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 csak kis mértékben 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 „logikai platformjáték” kategóriába sorolható, ami egy népszerű játéktípus, így viszonylag nagy a célközönsége.

2.1.3 Definíciók, rövidítések

architekturális kép: A szoftver belső felépítését szemléltető ábra.

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

BME: Budapesti Műszaki és Gazdaságtudományi 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.

Google Drive: Interneten keresztül elérhető felhőalapú tárhelyszolgáltatás, melyen keresztül verziókezelés is megvalósítható.

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

HSZK: A Budapesti Műszaki és Gazdaságtudományi Egyetemen belül működő Hallgatói Számítógép Központ rövidítése.

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

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

logikai platformjáték: olyan játéktípus, amiben a játékosnak különböző tárgyakat/eszközöket/erőforrásokat kell gyűjtenie az előrehaladáshoz, miközben ezek megszerzéséhez, illetve az akadályok leküzdéséhez a logikáját használja.

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.

Slack: Egy internetes felület, amin keresztül fejlesztőcsapatok tarthatják egymással a kapcsolatot.

StarUML 2: Egy UML diagramok készítését lehetővé tevő szoftver.

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

szoftver: Számítógépen futtatható program.

UML: Unified Modeling Language rövidítése, egy rendszermodellező 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.

WhiteStar UML: Egy UML diagramok készítését lehetővé tevő szoftver.

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

2.1.4 Hivatkozások

Szoftver Projekt Laboratórium https://www.iit.bme.hu/~projlab/

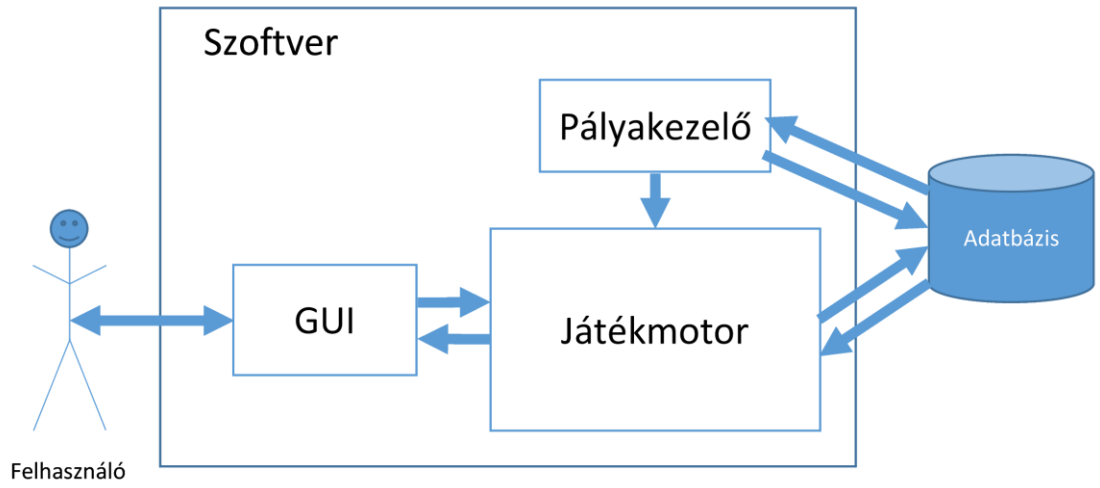
2.1.5 Összefoglalás

A dokumentum további részeiben található:

- Áttekintés, ami nagy vonalakban bemutatja a szoftvert
- A szoftverrel kapcsolatos követelmények leírása, később ezek figyelembe vételével 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



2.2.1.1 Architekturális kép

A játékszoftver alapja a játékmotor. A motor tartja számon, hogy a játékos a kétdimenziós játéktérben hol helyezkedik el, hány ZPM-et gyűjtött össze, mennyi ideje van még mielőtt véget érne a játék, éppen milyen portálok vannak nyitva, vagy hogy éppen cipel-e dobozt O'Neill ezredes. A motor gondoskodik az ajtók és mérlegek megfelelő viselkedéséről, nyilván tartja a szakadékok elhelyezkedését, tehát lényegében minden virtuális térbeli dolgot ez a rendszer menedzsel. Hozzáfér az adatbázishoz is, melyben a működéséhez szükséges információkat tárol, ill. olvas ki (például mentett játékok, vagy beállítások).

A felhasználóval a játékmotor a GUI-n keresztül kommunikál, mely a szoftver grafikus rendszere. A játékos a GUI-n keresztül küldhet parancsokat a motor felé, mely visszajelzésként adatokat küld a grafikus rendszerhez, hogy az kirajzolhassa a játék virtuális világát a képernyőre.

A játék pályakezelő rendszere foglalkozik a labirintusok (pályák, missziók, stb.) előkészítésével, tárolásával, ill. a játékmotornak is hozzáférést biztosít ezekhez. Hozzáfér az adatbázishoz, melyben elmentheti a labirintusokat, majd amikor ismét szükség lesz rájuk, innen tölti be őket a memóriába.

A játéknak nincsenek tervben többjátékos funkciói, így az hálózatot sem használ a jelenlegi felállás szerint, azonban szüksége van valamennyi helyre a számítógép háttértárán a játék grafikus elemei, hangfájljai, labirintusai, mentett beállításai, stb. számára.

2.2.2 Funkciók

A szoftver célja a felhasználó szórakoztatása. Egy felülnézetes, logikai, némileg ügyességi játékról van szó. A játékos feladata, hogy összegyűjtse a labirintusban elszórt, pályánként 20 ZPM-et. A játékos játékbeli materializációja O'Neill ezredes, akit a játékos a labirintus keretein belül mozgathat. A labirintus falain természetesen nem mehet át O'Neill, azok elmozdíthatatlan akadályként működnek.

A játékban a felhasználó győzedelmeskedik, ha összegyűjti a ZPM-eket és vereséget szenved, ha O'Neill-lel szakadékba esik, vagy ha kifut a megadott időlimitből. Vereség esetén Anubisz, a történet szerinti főgonosz sziluettje jelenik meg a képernyőn, illetve az a felirat, hogy "Vesztettél, Anubisz leigázta a Földet!".

A ZPM-ek összegyűjtése egyszerű feladat lenne, ha a játéktérként szolgáló labirintus nyílt területein helyezkednének el, ezért azok elzárt helyeken is fellelhetőek. Ezeket az elzárt helyeket ajtók választják el a játékostól, melyeket úgy lehet kinyitni, hogy a nekik megfelelő, nem mozgatható mérlegre a felhasználó az ezredessel egy dobozt helyez el. Ezek a mérlegek az ezredes súlyától is kinyitják az ajtókat, viszont azzal, hogy megszűnik a nyomás rajtuk attól, hogy a játékos O'Neill-lal leáll róluk, újra bezárulnak az azokhoz tartozó ajtók. Az ajtók a mérlegüktől, a mérlegek meg az azoktól legközelebbi doboztól általában távol lelhetőek fel. Ezeknek a távolságoknak az áthidalására, illetve a logikai paletta színesítésére O'Neillt felruháztuk egy olyan eszközzel, mely csillagkapukat, azaz féregjárat-végződéseket lövell ki magából.

Ezek a csillagkapuk a nehezítés kedvéért nem minden felületre felhelyezhetőek (például ajtókról lepattan), viszont ha két végződés el van helyezve a pályán, a játékos átsétálhat rajtuk a nagyobb távolságokat vagy bonyolult járatokat áthidalva. Fontos, hogy a két csillagkapu sárga vagy kék színű lehet, illetve színenként csak egy létezhet a pálya falfelületein. Ha egy olyan színű csillagkaput lő ki a felhasználó érvényes falfelületre, amilyen színűből egy fellelhető már, akkor a régi járatvégződés megszűnik létezni. Fontos, hogy a játékos a csillagkapuvetővel nem lőhet keresztül féregjáratokon. A korábban említett szakadék fölött is természetesen átmennek a csillagkapu-lövedékek, hogy az azok által elzárt részekre is eljuthasson a játékos O'Neill-lal.

A dobozok használatára is vonatkoznak szabályok. A játékos csak úgy vehet fel dobozt az ezredessel, ha közvetlenül előtte áll. A felvett dobozokat a játékos le tudja tenni. A dobozok átvihetőek a csillagkapuk által meghatározott féregjáratokon, viszont O'Neill ezredeshez hasonlóan, azok is megsemmisülnek a szakadékba zuhanva.

Miután O'Neill összegyűjtötte az adott pályán lévő 20 ZPM-et, a labirintus közepén megjelenik egy intergalaktikus szingularitás-mátrix, melybe belépve a következő pályára kerül. Természetesen a ZPM-ek összegyűjtése után is fenn áll az időlimit, így ha a játékos nem irányítja időben az ezredest a szingularitás-mátrixba, az bezárul és a játék negatív befejezéssel ér véget.

2.2.3 Felhasználók

A felhasználóknak a szoftver használatához az irányításhoz szükséges tevékenységek ismeretén túl -- amely könnyen és gyorsan elsajátítható -- nincs szükségük bármilyen speciális előismeretre.

2.2.4 Korlátozások

Az elkészítendő szoftver futtatható kell, hogy legyen a Hallgatói Számítógép Központ gépein, ezért nem hivatkozhat olyan külső csomagra, amely nem található meg az Oracle JDK-ban.

2.2.5 Feltételezések, kapcsolatok

Feladatkiírás https://www.iit.bme.hu/~projlab/

2.3 Követelmények

2.3.1 Funkcionális követelmények

Azonosító	Leírás	Ellenőrzés	Prioritás	Forrás	Use-case	Komment
2.3.1.1	A felhasználó tudja előre, hátra, illetve jobbra és balra mozgatni O'Neillt.	bemutatás	alapvető	megrendelő	Mozgás	
2.3.1.2	A felhasználó a dobozokat fel tudja venni, amennyiben azok előtte vannak, és maga elé le tudja rakni azokat.	bemutatás	alapvető	megrendelő	Doboz mozgatása	
2.3.1.3	A felhasználó be tudja gyűjteni a ZPM-eket.	bemutatás	alapvető	megrendelő	ZPM gyűjtés	
2.3.1.4	Az ajtókat lehetősége van kinyitni a felhasználónak.	bemutatás	alapvető	megrendelő	Ajtó nyitás	
2.3.1.5	A mérlegek érzékelik, ha valamilyen súly kerül rájuk.	bemutatás	alapvető	megrendelő		
2.3.1.6	A felhasználó tud csillagkaput lőni, melyből egyszerre maximum kettő lehet a pályán, színenként egy-egy.	bemutatás	alapvető	megrendelő	Kék CSK lövés/ Sárga CSK lövés	
2.3.1.7	A felhasználó kiválaszthatja, hogy sárga vagy kék csillagkaput lő.	bemutatás	fontos	megrendelő	Kék CSK lövés/ Sárga CSK lövés	
2.3.1.8	A játék véget ér, ha a felhasználó begyűjti az összes ZPM-et.	bemutatás	fontos	megrendelő	Játék vége	
2.3.1.9	A játék véget ér, ha O'Neill lezuhan és meghal.	bemutatás	fontos	megrendelő	Játék vége	
2.3.1.10	A játék véget ér, ha a felhasználó túllépi az időkeretet.	bemutatás	opcionális	csapat	Játék vége	

2.3.2 Erőforrásokkal kapcsolatos követelmények

Azonosító	Leírás	Ellenőrzés	Prioritás	Forrás	Komment
2.3.2.1	Monitor	nincs	alapvető	csapat	
2.3.2.2	Egér	nincs	alapvető	csapat	
2.3.2.3	Billentyűzet	nincs	alapvető	csapat	
2.3.2.4	GitHub	nincs	alapvető	csapat	git szolgáltató
2.3.2.5	Google Docs	nincs	alapvető	csapat	csoportmunkára alkalmas szövegszerkesztő
2.3.2.6	Office 365	nincs	alapvető	csapat	csoportmunkára alkalmas szövegszerkesztő
2.3.2.7	Slack	nincs	alapvető	csapat	fejlett üzenetküldő platform
2.3.2.8	Eclipse	nincs	opcionális	csapat	Java fejlesztőkörnyezet
2.3.2.9	IntelliJ IDEA	nincs	opcionális	csapat	Java fejlesztőkörnyezet
2.3.2.10	vim	nincs	opcionális	csapat	szövegszerkesztő
2.3.2.11	Enterprise Architect	nincs	opcionális	csapat	UML modellezés
2.3.2.12	WhiteStar UML	nincs	opcionális	csapat	UML modellezés
2.3.2.13	StarUML2	nincs	opcionális	csapat	UML modellezés
2.3.2.14	HSZK-s számítógép	bemutatás	alapvető	megrendelő	
2.3.2.15	Oracle JDK8	bemutatás	alapvető	megrendelő	

2.3.3 Átadással kapcsolatos követelmények

Azonosító	Leírás	Ellenőrzés	Prioritás	Forrás	Komment
2.3.3.1	Szkeleton átadás	bemutatás	alapvető	megrendelő	márc. 21. 18:00
2.3.3.2	Protó átadás	bemutatás	alapvető	megrendelő	ápr. 18. 18:00
2.3.3.3	Grafikus átadás (teljes program átadása)	bemutatás	alapvető	megrendelő	máj. 9. 18:00
2.3.3.4	A programnak működnie kell a BME HSZK számítógépein	bemutatás	alapvető	megrendelő	-

2.3.4 Egyéb nem funkcionális követelmények

Egyéb nem funkcionális követelmények nincsenek.

2.4 Lényeges use-case-ek

2.4.1 Use-case leírások

Use-case neve	Doboz mozgatása
Rövid leírás	A dobozt mozgatja
Aktorok	Játékos
Forgatókönyv	Ha a doboz az ezredes előtt van akkor fel tudja venni, és amikor akarja akkor maga elé le is tudja tenni.

Use-case neve	ZPM gyűjtés
Rövid leírás	Begyűjti az adott ZPM-et
Aktorok	Játékos
Forgatókönyv	Ha a ZPM az ezredes előtt van akkor begyűjti.

Use-case neve	Kék CSK lövés
Rövid leírás	Kilő egy kék csillagkaput
Aktorok	Játékos
Forgatókönyv	A kiválasztott helyre egyenes vonalban kilő egy kék csillagkaput.

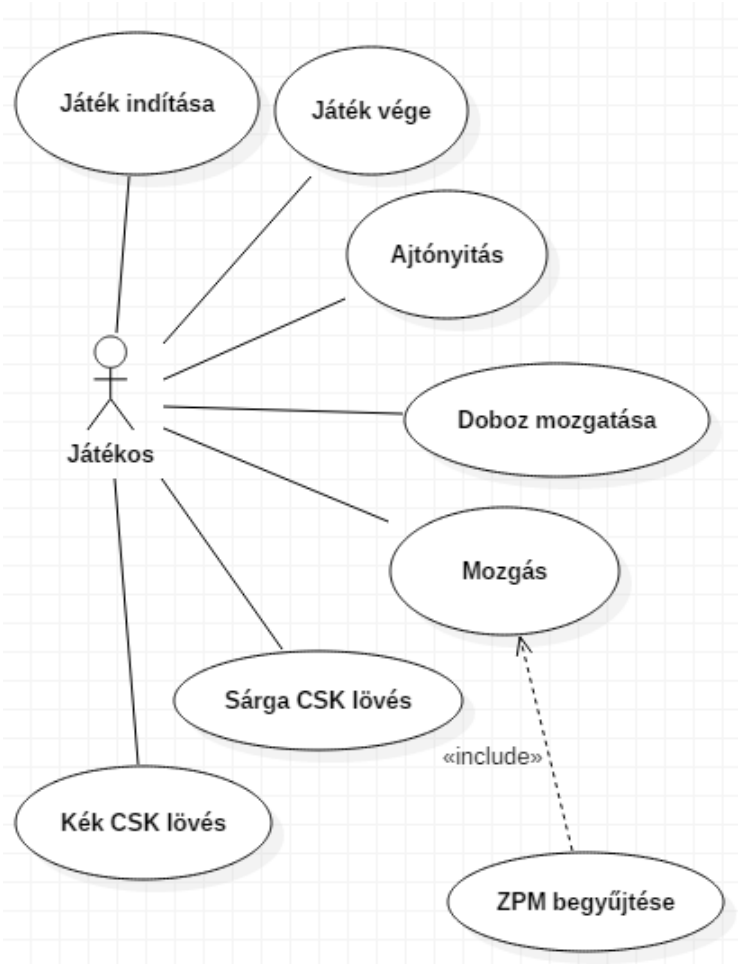
Use-case neve	Sárga CSK lövés
Rövid leírás	Kilő egy sárga csillagkaput.
Aktorok	Játékos
Forgatókönyv	A kiválasztott helyre egyenes vonalban kilő egy sárga csillagkaput.

Use-case neve	Irányítás
Rövid leírás	O'Neill mozgatása
Aktorok	Játékos
Forgatókönyv	A játékos billentyűk segítségével tudja mozgatni O'Neill ezredest négy irányban.

Use-case neve	Ajtónyitás
Rövid leírás	Kinyílik az ajtó
Aktorok	Játékos
Forgatókönyv	Ha az ajtóhoz tartozó mérlegre súly kerül akkor az adott ajtó kinyílik.

Use-case neve	Játék vége
Rövid leírás	A játék véget ér
Aktorok	Játékos
Forgatókönyv	A játék véget ér akkor, ha: 1) a játékos összegyűjtötte az összes ZPM-et. 2) a játékos leesett a szakadékba és meghalt. 3) a játékos túllépte az időkeretet.

2.4.2 Use-case diagram



2.5 Szótár

ajtó: A labirintus falain biztosítanak átjárást, és a játék kezdetén zárva vannak. Az ezredes nem tud rajtuk átjárót nyitni. Egy kapcsolóval (mérleg) lehet őket kinyitni.

csillagkaput lőni: O'Neill ezredes fegyvere egy különleges darab, mely átjárót képes nyitni a falakon, mely egy szintén a falra lőtt kijárattal kapcsolódik össze, ezen az átjárón az ezredes bármilyen irányban keresztülmehet.

doboz: A játékban nehezékként funkcionál, melyet O'Neill ezredes fel tud venni, le tud rakni, illetve mozgatni is tud. Szerepe az, hogy egy mérlegre helyezzék.

GUI: A szoftver azon rendszere, ami megjeleníti a játékvilágot a felhasználó számára, illetve közvetíti parancsait a játékmotor felé. Más néven grafikus rendszer.

játék elvesztése: A játék egy lehetséges végződése, akkor következik be, ha O'Neill ezredes meghal, vagy lejárt az idő.

játék megnyerése: A játék egy lehetséges végződése, akkor következik be, ha O'Neill ezredes minden ZPM-et összegyűjtött a labirintusban.

labirintus: A játék pályái, missziói. Szövegkörnyezet függvényében jelentheti a konkrét memóriában/háttértáron tárolt adatokat, vagy absztrakt módon a játékvilág egyes helyszíneit.

meghal: Ha O'Neill ezredes beleesik egy szakadékba, akkor meghal. Ilyenkor a játék véget ér, a játékos vesztett.

mérleg: Lényegében egy kapcsoló. Dobozt lehet ráhelyezni, vagy rá lehet állni O'Neill ezredessel, ekkor a mérleg érzékeli a súlyt, és kinyit egy ajtót a labirintusban.

