): Copy konstruktor.
  - + move(): boolean: Az ellenséget a sebességének megfelelő mértékben mozgatja a célja irányába. Ha az ellenség életereje 0 vagy kisebb, akkor igazzal tér visza, egyébként hamissal.
  - + damage(amount: Double): boolean: Csökkenti az ellenség életerejét amount-al, igazzal tér vissza ha az ellenség meghalt.
  - + getPosition(): Vector: A position attribútum értékével tér vissza.
  - + getEnemyType(): **EnemyType**: Visszaadja az ellenség típusát.
  - + getID(): int: Visszaadja az ellenség ID-ját.
  - + getDistance(): Double: Visszaadja az ellenség céltól való távolságát a legrövidebb úton haladva.
  - + setSlowingFactor(sf: Double): void: Beállítja a slowingFactor-t sf-re.
  - + setNextWaypoint(w: **Waypoint**): void: Beállítja a következő **Waypoint**-ot.
  - + split(d: Double): Enemy: Megsebzi az ellenséget d-vel, majd klónozza az objektumot és ezzel tér vissza.

8.1.2. EnemyType

- Felelősség
  Leírja egy bizonyos típusú (fajú) ellenség alapvető tulajdonságait. Egy-egy példányára hivatkozik az összes ellenség, amelyeknek ezáltal meghatározza a viselkedését. Az osztályból nem lehet példányosítani, csakis a statikus tagként szereplő objektumokat lehet felhasználni.

- Attribútumok
  - - magic: int: Az ilyen fajtájú ellenségek ára varázserőben.
  - - initialHealth: Double: Az ilyen fajtájú ellenségek kezdeti életereje.
  - - normalSpeed: Double: Az ilyen fajtájú ellenségek akadályoztatás nélküli haladási sebessége.
  - + EnemyType dwarf: Csak olvasható törp típus.
  - + EnemyType elf: Csak olvasható tünde típus.
  - + EnemyType hobbit: Csak olvasható hobbit típus.
  - + EnemyType human: Csak olvasható ember típus.
- Metódusok
  - + getHealth(): Double: Visszaadja az initialHealth attribútum értékét.
  - + getSpeed(): Double: Visszaadja a normalSpeed attribútum értékét.

8.1.3. Fog

- Felelősség
  Ez az osztály felelős a ködért, ami a 7.0 fejezetben lett specifikálva. Az osztály statikus metódusokkal biztosítja a köd működését és ki- és bekapcsolását.
- Attribútumok
  - - isSet: boolean: Ez tárolja, hogy be van-e kapcsolva a játékban a köd.
- Metódusok
  - + getRangeMultiplier(): Double: Ha be van kapcsolva a köd akkor egy <1 számmal tér vissza, amivel csökkenti a tornyok látótávát, ha nincs bekapcsolva akkor 1-el tér vissza.
  - + setFog(b: boolean): void: b paraméter értékére állítja az isSet attribútumot.

8.1.4. Game

- Felelősség
  A többi osztály nyilvántartása és összekötése, a játékbeli események vezérlése. A felhasználói felülettől érkező parancsok végrehajtása, és a játék állapotának rendelkezésre bocsájtása a kijelzéshez.
- Attribútumok
  - FPS: int: Másodpercenként hányszor fut le a játék főciklusa.
  - - map: **Map**: Referencia a kiválasztott pályára, amin a játék folyik.
  - - mission: **Mission**: Referencia a kiválasztott misszióra, amely alapján zajlik a játék.
  - - enemies: List<**Enemy**>: Az összes jelenleg élő ellenség található meg benne.
  - - projectiles: List<**Projectile**>: Eltárolja a jelenleg játékban lévő lövedékeket.
  - - towers: List<**Tower**>: Eltárolja a játékos megépített tornyait.
  - - obstacles: List<**Obstacle**>: Eltárolja a játékos megépített akadályait.
  - - magic: int: A játékos jelenlegi varázsereje.
- Metódusok
  - + run(): boolean: Ez a metódus futtatja a főciklust, amelyben maga a játék működik. Ez a metódus hívja meg az ellenségek, lövedékek léptető metódusait. Meghívja a tornyok attack metódusát az ellenségek listájával. Az ellenségeknek beállítja a lassítást, ha akadályba léptek.

  - + buildTower(position: Vector): void: Épít egy tornyot a paraméterül kapott helyen lévő mezőre, ha a pozíció a pályán belül, nem úton van nem ütközik másik toronnyal.
  - + buildObstacle(position: Vector): void: Épít egy akadályt a paraméterül kapott helyen lévő mezőre, ha a megadott pozíció úton, pályán belül van és nem ütközik másik akadállyal..
  - + addGem(position: Vector, gem: **TowerGem**): void: A paraméterként kapott helyen lévő toronyra rárakja a paraméterként kapott varázskövet.
  - + addGem(position: Vector, gem: **ObstacleGem**): void: A paraméterként kapott helyen lévő akadályra rárakja a paraméterként kapott varázskövet.
  - + addEnemy(en: Enemy): void: Hozzáadja a paraméterként kapott ellenséget az enemies listába.
  - + collidesWithTower(p: Vector): boolean: megadja, hogy ha p helyre építenénk tornyot, az belelógna-e egy már megépített toronyba.
  - +getCollidingTower(pos: Vector): Tower: Visszatér a pos helyén levő toronnyal.
  - + collidesWithObstacle(p: Vector): boolean: megadja, hogy ha p helyre építenénk akadályt, az belelógna-e egy már megépített akadályba.
  - + getCollidingObsacle(pos: Vector): Obstacle: Visszatér a pos helyén levő akadállyal.
  - + getMagic(): int: Visszatér a magic-el.
  - - getEnemyById(enemyID: int): Enemy: ID alapján visszatér az ellenséggel.
  - + giveup(): void: A játék feladása.
  - + slowEnemies(): void: Lelassítja az ellenségeket.
  - - step(): boolean: A játék logikáját egy lépéssel előrébb viszi. Igazzal tér vissza ha sikeres volt.

8.1.5. Gem

- Felelősség
  Egy általános varázskő tulajdonságainak tárolása. Absztrakt osztály.
- Attribútumok
  - - cost: int: A varázskő ára varázserőben.
  - - rangeMultiplier: Double: Megadja, hogy a varázskővel ellátott toronynak hányszorosára nő a hatótávolsága.
- Metódusok
  - + getRangeMultiplier(): Double: Visszaadja a varázskő hatótávolság szorzóját.

8.1.6. Projectile

- Felelősség
  Követni a cél ellenséget, majd sebezni ha eléri.
- Attribútumok
  - - damage: Double: A lövedék sebzése, ennyivel csökkenti a cél ellenség életerejét amikor eltalálja.
  - - position: Vector: A lövedék pozíciója.
  - - speed: Double: A lövedék sebessége.
  - - target: **Enemy**: A lövedék cél **Enemy**-je.
- Metódusok

  - + Projectile(**Enemy** enemy, Vector position, double speed): Konstruktor, átveszi a cél **Enemy**-t, a kezdő pozíciót és sebességet.
  - + step(): boolean: speed-el mozgatja a lövedéket az ellenség irányába. Ha eltalálta az ellenséget vagy az ellenség már meghalt, akkor true-t ad vissza, különben false-t.
  - + getPosition(): Vector: Visszaadja a lövedék pozícióját.

8.1.7. SplitterProjectile

- Felelősség
  Követni a cél ellenséget, majd kettévágni ha eléri.
- Ősosztályok
  Projectile
- Attribútumok
  - - game: Game: Egy referencia a játék objektumra. Erre a Game.addEnemy callback miatt van szükség.
- Metódusok
  - + Projectile(**Enemy** enemy, Vector position, double speed, Game game): Konstruktor, átveszi a cél **Enemy**-t, a kezdő pozíciót, sebességet és egy referenciát a játékra.

8.1.8. Waypoint

- Felelősség
  Útvonalat kijelölni az ellenségeknek, úgy, hogy megadja a pozícióját, amely felé az ellenségek mehetnek, valamint a következő **Waypoint**-ot ami felé menniük kell, ha egyszer elérték ezt a **Waypoint**-ot.
- Attribútumok
  - - position: Vector: A **Waypoint** pozíciója a pályán.
  - - distance: double: A **Waypoint** távolságát a céltól tárolja.
  - ID: int: Identifikálja az egyes **Waypoint**-okat.
  - - nextWaypoints: List <Waypoint, double>: A következő **Waypoint**-okat és a hozzájuk tartozó valószínűségeket
- Metódusok
  - + Waypoint(pos: Vector, ID: int): Konstruktor, mely adott helyen adott ID-val létrehoz egy **Waypoint**-ot.
  - + getDistance():double: Visszaadja a distance attribútumot.
  - + getNextWaypoint(): **Waypoint**: Visszatér a nextWaypoints listából véletlenszerűen kiválasztott **Waypoint**-al
  - + getPosition(): Vector: visszatér a position attribútummal.
  - + getID(): int: Visszatér az ID-val.
  - + listNextWaypoints(): List<**Waypoint**>: A nextWaypoints-ot listaként adja vissza.
  - + setDistance(): double: Rekurzívan bejárja a pályát, és beállítja a céltól való távolságukat.
  - + setID(ID: int): void: Beállítja az ID-t a kapott értékre.
  - + setNextWaypoint(wp: **Waypoint**, d: double): void: Hozzáadja a nextWaypointokhoz a wp-t d valószinűséggel.

8.1.9. Map

- Felelősség
  Betölt egy XML pályaleíró fájlt, és ez alapján felépíti a pályát Waypointok-ból; elérhetővé teszi a Waypoint-okat id alapján; ellenőrizni tudja, hogy egy adott helyre építhető-e torony vagy akadály.
- Attribútumok
  - - roadRadius: double: Az utak átmérője.
  - - waypoints: HashMap<Integer, Waypoint>: A Waypoint-okat tárolja id alapján.
- Metódusok
  - Map(file: String): Létrehoz egy Map osztályt a paraméterként megadott pályaleíró fájlból.
  - IsInRoadRange(pos: Vector, range: double): boolean: Visszatér azzal, hogy a pos úton van-e.
  - getWaypointByID(id: int): Waypoint: Visszaadja az id-hez tartozó Waypoint-ot.
  - canBuildObstacle(pos: Vector): boolean: Visszaadja, hogy a pos helyre építhető-e akadály.
  - canBuildTower(pos: Vector): boolean: Visszaadja, hogy a pos helyre építhető-e torony.

8.1.10. Mission

- Felelősség
  Betölt egy XML küldetésleíró fájlt, és létrehoz az összes ellenséghez egy Enemy objektumot, amiket eltárol, hogy később le lehessen kérni tőle a következő ellenséget.
- Attribútumok
  - -spawnList: List<Spawn>: Az Enemy-ket és a hozzájuk tartozó spawn időket tárolja.
  - name: String: A misszió neve.
- Metódusok
  - Mission(file: String, map: Map): Létrehoz egy Mission osztályt a paraméterként megadott küldetésleíró fájlból.
  - getNextEnemy(): Enemy: Visszaadja a következő ellenséget amit el kell indítani a pályán, vagy null.

8.1.11. Tower

- Felelősség
  Felelős **Projectile**-ok létrehozásához, azok megfelelő felparaméterezésével. Továbbá felelős azért, hogy **Projectile**-okat csak a megadott időközönként lőjjön ki.
- Attribútumok
  - cost: int: A torony ára varázserőben.
  - range: double: A távolság amire tud lőni.
  - fireRate: double: A lövési gyakoriság.
  - - cooldown: double: Mennyi idő van még a következő lövésig.

  - - critical: boolean: Kritikusat sebez-e.
  - - gem: **TowerGem**: Eltárol egy referenciát egy **Gem** típusú objektumra, ami meghatározza, hogy az adott épület milyen echant alatt áll.
  - - damage: **HashMap<EnemyType**, double>: Megadja mekkora az adott típusú ellenfélre kifejtett hatása a toronynak.
  - - position: Vector: Visszatér az épület koordinátáival.

- Metódusok
  - + attack(List **<Enemy>**): **Projectile**: Először megnézi, hogy lőhet-e, ha nem akkor semmivel se tér vissza. Ha igen akkor végignézi a kapott listában az ellenségeket, és amelyik a hatótávolságán belül van, és a legközelebb a célhoz, arra kilő egy **Projectile**-t, majd a visszatérési értékében visszaadja azt. A **Projectile**-t felparaméterezi az ellenséghez megfelelő sebzési adatokkal.
  - + doesCollide(pos: Vector): Visszatér azzal, hogy a Vector ütközik-e a toronnyal.
  - + getCost(): int: Visszatér a cost attribútummal.
  - + getGem(): **Gem**: Visszaadja az épületen található varázskövet.
  - + setGem(**TowerGem** gem): void: Beállítja az épületen lévő varázskövet. Ha már volt az épületen varázskő, akkor az előző megszűnik.
  - + getPosition(): Vector: Visszaadja a position attribútumot.
  - + getRange(): double: Visszatér a range-el.

8.1.12. Obstacle

- Felelősség
  Felelős, az ellenfelek lassításáért, úgy, hogy meg kell tudnia mondani a pozícióját, valamint, hogy az adott ellenséget mennyire lassítja.

- Attribútumok
  - cost: int: Az akadály ára varázserőben.
  - - gem: **ObstacleGem**: Eltárol egy referenciát egy **Gem** típusú objektumra, ami meghatározza, hogy az adott épület milyen echant alatt áll.
  - - slowingFactor: **Map<EnemyType**, double>: Megadja mekkora az adott típusú ellenfélre kifejtett hatása az akadálynak.
  - - position: **Vector**: Az akadály koordinátáit tárolja.
  - - range: double: Az akadály hatótávolsága.

- Metódusok
  - + Obstacle(pos: Vector): Létrehoz egy **Obstacle** objektumot, a pozícióját pos-ra állítva.
  - + doesCollide(pos: Vector): boolean: Visszatér azzal, hogy ütközik-e a pos az akadállyal.
  - + getCost(): int: Visszatér a torony árával.
  - + getSlowingFactor(**Enemy** enemy): double: Visszatér azzal az értékkel, amivel az adott ellenfelet lassítja.
  - + getGem(): **Gem**: Visszaadja az épületen található varázskövet.
  - + setGem(**Gem** gem): void: Beállítja az épületen lévő varázskövet. Ha már volt az épületen varázskő, akkor az előző megszűnik.
  - + getPosition(): Vector: Visszaadja a position attribútumot.
  - + getRange(): double: Visszaadja az akadály hatótávolságát.

8.1.13. ObstacleGem

- Felelősség
  Egy akadályra rakható varázskő tulajdonságainak tárolása.
- Ősosztályok
  Gem
- Attribútumok
  - yellow: ObstacleGem: Sárga ObstacleGem.
  - orange: ObstacleGem: Narancssárga ObstacleGem.
  - - speed: HashMap<**EnemyType**, Double>: Megadja, hogy a varázskővel elátott akadályon áthaladó adott típusú ellenség sebessége hányadára csökken.
- Metódusok
  - + getSpeedMultiplier(**EnemyType** enemyType): double: Visszaadja varázskő sebesség szorzóját egy adott típusú ellenséghez.