O'Neill ezredes: A játék főhőse, őt irányíthatja a játékos/felhasználó kalandjai során.

repository: Egy központi adattároló hely.

szakadék: A labirintusban található árkok, melyekbe ha O'Neill ezredes belelép, meghal.

ZPM: Zero Point Module, magyarul Zéró Pont Modul, a szoftverben található egység, melynek összegyűjtése a pálya teljesítésének feltétele.

2.6 Projekt terv

2.6.1 Csapat

A csapat 6 fős. A feladatokat próbáljuk a tagok képességei és preferenciái szerint szétosztani, tehát van, aki sokkal szebb dokumentumokat tud szerkeszteni, van aki pedig több szoftverfejlesztési tapasztalattal rendelkezik.

Név	**Felelősség**
Bokros Bálint	csapatkapitány, kódolás, dokumentáció
Hegedüs Fanni	kódolás, dokumentáció
Jáhn Erik	kódolás, documentáció
Siket Melinda Tekla	kódolás, dokumentáció
Tóth Kristóf	kódolás, dokumentáció
Varga Péter	kódolás, dokumentáció

2.6.2 Kommunikáció

Verziókezelés: A csapat a Git verziókezelő szoftvert használja, melyhez a központi repositoryt a GitHub biztosítja. Ez jó, hiszen fontos, hogy a csapat minden tagja hozzáférjen a kód legfrissebb változatához, illetve valahogyan meg kell oldani, hogy egyszerre többen is dolgozhassanak a projekten, amire a Git az egyik legjobb megoldás. Ezen kívül a változásokat is könnyen nyomon követhetővé teszi.

Facebook: A csapat a Facebook messenger szolgáltatását használja napközben az egymással való kapcsolattartásra.

Slack: A tagok a Slack nevű online kollaborációs eszközt használják a "hivatalos" kommunikáció nagy részének lebonyolításához, a feladatok koordinálásához, meetingek szervezéséhez és mindenféle egyébhez.

Meetingek: A csapat heti szinten egy vagy több nagy közös megbeszélést tart a munka szétosztásához, a haladás ellenőrzése, ill. közös döntések meghozásának céljából. Ez történhet személyesen, vagy valamilyen VoIP szoftver (pl. Skype) segítségével.

2.6.3 Használt szoftverek

Verziókezelés: Git & GitHub, fentebb részletesebben kifejtve

Fejlesztőkörnyezet: A csapattagok a preferenciájuknak megfelelő fejlesztőkörnyezetet használják, ez a legtöbb esetben az Eclipse, IntelliJ IDEA, de akár VIM is lehet.

Dokumentáció: A dokumentáció írására a Google Drive ill. az Office 365 szolgáltatásokat használjuk.

2.6.4 Határidők

	Dátum	Feladat
2.6.4.1	febr. 21.	24 h - csapatok regisztrációja
2.6.4.2	febr. 29.	Követelmény, projekt, funkcionalitás - beadás
2.6.4.3	márc. 7.	Analízis modell kidolgozása 1. - beadás
2.6.4.4	márc. 14.	Analízis modell kidolgozása 2. - elektronikus feltöltés. A nyomtatott változatot a márc. 16-án kell a laborvezetőnek odaadni.
2.6.4.5	márc. 21.	Szkeleton tervezése - beadás
2.6.4.6	márc. 29.	Szkeleton - beadás
2.6.4.7	ápr. 4.	Prototípus koncepciója - beadás
2.6.4.8	ápr. 11.	Részletes tervek - beadás
2.6.4.9	ápr. 18.	
2.6.4.10	ápr. 25.	Prototípus - beadás
2.6.4.11	máj. 2.	Grafikus felület specifikációja - beadás
2.6.4.12	máj. 9.	
2.6.4.13	máj. 17.	Grafikus változat - beadás
2.6.4.14	máj. 20.	Összefoglalás - beadás

A feladat megoldása 3 lépcsőben történik.

Szkeleton: Célja annak bizonyítása, hogy a szoftverben együttműködő objektumok valóban a megoldandó feladatot modellezik. Ez a szoftver egy kezdetleges verziója, melynek minden a végső verzióban is szereplő objektum a részét képzi. Ezen objektumok még nincsenek implementálva, csak az interfészük definiált. A program lefuttatása esetén minden objektum a konzolra kiírja a saját nevét (ez az egyszerűség jegyében történik a konzolra), ill. meghívja azon metódusokat, melyek szükségesek a feladata elvégzéséhez. Feltételes elágazásoknál valamilyen módon láttatni kell, hogy pontosan mi is történik. Ennek a verziónak alkalmasnak kell lennie szekvenciadiagramok ellenőrzésére.

Prototípus: A grafikus interfészt leszámítva teljes szoftver - célja demonstrálni, hogy a program készen van, a specifikációnak megfelelően működik. A megjelenítést szolgáló részek kivételével

az objektumok valamennyi metódusa implementálva van és azok véglegesek (leszámítva persze az esetleges későbbi javításokat a szoftver életciklusa során). Megjelenítésre a konzolt használja. Lehetőséget teremt a rendszer tesztelésére.

Teljes (grafikus) változat: A prototípustól csak annyiban különbözik, hogy már van grafikus felülete, ami egy játékszoftver esetén nyilván nagy változás, de a belső működés nem változik, csak a felhasználóval való kommunikáció minősége/módja.

2.7 Napló

Kezdet	Időtartam	Résztvevők	Leírás
2016. 02. 26. 15:00	2 óra	Bokros Hegedüs Jáhn Siket Tóth Varga	Értekezlet. A feladat értelmezése, az alapvető követelmények közös összegyűjtése, a feladatok szétosztása
2016. 02. 27. 23:20	1 óra	Bokros	Tevékenység. Bokros elkészíti a dokumentum rá kirótt részeit (2.2.3, 2.2.4, 2.3.2)
2016. 02. 28. 11:20	40 perc	Jáhn	Tevékenység. Jáhn elkészíti a dokumentum rá kirótt részét (2.2.2)
2016. 02. 28. 11:30	1,4 óra	Tóth	Tevékenység. Tóth elkészíti a dokumentum rá kirótt részeit (2.2.1, 2.6)
2016. 02. 28. 12:15	1,5 óra	Varga	Tevékenység. Varga elkészíti a dokumentum rá kirótt részeit (2.1, 2.3.3, 2.3.4)
2016. 02. 28. 16:00	50 perc	Siket	Tevékenység. Siket elkészíti a dokumentum rá kirótt részét (2.3.1)
2016. 02. 28. 16:30	1,5 óra	Hegedüs	Tevékenység. Hegedüs elkészíti a dokumentum rá kirótt részét (2.4)
2016. 02. 28. 17:00	50 perc	Bokros	Tevékenység. Formázás ellenőrzése, javítása, tördelés, fedlap elkészítése.

3 Analízis modell kidolgozása

3.1 Objektum katalógus

3.1.1 Box

Ez az osztály a játékvilágban található dobozoknak feleltethető meg. A játékos O'Neillt irányítva felveheti, illetve lerakhatja őket a megfelelő mezőkre. Van súlyuk, a játék mechanikájának a szempontjából elsősorban ajtónyitásra alkalmasak, így ha O'Neill rátesz egy dobozt egy mérlegre (*Scale*), akkor a hozzá tartozó kapu a *Scale* osztálynál leírtak értelmében kinyílik.

3.1.2 Direction

A többi osztály ezt az enumot használja az irányok tárolására (*NORTH*, *SOUTH*, *EAST*, *WEST*).

3.1.3 Dungeon

Felelőssége a játéktér (pálya) felépítése és lebontása: ez gyakorlatilag abból áll, hogy a játék kezdetekor beállítgatja a játék összes mezőjén (*Tile*), hogy milyen másik mezőkkel szomszédos. A játék során számon tartja, hogy O'Neill hány *ZPM*-et gyűjtött eddig össze, majd amikor annak eljön az ideje, lezárja a játékot.

3.1.4 Field

Olyan mezőket reprezentáló osztály, melyekre O'Neill rá tud lépni. A játék kezdetén is egy ilyen mezőn találja magát a játékos. Egyszerre egy darab valamilyen felvehető objektum (*Pickable*) lehet egy *Field* típusú mezőn, ez lehet például *ZPM* vagy doboz (*Box*). Ha egy *Field* mezőn *ZPM* van, akkor ha O'Neill rálép felveszi a *ZPM*-et, azonban ha egy doboz van rajta, akkor oda nem lehet lépni, csak akkor ha a dobozt felveszi, majd odébb viszi a játékos. Ismeri a saját szomszédságát mind a négy irányban.

3.1.5 FlowOfTime

Az idő előrehaladtatását végző osztály. A játék során az idő halad, ami fontos, mivel egy bizonyos idő elteltével O'Neill sajnálatos módon meghal, így a játékos rá van kényszerítve, hogy gyorsan szedje össze a *ZPM* modulokat.

3.1.6 Game

A játékmotor alapja, kezeli a játékos által végrehajtott eseményeket (billentyűlenyomások, stb.). Magába foglalja a játék logikáját és összeköti a többi osztály interakcióját.

3.1.7 Gap

A pályán található szakadékokat modellező osztály. Ha a játékos O'Neillt irányítva rálép egy ilyen típusú mezőre, a játék rögtön véget ér, ugyanis O'Neill lezuhan és szörnyet hal (természetesen attól függetlenül, hogy éppen visz-e valamit). Nem lehet rajta *ZPM* és ha a játékos dobozt (*Box*) rakna rá, akkor az leesik, azaz megsemmisül.

3.1.8 Gate

A játékban lévő kapukat reprezentáló osztály. Minden egyes *Gate*-hez tartozik egy darab *Scale* objektum, aminek segítségével a kapu kinyitható. Amennyiben egy kapu zárva van, azon O'Neill nem tud semmilyen módon átmenni, ehhez ki kell valahogyan nyitnia azt. Nem lehet rá

csillagkaput lőni. A játék kezdetekor a kapuk lehetnek csukott, vagy nyitott állapotban is, attól függően, hogy a hozzájuk tartozó mérlegen alapból van-e valamilyen súly.

3.1.9 ONeill

A játékban irányítható hőst modellező osztály. O'Neill egy ezredes és mint felelősségteljes tiszt, mindig tudja, hogy éppen hol van. Hosszú évek alatt megunta, hogy a falak állandóan az útjába állnak: kifejlesztette a csillagkapu kilövés képességét. Ez kék, illetve sárga csillagkapuk formájában lehetséges, bővebb leírás erről a mechanizmusról a *Stargate* osztálynál található. A játékos a karaktert 4 irányba mozgathatja, fel, le, jobbra és balra. Ugrálni persze tud, de mivel az fárasztó és nem elég elegáns hozzá, ezért olyan, mintha nem is tudna. Képes a pályákon szétszórt *ZPM*-eket felvenni, amiből ha mindent megszerezte, megnyeri a játékot. A játéktérben található dobozokat fel tudja venni, ha azok pont előtte vannak, majd le is tudja őket tenni valahova maga elé. Amennyiben egy bizonyos időn belül nem tudja összeszedni az összes a pályán elhelyezkedő *ZPM*-et, meghal, így a játék véget ér.

3.1.10 Scale

A kapuk kinyitásához használt mérlegek osztálya. Minden ilyen mérleghez tartozik egy kapu (*Gate*) a pályán, ami nyitható vele. Egy mérleg akkor nyitja ki a hozzá tartozó kaput, hogyha a játékos súlyt helyez rá. Ez történhet például úgy, hogy O'Neill karaktere rááll a mérlegre, vagy mondjuk rátesz egy dobozt (*Box*). Lehet a mérlegeken *ZPM* modul, amit a játékos hasonló módon vehet fel, mint normál esetben.

3.1.11 Stargate

Az O'Neill karaktere által kilőhető csillagkapukat modellező osztályok. Egy kilőtt csillagkapu lehet kék vagy sárga. A kilövésük egy olyan megkötéssel történik, hogy egyszerre egy időben csak egy darab létezhet a játékban az azonos színű csillagkapukból, tehát például létezhet egyszerre egy kék és sárga, de ugyanazon színből több már nem. Ha a pályán egy időben létezik egy kék és egy sárga csillagkapu, akkor a téridő átszakad köztük: féreglyuk keletkezik. Amennyiben O'Neill belemegy egy valamilyen színű csillagkapuba, az átteleportálja őt a másik színű csillagkapuhoz, amennyiben az létezik. Ha éppen ilyen csillagkapu nem létezik, akkor nem lehet belépni a csillagkapuba és semmi nem történik, mivel nem nyílt féreglyuk. Az ilyen féreglyukak hasznos tulajdonsága, hogy O'Neill akkor is át tud menni rajtuk, hogyha éppen cipel valamit. Ugyan a *Stargate* osztály absztrakt mégis ezt dokumentálom a katalógusban, mivel a *BlueStargate* és *YellowStargate* osztályok csak a színükben térnek el egymástól.

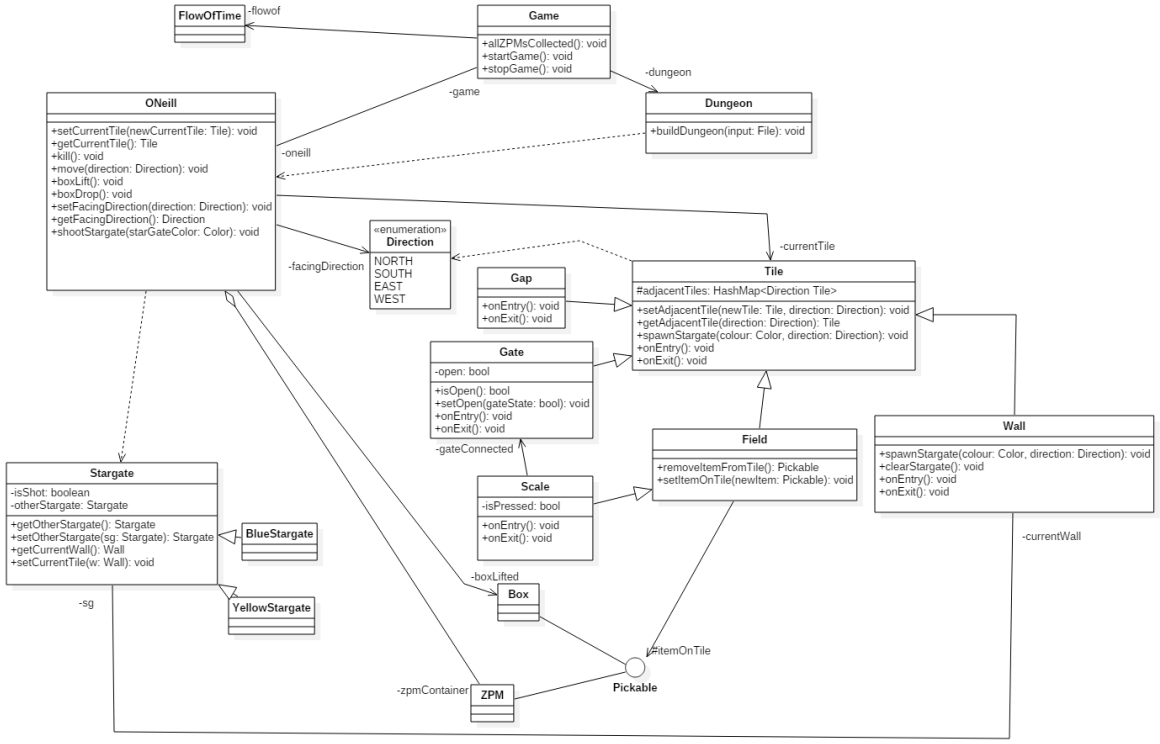
3.1.12 Wall

A pályán található falakat reprezentáló osztály. A falakon O'Neill nem tud átmenni, de az évek alatt kifejlesztette a csillagkapu lövés képességét: a falakra tud csillagkapukat lőni. A falakon nem lehet *ZPM* vagy doboz (*Box*). A pályák szélén tipikusan ilyen falak vannak, hogy valamennyire korlátozzák a játékos mozgását. A *Field* osztályhoz hasonlatosan ismeri a szomszédságát négy irányban.

3.1.13 ZPM

A játékban található összegyűjtendő tárgyak osztálya. A játék megnyerésének feltétele az, hogy O'Neill összegyűjtse az összes, pályán szétszórt *ZPM*-et, hogy az atlantiszi űrhajó energiaellátását meg tudja oldani. Ez úgy történik, hogy a játékos O'Neill karakterét egy olyan mezőre irányítja, ahol *ZPM* lebeg, ekkor O'Neill felveszi azt. Ugyan *ZPM* modulok lehetnek mérlegeken (*Scale*), a *ZPM* természeténél fogva lebeg, így nem nyomja le a mérleget, tehát a hozzá tartozó kaput (*Gate*) sem nyitja ki.

3.2 Statikus struktúra diagramok



3.3 Osztályok leírása

3.3.1 Box

- **Felelősség**

Az O'Neill által felvehető és letehető dobozokat megvalósító osztály.

- **Ősosztályok**

Csak az *Object*-ből származik le.
Legősebb osztály: Object

- **Interfészek**

A *Pickable* interfészt valósítja meg.

- **Attribútumok**

Nincsenek attribútumai.

- **Metódusok**

3.3.2 BlueStargate

- **Felelősség**

A kék csillagkaput megvalósító osztály. Singleton, azaz egy példánya létezik csak, és ez mozog a pályán. Az implementáció nagyrésze a *YellowStargate*-tel közös *Stargate* ősosztályban található.

- **Ősosztályok**

Legősebb osztály: Stargate

- **Interfészek**

- **Attribútumok**
    - Color: a kapu színe.

3.3.3 Direction

- **Felelősség**

A négy irányt (*NORTH, SOUTH, EAST, WEST*) tároló enum, a többi osztály ezt használja a szomszédos mezők azonosítására.

3.3.4 Dungeon

- **Felelősség**

Az osztály felelőssége a pálya felépítése és O'Neill elhelyezése azon a játék kezdetekor, valamint a játék lezárása, amikor O'Neill sikeresen összegyűjtötte az összes *ZPM*-et.

- **Ősosztályok**

Az osztály egyetlen őse az *Object*.

- **Interfészek**

Az osztály nem valósít meg interfészt.

- **Attribútumok**

- **Metódusok**
    - **void ZPMCollected():** Akkor kerül meghívásra, amikor O'Neill az egyik *ZPM*-et begyűjtötte. Ez a metódus zárja le a játékot győzelem esetén.
    - **void buildDungeon(File input):** A bemeneti fájlban tárolt információk alapján felépíti a pályát, és elhelyezi O'Neillt rajta.

3.3.5 Field

- **Felelősség**

A pályán belüli bejárható mezőket megvalósító osztály. O'Neill ezekre a mezőkre szabadon tud lépni, illetve a játékot is egy ilyen mezőn kezdi. Ezeken továbbá felvehető tárgyak is vannak, mint például a *ZPM* vagy a *Box*. Egy ilyen mezőn maximum egy felvehető tárgy lehet.

- **Ősosztályok**

A *Field* a *Tile* absztrakt osztály leszármazottja, ami megvalósítja annak nem implementált függvényeit.
Legősebb osztály: Tile

- **Interfészek**

Ez az osztály nem valósít meg interface-t.

- **Attribútumok**
    - Pickable itemOnTile

- **Metódusok**
    - **void SpawnStargate(Color colour, Direction direction):** A megadott irányban lévő szomszédos *Tile*-ra meghívja arra ugyanezt a függvényt a megadott színnel és iránnyal.
    - **Pickable removeItemFromTile():** Átadja a függvény hívójának az *itemOnTile* attribútumának referenciáját és törli azt.
    - **void setItemOnTile(Pickable newItem):** Ha nincs *Pickable* típusú objektuma már a mezőnek, a paraméterül kapottat teszi meg *itemOnTile*-nak.
    - **void onEntry:** O'Neill vagy egy *Box* rákerül a mezőre.
    - **void onExit:** O'Neill vagy egy *Box* elhagyja a mezőt.

3.3.6 FlowOfTime

- **Felelősség**

A játék kezdetekor elkezdi az eltelt idő mérését, és egy idő után (mikor már elviselhetetlen az ezredes számára a sugárterhelés) meghívja O'Neill *kill()* metódusát.

- **Ősosztályok**

A *Timer*-ból származik le.
Legősebb osztály: *Timer*

- **Interfészek**

Ez az osztály nem valósít meg semmilyen interfészt.

- **Attribútumok**

Nincsenek attribútomok.

- **Metódusok**
    - **void start():** elindítja az időmérőt.
    - **void stop():** leállítja az időmérőt.

3.3.7 Game

- **Felelősség**

Az események kezelése a játékban, illetve az összes többi osztály összekötése. A játékossal való kommunikáció.

- **Ősosztályok**

Az osztály egyetlen őse az *Object*.
Legősebb osztály: Object

Interfészek
Ez az osztály nem valósít meg interfészt

Attribútumok
- **Dungeon dungeon:** A pálya
- **ONeill oneill:** O'Neill ezredes
- **FlowOfTime flowoftime:** Az időzítő, ami számolja az idő múlását

Metódusok
- **void startGame():** elindítja a játékot
- **void stopGame():** leállítja a játékot
- **void allZPMsCollected():** értesíti a *Game* osztályt, hogy a játék véget ért, mert megvan a *ZPM* (O'Neill hívja)

3.3.8 Gap

- **Felelősség**

A pályán belüli szakadékokat megvalósító osztály. Amennyiben ilyen objektumra lép O'Neill ezredes, szörnyet hal. Ha a játékos egy *Box*-ot ilyen objektumra dobna, az megsemmisül.

- **Ősosztályok**

A *Gap* a *Tile* absztrakt osztály leszármazottja, ami megvalósítja annak nem implementált függvényeit.
Legősebb osztály: Tile

- **Interfészek**

Ez az osztály nem valósít meg interface-t.

- **Attribútumok**

- **Metódusok**
  - **void SpawnStargate(Color colour, Direction direction):** A megadott irányban lévő szomszédos *Tile*-ra meghívja arra ugyanezt a függvényt a megadott színnel és iránnyal.
  - **void onEntry:** Meghívja O'Neill vagy a *Box* destroy függvényét.
  - **void onExit:** Üres függvénytörzs.

3.3.9 Gate

- **Felelősség**

A pályán belüli kinyitható ajtókat megfeleltető osztály. Az ajtó lehet nyitott vagy zárt állapotban. Minden *Gate* objektumot ismer egy *Scale* objektum, amire súly helyezésével kinyílik a *Gate* objektum.

- **Ősosztályok**

A *Gate* a *Tile* absztrakt osztály leszármazottja, ami megvalósítja annak nem implementált függvényeit.
Legősebb osztály: Tile

- **Interfészek**

Ez az osztály nem valósít meg interface-t.

- **Attribútumok**
    - bool open

- **Metódusok**
    - **void SpawnStargate(Color colour, Direction direction):** Ha az ajtó nyitva van és a direction irányában *Field* található, meghívja arra ugyanezt a függvényt a megadott színnel és iránnyal. Ha az ajtó zárva van, visszatér a függvényből.
    - **bool isOpen():** Az ajtó nyitottságát lekérdező függvény, az *open* attribútum értékét adja vissza. True esetén nyitva, false esetén zárva van az ajtó.
    - **void setOpen(bool gateState):** Az ajtó nyitottságát beállító függvény. A paraméterként kapott értékre állítja az *open* attribútum értékét. True esetén nyitva, false esetén zárva van az ajtó.
    - **void onEntry:** Ha az *open* értéke true, egy *Field*-ként viselkedve O'Neill vagy egy *Box* rákerülhet az objektumra. Ha az *open* értéke false, egy *Wall*-ként viselkedve visszatér.
    - **void onExit:** Ha az *open* értéke true, egy Field-ként viselkedve O'Neill vagy egy *Box* elhagyja az objektumot. Ha az *open* értéke false, egy *Wall*-ként viselkedve visszatér.

3.3.10 ONeill

- **Felelősség**

Az ezredest megvalósító osztály. Képes észak, dél, kelet nyugati irányban mozogni a *Tile*-ok közt, *ZPM*-eket gyűjteni, illetve dobozokat felemelni, mozgatni, majd letenni.

- **Ősosztályok**

Ez az osztály nem örököl és nem is örökít, így csak az *Object* osztályból származik.
Legősebb osztály: Object

- **Interfészek**

Ez az osztály nem valósít meg interface-t.

- **Attribútumok**
    - Tile currentTile
    - Game game
    - ArrayList<ZPM> zpmContainer
    - Direction facingDirection
    - Box boxLifted

- **Metódusok**
    - **void setCurrentTile(Tile newCurrentTile):** Beállítja, hogy O'Neill melyik pályadarabon tartózkodik
    - **Tile getCurrentTile():** Visszaadja, hogy O'Neill melyik pályadarabon tartózkodik.
    - **void shootStargate(Color color):** Egy paraméterként kapott színű csillagkaput *(Stargate)* lő az aktuális irányba, ami az első útjába eső felületen nyílik meg.
    - **void kill():** O'Neill ezredes meghal és ezzel vége a játéknak.

- **void move(Direction direction):** O'Neillt a megadott irányban lévő pályadarabra mozgató függvény.
- **void boxLift():** O'Neill megpróbál felvenni abból az irányból dobozt, amerre néz. Ha van doboz az adott pályadarabon, a *boxLifted* attribútumának teszi, ha nincs, nem történik semmi.
- **void boxDrop():** O'Neill lerakja a dobozát. Ha van nála doboz, arra a pályadarabra rakja, amerre néz, ha nincs, nem történik semmi.
- **void setFacingDirection(Direction direction):** Beállítja, merre nézzen O'Neill.
- **Direction getFacingDirection():** Visszaadja, merre néz O'Neill.

3.3.11 Pickable

Felelősség

Közös interfésze a *Box* és a *ZPM* osztályoknak.

Ősosztályok

Legősebb osztály : Object.

3.3.12 Scale

- ## Felelősség

A pályán lévő mérlegeket megvalósító osztály. Ha súly kerül rá, kinyitja a hozzá tartozó ajtót *(Gate).*

- ## Ősosztályok

A *Scale* a *Field* osztály leszármazottja.
Legősebb osztály: Tile, Ősosztály1: Field

- ## Interfészek

Ez az osztály nem valósít meg interface-t.

- ## Attribútumok
    - Gate gateConnected
    - bool isPressed

- ## Metódusok
    - **void SpawnStargate(Color colour, Direction direction):** A megadott irányban lévő szomszédos *Tile*-ra meghívja arra ugyanezt a függvényt a megadott színnel és iránnyal.
    - **void onEntry():** True-ba állítja az *isPressed* attribútumot.
    - **void onExit():** False-ba állítja az *isPressed* attribútumot

3.3.13 Stargate

- ## Felelősség

Absztrakt osztály, itt kerülnek megvalósításra a *BlueStargate*, illetve a *YellowStargate* azonos attribútumai, illetve metódusai.

- **Ősosztályok**

Legősebb osztály: Object

- **Interfészek**

Nem valósít meg interfészt.

- **Attribútumok**
  - **Map<Color, Stargate> stargates:** tároja mindkét *Stargate* referenciáját
  - **boolean isShot:** nyilvántartja, hogy a *Stargate* éppen ki van-e lőve a pályára
  - **Tile currentTile:** nyilvántartja, hogy a *Stargate* éppen melyik *Tile*-ra van kilőve
  - **Stargate otherStargate:** tárolja a másik *Stargate* referenciáját

- **Metódusok**
  - **Stargate getStargate(Color c):** Visszaadja a paraméterként kapott színű *Stargate* referenciáját
  - **Stargate getOtherStargate():** Visszaadja a másik *Stargate* referenciáját
  - **Stargate setOtherStargate(Stargate sg):** Beállítja a másig *Stargate* referenciáját és visszaadja azt.
  - **Tile getCurrentTile():** Visszaadja az aktuális *Tile* referenciáját.
  - **Tile setCurrentTile(Tile t):** Beállítja az aktuális *Tile* referenciáját és visszaadja azt.
  - **boolean isOpen():** Akkor tér vissza igazzal, ha mindkét *Stargate isShot* attribútuma igaz, egyébként hamis.
  - **void teleport(Direction incomingDirection):** Amennyiben a csillagkapu nyitva van, átteleportálja ONeill-t a másik csillagkapu mellé, a beérkező iránnyal ellentétes irányba. (Azaz amennyiben O'Neill keleti irányból belép a kék csillagkapun, a sárga csillagkapu nyugati szomszédos mezején fog landolni.) Amennyiben O'Neill nem léphet át a féreglyukon, mert az zárva van, vagy olyan mezőhöz vezetne, amelyre nem léphet, visszakerül az eredeti mezőre, ahonnan elindult.

3.3.14 Tile

- **Felelősség**

A pályák "építőkövei". A *Tile* egy absztrakt ősosztály, amiből a különböző pályaelemek, mint például a *Wall* vagy a *Field* származnak le. Minden *Tile* ismeri a négy irányban (észak, dél, kelet, nyugat) szomszédját, ha a pálya szélén található, akkor null az adott irányban a szomszédos objektum.

- **Ősosztályok**

Ez az absztrakt osztály a hierarchia csúcsán áll, így nem származik csak az *Object* ősosztályból.
Legősebb osztály: Object

- **Interfészek**

Ez az absztrakt osztály nem valósít meg interface-t.

- **Attribútumok**

- **Metódusok**
  - **void setAdjacentTile(Tile newTile, Direction direction):** beállítja az objektumtól megadott irányban lévő szomszédos *Tile* referenciáját a paraméterként megadottra.
  - **Tile getAdjacentTile(Direction direction):** visszaadja a paraméterként megadott irányban lévő *Tile* referenciáját.
  - **void SpawnStargate(Color colour, Direction direction):** Ebben az absztrakt osztályban nem megvalósított függvény. A fő célja, hogy az adott irányba megadott *Tile*-ra egy csillagkaput tegyen.
  - **void onEntry():** Akárhányszor O'Neill rálép, vagy egy *Box*-ot tesznek egy *Tile*-ra, meghívódik a függvény. Leszármazottonként más az implementációja.
  - **void onExit():** Akárhányszor O'Neill lelép, vagy egy *Box*-ot vesznek le egy *Tile*-ról, meghívódik a függvény. Leszármazottonként más az implementációja.

3.3.15 Wall

- **Felelősség**

A pályán fellelhető falak megvalósításáért felelő osztály. Ennek a példányaiban O'Neill vagy a *Box* típusú objektumok megakadnak.

- **Ősosztályok**

A *Wall* a *Tile* absztrakt osztály leszármazottja, ami megvalósítja annak nem implementált függvényeit.
Legősebb osztály: Tile

- **Interfészek**

Ez az osztály nem valósít meg interface-t.

- **Attribútumok**
  - Stargate sg

- **Metódusok**
  - **void spawnStargate(Color colour, Direction direction):** Ha van már csillagkapu az objektumon (az sg attribútum nem null), visszatér. Ha nincs, felveszi az adott színű csillagkaput az sg attribútumnak.
  - **void clearStargate():** Törli az *sg* attribútum referenciáját.
  - **void onEntry():** A függvény visszatér, a *Box* objektum vagy O'Neill nem lép a mezőre.
  - **void onExit():** Üres függvénytörzs.

3.3.16 ZPM

- **Felelősség**

O'Neill által gyűjtendő tárgyakat megvalósító osztály.

- **Ősosztályok**

Osztályból nem származik, így legősebb osztálya az *Object*.

Legősebb osztály: Object

- **Interfészek**

A *Pickable* interfészt valósítja meg.

- **Attribútumok**

Nincsenek attribútumai.

- **Metódusok**

Ninncsenek metódusai

3.3.17 YellowStargate

- **Felelősség**

A sárga csillagkaput megvalósító osztály. Singleton, azaz egy példánya létezik csak, és ez mozog a pályán. Az implementáció nagyrésze a *BlueStargate*-tel közös *Stargate* ősosztályban található.

- **Ősosztályok**

Legősebb osztály: Stargate

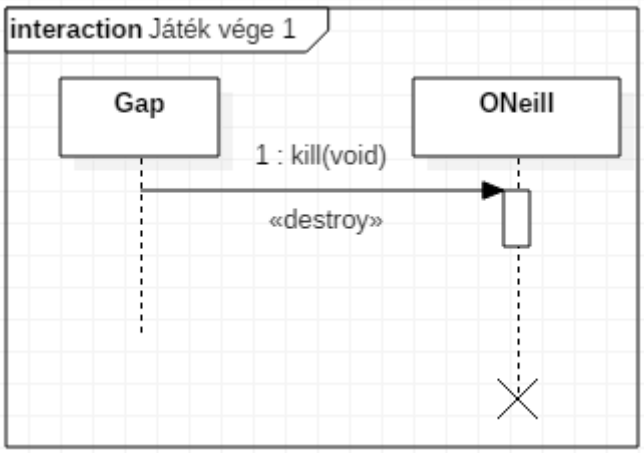
- **Interfészek**

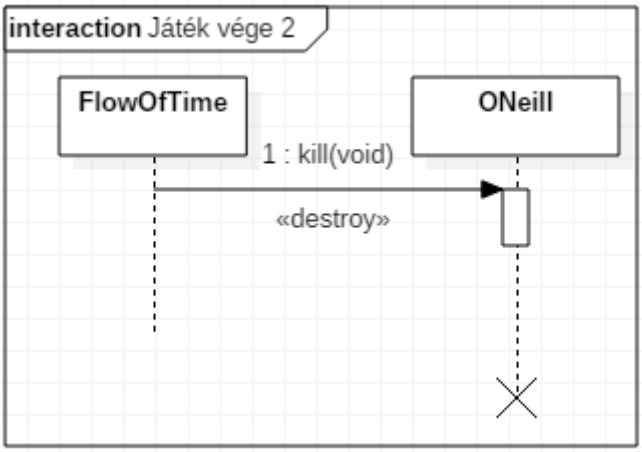
- **Attribútumok**

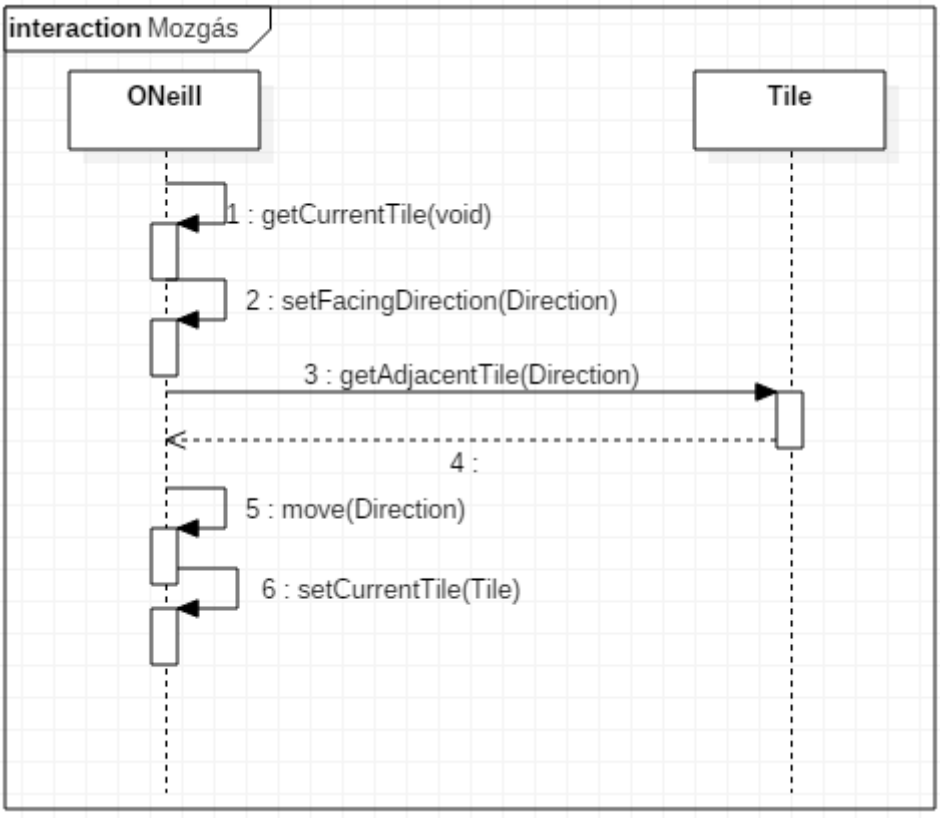
  - Color: a kapu színe.