8.1.14. TowerGem

- Felelősség
  Egy toronyra rakható varázskő tulajdonságainak tárolása.
- Ősosztályok
  Gem
- Attribútumok
  - red: TowerGem: Piros TowerGem.
  - green: TowerGem: Zöld TowerGem.
  - blue: TowerGem: Kék TowerGem.
  - - rate: double: Megadja, hogy a varázskővel ellátott toronynak hányszorosára nő a tüzelési sebessége.
  - - damage: HashMap<**EnemyType**, double>: Megadja, hogy a varázskővel ellátott toronynak hányszorosára nő a sebzése egy adott típusú ellenséggel szemben.
- Metódusok
  - + getRateMultiplier(): double: Visszaadja a varázskő tüzelési sebesség szorzójáz.
  - + getDamageMultiplier(**EnemyType** enemyType): double: Visszaadja varázskő sebzés szorzóját egy adott típusú ellenséghez.

8.2. A tesztek részletes tervei, leírásuk a teszt nyelvén

8.2.1. Alapvető működés

- Leírás
  Ennek a tesztnek az alapegységek működésének tesztelése a célja.
- Ellenőrzött funkcionalitás, várható hibahelyek
  Egy pálya és egy misszió betöltése, egy torony és egy akadály építése, egy időegység léptetése, valamint a tornyok, lövedékek, és ellenségek listázása, ezután pedig az épületek megerősítése, és az ellenség útválasztása kerül ellenőrzésre. Lehetséges hibák: A torony vagy az akadály nem épül meg. Az ellenség

nem követi a számára kijelölt utat. Az épületek erősítése nem történik meg megfelelően. A torony nem sebzi a mellette elhaladó ellenséget, vagy létre sem jön lövedék.

- Bemenet
  loadMap basic_test_map
  loadMission basic_test_mission
  buildTower 2 5
  buildObstacle 10 4
  step 1
  listTowers
  listObstacles
  listProjectiles
  listEnemies
  enchant 1 2 5
  enchant 2 10 4
  setWaypoint 3
  step 100
  listTowers
  listObstacles
  listEnemies

- Elvárt kimenet
  1 (2;5) -
  1 (10;4) -
  1 (2;5) 1 false
  1 100 (0;4)
  1 (2;5) 1
  1 (10;4) 2
  1 60 (20;6)

8.2.2. Köd ellenőrzése

- Leírás
  Ennek a tesztnek a köd működésének tesztelése a célja.

- Ellenőrzött funkcionalitás, várható hibahelyek
  A ködben létrejövő látótávolság-csökkenés hatását vizsgálja. Lehetséges hiba, hogy a köd nem csökkenti megfelelő mértékben a látótávolságot, így az az ellenség is sebződik, amelyik túl messze van a toronytól.

- Bemenet
  loadMap fog_test_map
  loadMission fog_test_mission
  buildTower 2 5
  setFog 1
  step 100
  listEnemies

- Elvárt kimenet
  1 100 (20;10)

8.2.3. Erős lövés ellenőrzése

- Leírás
  Ennek a tesztnek az erős lövedék hatásának tesztelése a célja.
- Ellenőrzött funkcionalitás, várható hibahelyek
  A tornyokban elvétve előforduló erős lövés hatását teszteli. Lehetséges hiba, hogy az ellenség nem osztódik ketté.
- Bemenet
  loadMap fog_test_map
  loadMission fog_test_mission
  buildTower 2 5
  setCritical 1 step 100
  listEnemies
- Elvárt kimenet
  1 40 (20;10)
  2 40 (20;10)

8.3. A tesztelést támogató programok tervei

A tesztelést végző segédprogram a specifikált formátumban megírt tesztfájlokat olvassa be, elindítja az alkalmazást, megadja neki az előírt bemenetet, majd a kapott kimenetet összehasonlítja a megaditt elvárt kimenettel, és amelyik sorban eltérést észlel, ott jelzi a különbséget. Ha nincs eltérés, a teszt sikeresen lefutott.

8.3.1. attack_damage_gem

Létrehoz egy tornyot olyan távolságban az úttól, hogy ne legyen a hatósugarában, de egy piros varázskővel már elérje. Majd rárak egy piros varázskövet a toronyra.
Bemenet:
loadMap test.map
loadMission attack.mission
buildTower 7 1
enchant red 7 1
listTowers
step 34
listEnemies
step 1
listEnemies

exit

Elvárt kimenet:
(7.0;1.0) red
1 75.0 (8.6;8.6)
1 45.0 (8.8;8.8)

8.3.2. attack_one

A teszt misszióban egy ellenség van. A teszt épít egy tornyot, majd megfelelő mennyiségű lépés után a torony elindít egy projectile-t az ellenség felé.
Bemenet:
loadMap test.map
loadMission attack.mission
buildTower 7 1
listTowers
step 30
listEnemies
listProjectiles
step 1
listEnemies
listProjectiles
step 1
listEnemies
listProjectiles
step 1
listEnemies
listProjectiles
step 1
listEnemies
listProjectiles
step 1
listEnemies
listProjectiles

exit

Elvárt kimenet:

(7.0;1.0) -
1 75.0 (7.5;7.5)
1 75.0 (7.8;7.8)
1 75.0 (8.0;8.0)
(7.0;1.0) 1 false
1 75.0 (8.3;8.3)
(7.6;4.3) 1 false
1 75.0 (8.6;8.6)
(8.3;7.5) 1 false
1 55.0 (8.8;8.8)

8.3.3. buildobstacle

Épít egy akadályt útra, majd kilistázza.
Bemenet:
loadMap test.map
loadMission test.mission
buildObstacle 5 5

listObstacles
exit
Elvárt kimenet:
(5.0;5.0) -

8.3.4. buildobstacle_wrong

Azt teszteli, hogy akadályt nem lehet nem útra rakni.
Bemenet:
loadMap test.map
loadMission test.mission
buildObstacle 5 20
listObstacles
exit
Elvárt kimenet:

8.3.5. buildtower

Torony építését teszteli.
Bemenet:
loadMap test.map
loadMission test.mission
buildTower 5 20
listTowers
exit

Elvárt kimenet:
(5.0;20.0) -

8.3.6. buildtower_wrong

Azt teszteli, hogy tornyot nem lehet útra rakni.
Bemenet:
loadMap test.map
loadMission test.mission
buildTower 5 5
listTowers

exit
Elvárt kimenet:

8.3.7. critical

Szétvágó lövedéket teszteli.
Bemenet:
loadMap test.map
loadMission attack.mission

setCritical 1
buildTower 7 1
listTowers
step 30
listEnemies
listProjectiles
step 1
listEnemies
listProjectiles
step 1
listEnemies
listProjectiles
step 1
listEnemies
listProjectiles
step 1
listEnemies
listProjectiles
step 1
listEnemies
listProjectiles

exit
Elvárt kimenet:
(7.0;1.0) -
1 75.0 (7.5;7.5)
1 75.0 (7.8;7.8)
1 75.0 (8.0;8.0)
(7.0;1.0) 1 true
1 75.0 (8.3;8.3)
(7.6;4.3) 1 true
1 75.0 (8.6;8.6)
(8.3;7.5) 1 true
1 55.0 (8.8;8.8)
2 55.0 (8.8;8.8)

8.3.8. elagazodas_balra

Elágazás tesztelése. Elágazásnál, ahol két irányba mehet balra megy (igazából inkább lefelé).
Bemenet:
loadMap elagazodas.map
loadMission one_enemy.mission
step 1
setWaypoint 0 3
step 70
listEnemies
exit

Elvárt kimenet:
0 100.0 (11.1;18.5)

8.3.9. elagazodas_jobbra

Elágazás tesztelése. Elágazásnál, ahol két irányba mehet jobbra megy.
Bemenet:
loadMap elagazodas.map
loadMission one_enemy.mission
step 1
setWaypoint 0 4
step 70
listEnemies
exit

Elvárt kimenet:
0 100.0 (18.5;11.1)

8.3.10. enemy_win

Az ellenség nyerését teszteli.
Bemenet:
loadMap test.map
loadMission test.mission
step 100
listEnemies
exit

Elvárt kimenet:
Enemy winz

8.3.11. fog

Ködöt teszteli. A teszt program olyan tornyot épít, ami rálát egy útra, de a köd bekapcsolásával már nem.
Bemenet:
loadMap test.map
loadMission attack.mission
setFog 1
buildTower 7 1
listTowers
step 30
listEnemies
listProjectiles
step 1
listEnemies
listProjectiles
step 1
listEnemies
listProjectiles
step 1

listEnemies
listProjectiles
step 1
listEnemies
listProjectiles
step 1
listEnemies
listProjectiles

exit

Elvárt kimenet:
(7.0;1.0) -
1 75.0 (7.5;7.5)
1 75.0 (7.8;7.8)
1 75.0 (8.0;8.0)
1 75.0 (8.3;8.3)
1 75.0 (8.6;8.6)
1 75.0 (8.8;8.8)

8.3.12. loadmap

Pályabetöltés tesztelése.
Bemenet:
loadMap test.map
loadMission test.mission
exit

Elvárt kimenet:

8.3.13. no_obstacle

obstacle teszesettel való összehasonlításra. A két teszt ugyanazt végzi, csak az obstacle-ben van egy akadály.
Bemenet:
loadMap test.map
loadMission attack.mission
listTowers
step 15
listEnemies
listProjectiles
step 5
listEnemies
listProjectiles
step 5
listEnemies
listProjectiles
step 5
listEnemies

listProjectiles
step 5
listEnemies
listProjectiles
step 5
listEnemies
listProjectiles
step 5
listEnemies
listProjectiles
step 5
listEnemies
listProjectiles

exit

Elvárt kimenet:
1 75.0 (3.6;3.6)
1 75.0 (4.9;4.9)
1 75.0 (6.2;6.2)
1 75.0 (7.5;7.5)
1 75.0 (8.8;8.8)
1 75.0 (10.1;10.1)
1 75.0 (11.4;11.4)
1 75.0 (12.7;12.7)

8.3.14. obstacle

Akadály tesztelése. A pályára épített akadály lassítja az ellenséget.
Bemenet:
loadMap test.map
loadMission attack.mission
buildObstacle 10 10
listTowers
step 15
listEnemies
listProjectiles
step 5
listEnemies
listProjectiles
step 5
listEnemies
listProjectiles
step 5
listEnemies
listProjectiles
step 5
listEnemies
listProjectiles
step 5

listEnemies
listProjectiles
step 5
listEnemies
listProjectiles
step 5
listEnemies
listProjectiles

exit

Elvárt kimenet:
1 75.0 (3.6;3.6)
1 75.0 (4.9;4.9)
1 75.0 (6.2;6.2)
1 75.0 (7.0;7.0)
1 75.0 (7.6;7.6)
1 75.0 (8.3;8.3)
1 75.0 (8.9;8.9)
1 75.0 (9.6;9.6)

8.3.15. one_enemy_move

Egy ellenség mozgatásánál a tesztelése.
Bemenet:
loadMap test.map
loadMission one_enemy.mission
step 31
listEnemies
exit

Elvárt kimenet:
0 100.0 (7.1;7.1)

8.3.16. elagazodas.map tartalma

```
<map>
    <name>Test_Multi_Waypoint</name>
    <waypoint>
        <id>1</id>
        <coords>
            <x>0</x>
            <y>0</y>
        </coords>
    </waypoint>
    <waypoint>
        <id>2</id>
        <coords>
```

```
            <x>10</x>
            <y>10</y>
        </coords>
    </waypoint>
    <waypoint>
        <id>3</id>
        <coords>
            <x>10</x>
            <y>20</y>
        </coords>
    </waypoint>
    <waypoint>
        <id>4</id>
        <coords>
            <x>20</x>
            <y>10</y>
        </coords>
    </waypoint>
    <waypoint>
        <id>5</id>
        <coords>
            <x>20</x>
            <y>20</y>
        </coords>
    </waypoint>


    <route>
        <a>1</a>
        <b>2</b>
        <chance>1</chance>
    </route>

    <route>
        <a>2</a>
        <b>3</b>
        <chance>0.5</chance>
    </route>
    <route>
        <a>2</a>
        <b>4</b>
        <chance>0.5</chance>
    </route>
    <route>
        <a>3</a>
        <b>5</b>
        <chance>1</chance>
    </route>
    <route>
        <a>4</a>
```

```
        <b>5</b>
        <chance>1</chance>
    </route>
</map>
```

8.3.17. test.map tartalma

```
<map>
    <name>Test</name>
    <waypoint>
        <id>1</id>
        <coords>
            <x>0</x>
            <y>0</y>
        </coords>
    </waypoint>
    <waypoint>
        <id>2</id>
        <coords>
            <x>100</x>
            <y>100</y>
        </coords>
    </waypoint>
    <route>
        <a>1</a>
        <b>2</b>
        <chance>1</chance>
    </route>
</map>
```

8.3.18. attack.mission tartalma

```
<mission>
    <name>Test</name>
    <enemy>
        <id>1</id>
        <waypointID>1</waypointID>
        <type>hobbit</type>
        <time>0</time>
    </enemy>
</mission>
```

8.3.19. one_enemy.mission tartalma

```
<mission>
    <name>Test</name>
    <enemy>
        <id>0</id>
```

```
        <waypointID>1</waypointID>
        <type>human</type>
        <time>0</time>
    </enemy>
</mission>
```

8.3.20. test.mission tartalma

```
<mission>
    <name>Test</name>
    <enemy>
        <id>1</id>
        <waypointID>1</waypointID>
        <type>hobbit</type>
        <time>0</time>
    </enemy>
    <enemy>
        <id>2</id>
        <waypointID>1</waypointID>
        <type>hobbit</type>
        <time>0</time>
    </enemy>
    <enemy>
        <id>3</id>
        <waypointID>1</waypointID>
        <type>hobbit</type>
        <time>0</time>
    </enemy>
    <enemy>
        <id>4</id>
        <waypointID>1</waypointID>
        <type>hobbit</type>
        <time>0</time>
    </enemy>
    <enemy>
        <id>5</id>
        <waypointID>1</waypointID>
        <type>hobbit</type>
        <time>0</time>
    </enemy>
</mission>
```

8.4. A tesztelést támogató programok tervei

A tesztelést végző segédprogram a specifikált formátumban megírt tesztfájlokat olvassa be, elindítja az alkalmazást, megadja neki az előírt bemenetet, majd a kapott kimenetet összehasonlítja a megaditt elvárt kimenettel, és amelyik sorban eltérést észlel, ott jelzi a különbséget. Ha nincs eltérés, a teszt sikeresen lefutott.

8.5. Napló

Kezdet	Időtartam	Résztvevők	Leírás
2014.04.06. 20:01	2 óra	**Tallér**	Osztályok leírásának elkészítése.
2014.04.06. 20:03	2 óra	**Szabó**	Tesztek megírása.
2014.04.06. 20:03	2 óra	**Török**	Tesztek megírása.
2014.04.06. 20:07	2 óra	**Nusser**	Osztályok leírásának elkészítése.