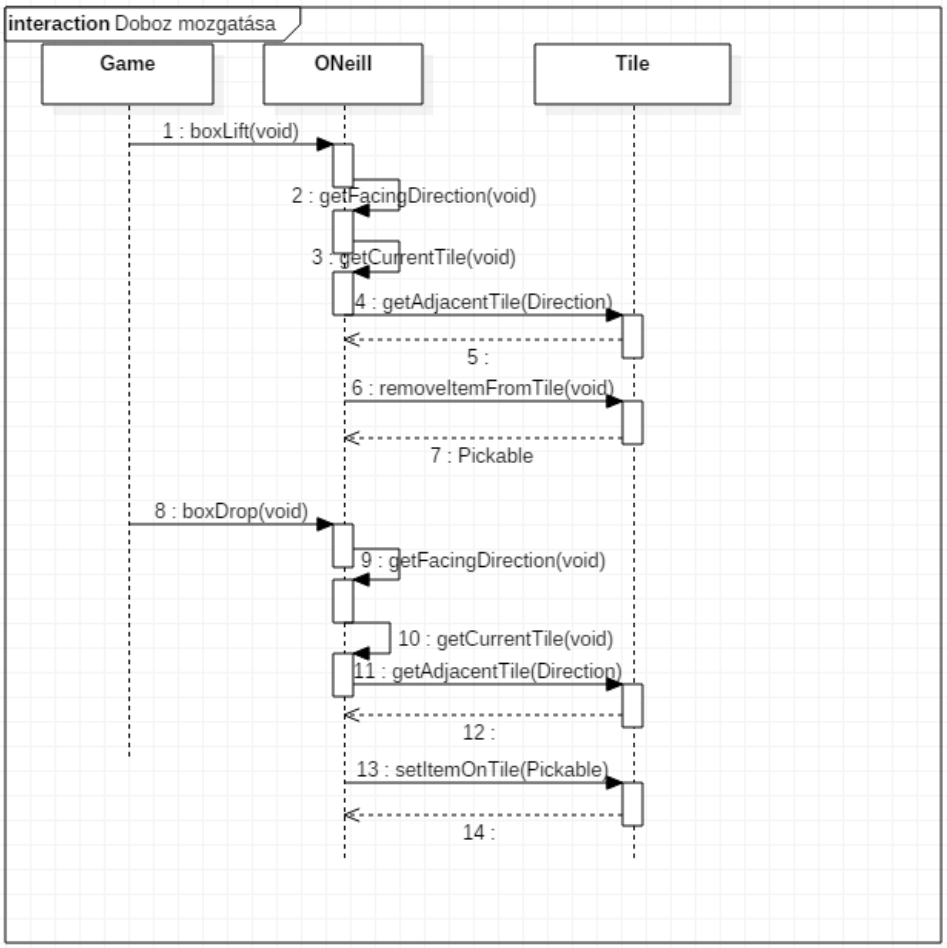
3.4 Szekvencia diagramok

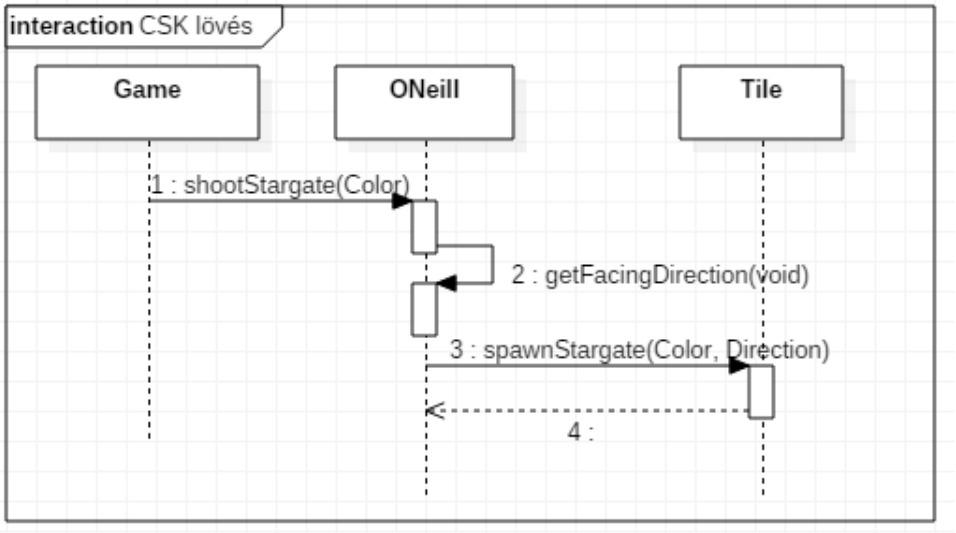
ONeill

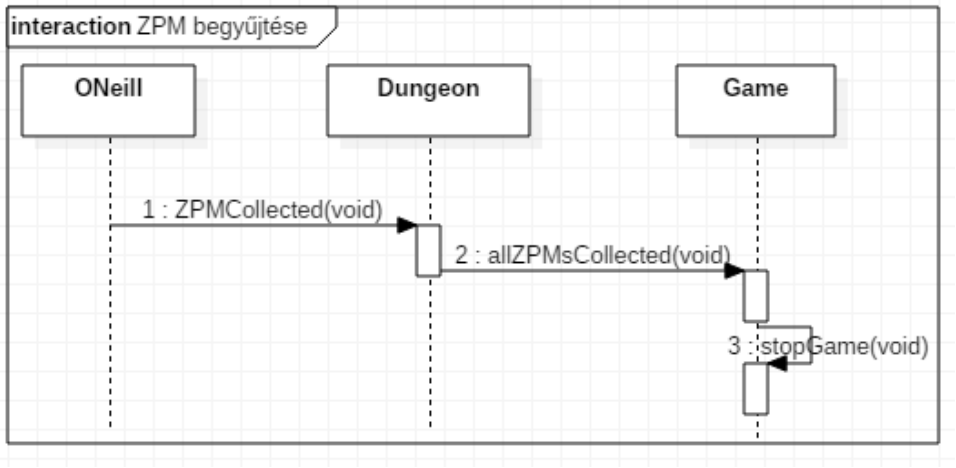

interaction Játék vége 1
Gap
ONeill
1 : kill(void)
«destroy»


interaction Játék vége 2
FlowOfTime
ONeill
1 : kill(void)
«destroy»


interaction Mozgás
ONeill
Tile
1 : getCurrentTile(void)
2 : setFacingDirection(Direction)
3 : getAdjacentTile(Direction)
4 :
5 : move(Direction)
6 : setCurrentTile(Tile)


interaction Doboz mozgatása
Game
ONeill
Tile
1 : boxLift(void)
2 : getFacingDirection(void)
3 : getCurrentTile(void)
4 : getAdjacentTile(Direction)
5 :
6 : removeItemFromTile(void)
7 : Pickable
8 : boxDrop(void)
9 : getFacingDirection(void)
10 : getCurrentTile(void)
11 : getAdjacentTile(Direction)
12 :
13 : setItemOnTile(Pickable)
14 :


interaction CSK lövés
Game
ONeill
Tile
1 : shootStargate(Color)
2 : getFacingDirection(void)
3 : spawnStargate(Color, Direction)
4 :



3.5 State-chartok

3.6 Napló

Kezdet	Időtartam	Résztvevők	Leírás
2016.03.06. 14:30	1,5 óra	Bokros Hegedüs Jáhn Siket Tóth Varga	Értekezlet. A modell kialakításának kezdeti lépései.
2016.03.05 18:00	2 óra	Bokros Hegedüs Jáhn Siket Tóth Varga	Értekezlet. A modell kialakításának folytatása.
2016.03.05. 20:00	4 óra	Jáhn	Tevékenység Jáhn megírja a Tile absztakt ősosztály és annak leszármazottjainak leírását.
2016.03.06. 11:00	1 óra	Bokros	Tevékenység Bokros megírja a Dungeon és a Direction leírását.
2016.03.06. 12:00	1,3 óra	Tóth	Tevékenység: Tóth megírja az objektum katalógusban lévő osztályok feléhez tartozó leírást
2016.03.06. 16:00	2 óra	Bokros	Tevékenység: Bokros megírja a Stargate, a YellowStargate és a BlueStargate leírását.
2016.03.06. 16:00	30 perc	Siket	Tevékenység: Siket lektorálja az eddig leírtakat.
2016.03.06. 16:30	1,7 óra	Tóth	Tevékenység: Tóth befejezi az objektum katalógust és ellenőrzi a dokumentum eddigi tartalmát.
2016.03.06. 18:00	1,5 óra	Hegedüs	Tevékenység: Hegedüs megírja a Pickable, ZPM és Box osztályok leírását és ellenőrzi a többi osztály leírását.
2016.03.06 18:00	3óra	Siket	Tevékenység: Siket megszerkeszti az osztálydiagramot.
2016.03.06. 21:00	3 óra	Varga	Tevékenység: Varga ellenőrzi a dokumentumot, majd megszerkeszti a use-case diagramot, illetve a szekvenciadiagramokat az ONeill osztályhoz.
2016.03.06 22:00	45 perc	Tóth	Tevékenység: Tóth megírja a Game osztály leírását
2016.03.06 23:15	30 perc	Bokros	Tevékenység. Bokros ellenőrzi és javítja a dokumentum formázását, valamint elkészíti a fedlapot.
2016.03.06 23:30	35 perc	Tóth Jáhn	Tevékenység: Tóth és Jáhn további módosításokat végez a dokumentumon.
2016.03.07. 1:15	30 perc	Varga	Tevékenység:

			Varga átvizsgálja a dokumentumot, javításokat végez.
2016.03.07. 1:45	15 perc	Siket	Tevékenység: Siket egységesíti a dokumentum szövegének formázását.

4. Analízis modell kidolgozása 2

4.1 Objektum katalógus

4.1.1 Box

Ez az osztály a játékvilágban található dobozoknak feleltethető meg. A játékos O'Neillt irányítva felveheti, illetve lerakhatja őket a megfelelő mezőkre. Van súlyuk, a játék mechanikájának szempontjából elsősorban ajtónyitásra alkalmasak, így ha O'Neill rátesz egy dobozt egy mérlegre (*Scale*), akkor a hozzá tartozó kapu (*Gate*) a *Scale* osztálynál leírtak értelmében kinyílik.

4.1.2 Direction

A többi osztály ezt az enumot használja az irányok tárolására (*NORTH*, *SOUTH*, *EAST*, *WEST*).

4.1.3 Dungeon

Felelőssége a játéktér (pálya) felépítése és lebontása: ez gyakorlatilag abból áll, hogy a játék kezdetekor beállítgatja a játék összes mezőjén (*Tile*), hogy milyen másik mezőkkel szomszédos.

4.1.4 Field

Olyan mezőket reprezentáló osztály, melyekre O'Neill rá tud lépni. A játék kezdetén is egy ilyen mezőn találja magát a játékos. Egyszerre egy darab valamilyen felvehető objektum (*Pickable*) lehet egy *Field* típusú mezőn, ez lehet például *ZPM* vagy doboz (*Box*). Ha egy *Field* mezőn *ZPM* van, akkor ha O'Neill rálép, felveszi a *ZPM*-et, azonban ha egy doboz van rajta, akkor oda nem lehet lépni, csak akkor ha a dobozt felveszi, majd odébb viszi a játékos. Ismeri a saját szomszédságát mind a négy irányban.

4.1.5 FlowOfTime

Az idő előrehaladását ellenőrző osztály. A játékélmény növelése érdekében a játékos rá van kényszerítve, hogy megadott időn belül gyűjtse össze a *ZPM*-eket. Ha az időkeretet túllépi, akkor O'Neill meghal és a játék véget ér.

4.1.6 Game

A játékmotor alapja, kezeli a játékos által végrehajtott eseményeket (billentyűlenyomások, stb.). Magába foglalja a játék logikáját és összeköti a többi osztály interakcióját, a megfelelő események bekövetkezésekor lezárja a játékot.

4.1.7 Gap

A pályán található szakadékokat modellező osztály. Ha a játékos O'Neillt irányítva rálép egy ilyen típusú mezőre, akkor O'Neill meghal, a játék pedig véget ér. Nem lehet rajta *ZPM,* és ha a játékos egy dobozt (*Box*) rakna rá, akkor az leesik, azaz megsemmisül. Ismeri a saját szomszédságát mind a négy irányban.

4.1.8 Gate

A játékban lévő kapukat reprezentáló osztály. Minden egyes *Gate*-hez tartozik egy darab *Scale* objektum, aminek segítségével a kapu kinyitható, ha a mérlegre valamilyen súly kerül. Amennyiben egy kapu zárva van, azon O'Neill nem tud semmilyen módon átmenni, ehhez ki kell valahogyan nyitnia azt. Nem lehet rá csillagkaput lőni. A játék kezdetekor a kapuk lehetnek

csukott, vagy nyitott állapotban is, attól függően, hogy a hozzájuk tartozó mérlegen alapból van-e valamilyen súly. Ismeri a saját szomszédságát mind a négy irányban.

4.1.9 ONeill

A játékban irányítható hőst modellező osztály. Mindig nyilvántartja, hogy éppen hol van. A játékos tud általa csillagkapukat lőni, hogy megkönnyítse a pályán való mozgást, illetve hogy eljusson esetlég másképp elérhetetlen helyekre. Ez kék, illetve sárga csillagkapuk formájában lehetséges, bővebb leírás erről a mechanizmusról a *Stargate* osztálynál található. A játékos a karaktert 4 irányba mozgathatja, fel, le, jobbra és balra. Képes a pályákon szétszórt *ZPM*-eket felvenni, amiből ha mindet megszerezte, megnyeri a játékot. A játéktérben található dobozokat fel tudja venni, ha azok pont előtte vannak, majd le is tudja őket tenni valahova maga elé. Amennyiben egy megadott időn belül nem tudja összeszedni az összes, a pályán található *ZPM*-et, vagy egy szakadék mezőre (*Gap*) lép, meghal, így a játék véget ér.

4.1.10 Scale

A kapuk kinyitásához használt mérlegek osztálya. Minden ilyen mérleghez tartozik egy kapu (*Gate*) a pályán, ami nyitható vele. Egy mérleg akkor nyitja ki a hozzá tartozó kaput, hogyha valamilyen súly van rajta. Ez történhet például úgy, hogy O'Neill karaktere rááll a mérlegre, vagy mondjuk rátesz egy dobozt (*Box*). Lehet a mérlegeken *ZPM* modul, amit a játékos hasonló módon vehet fel, mint normál esetben, azonban a *ZPM*-ek a mérleg szempontjából nem rendelkeznek súllyal, így nem nyitják a mérleghez tartozó kaput. Ismeri a saját szomszédságát mind a négy irányban.

4.1.11 Stargate

Az O'Neill karaktere által kilőhető csillagkapukat modellező osztály. Egy kilőtt csillagkapu lehet kék vagy sárga. A kilövés egy olyan megkötéssel történik, hogy egyszerre egy időben csak egy darab létezhet a játékban egy adott színű csillagkapuból, tehát például létezhet egyszerre egy kék és egy sárga, de ugyanazon színből több már nem. Ha a pályán egy időben létezik egy kék és egy sárga csillagkapu, akkor a tér átjárható lesz közöttük. Amennyiben O'Neill belemegy egy valamilyen színű csillagkapuba, az átteleportálja őt a másik színű csillagkapuhoz, amennyiben az létezik. Ha éppen ilyen csillagkapu nem létezik, akkor nem lehet belépni a csillagkapuba és semmi nem történik. O'Neill akkor is át tud járni a csillagkapuk között, ha visz valamit.

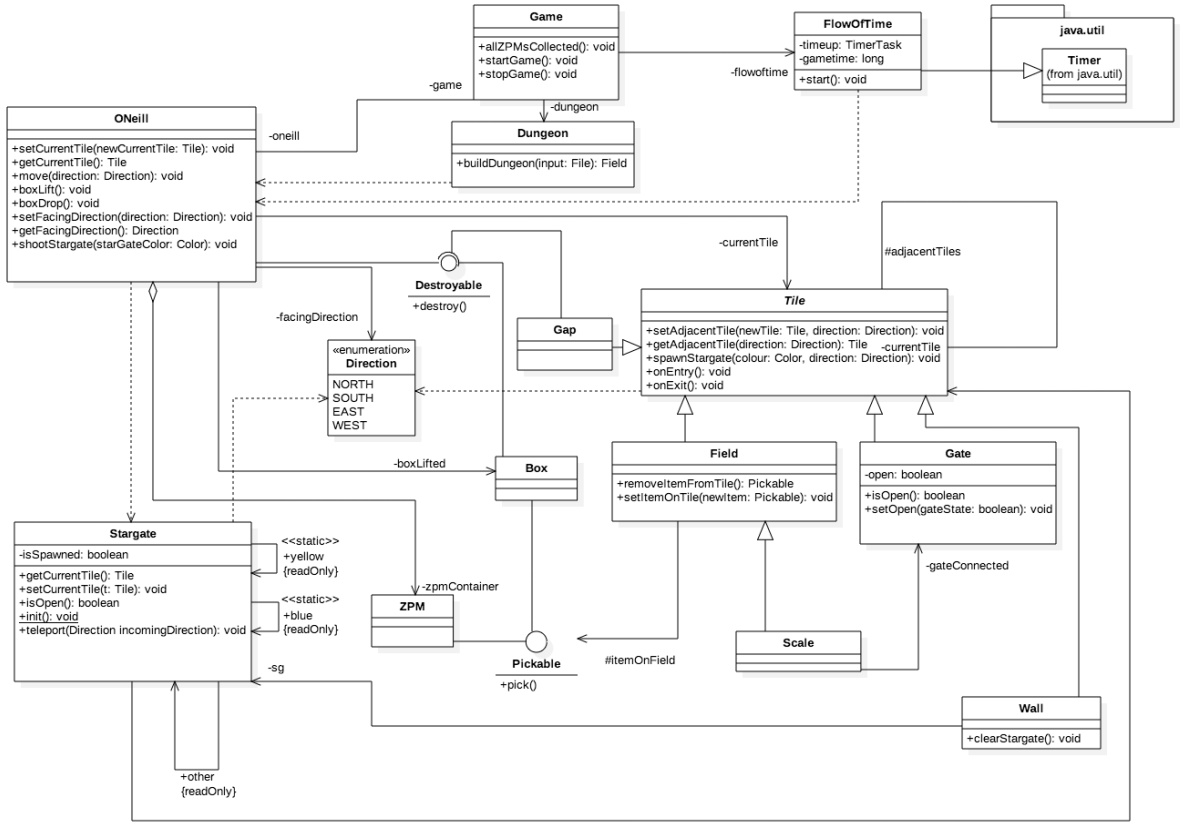
4.1.12 Wall

A pályán található falakat reprezentáló osztály. A falakon O'Neill nem tud átmenni, de a falakra tud csillagkapukat lőni. A falakon nem lehet *ZPM* vagy doboz (*Box*). A pályák szélén tipikusan ilyen falak vannak, hogy valamennyire korlátozzák a játékos mozgását. Ismeri a szomszédságát mind a négy irányban.

4.1.13 ZPM

A játékban található összegyűjtendő tárgyak osztálya. A játék megnyerésének feltétele az, hogy O'Neill összegyűjtse az összes, pályán szétszórt *ZPM*-et. Ez úgy történik, hogy a játékos O'Neill karakterét egy olyan mezőre irányítja, ahol *ZPM* lebeg, ekkor O'Neill felveszi azt. Ugyan *ZPM* modulok lehetnek mérlegeken (*Scale*), a *ZPM* természeténél fogva lebeg, így nem nyomja le a mérleget, tehát a hozzá tartozó kaput (*Gate*) sem nyitja ki.

4.2 Statikus struktúra diagramok



4.3 Osztályok leírása

4.3.1 Box

- **Felelősség**

Az O'Neill által felvehető és letehető dobozokat megvalósító osztály.

- **Ősosztályok**

Ez az osztály nem származik más osztályból.

- **Interfészek**

A *Pickable* és a *Destroyable* interfészt valósítja meg.

- **Attribútumok**

Nincsenek attribútumai.

- **Metódusok**

  - **void destroy()**: megsemmisül a doboz.
  - **void pick():** A felvételt jelzi. Ez GUI-val együtt egy hangeffektet is jelenthet.

4.3.2 Direction

- **Felelősség**

A négy irányt (*NORTH, SOUTH, EAST, WEST*) tároló enum, a többi osztály ezt használja a szomszédos mezők azonosítására.

4.3.3 Destroyable

- **Felelősség**

Az interfész arra ad lehetőséget, hogy a *Box* és *ONeill* osztályok hasonló megsemmisülésükből fakadóan azonos nevű függvény által pusztulhassanak el.