10. Prototípus beadása

10.1. Fordítási és futtatási útmutató

10.1.1. Fájllista

Fájl neve	Méret	Keletkezés ideje	Tartalom
Enemy.java	3 473 byte	2014.03.31 10:16	Az ellenségeket megvalósító osztály.
EnemyType.java	1 042 byte	2014.03.31 10:16	Az ellenségek típusait leíró osztály.
Fog.java	513 byte	2014.04.21 11:46	A köd viselkedését meghatározó osztály.
Game.java	11 727 byte	2014.03.31 10:16	A játék mechanikáját tartalmazó osztály.
Gem.java	357 byte	2014.03.31 10:16	A varázskövek közös absztrakt ősosztálya.
Map.java	3 628 byte	2014.03.31 10:16	Egy térképet tároló osztály.
Mission.java	2 390 byte	2014.03.31 10:16	Egy küldetés menetét leíró osztály.
Obstacle.java	2 116 byte	2014.03.31 10:16	Egy akadályt megvalósító osztály.
ObstacleGem.java	821 byte	2014.03.31 10:16	Az akadályokra rakható varázskő osztálya.
Pair.java	225 byte	2014.04.21 13:24	Egy egyszerű generikus pár.
Projectile.java	1 228 byte	2014.03.31 10:16	A lövedékeket működtető osztály.
Spawn.java	231 byte	2014.04.22 6:59	Egy ellenség megjelenését tároló osztály.
SplitterProjectile.java	760 byte	2014.04.21 15:42	Speciális, az ellenséget kettévágó Projectile.
Tower.java	3 696 byte	2014.03.31 10:16	Egy tornyot leíró osztály.
TowerGem.java	928 byte	2014.03.31 10:16	A tornyokat erősítő varázsköveket tárolja.
Vector.java	1 356 byte	2014.03.31 10:16	Egy két dimenziós valós vektor.
Waypoint.java	2 319 byte	2014.03.31 10:16	Az ellenségek útvonalainak egy csomópontja.
Main.java	4 548 byte	2014.04.22 11:09	A teszter segédprogram főosztálya.

10.1.2. Fordítás

```
javac -encoding utf8 -d bin src/szoftlab4/*.java src/Tester/*.java
```

10.1.3. Futtatás

Teszter indítása

```
cd tests
java -cp ../bin Tester.Main
```

Program indítása, ekkor a standard bementetről lehet parancsokat adni a programnak

```
cd bin
java szoftlab4.Main
```

Ilyenkor lehetséges előre elkészített teszteseteket is lefuttatni, pl így:

```
cd tests
java -cp ../bin szoftlab4.Main < teszteset.in
```

Ez azért lehet érdekes, mert ilyenkor látjuk a program kimenetét, nem csak a teszter mondja, hogy rendben lefutott a teszt. A tesztesetek egy .in és egy .out fájlból állnak, a .in a bemenetet tartalmazza, a .out meg az elvárt kimenetet.

10.2. Tesztek jegyzőkönyvei

10.2.1. attackdamagegem

Tesztelő neve	**Tallér Bátor**
Teszt időpontja	2014.04.22.

10.2.2. attackone

Tesztelő neve	**Tallér Bátor**
Teszt időpontja	2014.04.22.

Tesztelő neve	**Tallér Bátor**
Teszt időpontja	2014.04.22.
Teszt eredménye	Nem támadta meg az ellenséget
Lehetséges hibaok	Nem rakja be a projectile-t a projectiles listába
Változtatások	Hozzáadjuk a listához a létrejövő projectilet.

10.2.3. buildobstacle

Tesztelő neve	**Tallér Bátor**
Teszt időpontja	2014.04.22.

10.2.4. buildobstaclewrong

Tesztelő neve	**Tallér Bátor**
Teszt időpontja	2014.04.22.

10.2.5. buildtower

Tesztelő neve	**Tallér Bátor**
Teszt időpontja	2014.04.22.

10.2.6. buildtowerwrong

Tesztelő neve	**Tallér Bátor**
Teszt időpontja	2014.04.22.

10.2.7. critical

Tesztelő neve	**Tallér Bátor**
Teszt időpontja	2014.04.22.

10.2.8. elagazodasbalra

Tesztelő neve	**Török Attila**
Teszt időpontja	2014.04.22.

10.2.9. elagazodasjobbra

Tesztelő neve	**Török Attila**
Teszt időpontja	2014.04.22.

Tesztelő neve	**Török Attila**
Teszt időpontja	2014.04.22.
Teszt eredménye	Az ellenség továbbra is véletlenszerűen haladt.
Lehetséges hibaok	Rossz map file.
Változtatások	Map file átírása.

10.2.10. enemywin

Tesztelő neve	**Török Attila**
Teszt időpontja	2014.04.22.

10.2.11. fog

Tesztelő neve	**Török Attila**
Teszt időpontja	2014.04.22.

10.2.12. loadmap

Tesztelő neve	**Török Attila**
Teszt időpontja	2014.04.22.

10.2.13. noobstacle

Tesztelő neve	**Török Attila**
Teszt időpontja	2014.04.22.

10.2.14. obstacle

Tesztelő neve	**Török Attila**
Teszt időpontja	2014.04.22.

10.2.15. oneenemymove

Tesztelő neve	**Török Attila**
Teszt időpontja	2014.04.22.

10.3. Értékelés

Tag	Munka százalékban	Aláírás
Nusser	25 %	
Szabó	27 %	
Tallér	29 %	
Török	19 %	

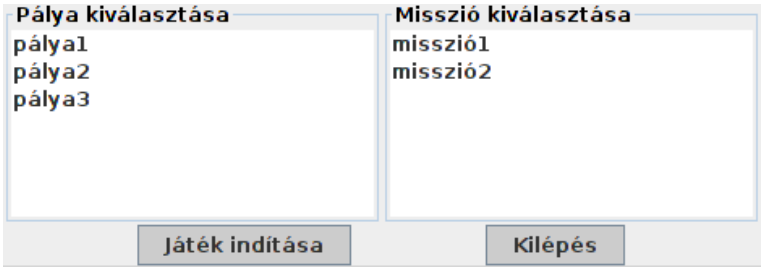
10.4. Napló

Kezdet	Időtartam	Résztvevők	Leírás
2014.04.21. 18:00	4 óra	**Tallér**	Tevékenység: Osztályok implementációja.
2014.04.21. 18:00	4 óra	**Török**	Tevékenység: Osztályok implementációja.
2014.04.21. 18:00	4 óra	**Szabó**	Tevékenység: Osztályok implementációja.
2014.04.21. 18:00	4 óra	**Nusser**	Tevékenység: Osztályok implementációja.
2014.04.22. 10:00	2 óra	**Tallér** **Török**	Tevékenység: Tesztelés

11. Grafikus felület specifikációja

11.1. A grafikus interfész

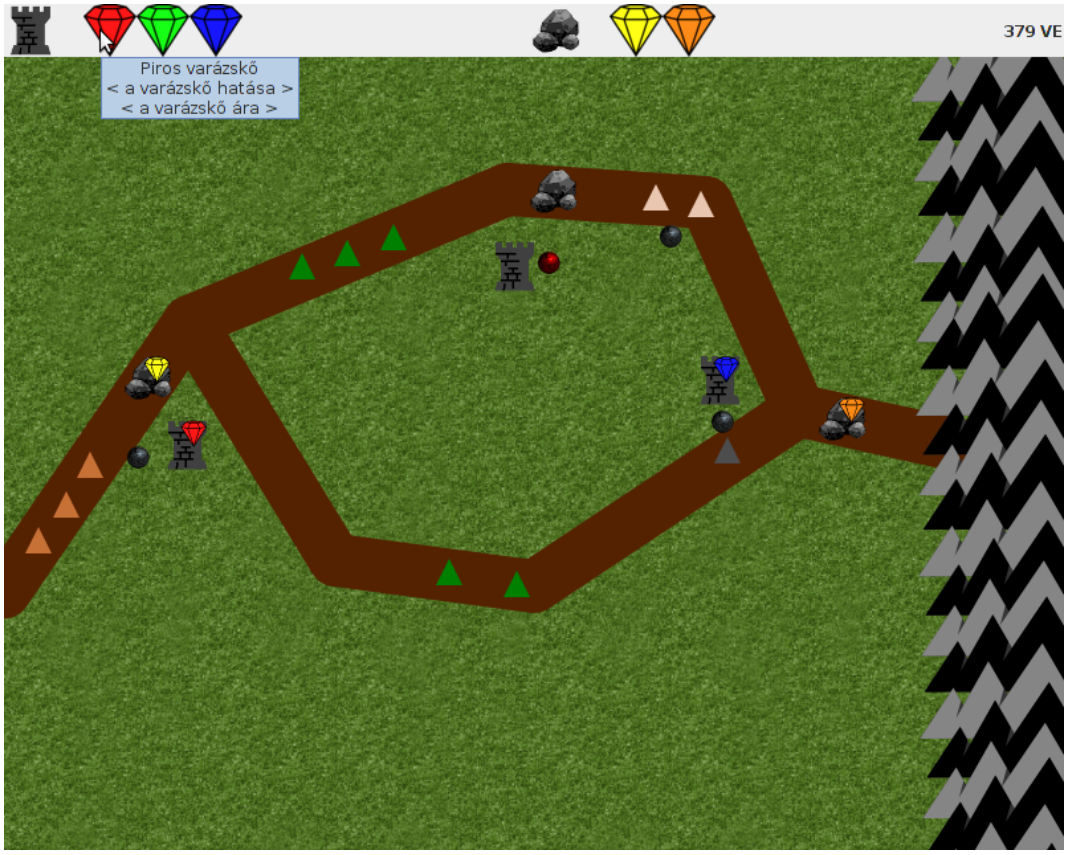
A szoftvert elindítva megjelenik egy grafikus ablak a következő tartalommal:



11.1. ábra. A menü kinézete

Értelemszerűen a bal oldali listában az elérhető pályák közül választhatja ki a felhasználó azt, amelyiken játszani szeretne, a jobb oldaliban pedig a kiválasztott pályához tartozó missziók közül azt, amelyiket el kívánja indítani.

Az alul látható gombok közül a „Kilépés” feliratú bezárja az alkalmazást, a „Játék indítása” feliratú pedig a fent kijelölt paraméterekkel elindítja a játékot. Ekkor az ablak átméreteződik, tartalma kicserélődik erre:



11.2. ábra. A játék képe

Az ablak felső részében a menüsáv, alatta pedig a játéktér foglal helyet.
A menüsávban lévő kis ikonok segítségével a következő funkciók érhetők el (balról jobbra haladva):

- Torony ikon: Először erre, majd a játéktéren egy megfelelő helyre (nem útra) kattintva egy torony építhető
- Piros, zöld és kék varázskő ikonok: Először ezekre, majd a játéktéren egy toronyra kattintva egy torony erősíthető a kiválasztott varázskővel
- Akadály ikon: Először erre, majd a játéktéren egy megfelelő helyre (útra) kattintva egy akadály építhető
- Sárga és narancssárga varázskő ikonok: Először ezekre, majd a játéktéren egy akadályra kattintva egy akadály erősíthető a kiválasztott varázskővel

A menüsáv jobb szélén pedig a jelenleg rendelkezésre álló varázserőnket olvashatjuk le.

Ahogyan a fenti ábrán is látszik, az egérmutatót a varázskő ikonokon tartva egy felugró eszköztipp ablakban megjelenik néhány sor szöveg, amely leírja a kiválasztott varázskő nevét, tulajdonságait, és varázserőben mért árát.

A játéktér jeleníti meg a csata pillanatnyi állapotát. A barna vonalak az ellenség útvonalai, amelyeken az ablak jobb szélen látható Végzet Hegye felé tartanak.

Az egyes ellenségeket különbözű színű háromszögek szimbolizálják. A sötétebb narancssárga embert, a halványrózsaszín hobbitot, a zöld tündét, a szürke pedig törpöt jelent.

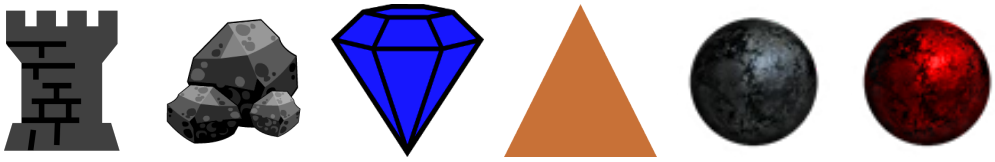
A tornyok magától értetődő ábrával jelennek meg, az akadályok az utakon pedig egy három szikladarabot ábrázoló kép formájában.

Azon tornyok és akadályok képe fölött, amelyek már meg lettek erősítve varázskővel, a rajtuk lévő kő ikonja megjelenik a jobb felső sarokban.

A tornyok által az ellenségekre kilőtt lövedékek a fekete, míg a különleges, kettévágó lövedékek a piros golyók.

Az alkalmanként leszálló köd az egész játékteret beborító halvány, szürkés átfedés formájában fog megjelenni.