- **Ősosztályok**

Interfész lévén nem származik osztályból.

- **Interfészek**

Ez az interfész nem valósít meg interfészt.

- **Attribútumok**

Interfész lévén nincsenek attribútumai.

- **Metódusok**
  - **void destroy():** meghívva különféle módon semmisül meg az objektum, amin meghívták.

4.3.4 Dungeon

- **Felelősség**

Az osztály felelőssége a pálya felépítése és O'Neill elhelyezése azon a játék kezdetekor.

- **Ősosztályok**

Ez az osztály nem származik más osztályból.

- **Interfészek**

Az osztály nem valósít meg interfészt.

- **Attribútumok**

Az osztálynak nincsenek attribútumai.

- **Metódusok**
  - **Field buildDungeon(File input):** A bemeneti fájlban tárolt információk alapján felépíti a pályát. Visszatér annak a mezőnek a referenciáján, amelyen O'Neill áll.

4.3.5 Field

- **Felelősség**

A pályán belüli bejárható mezőket megvalósító osztály. O'Neill ezekre a mezőkre szabadon tud lépni, illetve a játékot is egy ilyen mezőn kezdi. Ezeken továbbá felvehető tárgyak is vannak, mint például a *ZPM* vagy a *Box*. Egy ilyen mezőn maximum egy felvehető tárgy lehet.

- **Ősosztályok**

A *Field* a *Tile* absztrakt osztály leszármazottja, ami megvalósítja annak nem implementált függvényeit.
Legősebb osztály: Tile

- **Interfészek**

Ez az osztály nem valósít meg interfészt.

- **Attribútumok**
  - **Pickable itemOnField:** A *Field*-en lévő *Pickable* objektum.

- **Metódusok**
  - **void spawnStargate(Stargate sg, Direction direction):** A megadott irányban lévő szomszédos *Tile*-ra meghívja arra ugyanezt a függvényt a megadott csillagkapuval és iránnyal.
  - **Pickable removeItemFromTile():** Átadja a függvény hívójának az *itemOnTile* attribútumának referenciáját és törli azt.
  - **void setItemOnTile(Pickable newItem):** Ha nincs *Pickable* típusú objektuma már a mezőnek, a paraméterül kapottat teszi meg *itemOnTile*-nak.
  - **void onEntry():** Ha nincs rajta az objektumon *Box*, O'Neill vagy egy *Box* rákerül a mezőre.
  - **void onExit():** O'Neill vagy egy *Box* lekerül a mezőről.

4.3.6 FlowOfTime

- **Felelősség**

A játék kezdetekor elkezdi az eltelt idő mérését, és egy előre megadott idő után meghívja *ONeill destroy()* metódusát.

- **Ősosztályok**

A java.util.Timer osztályból származik.

- **Interfészek**

Ez az osztály nem valósít meg semmilyen interfészt.

- **Attribútumok**

  - **TimerTask timeup:** Az idő leteltével meghívandó függvényt tartalmazó objektum, ami meghívja *ONeill destroy()* metódusát.
  - **long gametime:** Ennyi ideje van a játékosnak összegyűjteni az összes *ZPM*-et (milliszekundumban mérve).

- **Metódusok**

  - **void start():** elindítja az időmérőt.

4.3.7 Game

- **Felelősség**

Az események kezelése a játékban, illetve az összes többi osztály összekötése. A játékossal való kommunikáció.

- **Ősosztályok**

Ez az osztály nem származik más osztályból.

Interfészek

Ez az osztály nem valósít meg interfészt.

Attribútumok

- **Dungeon dungeon:** A pálya.
- **ONeill oneill:** O'Neill ezredes.
- **FlowOfTime flowoftime:** Az időzítő, ami számolja az idő múlását.

Metódusok

- **void startGame():** elindítja a játékot
- **void stopGame():** leállítja a játékot
- **void allZPMsCollected():** értesíti a *Game* osztályt, hogy a játék véget ért, mert megvan a *ZPM* (O'Neill hívja)

4.3.8 Gap

- **Felelősség**

A pályán belüli szakadékokat megvalósító osztály. Amennyiben ilyen objektumra lép O'Neill ezredes, szörnyet hal. Ha a játékos egy *Box*-ot ilyen objektumra dobna, az megsemmisül.

- **Ősosztályok**

A *Gap* a *Tile* absztrakt osztály leszármazottja, ami megvalósítja annak nem implementált függvényeit.
Legősebb osztály: Tile

- **Interfészek**

Ez az osztály nem valósít meg interfészt.

- **Attribútumok**

- **Metódusok**

  - **void spawnStargate(Stargate sg, Direction direction):** A megadott irányban lévő szomszédos *Tile*-ra meghívja arra ugyanezt a függvényt a megadott csillagkapuval és iránnyal.
  - **void onEntry():** Meghívja *ONeill v*agy a *Box destroy()* függvényét.
  - **void onExit():** Kivételt dob a függvény.

4.3.9 Gate

- **Felelősség**

A pályán belüli kinyitható ajtókat megfeleltető osztály. Az ajtó lehet nyitott vagy zárt állapotban. Minden *Gate* objektumot ismer egy *Scale* objektum, amire súly helyezésével kinyílik a *Gate* objektum.

- **Ősosztályok**

A *Gate* a *Tile* absztrakt osztály leszármazottja, ami megvalósítja annak nem implementált függvényeit.
Legősebb osztály: Tile

- **Interfészek**

Ez az osztály nem valósít meg interfészt.

- **Attribútumok**

  - **boolean open:** A kapu nyitottságát jelzi. True esetén nyitott, false esetén zárt.

- **Metódusok**

  - **void spawnStargate(Stargate sg, Direction direction):** Ha az ajtó nyitva van és a direction irányában *Field* található, meghívja arra ugyanezt a függvényt a megadott csillagkapuval és iránnyal. Ha az ajtó zárva van, visszatér a függvényből.
  - **boolean isOpen():** Az ajtó nyitottságát lekérdező függvény, az *open* attribútum értékét adja vissza. True esetén nyitva, false esetén zárva van az ajtó.

- **void setOpen(boolean gateState):** Az ajtó nyitottságát beállító függvény. A paraméterként kapott értékre állítja az *open* attribútum értékét. True esetén nyitva, false esetén zárva van az ajtó.
- **void onEntry():** Ha az *open* értéke true, egy *Field*-ként viselkedve O'Neill vagy egy *Box* rákerülhet az objektumra. Ha az *open* értéke false, egy *Wall*-ként viselkedve visszatér.
- **void onExit():** Ha az *open* értéke true, egy *Field*-ként viselkedve O'Neill vagy egy *Box* elhagyja az objektumot. Ha az *open* értéke false, egy *Wall*-ként viselkedve kivételt dob.

4.3.10 ONeill

- **Felelősség**

Az ezredest megvalósító osztály. Képes észak, dél, kelet nyugati irányban mozogni a *Tile*-ok közt, *ZPM*-eket gyűjteni, illetve dobozokat felemelni, mozgatni, majd letenni.

- **Ősosztályok**

Ez az osztály nem származik más osztályból.

- **Interfészek**

ONeill a *Destroyable* interfészt valósítja meg.

- **Attribútumok**

  - **Tile currentTile:** annak a mezőnek a referenciája, amin O'Neill éppen tartózkodik.
  - **Game game:** az aktuális játék referenciája.
  - **ArrayList<ZPM> zpmContainer:** egy lista, amiben az összegyűjtött ZPM-eket tároljuk.
  - **Direction facingDirection:** az aktuális irány, amerre O'Neill néz.
  - **Box boxLifted:** ha éppen doboz van nála, akkor a doboz referenciája.

- **Metódusok**

  - **void setCurrentTile(Tile newCurrentTile):** Beállítja, hogy O'Neill melyik pályadarabon tartózkodik.
  - **Tile getCurrentTile():** Visszaadja, hogy O'Neill melyik pályadarabon tartózkodik.
  - **void shootStargate(Stargate sg):** Egy paraméterként kapott csillagkaput *(Stargate)* lő az aktuális irányba, ami az első útjába eső felületen nyílik meg.
  - **void destroy():** O'Neill ezredes meghal és ezzel vége a játéknak.
  - **void move(Direction direction):** O'Neillt a megadott irányban lévő pályadarabra mozgató függvény.
  - **void boxLift():** O'Neill megpróbál felvenni abból az irányból dobozt, amerre néz. Ha van doboz az adott pályadarabon, a *boxLifted* attribútumának teszi, ha nincs, nem történik semmi.
  - **void boxDrop():** O'Neill lerakja a dobozát. Ha van nála doboz, arra a pályadarabra rakja, amerre néz, ha nincs, nem történik semmi.
  - **void setFacingDirection(Direction direction):** Beállítja, merre nézzen O'Neill.
  - **Direction getFacingDirection():** Visszaadja, merre néz O'Neill.

4.3.11 Pickable

Felelősség

Közös interfésze a *Box* és a *ZPM* osztályoknak.

Ősosztályok

Interfész lévén nem származik osztályból.

Attribútumok

Nincsenek attribútumai.

Metódusok

- **void pick()**: A megvalósító objektumok felvételéről küld jelzést.

4.3.12 Scale

- ## Felelősség

A pályán lévő mérlegeket megvalósító osztály. Ha súly kerül rá, kinyitja a hozzá tartozó ajtót *(Gate).*

- ## Ősosztályok

A *Scale* a *Field* osztály leszármazottja.
Legősebb osztály: Tile, Ősosztály1: Field

- ## Interfészek

Ez az osztály nem valósít meg interfészt.

- ## Attribútumok
  - **Gate gateConnected:** A mérleghez kapcsolódó kapu amit nyithat illetve zárhat.

- ## Metódusok
  - **void spawnStargate(Stargate sg, Direction direction):** A megadott irányban lévő szomszédos *Tile*-ra meghívja arra ugyanezt a függvényt a megadott csillagkapuval és iránnyal.
  - **void onEntry():** Meghívja a *gateConnected* attribútum *setOpen* függvényét true paraméterrel, illetve ha nincs rajta az objektumon *Box*, O'Neill vagy egy *Box* rákerülhet a területre.
  - **void onExit():** Meghívja a *gateConnected* attribútum *setOpen* függvényét false paraméterrel.

4.3.13 Stargate

- **Felelősség**

Ez az osztály valósítja meg a két csillagkaput. Mivel mindkét csillagkapuból csak egy létezhet, ezeket a játék elején létrehozza, majd a játék során csak ezek pályán való elhelyezkedése módosulhat.

- **Ősosztályok**

Ez az osztály nem származik más osztályból.

- **Interfészek**

Nem valósít meg interfészt.

- **Attribútumok**

  - **static final Stargate yellow:** referencia a sárga csillagkapura.
  - **static final Stargate blue:** referencia a kék csillagkapura.
  - **boolean isSpawned:** nyilvántartja, hogy a *Stargate* éppen ki van-e lőve a pályára.
  - **Tile currentTile:** nyilvántartja, hogy a *Stargate* éppen melyik *Tile*-ra van kilőve.
  - **final Stargate other:** referencia a másik *Stargate*-re.

- **Metódusok**

  - **Tile getCurrentTile():** Visszaadja az aktuális *Tile* referenciáját.
  - **void setCurrentTile(Tile t):** Beállítja az aktuális *Tile* referenciáját.
  - **boolean isOpen():** Akkor tér vissza igazzal, ha mindkét *Stargate isSpawned* attribútuma igaz, egyébként hamis.
  - **static void Init():** létrehozza a két csillagkaput, és beállítja a referenciáikat egymásra
  - **void teleport(Direction incomingDirection):** Amennyiben a csillagkapu nyitva van, átteleportálja O'Neill-t a másik csillagkapu mellé, a beérkező iránnyal ellentétes irányba. (Azaz amennyiben O'Neill keleti irányból belép a kék csillagkapun, a sárga csillagkapu nyugati szomszédos mezején fog landolni.) Amennyiben O'Neill nem léphet át a féreglyukon, mert az zárva van, vagy olyan mezőhöz vezetne, amelyre nem léphet, visszakerül az eredeti mezőre, ahonnan elindult.

4.3.14 Tile

- **Felelősség**

A pályák "építőkövei". A *Tile* egy absztrakt ősosztály, amiből a különböző pályaelemek, mint például a *Wall* vagy a *Field* származnak le. Minden *Tile* ismeri a négy irányban (észak, dél, kelet, nyugat) szomszédját, ha a pálya szélén található, akkor null az adott irányban a szomszédos objektum.

- **Ősosztályok**

Ez az absztrakt osztály a hierarchia csúcsán áll, így nem származik más osztályból.

- **Interfészek**

Ez az absztrakt osztály nem valósít meg interfészt.

- **Attribútumok**

  - **HashMap<Direction, Tile> adjacentTiles:** Tárolja a négy irányban (Észak, Dél, Kelet, Nyugat) lévő szomszédos Tile-okat.

- **Metódusok**

  - **void setAdjacentTile(Tile newTile, Direction direction):** beállítja az objektumtól megadott irányban lévő szomszédos *Tile* referenciáját a paraméterként megadottra.
  - **Tile getAdjacentTile(Direction direction):** visszaadja a paraméterként megadott irányban lévő *Tile* referenciáját.
  - **void spawnStargate(Stargate sg, Direction direction):** Ebben az absztrakt osztályban nem megvalósított függvény. A fő célja, hogy az adott irányba megadott *Tile*-ra egy csillagkaput tegyen.
  - **void onEntry():** Akárhányszor O'Neill rálép, vagy egy *Box*-ot tesznek egy *Tile*-ra, meghívódik a függvény. Leszármazottonként más az implementációja**.**
  - **void onExit():** Akárhányszor O'Neill lelép, vagy egy Box-ot vesznek le egy Tile-ról, meghívódik a függvény. Leszármazottonként más az implementációja.

4.3.15 Wall

- **Felelősség**

A pályán fellelhető falak megvalósításáért felelő osztály. Ennek a példányaiban O'Neill vagy a *Box* típusú objektumok megakadnak.

- **Ősosztályok**

A *Wall* a *Tile* absztrakt osztály leszármazottja, ami megvalósítja annak nem implementált függvényeit.
Legősebb osztály: Tile

- **Interfészek**

Ez az osztály nem valósít meg interfészt.

- **Attribútumok**

  - **Stargate sg:** A falon lévő csillagkapu. Ha nincs a falon csillagkapu, null értéket tárol.

- **Metódusok**

  - **void spawnStargate(Stargate sg, Direction direction):** Ha van már csillagkapu az objektumon (az sg attribútum nem null), visszatér. Ha nincs, felveszi a megadott csillagkaput az sg attribútumnak.
  - **void clearStargate():** Törli az *sg* attribútum referenciáját.

- **void onEntry():** A függvény visszatér, a *Box* objektum vagy O'Neill nem lép a mezőre.
- **void onExit():** Kivételt dob a függvény.

4.3.16 ZPM

- **Felelősség**

O'Neill által gyűjtendő tárgyakat megvalósító osztály.

- **Ősosztályok**

Ez az osztály nem származik más osztályból.

- **Interfészek**

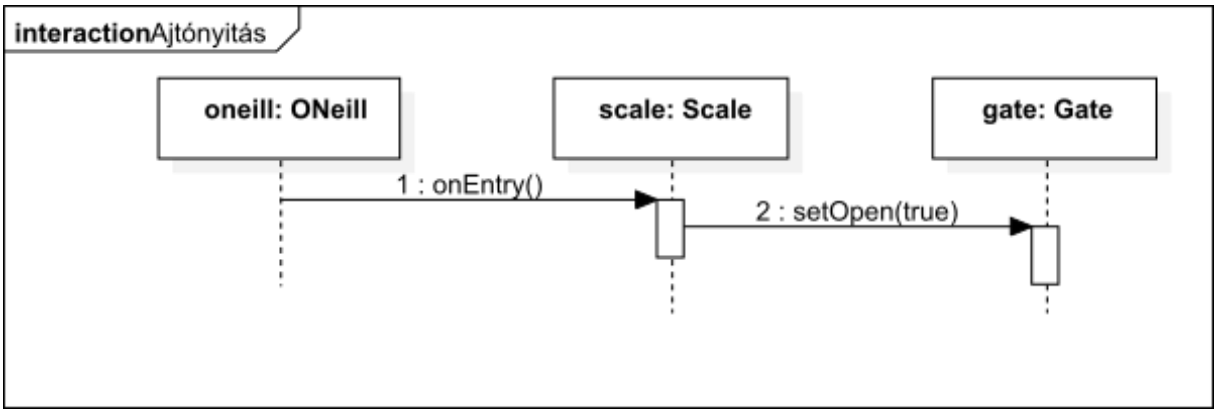
A *Pickable* interfészt valósítja meg.

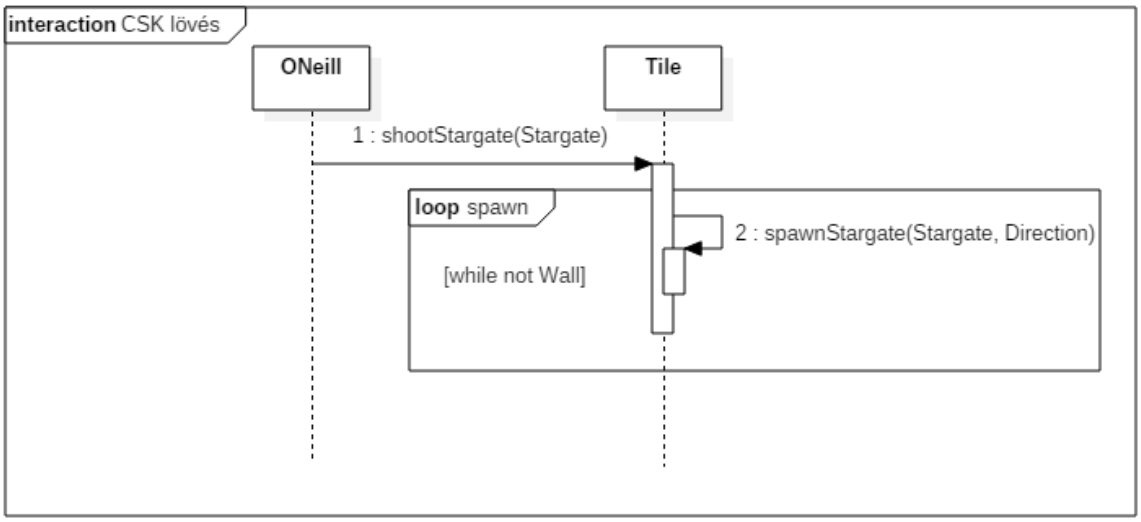
- **Attribútumok**

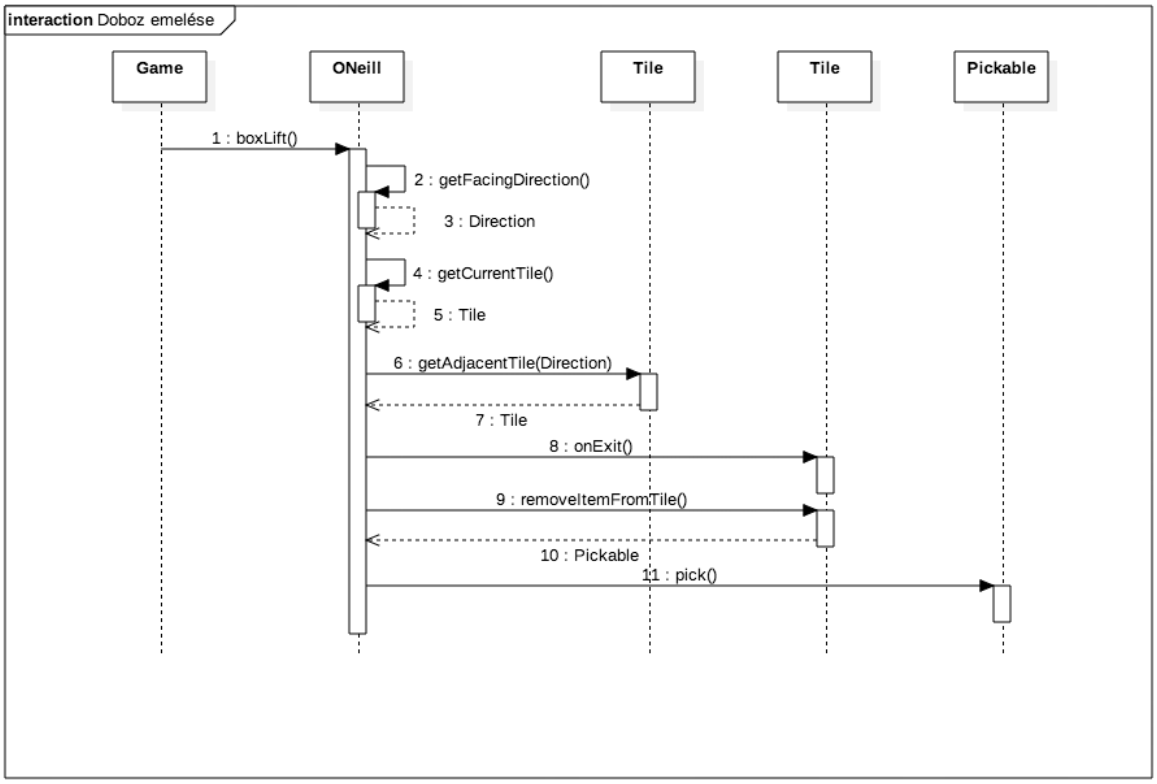
Nincsenek attribútumai.