Összefoglaló áttekintés a játéktér ikonjairól (nem méretarányosan):



(a) Torony (b) Akadály (c) Varázskő (d) Ellenség (e) Lövedék (f) Kettévágó lövedék

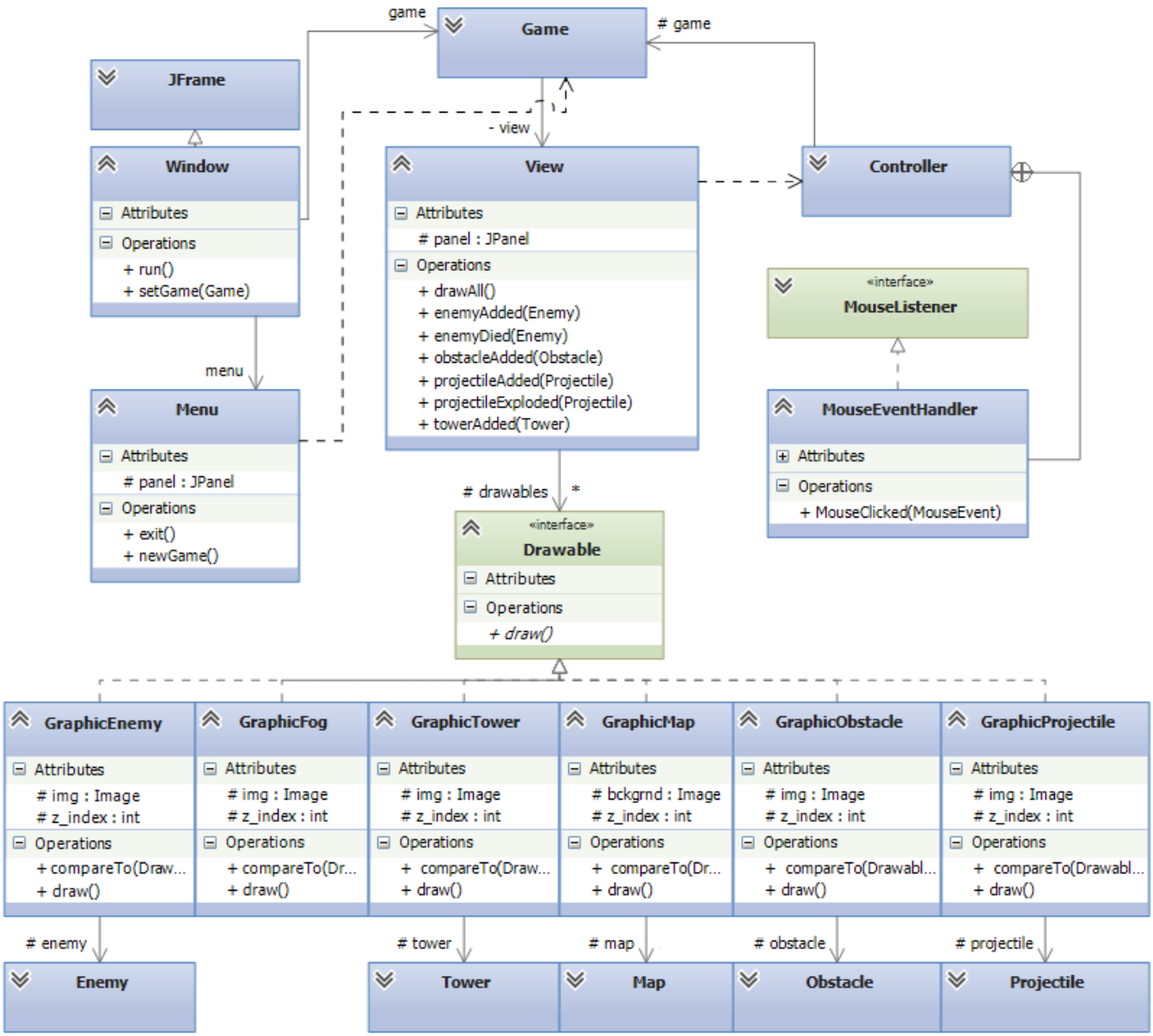
11.3. ábra. Áttekintés az ikonokról

11.2. A grafikus rendszer architektúrája

11.2.1. A felület működési elve

A grafikus felületet MVC architektúrának megfelelően próbáltuk megtervezni. Minden képernyőn megjelenő objektumnak van egy grafikus párja, ami megvalósítja a **Drawable** interfészt. A **View** osztály ilyen objektumokat tárol, és minden kirajzolásnál meghívja ezen objektumok draw metódusát.
A megjelenítés push alapú. Ha a modelben történt valami változás (amit hívhatunk egy eseménynek), akkor értesíti a **View**-t, hogy valami esemény történt, úgy hogy meghívja az eseménynek megfelelő metódust. Kirajzolásnál minden grafikus objektum, lekérdezi a **Game**-ben lévő párjától az adatait, majd ez alapján kirajzolja magát. Esemény esetén nem történik kirajzolás. Kirajzolás csakis a drawAll() metódus hívásra történik, amit a Game adott időközönként meghív.
Az MVC megvalósítás alól kivételt képez a program indulása, amikor is egy egyszerű menü jelenik, amin el lehet indítani a játékot, ezt a **Menu** osztály végzi. Ez az osztály annyira egyszerű, hogy felesleges ebben is egy MVC-t megvalósítani.
A **Controller** osztály a program elején, a **Game** és **View** objektumok létrehozása után feliratkozik a neki megfelelő eseményekre. Ez után már fogadja is az eseményeket.
Tervezés során ügyletünk arra is, hogy könnyen bővíthető, cserélhető legyen a rendszer. A **View** és **Graphic** osztályokból való származtatással új GUI-t lehet készíteni, anélkül, hogy meglévő kódban kelljen (sokat) változtatni.

11.2.2. A felület osztály-struktúrája



11.4. ábra. Grafikus felület osztálydigaramja

11.3. A grafikus objektumok felsorolása

11.3.1. Controller

- Felelősség
  A játék futása közben a felhasználótól érkező eseményeket kezeli, és értesíti róluk a Game-et. Az egyes események kezelését továbbadja a benne lévő eseménykezelő osztályoknak.
- Attribútumok
  - - game: Game: A kontrollerhez tartozó Game onjektum.
  - - view: View: A kontrollerhez tartozó View onjektum.
  - - mapMouseEvent: MapMouseDelegate: Ez tárolja mi az elvégzendő művelet a térképen történő eseményeknél
  - - activeButton: JButton: A jelenleg lenyomott gomb
- Metódusok
  - - nullActiveButton(): Kitörli a kiválasztott gombot, beállítja a gombok kinézetét

11.3.2. + Controller.MenuPanelMouseEvent

- Felelősség
  A menüsávon való kattintás lekezelése.
- Ősosztályok
  MouseAdapter
- Metódusok
  - + mousePressed(MouseEvent): Az egérkattintás eseménykezelője

11.3.3. + Controller.MapMouseDelegate «interface»

- Felelősség
  Egy interfész a térkép események kezelésére.
- Metódusok
  - + MapClicked(MouseEvent): Az egérkattintás eseménykezelője
  - + MapMoved(MouseEvent): Az egérmozgatás eseménykezelője

11.3.4. Drawable

- Felelősség
  Közös alapot biztosít a játékban az összes kirajzolható objektumnak.
- Interfészek
  - Comparable<Drawable>: A kirajzolási sorrend megállapítása miatt van erre szükség.
- Attribútumok
  - # img: Image: A kirajzolandó objektum képe.
  - # z_index: int: A kirajzolandó objektum mélységi elhelyezkedése.
- Metódusok

  - + *draw(Graphics)*: Kirajzolja az objektumot.
  - + compareTo(Drawable): int: Z-index szerint hasonlít össze két Drawable-t.
  - + drawRangeCircle(Graphics2D, Color, int x, int y, int radius): Egy kört rajzol ki az adott koordináták köré, a tornyok és akadályok hatótávolságának jelzésére van.

11.3.5. Game

- Attribútumok
  - - view: View: Egy referencia a grafikus megjelenítést végző osztályra.
  - + FPS: int: Másodpercenkénti képfrissítések száma.
  - - gameState: State: A játék állapotát tárolja.
- Metódusok
  - + getView: View: visszaadja a View-t
  - - initFPSCounter: Inicializája a vizuális FPS számlálót. Az FPS számláló a képernyő jobb felső sarkában látható
  - - setFog(): ki/be kapcsolja a ködöt. A kikapcsolt időtartamot egy 13 szórású 40 várhatóértékű, a bekapcsoltat egy 5 szórású 10 várhatóértékű normális eloszlás adja.
  - + toGameCoords(Vector): Vector: áttranszformál egy pozíciót játékbeli koordinátákra
  - + toMouseCoords(Vector): Vector: áttranszformál egy játékbeli koordinátát fizikai koordinátákra.

11.3.6. GemButton

- Felelősség
  Egy olyan Gomb, ami képes egy Gem-et tárolni.
- Ősosztályok
  JButton
- Attribútumok
  - - type : Gem: az eltárolt Gem
- Metódusok
  - + getGemType: Gem: visszatért az eltárolt Gem típusával

11.3.7. GraphicEnemy

- Felelősség
  Egy ellenség kirajzolása.
- Ősosztályok
  Drawable
- Attribútumok
  - # enemy: Enemy: A kirajzolandó ellenség.
- Metódusok
  - + draw(Graphics): Kirajzolja az ellenséget.
  - + equals(Object): boolean: Megadja, hogy két Graphic objektum egyenlő-e (ugyanazt a játékbeli objektumot reprezentálja).

11.3.8. GraphicFog

- Felelősség
  A köd kirajzolása.
- Ősosztályok
  Drawable
- Metódusok
  - + draw(Graphics): Kirajzolja a ködöt.

11.3.9. GraphicGem

- Felelősség
  Egy varázskő kirajzolása lerakás közben.
- Ősosztályok
  Drawable
- Attribútumok
  - # pos: Vector: A varázskő jelenlegi helye.
- Metódusok
  - + draw(Graphics): Kirajzolja a varázskövet.
  - + getPosition(): Vector: Visszaadja a varázskő helyét.

11.3.10. GraphicMap

- Felelősség
  A pálya kirajzolása.
- Ősosztályok
  Drawable
- Attribútumok
  - # map: Map: A kirajzolandó pálya.
  - # mountains: Image: A pálya jobb oldalán lévő hegyek képe.
- Metódusok
  - + draw(Graphics): Kirajzolja a pályát.
  - + equals(Object): boolean: Megadja, hogy két Graphic objektum egyenlő-e (ugyanazt a játékbeli objektumot reprezentálja).

11.3.11. GraphicObstacle

- Felelősség
  Egy akadály kirajzolása.
- Ősosztályok
  Drawable
- Attribútumok

  - # obstacle: Obstacle: A kirajzolandó akadály.
  - # gemImage: Image: Az akadályra rakott varázskő képe.

- Metódusok
  - + draw(Graphics): Kirajzolja az akadályt.
  - + equals(Object): boolean: Megadja, hogy két Graphic objektum egyenlő-e (ugyanazt a játékbeli objektumot reprezentálja).
  - + setGem(): Frissíti a kirajzolandó varázskő képét.

11.3.12. GraphicProjectile

- Felelősség
  Egy lövedék kirajzolása.
- Ősosztályok
  Drawable
- Attribútumok
  - # projectile: Projectile: A kirajzolandó lövedék.
- Metódusok
  - + draw(Graphics): Kirajzolja a lövedéket.
  - + equals(Object): boolean: Megadja, hogy két Graphic objektum egyenlő-e (ugyanazt a játékbeli objektumot reprezentálja).

11.3.13. GraphicTower

- Felelősség
  Egy torony kirajzolása.
- Ősosztályok
  Drawable
- Attribútumok
  - # tower: Tower: A kirajzolandó torony.
  - # gemImage: Image: A toronyra rakott varázskő képe.
- Metódusok
  - + draw(Graphics): Kirajzolja a tornyot.
  - + equals(Object): boolean: Megadja, hogy két Graphic objektum egyenlő-e (ugyanazt a játékbeli objektumot reprezentálja).
  - + setGem(): Frissíti a kirajzolandó varázskő képét.

11.3.14. Main

- Felelősség
  Elindítani a játékot, a parancssori argumentumoknak megfelelően beállítani néhány statikus változót.
- Metódusok
  - + main(String[]): A program belépési pontja.

11.3.15. Menu

- Felelősség
  A játék főmenüjét valósítja meg, amivel új játékot lehet indítani, ezen belül kiválasztani a pályát és a missziót.
- Attribútumok
  - # panel: JPanel: A menü kirajzolásához használt JPanel.
  - - mapListModel: AbstractListModel: pályákat tárolja
  - - missionListModel: AbstractListModel: küldetéseket tárolja
- Metódusok
  - + exit(): Bezárja a játék ablakait, és kilép a programból.
  - + newGame(): Új játék indítását kezdeményezi.
  - + run(): Elindítja a menü futását.
  - - everythingSelected(): boolean: Leellenőrzi, hogy pálya és misszió is ki van-e választva.

11.3.16. Resources

- Felelősség
  Eltárolja a játék során felhasznált képeket, azért, hogy csak egyszer kelljen őket betölteni.
- Attribútumok
  - MouseListener
  - + TowerImage: Image
  - + ObstacleImage: Image
  - + RedGemImage: Image
  - + GreenGemImage: Image
  - + BlueGemImage: Image
  - + YellowGemImage: Image
  - + OrangeGemImage: Image
  - + HumanImage: Image
  - + DwarfImage: Image
  - + ElfImage: Image
  - + HobbitImage: Image
  - + BackgroundImage: Image
  - + MountainsImage: Image
  - + ProjectileImage: Image
  - + SplitterProjectileImage: Image
  - + LZImage: Image
- Metódusok
  - + load: betölti a képeket, IOException-t dob ha valamelyik nem sikerült.

11.3.17. View

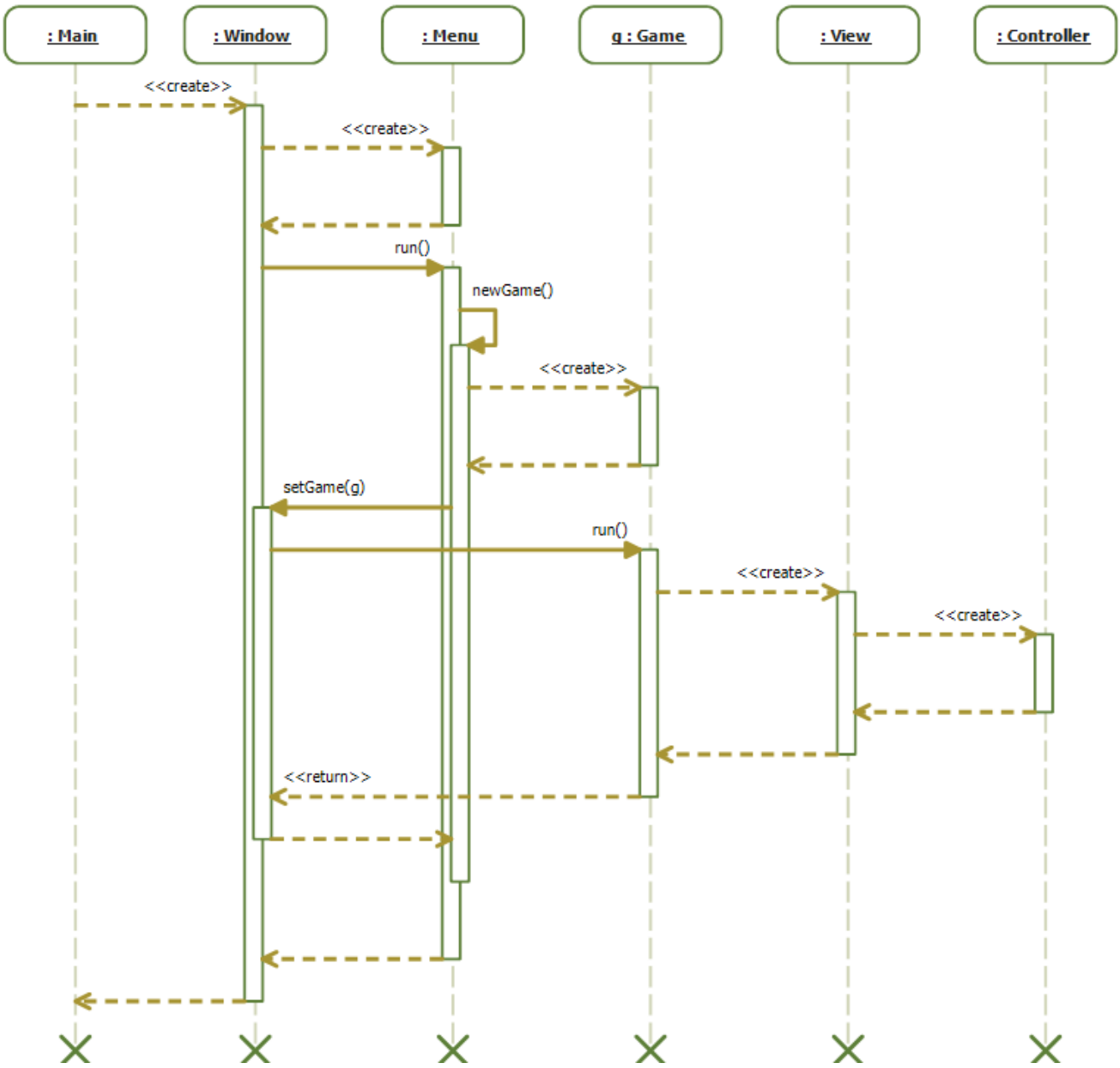
- Felelősség
  A játék kirajzolása, azaz a Game osztály állapotának megjelenítésa a képernyőn.
- Attribútumok
  - - panel: JPanel: A játék kirajzolásához használt JPanel
  - - drawables: List<Drawable>: A játékban kirajzolandó dolgok (ellenségek, tornyok, stb.) listája rendezve, hogy jó sorrenben legyen az objektumok kirajzolva.
  - - mapPanel: JPanel: A játéknak a játékterét (pálya, ellenségek, tornyok, stb.) megjelenítő JPanel.
  - - magicPanel: JPanel: A játék ezköztárát megjelenítő JPanel.
  - - placing: Drawable: Azt az objektumot tárolja, ami épp lerakás alatt van, így ekkor is meg lehet jeleníteni.
- Metódusok
  - + addDrawable(Drawable): Berak egy Drawable-t a kirajzolandó objektumok listájába.
  - + drawAll(): Kirajzolja a játék jelenlegi állását.
  - + enemyAdded(Enemy): A Game hívja meg amikor új ellenség kerül a pályára, frissíti a View adatstruktúráit.
  - + enemyDied(Enemy): A Game hívja meg amikor meghal egy ellenség, frissíti a View adatstruktúráit.
  - + gameLost(): Kirajzolja a vesztes játék végi üzenetet.
  - + gameWon(): Kirajzolja a nyertes játék végi üzenetet.
  - + getPanel(): Visszaadja a „panel” tagváltozót.
  - + magicChange(int): Megváltoztatja a kiírt varázserő értéket a paraméterként kapottra.
  - + obstacleAdded(Obstacle): A Game hívja meg amikor új akadály kerül a pályára, frissíti a View adatstruktúráit.
  - + obstacleEnchanted(Obstacle): Frissíti egy meglévő akadály aktív varázskövét.
  - - setButtonLook(JButton, ImageIcon): Beállítja egy JButton kinézetét.
  - + setPlacing(Drawable): Beállítja az éppen lerakás alatt álló objektumot.
  - + towerAdded(Tower): A Game hívja meg amikor új torony kerül a pályára, frissíti a View adatstruktúráit.
  - + towerEnchanted(Tower): Frissíti egy meglévő torony aktív varázskövét.
  - + projectileAdded(Projectile): A Game hívja meg amikor új lövedék kerül a pályára, frissíti a View adatstruktúráit.
  - + projectileExploded(Projectile): A Game hívja meg amikor egy lövedék megsemmisül, frissíti a View adatstruktúráit.
  - - winLoseScreen(String, Image): Kirajzolja a játék végi üzenetet és képet (nyerés/vesztés).

11.3.18. Window

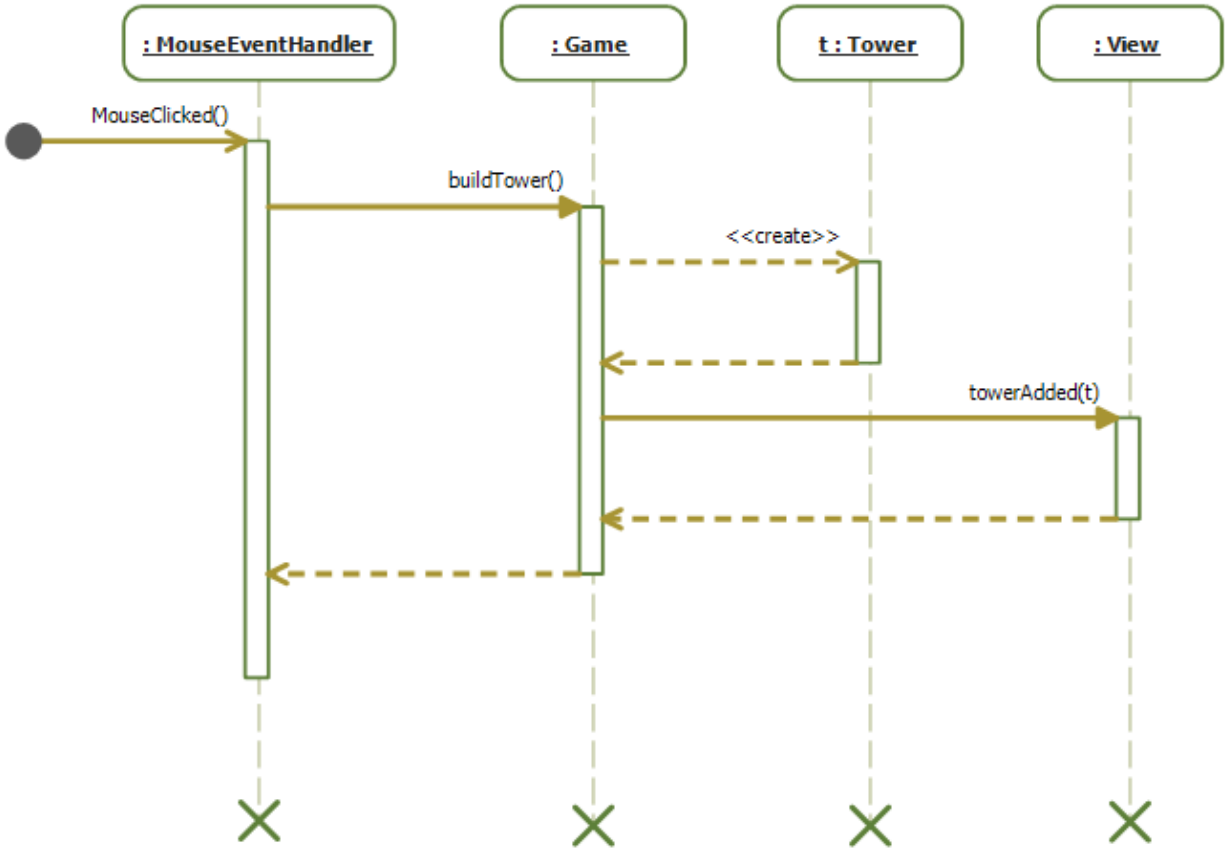
- Felelősség
  A játék fő ablaka, ez jeleníti meg a menüt és aztán magát a játékot.
- Ősosztályok
  JFrame

- Attribútumok
  - - menu: Menu: A játék menü objektumát tárolja.
  - - game: Game: A játék Game objektumát tárolja.
- Metódusok
  - + setGame(Game): Beállítja azt a Game objektumot, amit a kezelőfelület meg fog jeleníteni. A menü fogja meghívni a pálya- és misszióválasztás után. Amikor a játék készen áll az elindításra, jelez az erre az objektumra váró objektumoknak.
  - + runGame(): elindítja a játékot

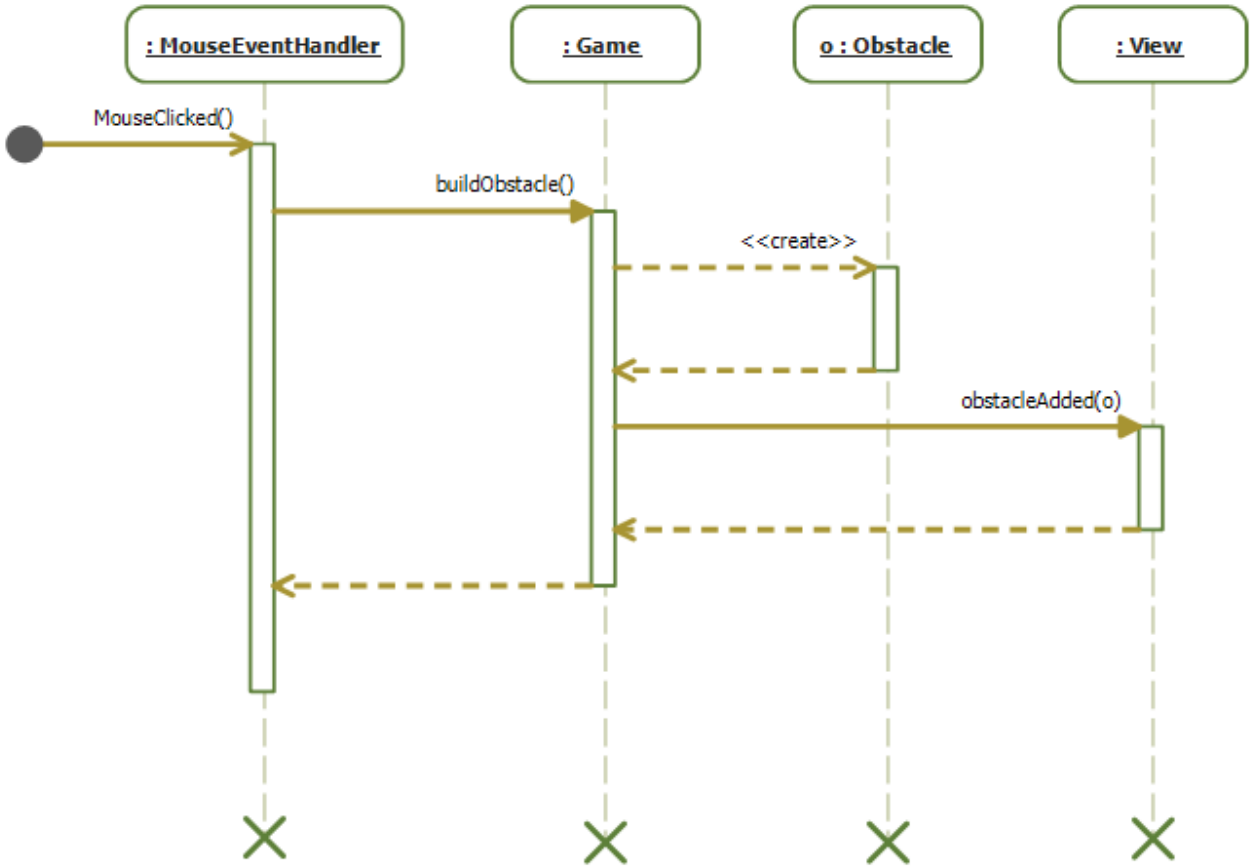
11.4. Kapcsolat az alkalmazói rendszerrel



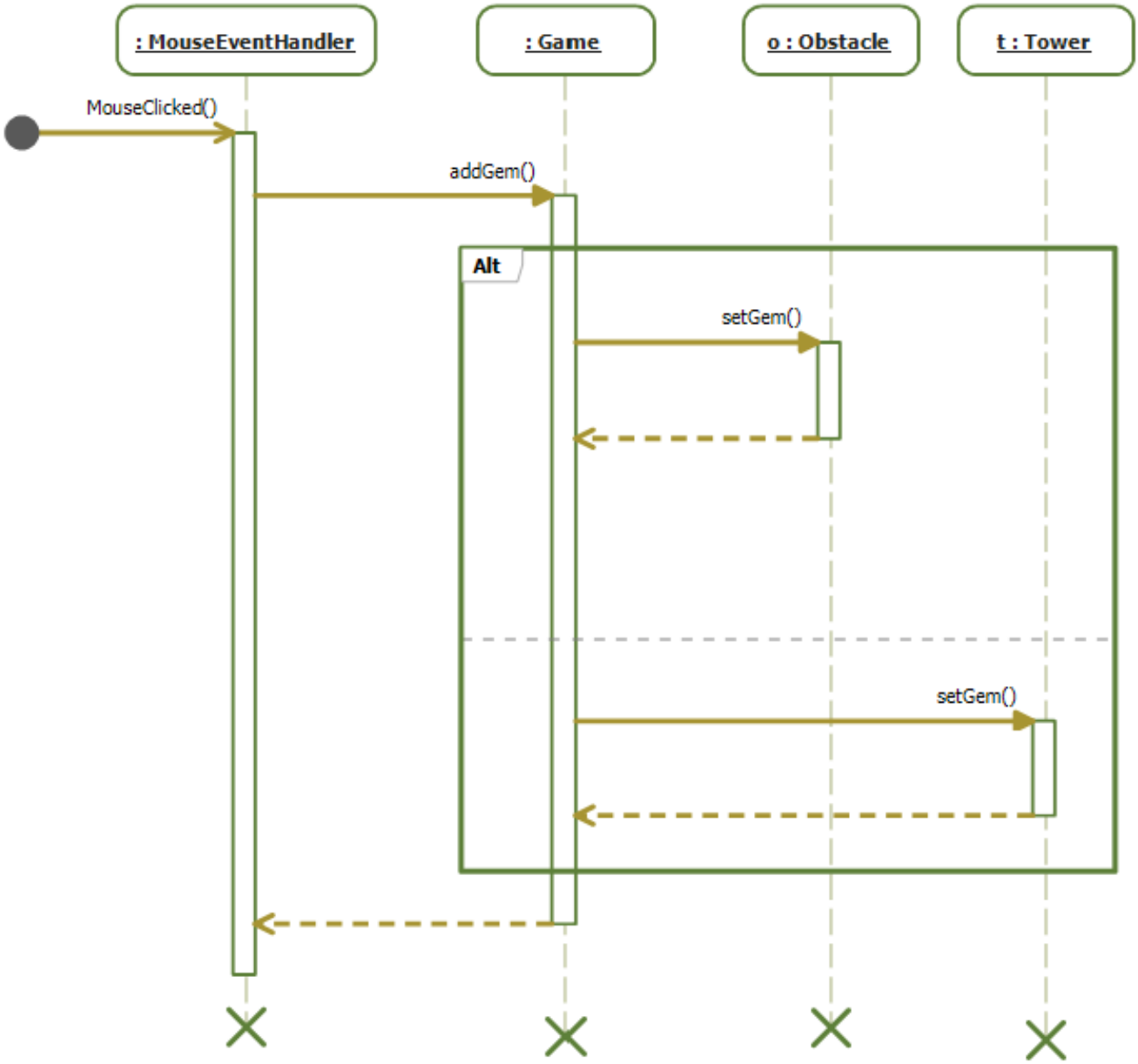
11.5. ábra. Játék inicializálása.