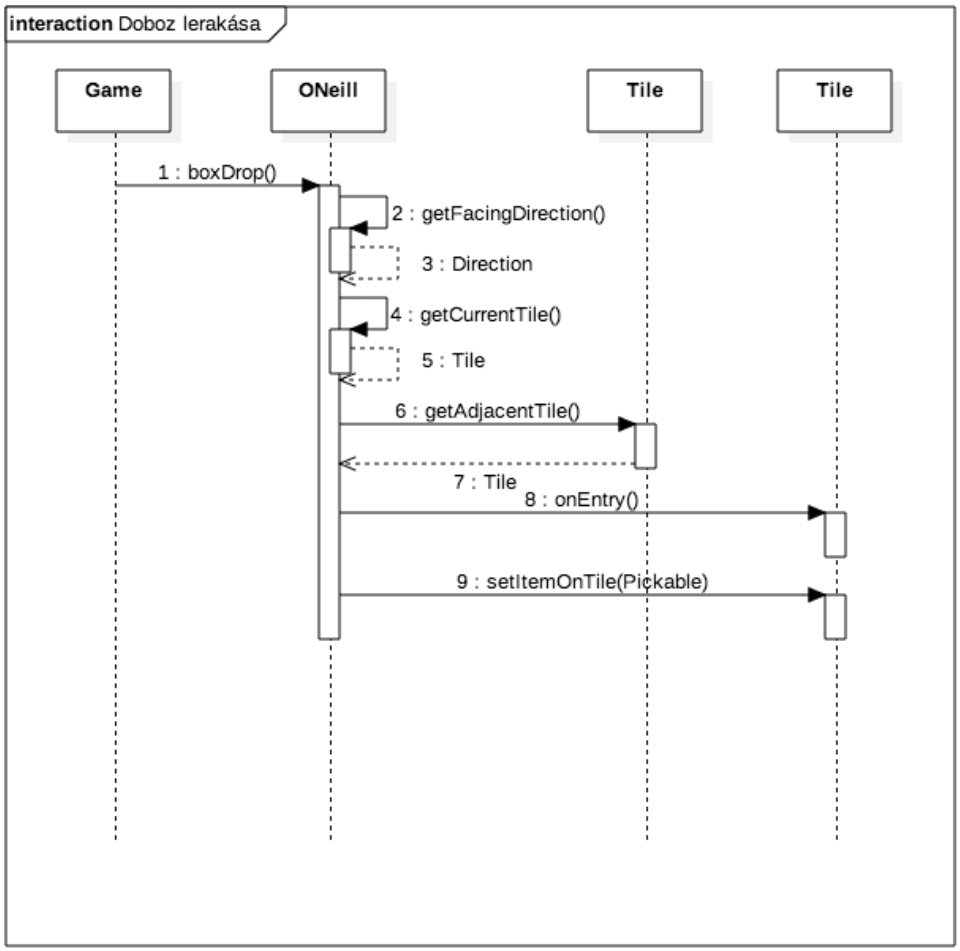
- **Metódusok**
    - **void pick():** A felvételt jelzi. Ez GUI-val együtt egy hangeffektet is jelenthet.

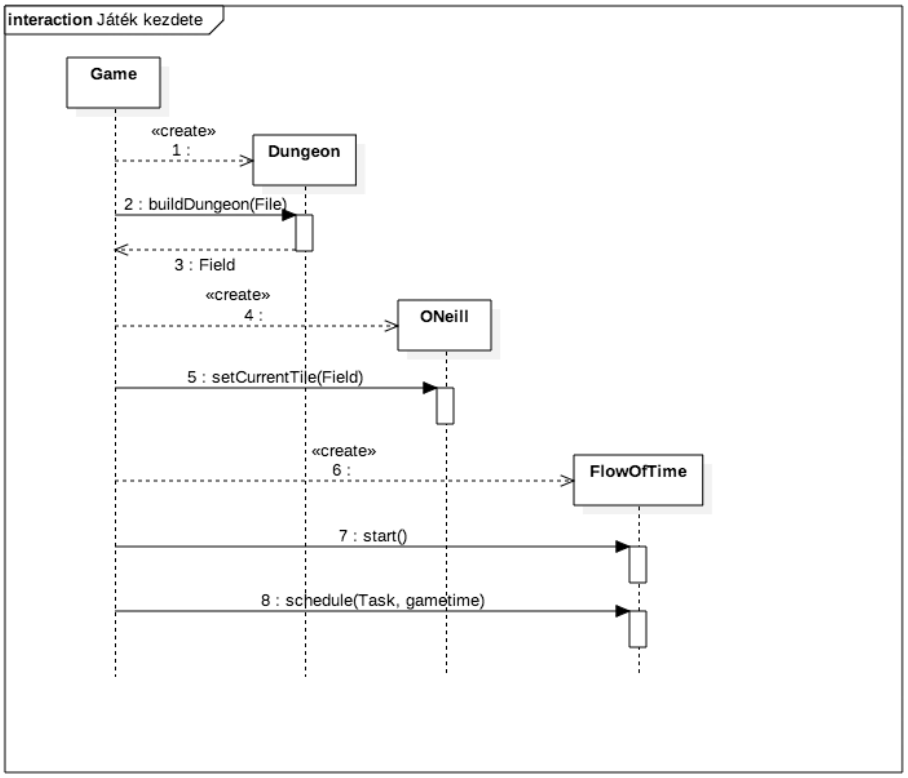
4.4 Szekvencia diagramok

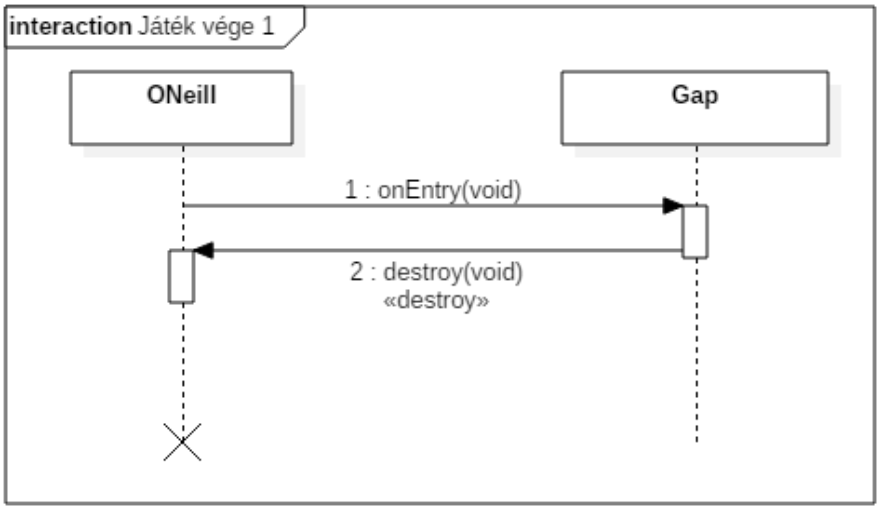


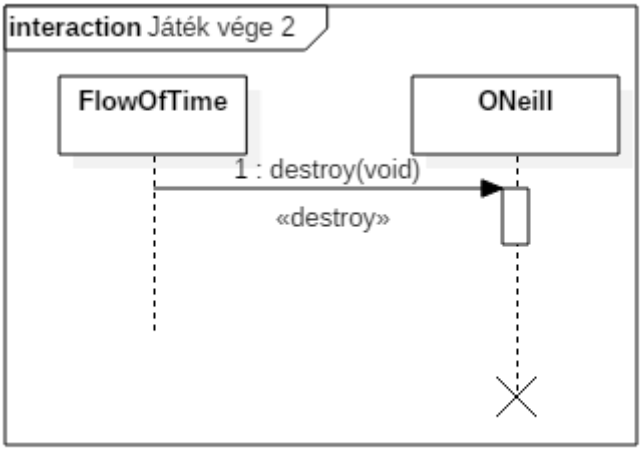


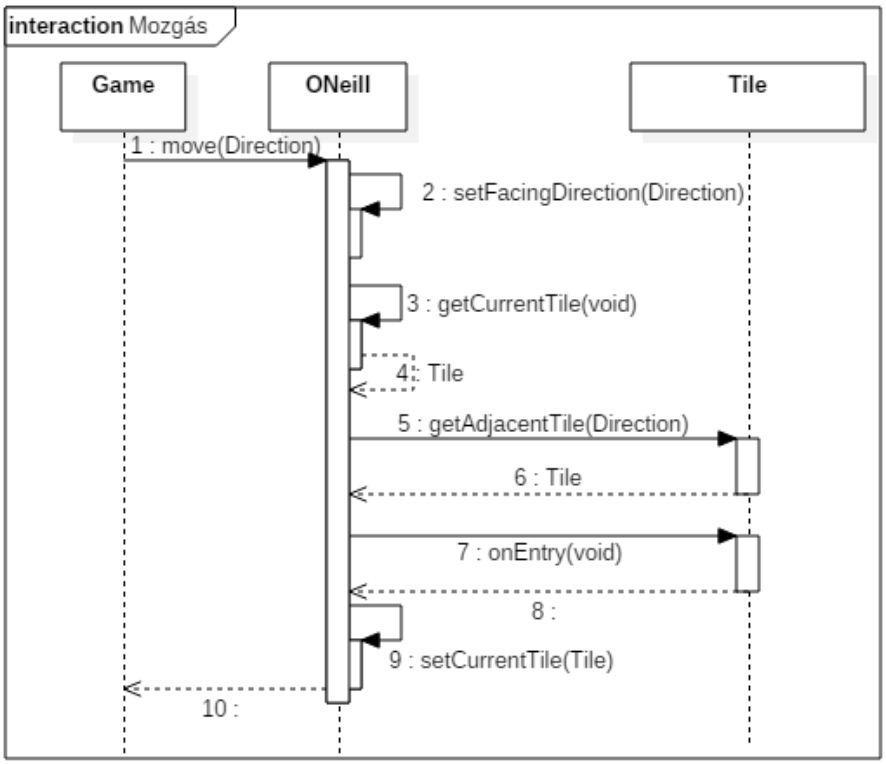

interaction Doboz emelése
Game
ONeill
Tile
Tile
Pickable
1 : boxLift()
2 : getFacingDirection()
3 : Direction
4 : getCurrentTile()
5 : Tile
6 : getAdjacentTile(Direction)
7 : Tile
8 : onExit()
9 : removeItemFromTile()
10 : Pickable
11 : pick()

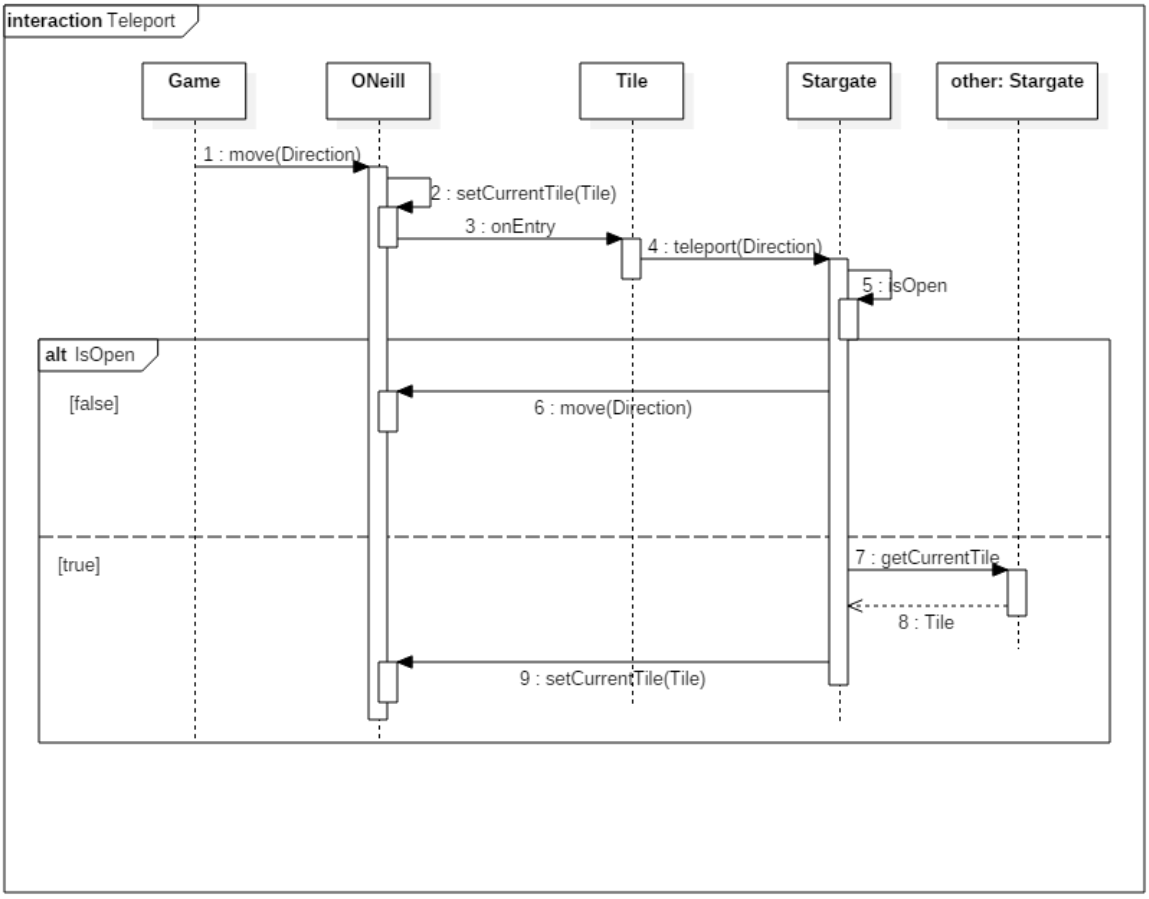

interaction Doboz lerakása
Game
ONeill
Tile
Tile
1 : boxDrop()
2 : getFacingDirection()
3 : Direction
4 : getCurrentTile()
5 : Tile
6 : getAdjacentTile()
7 : Tile
8 : onEntry()
9 : setItemOnTile(Pickable)

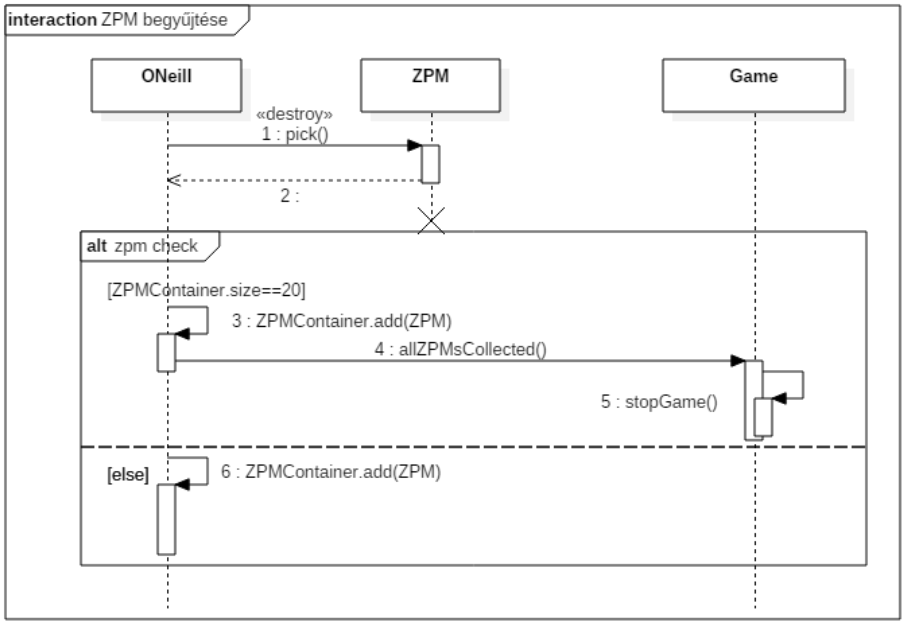

interaction Játék kezdete
Game
«create»
1 :
Dungeon
2 : buildDungeon(File)
3 : Field
«create»
4 :
ONeill
5 : setCurrentTile(Field)
«create»
6 :
FlowOfTime
7 : start()
8 : schedule(Task, gametime)


interaction Játék vége 1
ONeill
Gap
1 : onEntry(void)
2 : destroy(void)
«destroy»


interaction Játék vége 2
FlowOfTime
ONeill
1 : destroy(void)
«destroy»