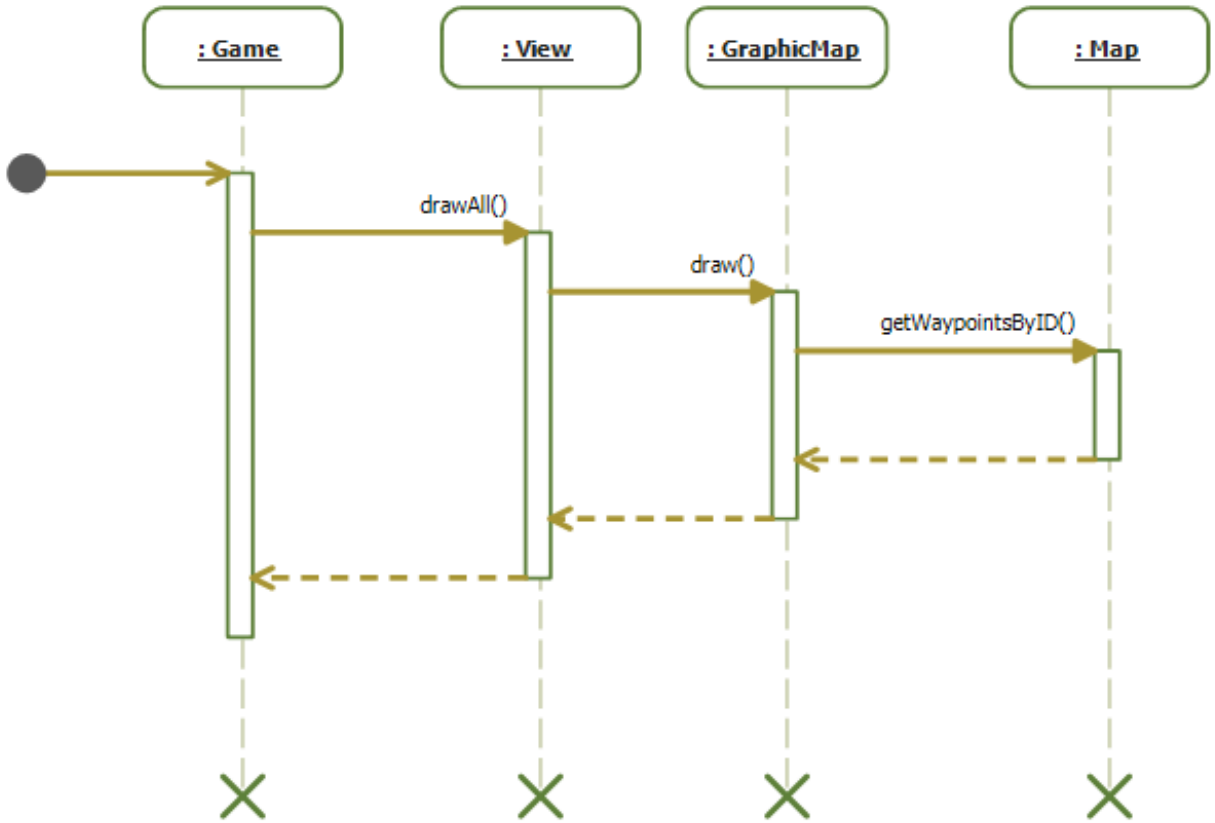
11.6. ábra. Egérkattintásra torony lerakása.



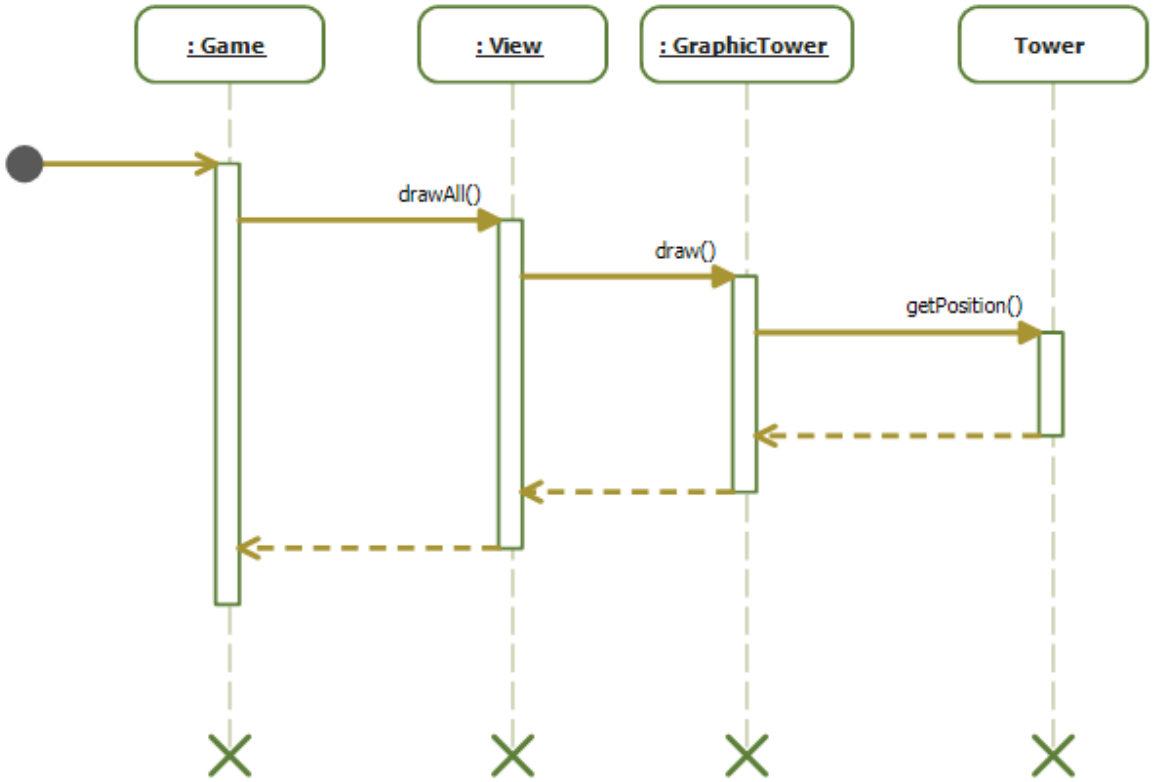
11.7. ábra. Egérkattintásra akadály lerakása.



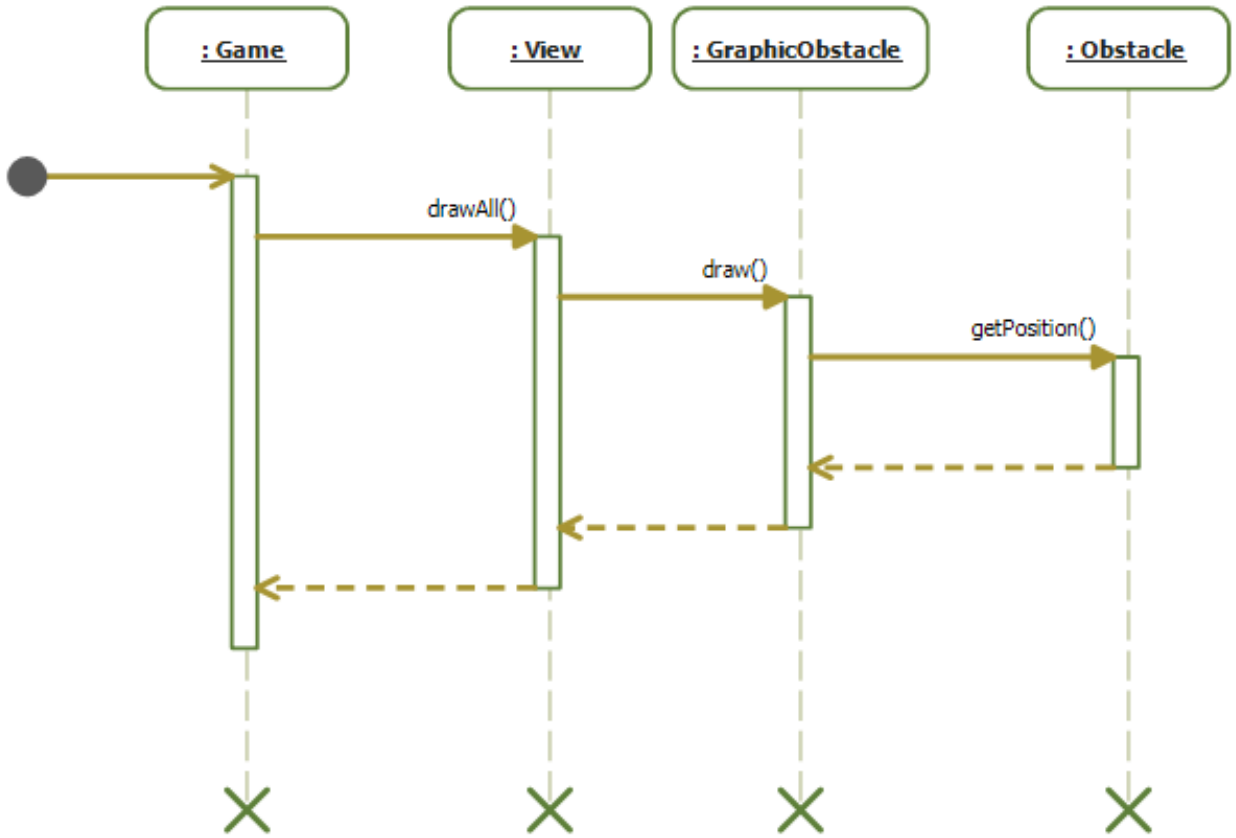
11.8. ábra. Egérkattintásra torony vagy akadály varázskővel felruházása.



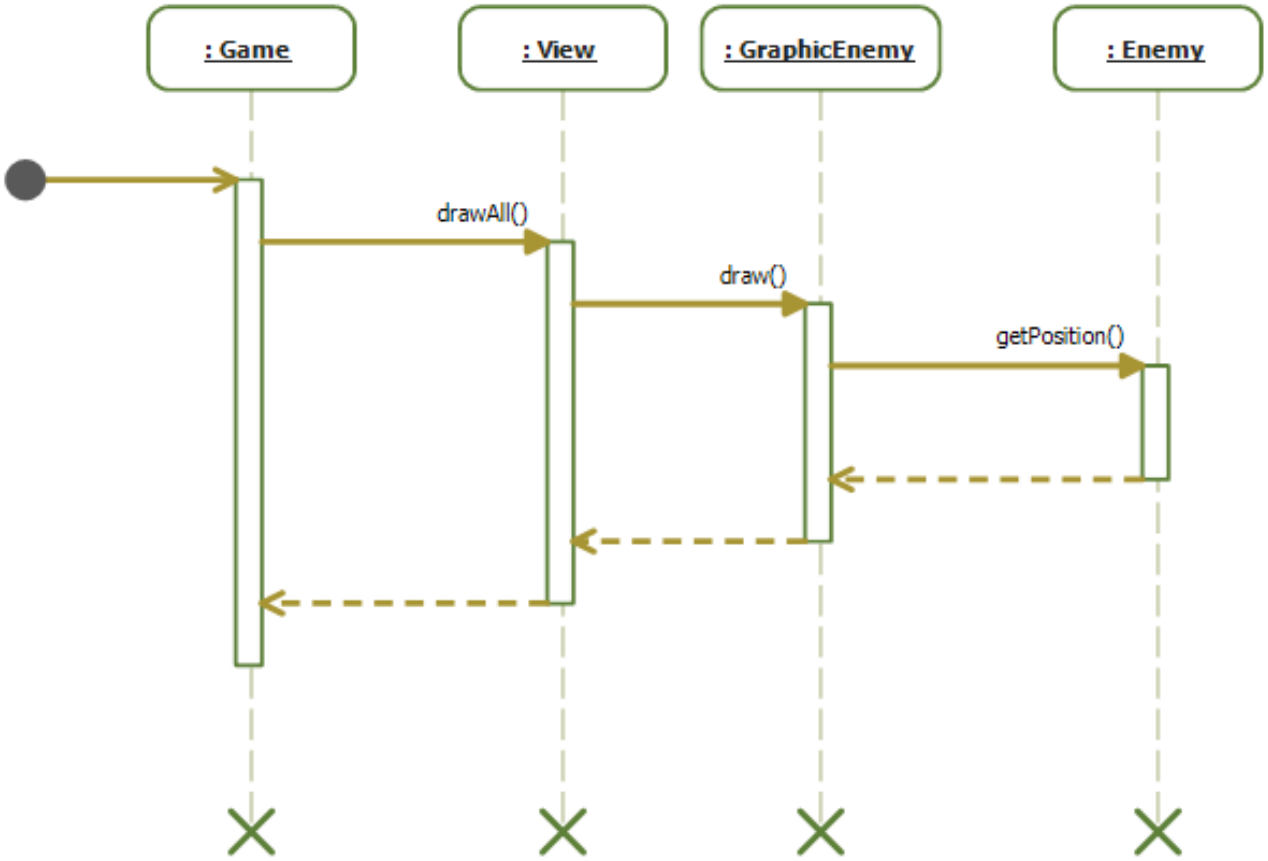
11.9. ábra. Az egész játék kirajzolása.(Map része)



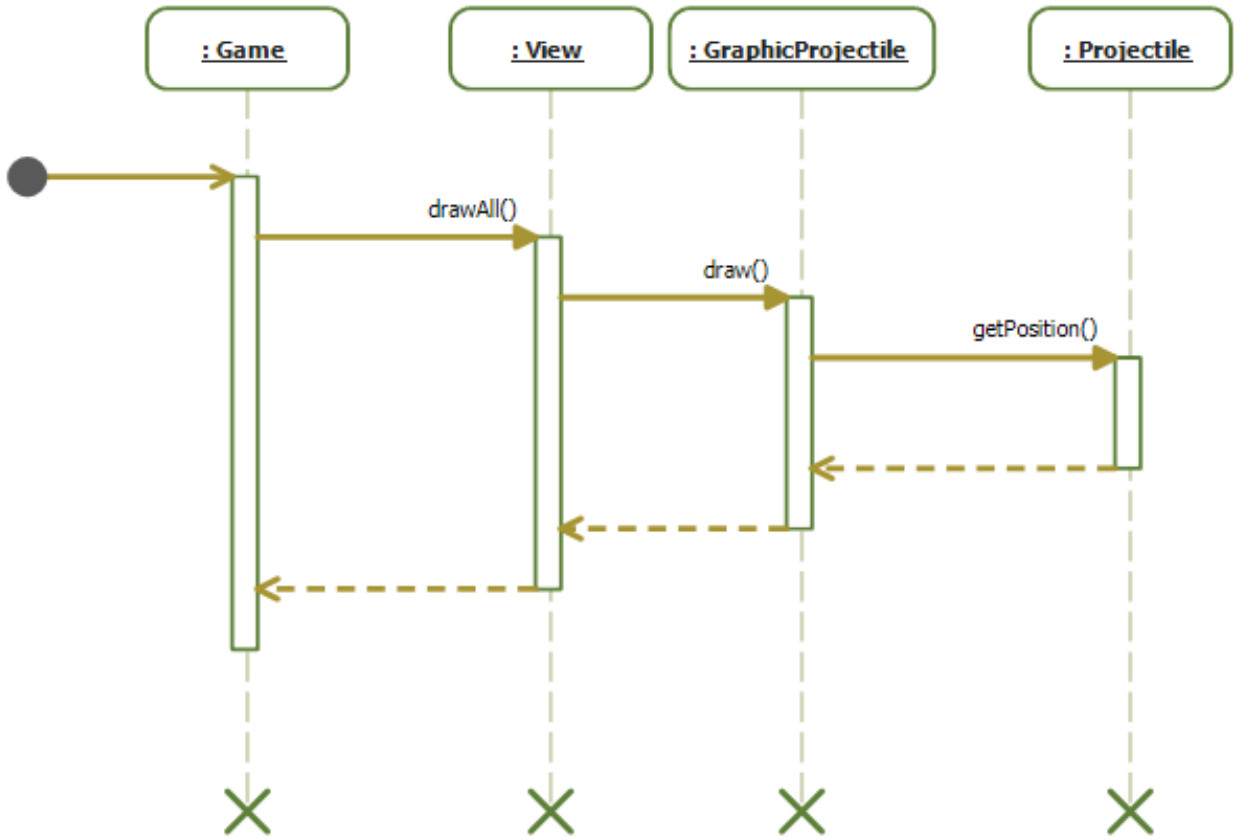
11.10. ábra. Az egész játék kirajzolása.(Tower része)