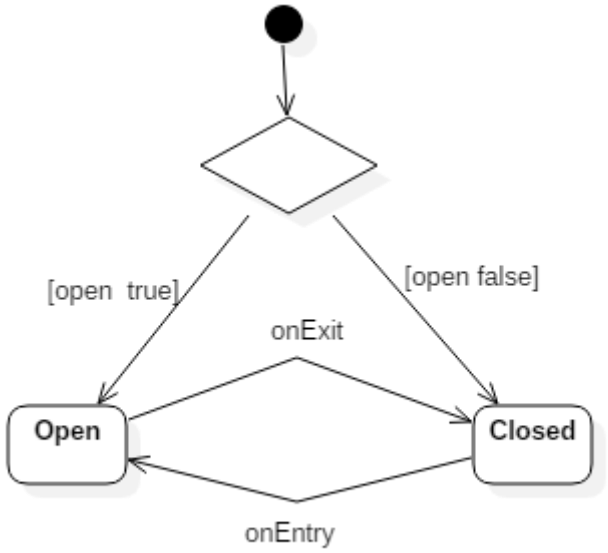

interaction Mozgás
Game
ONeill
Tile
1 : move(Direction)
2 : setFacingDirection(Direction)
3 : getCurrentTile(void)
4 : Tile
5 : getAdjacentTile(Direction)
6 : Tile
7 : onEntry(void)
8 :
9 : setCurrentTile(Tile)
10 :


interaction Teleport
Game
ONeill
Tile
Stargate
other: Stargate
1 : move(Direction)
2 : setCurrentTile(Tile)
3 : onEntry
4 : teleport(Direction)
5 : isOpen
alt IsOpen
[false]
6 : move(Direction)
[true]
7 : getCurrentTile
8 : Tile
9 : setCurrentTile(Tile)


interaction ZPM begyűjtése
ONeill
ZPM
Game
«destroy»
1 : pick()
2 :
alt zpm check
[ZPMContainer.size==20]
3 : ZPMContainer.add(ZPM)
4 : allZPMsCollected()
5 : stopGame()
[else]
6 : ZPMContainer.add(ZPM)

4.5 State-chartok

4.5.1 Gate



4.6 Napló

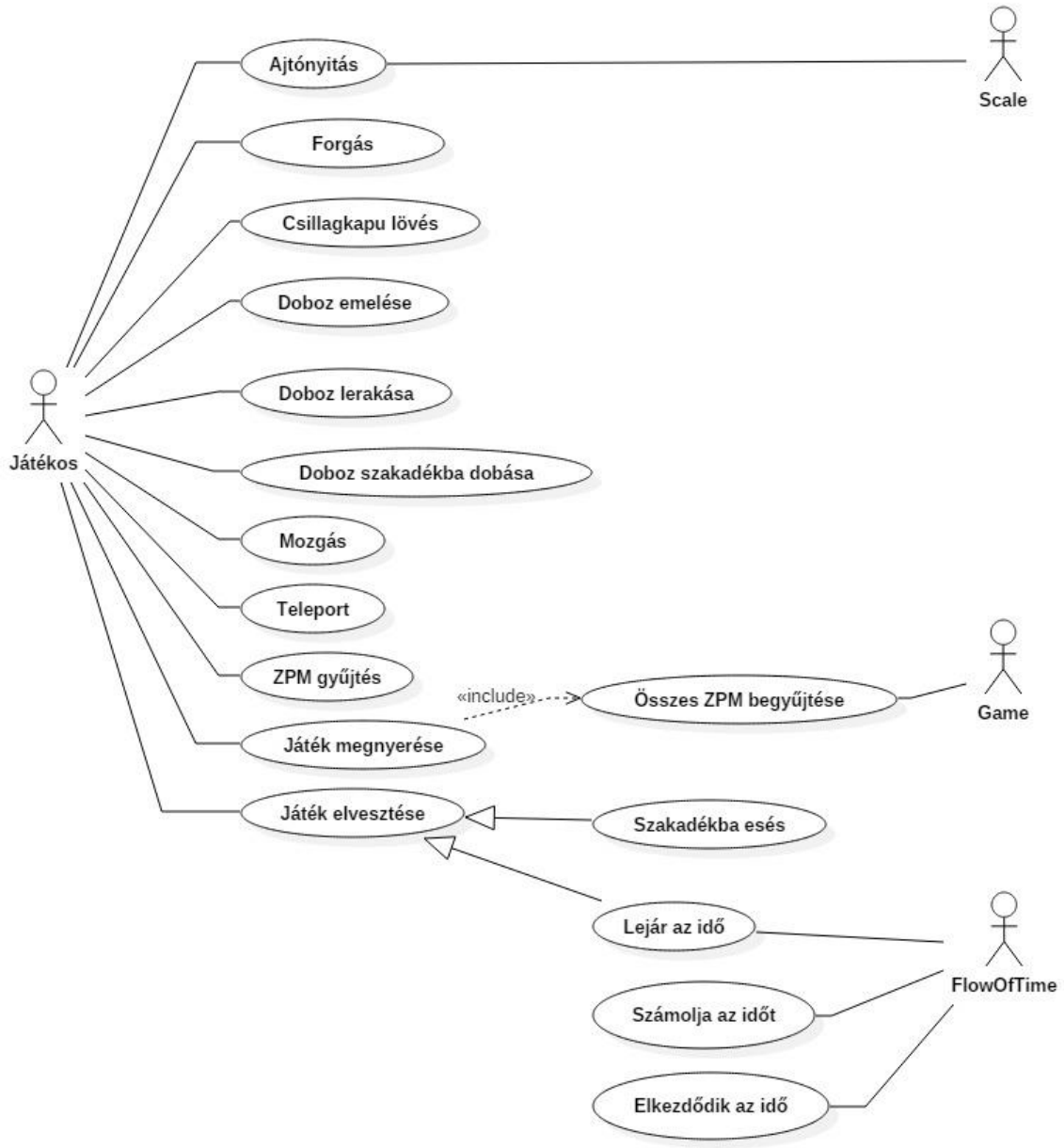
Kezdet	Időtartam	Résztvevők	Leírás
2016.03.12. 12:00	1,5 óra	Bokros Hegedüs Jáhn Siket Tóth Varga	Értekezlet. A modellen végrehajtandó módosítások és bővítések összegyűjtése.
2016.03.12. 13:00	4 óra	Jáhn	Tevékenység. Jáhn konzisztenssé teszi az osztályleírásokat a statikus modellel, illetve a modell hibáinak javításain dolgozik.
2016.03.12. 18:00	1,5 óra	Tóth	Tevékenység. Tóth módosítja a FlowOfTime osztályt, illetve változtatásokat eszközöl az objektum katalóguson.
2016.03.12. 16:00	2 óra	Bokros	Tevékenység. Bokros átalakítja a Stargate osztályt a megbeszélteknek megfelelően, majd a változásokat átvezeti a dokumentumba, és frissíti a többi osztály leírását, hogy azok együtt tudjanak működni a módosított Stargate osztállyal. Bokros elkészíti a Csillagkapu lövéshez tartozó dokumentumot.
2016.03.13 16:30	2 óra	Bokros Tóth	Tevékenység. Tóth és Bokros ellenőrzi az osztálydiagramot, és változásokat eszközöl rajta ott, ahol az szükséges.
2016.03.13. 18:00	20 perc	Tóth	Tevékenység. Tóth elkészíti a játék kezdete szekvenciadiagramot.
2016.03.13 20:00	4 óra	Tóth	Tevékenység. Tóth elkészíti a *Box* osztály szekvenciadiagramjait, szerkeszti az osztálydiagramot, változtatásokat visz végbe a dokumentumban, ill. lektorálja azt.
2016.03.13	20 perc	Bokros	Tevékenység. Bokros elkészíti a Teleport szekvenciadiagramot.
2016.03.13 21:00	15 perc	Jáhn	Tevékenység. Jáhn létrehozza az Ajtónyitás szekvenciadiagrammot.
2016.03.13. 21:30	30 perc	Siket	Tevékenység. Siket aktualizálja és lektorálja az objektum katalógust.
2016.03.13. 21:30	20 perc	Varga	Tevékenység. Varga elkészíti a Játék vége 1 és a Játék vége 2 szekvenciadiagramokat.

2016.03.13. 22:00	20 perc	Varga	Tevékenység. Varga elkészíti a Mozgás szekvenciadiagramot.
2016.03.13. 22:00	30 perc	Jáhn	Tevékenység. Jáhn létrehozza a Destroyable interfészt és módosítja az olyan osztályok leírását, melyek megvalósítják azt.
2016.03.13. 22:00	30 perc	Hegedüs	Tevékenység. Hegedüs módosítja a Box és A ZPM osztály, valamint a Pickable interfész leírását.
2016.03.13. 23:00	1 óra	Siket	Tevékenység. Siket megszerkeszti a Gate osztályhoz tartozó állapotgépet.
2016.03.14 00:00	20 perc	Bokros	Tevékenység. Bokros ellenőrzi és javítja a dokumentum formázását és tördelését.
2016.03.14. 01:00	20 perc	Siket	Tevékenység. Siket lektorálja a dokumentum szöveges részeit és egységesíti a szöveg formázását.
2016.03.14. 8:00	30 perc	Varga	Tevékenység. Varga elkészíti a ZPM begyűjtése szekvenciadiagramot, és átnézi a teljes dokumentumot.

5. Szkeleton tervezése

5.1 A szkeleton modell valóságos use-case-ei

5.1.1 Use-case diagram



5.1.2 Use-case leírások

Use-case neve	**Ajtónyitás**
Rövid leírás	Kinyílik az ajtó.
Aktorok	Játékos, Scale
Forgatókönyv	Ha az ajtóhoz tartozó mérlegre súly kerül (doboz, vagy O'Neill), akkor az kinyílik.

Use-case neve	**Forgás**
Rövid leírás	O'Neill a neki kellő irányba fordul.
Aktorok	Játékos
Forgatókönyv	O'Neill derékszögben tud forogni jobbra vagy balra, hogy arra tudjon nézni amerre kell neki.

Use-case neve	**Csillagkapu lövés**
Rövid leírás	Kilövünk egy csillagkaput.
Aktorok	Játékos
Forgatókönyv	A játékos a megfelelő irányba áll, majd kilő egy csillagkaput, ami ha jó falfelületre érkezik kinyílik.

Use-case neve	**Doboz emelése**
Rövid leírás	O'Neill felemeli a dobozt.
Aktorok	Játékos
Forgatókönyv	O'Neill a megfelelő irányba nézve felveszi a dobozt.

Use-case neve	**Doboz lerakása**
Rövid leírás	O'Neill lerakja a dobozt.
Aktorok	Játékos
Forgatókönyv	O'Neill a neki kellő mezőre lerakja a dobozt.

Use-case neve	**Doboz szakadékba dobása**
Rövid leírás	O'Neill beledobja a szakadékba a dobozt.
Aktorok	Játékos
Forgatókönyv	O'Neill megáll a szakadék mellett, majd beledobja a dobozt.

Use-case neve	**Mozgás**
Rövid leírás	O'Neill haladása.
Aktorok	Játékos
Forgatókönyv	O'Neill a megadott irányba lépked.

Use-case neve	**Teleport**
Rövid leírás	O'Neill az egyik helyről a másikra teleportál.
Aktorok	Játékos
Forgatókönyv	A egyik kilőtt csillagkapun át O'Neill átjut a másik kilőtt csillagkapuba.

Use-case neve	**ZPM gyűjtés**
Rövid leírás	O'Neill begyűjti a ZPM-et.
Aktorok	Játékos
Forgatókönyv	O'Neill a ZPM mellé sétál, irányba áll, majd begyűjti az adott ZPM-et.

Use-case neve	**Játék megnyerése**
Rövid leírás	A játékos megnyeri a játékot.
Aktorok	Játékos

Forgatókönyv	Ha O'Neill az időkereten belül összegyűjti az összes ZPM-et, akkor megnyeri a játékot.

Use-case neve	**Összes ZPM begyűjtése**
Rövid leírás	O'Neill összeszedi az összes ZPM-et.
Aktorok	Játékos, Game
Forgatókönyv	O'Neill összeszedi az összes ZPM-et, ezzel megnyeri a játékot.

Use-case neve	**Játék elvesztése**
Rövid leírás	O'Neill elveszti a játékot.
Aktorok	Játékos
Forgatókönyv	O'Neill elvesztheti a játékot, ha meghal (beleesik a szakadékba) vagy lejár az ideje.

Use-case neve	**Szakadékba esés**
Rövid leírás	O'Neill beleesik a szakadékba
Aktorok	Játékos
Forgatókönyv	O'Neill belesétál a szakadékba és meghal, ezzel elveszíti a játékot.

Use-case neve	**Lejár az idő**
Rövid leírás	A megadott idő letelik.
Aktorok	Játékos, FlowOfTime
Forgatókönyv	Lejár a megadott idő és ezért O'Neill elveszíti a játékot.

Use-case neve	**Számolja az időt**
Rövid leírás	Telik az idő.
Aktorok	FlowOfTime
Forgatókönyv	A megadott idő számlálódik.

Use-case neve	**Elkezdődik az idő**
Rövid leírás	Elkezdődik az idő számlálása.
Aktorok	FlowOfTime
Forgatókönyv	A játék indításakor elkezdi számolni az időt.

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 el lehet indítani a játékot, vagy ki lehet lépni. A felhasználó a játék vége után ide kerül vissza. A játék indítása után egy újabb menü jelenik meg, ebben irányítható O'Neill. Az X gomb megnyomásával mindig vissza lehet lépni ide. A játék indítása után látható menü felépítése a következő:

5.2.1 Lépés

- **Elfogadott bemenet:** W, A, S, D
- **Lehetséges kimenet:**
  - O'Neill [északi|nyugati|déli|keleti] irányba lép egyet, …

        - egy egyszerű mezőre,…
            - ami üres
            - amelyen egy [dobozt|ZPM-et] talál.
        - egy szakadékba és meghal.
        - egy nyitott kapura, amelyen sikeresen áthalad.
        - egy mérlegre, amellyel kinyit egy ajtót.
        - egy falra, amelyen [sárga|kék] csillagkapu van, így
            - átlép a hipertéren és a [kék|sárga] csillagkapu mellett landol
            - nem tud átlépni a hipertéren, mert a csillagkapu zárva van
    - O'Neill [északi|nyugati|déli|keleti] irányba lépne, de
        - egy zárt kapu útját állja.
        - egy fal útját állja.

5.2.2 Doboz felvétele

- **Elfogadott bemenet:** L
- **Lehetséges kimenet:**
    - O'Neill felvette a dobozt a tőle [északra|nyugatra|délre|keletre] lévő mezőről.
    - O'Neill nem vett fel dobozt, mivel az nem található meg a tőle [északra|nyugatra|délre|keletre] lévő mezőn.

5.2.3 Doboz lerakása

- **Elfogadott bemenet:** D
- **Lehetséges kimenet:**
    - O'Neill letette a dobozt a tőle [északra|nyugatra|délre|keletre] lévő mezőre.
    - O'Neill nem tette le a dobozt, mivel az nem tehető meg a tőle [északra|nyugatra|délre|keletre] lévő mezőn…
        - A mezőn jelenleg egy [másik doboz|ZPM] van.
        - A mező egy [fal|zárt kapu].

5.2.4 Elforgás

- **Elfogadott bemenet:** L, R
- **Lehetséges kimenet:**
    - O'Neill elfordult, [észak|nyugat|dél|kelet] felé néz, előtte egy [egyszerű|szakadék|kapu|mérleg|fal] mező látható,…
        - amelyen (egy) [doboz|ZPM|sárga csillagkapu|kék csillagkapu|semmi sem] található (, ami átjárható).

5.2.5 Nézés

- **Elfogadott bemenet:** W
- **Lehetséges kimenet:**
    - O'Neill előtt egy [egyszerű|szakadék|kapu|mérleg|fal] mező látható,…
        - amelyen (egy) [doboz|ZPM|sárga csillagkapu|kék csillagkapu|semmi sem] található (, ami átjárható).

5.2.6 Csillagkapu lövés

- **Elfogadott bemenet:** Y, B
- **Lehetséges kimenet:**

- A lövés irányában következő mező [egyszerű|szakadék|nyitott kapu|mérleg], a csillagkapu tovább halad a következő mezőre, és újabb választás következik a lehetséges kimenetelek közül.
- A lövés irányában következő mező [zárt kapu|fal, amin van csk], a csillagkapu lepattan, és a hipertér nem nyílik meg.
- A lövés irányában következő mező fal, amin nincs csillagkapu, így a kilőtt csillagkapu megjelenik rajta, és amennyiben a párja is ki van lőve, meg is nyílik a hipertér.

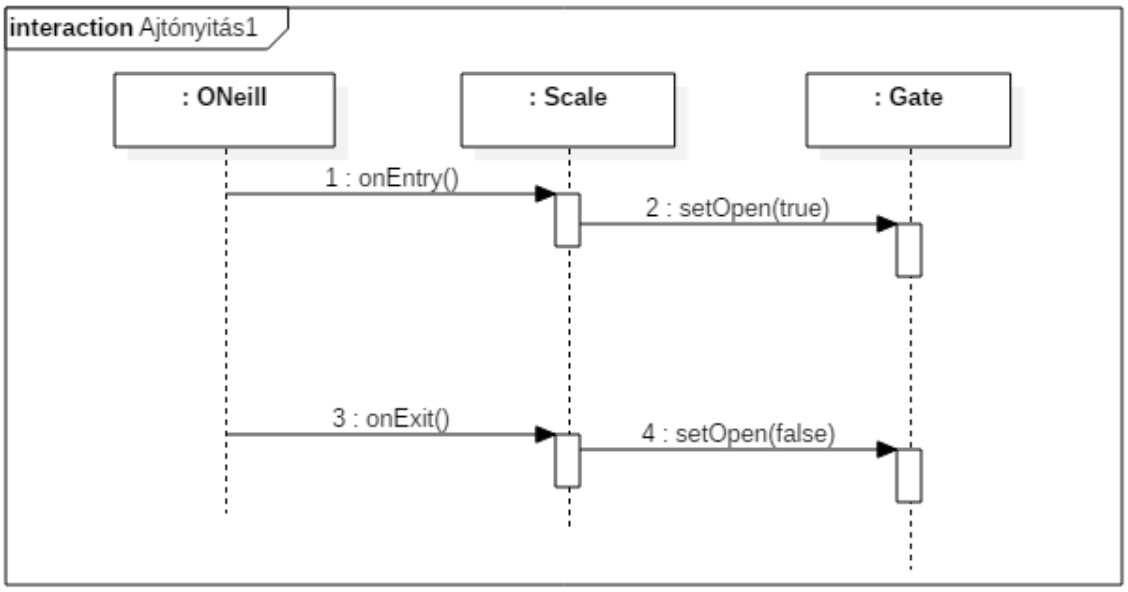
Mivel a szkeletonban a program váza mögött még nincsen ott az azt működtető logika, ezért a [opció1|…|opcióN] formában jelölt elágazások közti döntés a felhasználó feladata. Ehhez egy menüpont kiválasztása után az elfogadott bemenetek közti gombok közül valamelyik megnyomásával értelemszerűen lehet választani, majd a menüpontban elmerülve a további döntési helyzetekben az opciók sorszámával lehet választani. Eközben a képernyőn kiírásra kerülnek folyamatosan a hívott utasítások > jelöléssel, amennyiben meghívásra kerülnek és < jelöléssel amennyiben visszatérnek. Döntési helyzetben a ? jel jelzi, hogy a program felhasználói bemenetre vár.