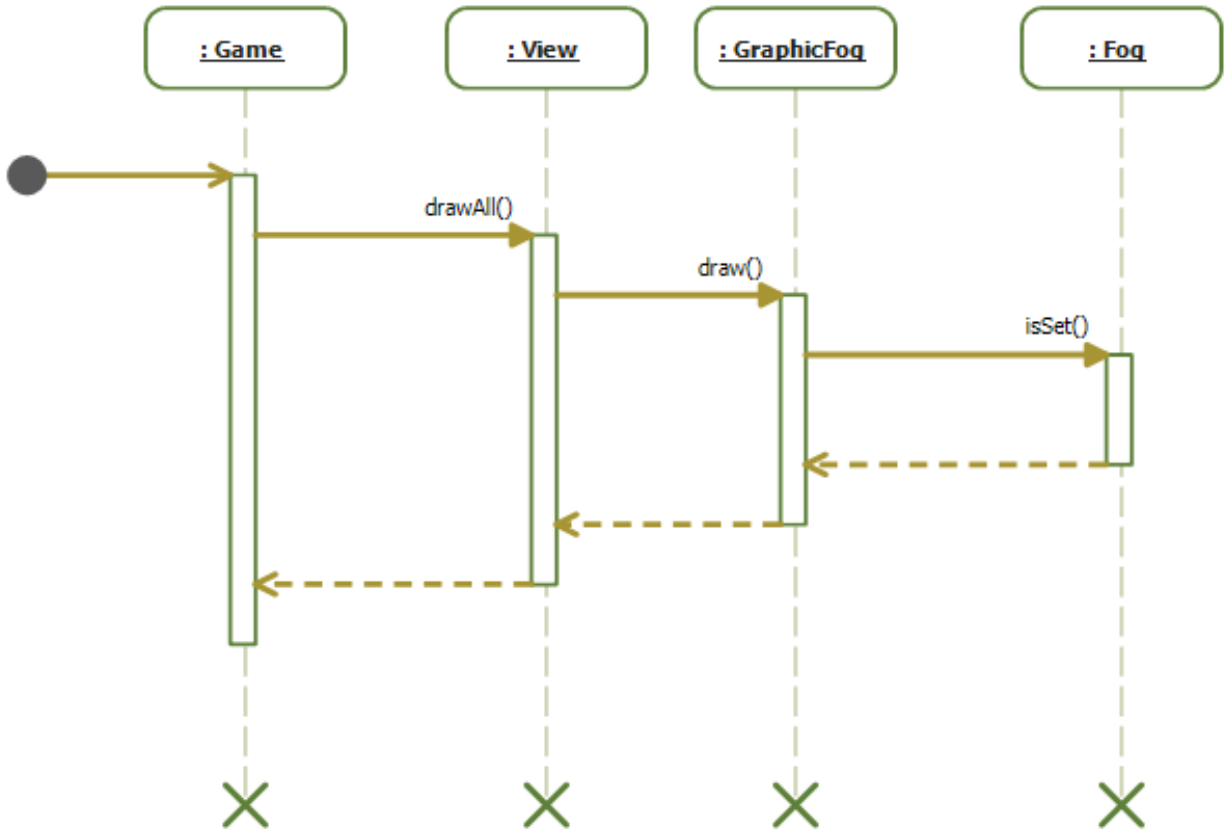
11.11. ábra. Az egész játék kirajzolása.(Obstacle része)



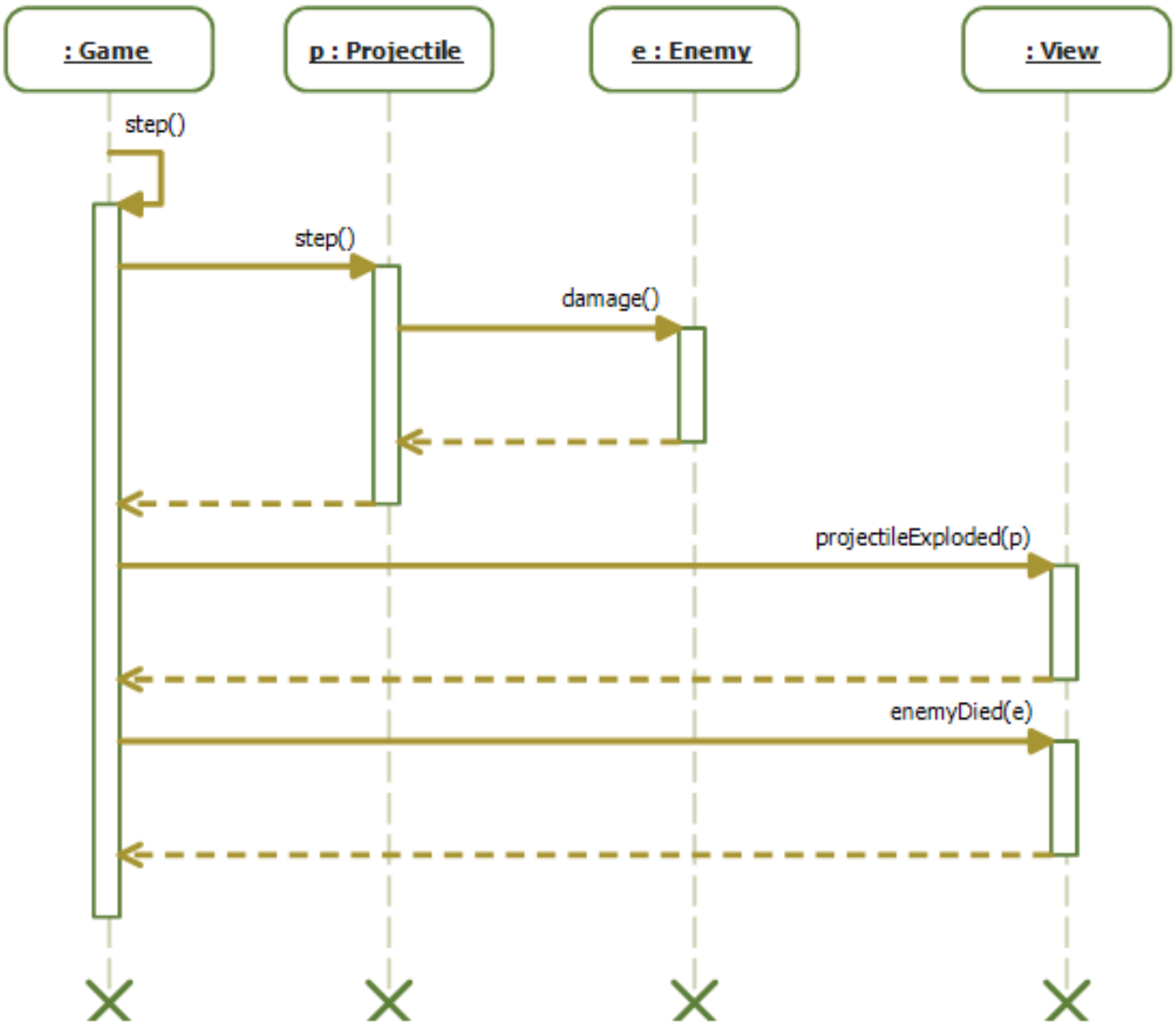
11.12. ábra. Az egész játék kirajzolása.(Enemy része)



11.13. ábra. Az egész játék kirajzolása.(Projectile része)



11.14. ábra. Az egész játék kirajzolása.(Fog része)



11.15. ábra. Ellenség és lövedék törlése.

11.5. Napló

Kezdet	Időtartam	Résztvevők	Leírás
2014.04.25. 14:00	0,5 óra	**Nusser** **Tallér** **Török**	Értekezlet. Grafikus felület architektúrájának és alap grafikus osztályok működésének megbeszélése.
2014.04.26. 11:00	0,5 óra	**Tallér**	Tevékenység: osztálydiagram 1. változatának elkészítése
2014.04.26. 12:00	0,5 óra	**Szabó** **Tallér** **Török**	Értekezlet. Architektúra megvitatása.
2014.04.27. 16:00	1 óra	**Tallér**	Tevékenység: osztálydiagram befejezése, hozzá tartozó dokumentáció megírása.
2014.04.27. 17:00	2 óra	**Szabó** **Tallér** **Török**	Tevékenység: A grafikus felület elemeinek, kinézetének összeállítása.
2014.04.27. 20:00	3 óra	**Török**	Tevékenység: A felhasználói felület megtervezése, dokumentálása.
2014.04.27. 22:00	3,5 óra	**Szabó**	Osztályleírások elkészítése.
2014.04.27. 21:00	3 óra	**Nusser**	Tevékenység: Szekvencia diagramok készítése.
2014.04.28. 9:00	0,5 óra	**Tallér**	Tevékenység: Dokumentációban hibák javítása, végső ellenőrzés.

13. Grafikus változat

13.1. Fordítási és futtatási útmutató

13.1.1. Fájllista

Fájl neve	Méret	Keletkezés ideje	Tartalom
icons/background.png	323033 byte	2014.05.03 19:15	A játéktér háttérképe.
icons/blue_gem.png	2298 byte	2014.05.03 17:48	A kék varázskövek ikonja.
icons/dwarf.png	304 byte	2014.05.04 19:06	A törpök ikonja.
icons/elf.png	308 byte	2014.05.04 19:06	A tündék ikonja.
icons/green_gem.png	2311 byte	2014.05.03 17:48	A zöld varázskövek ikonja.
icons/hobbit.png	319 byte	2014.05.04 19:06	A hobbitok ikonja.
icons/human.png	312 byte	2014.05.04 19:06	Az emberek ikonja.
icons/LZF.png	52032 byte	2014.05.05 16:29	A játék megnyerésekor megjelenő kép.
icons/obstacle.png	2430 byte	2014.05.03 17:48	Az akadályok ikonja.
icons/orange_gem.png	2526 byte	2014.05.03 17:48	A narancssárga varázskövek ikonja.
icons/projectile.png	1167 byte	2014.05.04 19:22	A lövedékek ikonja.
icons/red_gem.png	2302 byte	2014.05.03 17:48	A piros varázskövek ikonja.
icons/saurontower.png	43961 byte	2014.05.03 19:15	A Végzet Hegyét és Szauron tornyát ábrázoló kép.
icons/splitter_projectile.png	905 byte	2014.05.04 19:22	A kettévágó lövedékek ikonja.
icons/tower.png	989 byte	2014.05.03 17:48	A tornyok ikonja.
icons/yellow_gem.png	2443 byte	2014.05.03 17:48	A sárga varázskövek ikonja.
maps/demo.map	1228 byte	2014.05.04 20:42	A felületi terven is látható pályafájl.
maps/test.map	267 byte	2014.05.04 03:54	Egy minimális pályafájl.
missions/demo_advanced.mission	5401 byte	2014.05.10 15:00	Egy összetettebb missziófájl.
missions/demo_attack.mission	1251 byte	2014.05.04 20:42	Egy egyszerű missziófájl.
missions/demo_delay.mission	1250 byte	2014.05.05 01:19	Egy egyszerű missziófájl, hosszabb felkészülési idővel.
missions/test_attack.mission	139 byte	2014.05.04 03:54	Egy minimális missziófájl.
src/Controller.java	5456 byte	2014.05.03 15:20	Az MVC modell controller részét valósítja meg.
src/Drawable.java	2322 byte	2014.05.03 18:07	A kirajzolható objektumok közös ősosztálya.
src/Enemy.java	3362 byte	2014.03.20 20:33	Az ellenségeket megvalósító osztály.
src/EnemyType.java	998 byte	2014.03.21 12:57	Az ellenségtípusokat leíró osztály.
src/Fog.java	640 byte	2014.04.21 18:25	A ködöt működtető osztály.
src/Game.java	10064 byte	2014.03.20 09:55	A játék logikáját összefogó osztály.
src/Gem.java	409 byte	2014.03.21 12:57	A varázskövek közös ősosztálya.
src/GemButton.java	289 byte	2014.05.03 16:09	A gombok, amelyekkel varázskövet lehet rakni az épületekre.
src/GraphicEnemy.java	1205 byte	2014.05.03 18:07	Az ellenségek kirajzolásáért felelős osztály.
src/GraphicFog.java	432 byte	2014.05.03 21:32	A köd kirajzolásáért felelős osztály.
src/GraphicGem.java	1101 byte	2014.05.05 20:48	A varázskövek kirajzolásáért felelős osztály.
src/GraphicMap.java	2098 byte	2014.05.03 19:14	A pálya kirajzolásáért felelős osztály.

Fájl neve	Méret	Keletkezés ideje	Tartalom
src/GraphicObstacle.java	2290 byte	2014.05.03 18:16	Az akadályok kirajzolásáért felelős osztály.
src/GraphicProjectile.java	1016 byte	2014.05.03 18:07	A lövedékek kirajzolásáért felelős osztály.
src/GraphicTower.java	2401 byte	2014.05.03 18:16	A tornyok kirajzolásáért felelős osztály.
src/Main.java	2645 byte	2014.05.04 19:32	Az alkalmazást elindító osztály.
src/Map.java	3801 byte	2014.03.20 20:33	Egy pályát megvalósító osztály.
src/Menu.java	6174 byte	2014.05.04 03:53	A játék indító menüjét megvalósító osztály.
src/Mission.java	2354 byte	2014.03.20 20:33	Egy küldetést leíró osztály.
src/Obstacle.java	2206 byte	2014.03.21 12:57	Az akadályokat megvalósító osztály.
src/ObstacleGem.java	805 byte	2014.03.20 20:33	Az akadályokra tehető varázskövek.
src/Pair.java	209 byte	2014.04.21 13:10	Párokat tároló generikus osztály.
src/Projectile.java	1219 byte	2014.03.20 20:33	Egy lövedék viselkedését megvalósító osztály.
src/Resources.java	1949 byte	2014.05.05 21:58	Az összes játékban használt képet eltároló osztály.
src/Spawn.java	302 byte	2014.04.22 00:59	Egy konkrét ellenség típusát és játékba lépésének idejét tárolja.
src/SplitterProjectile.java	729 byte	2014.04.21 17:43	A kettévágó lövedékeket megvalósító osztály.
src/Tower.java	3646 byte	2014.03.20 20:33	Egy tornyot megvalósító osztály.
src/TowerGem.java	902 byte	2014.03.20 20:33	A tornyokra rakható varázskövek.
src/Vector.java	1397 byte	2014.03.20 20:33	Egy koordináta pontot megvalósító osztály.
src/View.java	10510 byte	2014.05.03 14:31	Az MVC modell view része.
src/Waypoint.java	2208 byte	2014.03.21 12:57	Az utakat kijelölő pontokat magvalósító osztály.
src/Window.java	795 byte	2014.05.04 03:53	A grafikus ablakot kezelő osztály.

13.1.2. Fordítás

A fordításhoz lépjünk be egy terminálban a projekt főkönyvtárába, majd adjuk ki a következő parancsot:

```
javac -d bin -encoding utf8 src/szoftlab4/*.java
```

13.1.3. Futtatás

A futtatáshoz pedig a fordítást követően:

```
java -cp bin szoftlab4.Main
```

A fenti parancs után szóközzel elválasztva fűzhetünk még tetszés szerint kétféle parancssori argumentumot. Egyikkel bekapcsolhatjuk az élsimítást (antialiasingot) a kirajzolásnál, a másikkal pedig a másodpercenként kirajzolt képkockák számát jeleníttethetjük meg a grafikus ablak jobb felső sarkában. Szokás szerint mindkét paraméternek van egy hosszú és egy rövid nevű változata.

A megszokott jelölést követve tehát a lehetséges opciók:

```
java -cp bin szoftlab4.Main [--antialiasing|-aa] [--showfps|-f]
```

13.2. Értékelés

Tag	Munka százalékban	Aláírás
Nusser	23 %	
Szabó	26 %	
Tallér	29 %	
Török	22 %	

13.3. Napló

Kezdet	Időtartam	Résztvevők	Leírás
2014.05.01. 16:00	1 óra	**Tallér**	Tevékenység: Game osztály átalakítása.
2014.05.03. 14:00	2,5 óra	**Tallér**	Tevékenység: View és Controller osztály elkészítése.
2014.05.03. 18:00	2 óra	**Nusser**	Tevékenység: A grafikus osztályok megvalósítása.
2014.05.03. 22:00	1 óra	**Tallér**	Tevékenység: Hibák javítása, apró módosítások.
2014.05.03. 20:00	1,5 óra	**Tallér**	Tevékenység: Javítások, felhasználó felület élesztése.
2014.05.04. 02:00	2 óra	**Török**	Tevékenység: A Window és a Menu osztályok megvalósítása.
2014.05.04. 19:00	4 óra	**Török**	Tevékenység: Egy pálya- és egy missziófájl elkészítése, valamint rengeteg apró javítás eszközölése a kódon.
2014.05.04. 20:00	2 óra	**Szabó**	Tevékenység: A View osztály frissítésének megvalósítása, útkirajzolás implementálása; kisebb javítások/-módosítások.
2014.05.04. 23:00	1 óra	**Nusser**	Tevékenység: Apróbb javítások a kódon.
2014.05.05. 14:00	2 óra	**Tallér**	Tevékenység: Javítások, felhasználó felület csiszolása, hibakeresés.
2014.05.05. 16:00	1 óra	**Nusser**	Tevékenység: Hibák javítása, kisebb fejlesztések.
2014.05.05. 01:00	1 óra	**Török**	Tevékenység: A grafikus felület fejlesztése, javítása.
2014.05.05. 19:00	2,5 óra	**Szabó**	Tevékenység: Az éppen építésre/lerakásra váró tornyok/akadályok/varázskövek folyamatos kirajzolásának implementálása, a képek betöltésének átrendezése.
2014.05.06. 14:00	0,5 óra	**Tallér**	Tevékenység: Javítások, felhasználó felület csiszolása, hibakeresés.
2014.05.10. 12:00	1 óra	**Tallér**	Tevékenység: Kommentezés, kisebb javítások.
2014.05.11. 12:00	4 óra	**Szabó**	Tevékenység: Mindenféle javítás, kommentezés.
2014.05.11. 17:00	2 óra	**Nusser**	Tevékenység: Kommentek írása.
2014.05.11. 23:00	4 óra	**Török**	Tevékenység: Kommentezés, a fejezet dokumentációjának megírása.

14. Összefoglalás

14.1. Projekt összegzés

Tag	Munkaidő (óra)
Nusser	46
Szabó	54.5
Tallér	58.5
Török	46.5
Összesen:	**205.5**

Fázis	Forrássor
Szkeleton	962
Protó	1686
Grafikus	2993

14.1.1. Nusser Ádám véleménye

- Mit tanultak a projektből konkrétan és általában?
  Ahogy hallottam, az iparban nincs mindig idő (soha), hogy ilyen részletesen megtervezzünk és dokumentáljunk egy programot, így jó volt legalább egyszer végigmenni a lépésin. Ezzel kicsit betekintést nyertünk szerintem a szoftveresek világába, amiből sokat tanultunk. A módszert ismertük, de így, hogy a gyakorlatban is láttuk, sokat segített abban, hogy valóban értelmét lássuk az egésznek. Hasznos volt még megismerni különböző programokat és nyelveket (Github, LateX, ...) melyek a jövőben is hasznosak lehetnek.
  Talán csapatmunkával kapcsolatban tanultuk a legkevesebbet, mert jól programozik és szorgalmas a csapat, így az egyéni munka sokkal több volt mint a csapatmunka, és míg ez nálunk működött, nagyon rossz vége is lehetett volna.

- Mi volt a legnehezebb és a legkönnyebb?
  A programváz tervezése volt talán a legnehezebb, hiszen nagyon nehéz olyan architektúrát tervezni, mely majd minden változtatás, és minden hozzáadott dolgot túléljen, és működjön nagyobb változtatások nélkül. A csapatban való dolgozás sem lett volna egyszerű, ha a csapattársaim nem lennének ilyen kiváló szakemberek, mert sajnos sokszor egyéni munkával oldottunk meg olyan feladatokat, melyeket sokkal egyszerűbben meg tudtunk volna oldani, ha leülünk és részletesebben átgondoljuk, megbeszéljük (ez főleg a vége felé igaz).
  A legegyszerűbb rész az implementáció volt, hiszen hetek óta azt terveztük, és mire oda értünk hogy kódolni kéne, már kentük-vágtuk, hogy mit fogunk nagyjából írni.

- Összhangban állt-e az idő és a pontszám az elvégzendő feladatokkal?
  Nagyrészt igen. Volt egy-két rész, amit ha az ember előtte jól megírt akkor később alig volt vele munka és mégis sok pontot ért, de ez így jó.

- Ha nem, akkor hol okozott ez nehézséget?
  Nem okozott.

- Milyen változtatási javaslatuk van?
  Szerintem a tervezéskor van egy-két olyan feladat, mely nagyon kicsi módosításokkal már szerepelt korábban, és így érdemi munkát nem ad, csak megy vele az idő (tudom, hogy a RUP része, de akkor is kihagyható lenne). Másrészt viszont a tesztelésre jobban kitérhetnénk, és kicsit jobban specifikált teszt feladatot kaphatnánk, mert az nem kap véleményem szerint annyi odafigyelést mint amennyit érdemelne.

- Milyen feladatot ajánlanának a projektre?
  Esetleg felül-nézetes, Rouge-like RPG-t lehetne feladni még a jövőben, mert nem túl nehéz megírni egy olyat, de jól meg kell tervezni, hogy könnyen bővíthető legyen.

14.1.2. Tallér Bátor véleménye

- Mit tanultak a projektből konkrétan és általában?
  Csapatban dolgozni nehéz, de egyben rengeteg dolgot megkönnyít. Ha valakire nincs kiadva feladat akkor az a feladat nem lesz megcsinálva. A sok tervezés és dokumentáció után már könnyű azt leprogramozni.

- Mi volt a legnehezebb és a legkönnyebb?
  A legnehezebb része a tervezés volt, konkrétan azon belül az, amikor kitaláltunk valamit majd később rájöttünk, hogy ez valamilyen ok miatt mégse megfelelő, és ezért újra kellett tervezni egyes részeket.
  Továbbá nehézséget okozott összehangolni a projektet és az időnket.

- Összhangban állt-e az idő és a pontszám az elvégzendő feladatokkal?
  Nagyjából összhangban állt. A súlyozással viszont nem értek egyet, mert pont a prototípussal dolgoztunk

a legkevesebbet, míg a szkeletonnál sokat terveztünk és a grafikusnál meg sokat kódoltunk.

- Ha nem, akkor hol okozott ez nehézséget?
  Nem okozott nehézségeket.

- Milyen változtatási javaslatuk van?
  A dokumentáció sablonokat, illetve a tesztelési dokumentációkat aktualizálni kéne. A tesztelősben konkrétan ott áll az xls tetején, hogy 2009/2010, továbbá olyan kérdések voltak benne, amik nem tartoztak az aktuális feladathoz.

  Véleményem szerint egy agilis módszertan jobb lenne a RUP helyett. Néhányszor azt éreztem, hogy a dokumentáció egyes részei feleslegesek (pl. állapotdiagram).

  Hercules fejlesztése. Az az oldal azon kívül, hogy feltöltsük a beadandókat semmire se jó. Legalább az elért pontszámainkat feltüntethetné.

- Milyen feladatot ajánlanának a projektre?
  Konkrét feladat nem jut az eszembe, de az ideihez hasonló teljesen megfelel a célra, mert nem csak egy unalmas program készül a végén, hanem egy működő játék.

14.1.3. Török Attila véleménye

- Mit tanultak a projektből konkrétan és általában?
  Konkrétan a LaTeX nyelv alapjait, valamint git használatát, és annak segítségével munka megosztását DVCS-en keresztül, ugyanis eddig is használtam már verziókezelő rendszert, de nem gitet, és csak egyedül. Általában pedig azt, hogy egy nagyobb szoftver megtervezése és lefejlesztése, dokumentálása vajon hogyan is történhet(ett), még ha ez a projekt talán - László Zoltán tanár úr szavaival élve - „kutyaólépítés toronydaruval” volt is.

- Mi volt a legnehezebb és a legkönnyebb?
  A legnehezebbnek a temérdek dokumentáció és számtalan diagram legyártását tartottam egy még nem létező dologról. Valahogy jobban tetszenek az iteratív fejlesztési módszerek, amik használatakor legalább a dokumentációval együtt fejlődik a program, még ha nem is a végleges verziója, de egy működő, futó prototípusa annak, és nem ilyen lineáris a dokumentációtól a kód felé haladás. Főleg a korai szakaszban való túlzottan részletekbe bocsátkozás kényszerítése idegenkedett tőlem, ugyanis úgy vélem, hogy az utolsó függvényparaméterig lehetetlen úgy megtervezni egy szoftvert, még egy ilyen kicsit is, hogy az végül illeszkedjen az eleinte elképzeltekre. Ha pedig a realizálódás során úgyis változni fognak a részletek, akkor egyáltalán miért írnánk le róluk egy légből kapott elképzelést? Persze a gondos, de viszonylag nagy vonalakban dolgozó architektúrális tervezést én is elengedhetetlennek tartom már a kezdetektől fogva.
  Legkönnyebb nekem maga a lényegi kódolás volt, ugyanis abban már van gyakorlatom, és szívesen is csinálom, mert akkor érzem csak úgy, mintha valami működőt teremtenék.

- Összhangban állt-e az idő és a pontszám az elvégzendő feladatokkal?
  A tárgyon belül kiosztott pontokat nem követtem minden feladatrésznél egészen pontosan nyomon, ezért erre nem tudok megfontolt választ adni, de az egyes csapattagok részvételi aránya és lelkesedése hétről hétre erősen ingadozhat, ezért a teher tetszés szerint megosztható.

- Milyen változtatási javaslatuk van?
  Irreális követelménynek tartom a 6-os verziójú Java használatát, ugyanis ez már nem letölthető a gyártó hivatalos honlapjáról, ilyen régi JDK-t pedig szinte lehetetlen bárhol máshol találni, és JRE-t is csak kétséges forrásokból sikerült. Arra való tekintettel pedig, hogy nemrégiben megjelent a 8-as verzió, nem

tartanám kizártnak, hogy egy év múlva már a 7-es sem lesz egykönnyen beszerezhető a HSZK pincéjében féltve őrzött lyukkártyáktól különböző médiumról.
Apróbb kellemetlenség volt még számomra a rendszeres hétfő reggeli beadási határidő.

- Milyen feladatot ajánlanának a projektre?
  Nagyszerű ötletnek tartanék egy síkbeli Minecraft-szerű, „ásós-szerkesztős-építős” játékot.

- Egyéb észrevételek, kritikák

  Egy kicsit neheztelem azt is (igaz, legjobb tudomásom szerint ez már nem az Önök hatásköre, ezért csak mint véleményt, és nem mint kritikát írom), hogy ezért az 50-60 óra tényleges munkáért mindössze két kredit jár, ami - a legkisebb túlzás nélkül - mindenféle mókás kötelezően választható tárgyakból egyetlen délután alatt megszerezhető, némi szerencsével akár 4-es, vagy jobb érdemjeggyel. Nem magát az abszolút mértéket tartom inkorrektnek, tehát nem az a kérdés, hogy 1 kredit épp 5 vagy 150 órányi munkát jelképez elméletben, hanem abban látom a problémát, hogy a tárgyak között hatalmas szórás van az egy kreditért elvégzendő munkában, holott a későbbiekben már teljesen egyformán van kezelve, attól függetlenül, hogy milyen tantárgyból származik. Pedig szerintem az vitathatatlan, hogy a két órányi előadásra beüléshez (esetleg az arról való ellógáshoz) szükséges erőfeszítés meg sem közelíti a két órányi szekvenciadiagram-rajzoláshoz kellőt.

14.1.4. Szabó Antal véleménye

- Mit tanultak a projektből konkrétan és általában?
  Jó csapatban dolgozni, mert ha vannak olyan dolgok, amiket valaki nem szeret csinálni, jó esetben akkor is szét lehet úgy osztani a munkát, hogy mindenkinek elviselhető legyen.
  Konkrétan a Git és a LaTeX használatában szereztem sok tapasztalatot, ami később jól jöhet.

- Mi volt a legnehezebb és a legkönnyebb?
  Az egyik legnehezebb a program véleményem szerint túl részletes előre megtervezése volt, bármilyen kód írása nélkül. Nagyvonalakban természetesen meg kell előre tervezni, de a túl részletes tervezés szerintem felesleges, mert később úgy is lesznek benne változások, módosítások, amik már csak kód írásakor derülnek ki. A másik (ezzel összefüggően) a töménytelen mennyiségű dokumentáció írása volt, amiben sokszor lényegében ugyanazt kellett többször leírni akár egy fejezeten belül, akár a fejezetek között.
  A legkönnyebb (illetve legkellemesebb) rész a kód megírása volt, mert azt szeretem csinálni.

- Milyen változtatási javaslatuk van?
  A dokumentáció sablonok és az ellenőrzőlapok aktualizálása; a dokumentációban lehetne kevesebb redundáns rész, illetve kevesebb olyan, ahova lényegében rizsázni kell.

- Milyen feladatot ajánlanának a projektre?
  Konkrét ötletem nincs, de valamilyen játék írása azért jó, mert nem egy unalmas dolgot kell elkészíteni, hanem egy érdekes, a végén játszható programunk lesz.