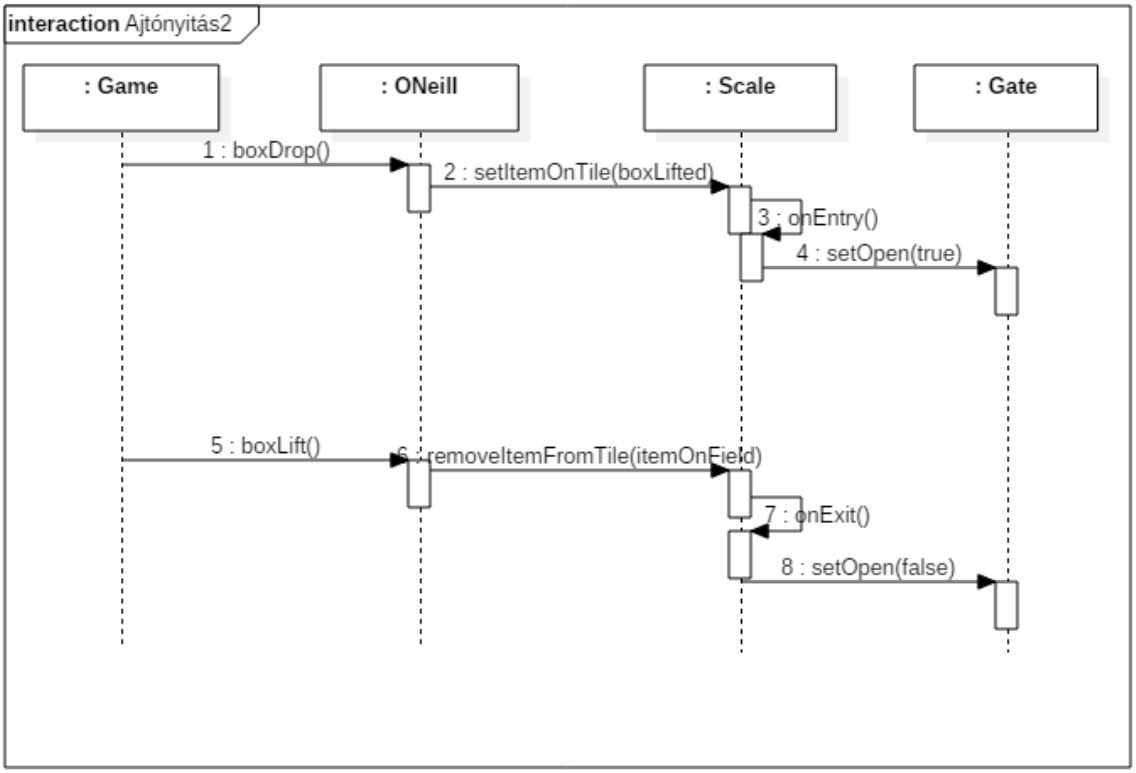
Példa az Lépés interakcióra:

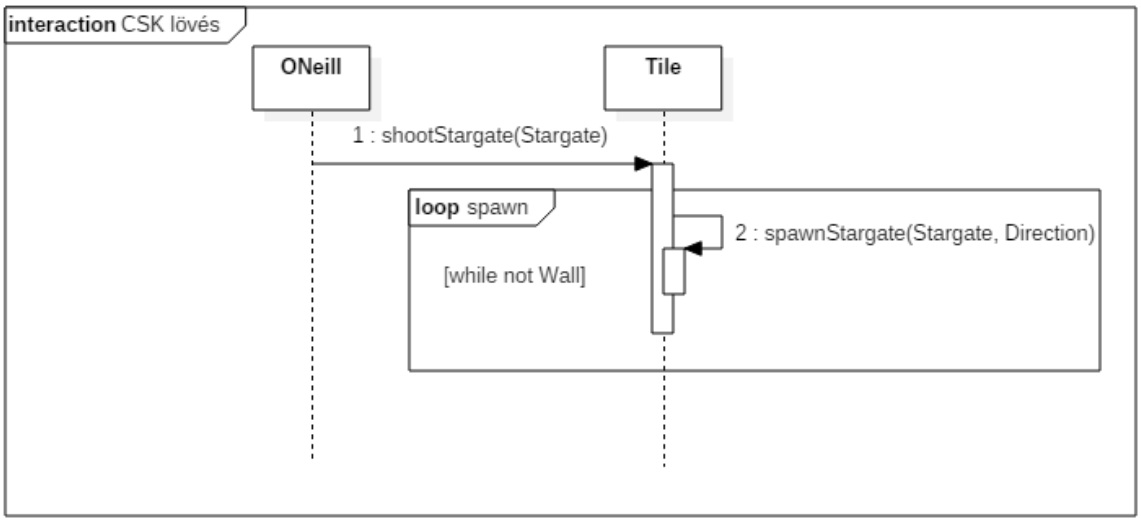
```
? Válassz a menüből: 1
– 1. Lépés
? Milyen irányba?
– W
? Milyen mezőre? [egyszerű|szakadék|kapu|mérleg|fal]
– 1
? Található valami a mezőn? [doboz|ZPM|semmi]
– 3
> [:Game].move(NORTH)
>       [:ONeill].setFacingDirection(NORTH)
<       [:ONeill].setFacingDirection(NORTH)
>       [currentTile:Tile].getAdjacentTile(NORTH)
<       [currentTile:Tile].getAdjacentTile(NORTH) : adjacent
>       [adjacent:Tile].onEntry()
>               [:ONeill].setCurrentTile(adjacent)
<               [:ONeill].setCurrentTile(adjacent)
<       [adjacent:Tile].onEntry()
< [:Game].move(NORTH)
```

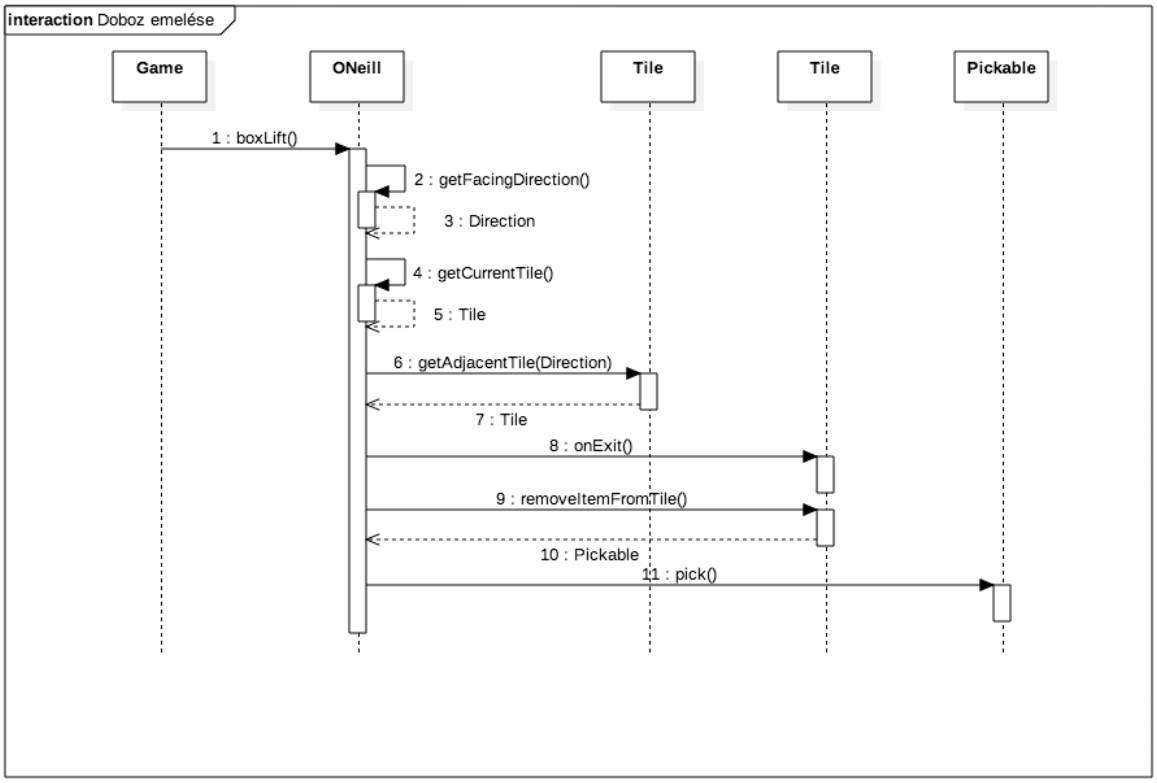
5.3 Szekvencia diagramok a belső működésre

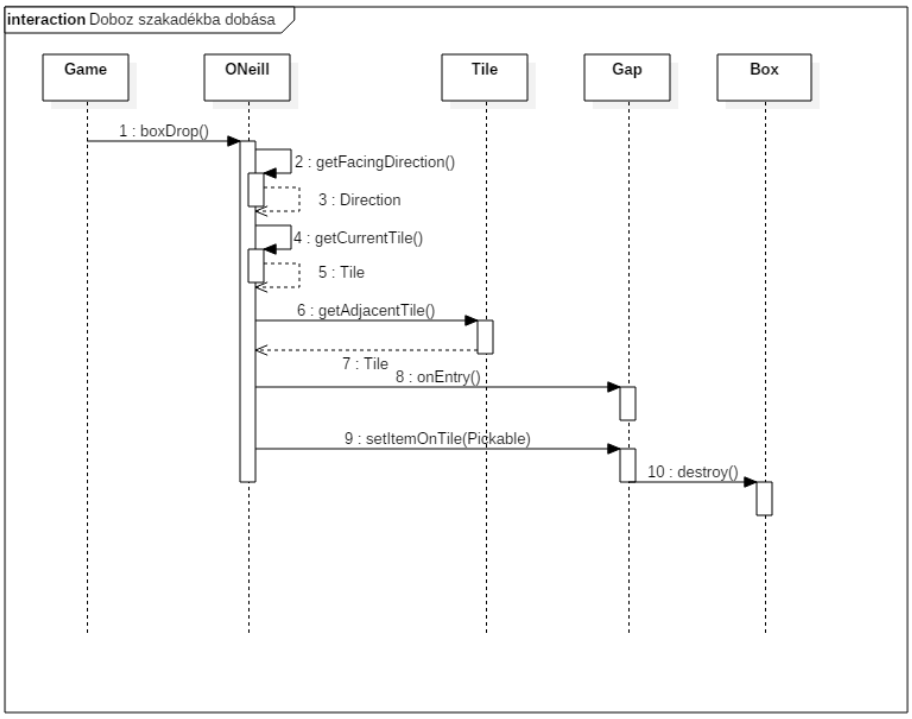
Az előző fejezet diagramjai közül a legtöbb továbbra is érvényes, amelyiken szükséges volt javítások történtek.

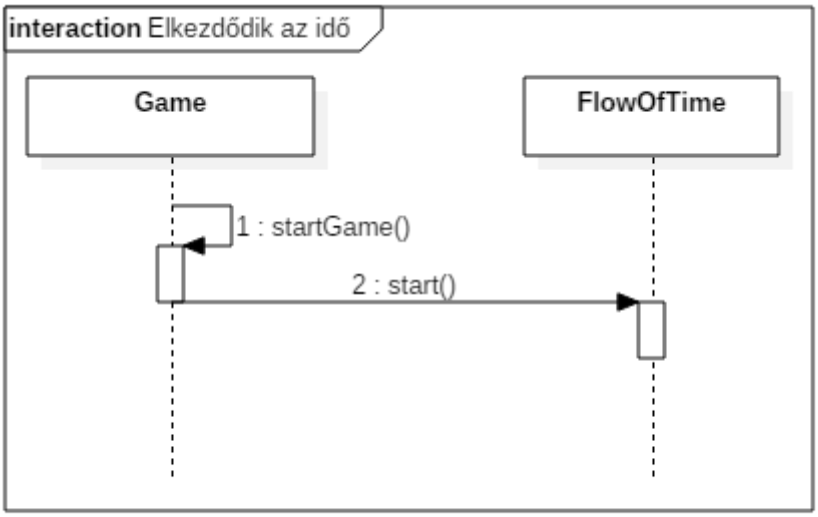

interaction Ajtónyitás1
: ONeill
: Scale
: Gate
1 : onEntry()
2 : setOpen(true)
3 : onExit()
4 : setOpen(false)

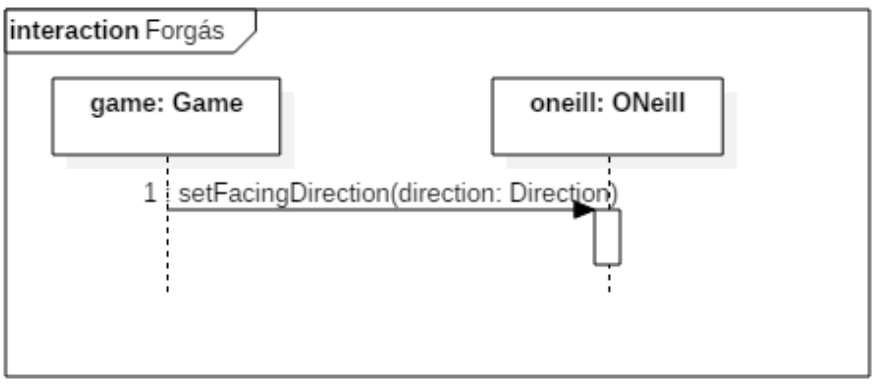

interaction Ajtónyitás2
: Game
: ONeill
: Scale
: Gate
1 : boxDrop()
2 : setItemOnTile(boxLifted)
3 : onEntry()
4 : setOpen(true)
5 : boxLift()
6 : removeItemFromTile(itemOnField)
7 : onExit()
8 : setOpen(false)

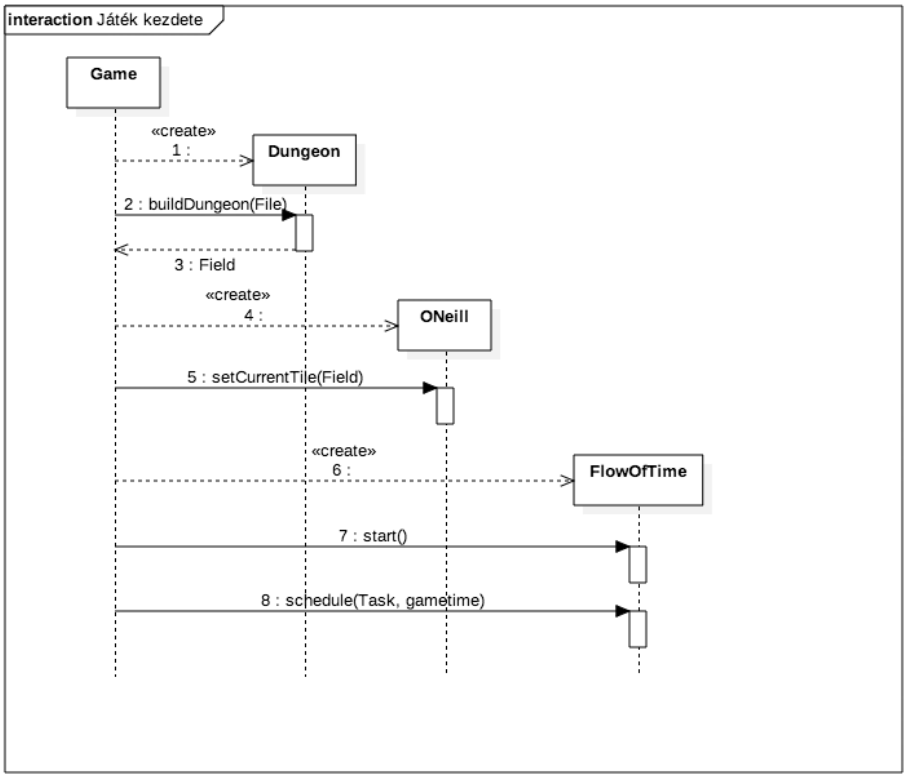

interaction CSK lövés
ONeill
Tile
1 : shootStargate(Stargate)
loop spawn
[while not Wall]
2 : spawnStargate(Stargate, Direction)

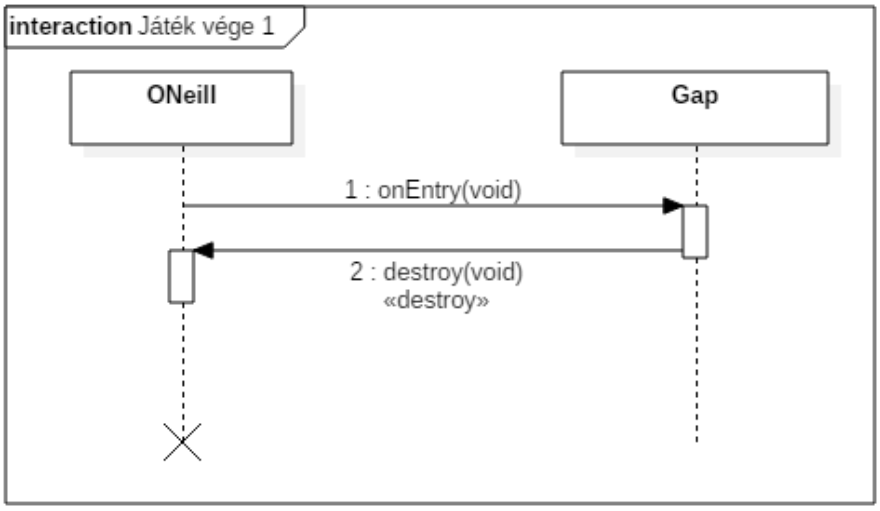

interaction Doboz emelése
Game
ONeill
Tile
Tile
Pickable
1 : boxLift()
2 : getFacingDirection()
3 : Direction
4 : getCurrentTile()
5 : Tile
6 : getAdjacentTile(Direction)
7 : Tile
8 : onExit()
9 : removeItemFromTile()
10 : Pickable
11 : pick()

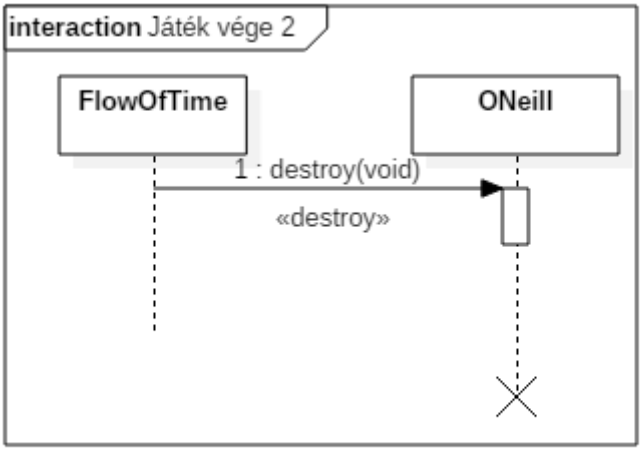

interaction Doboz szakadékba dobása
Game
ONeill
Tile
Gap
Box
1 : boxDrop()
2 : getFacingDirection()
3 : Direction
4 : getCurrentTile()
5 : Tile
6 : getAdjacentTile()
7 : Tile
8 : onEntry()
9 : setItemOnTile(Pickable)
10 : destroy()

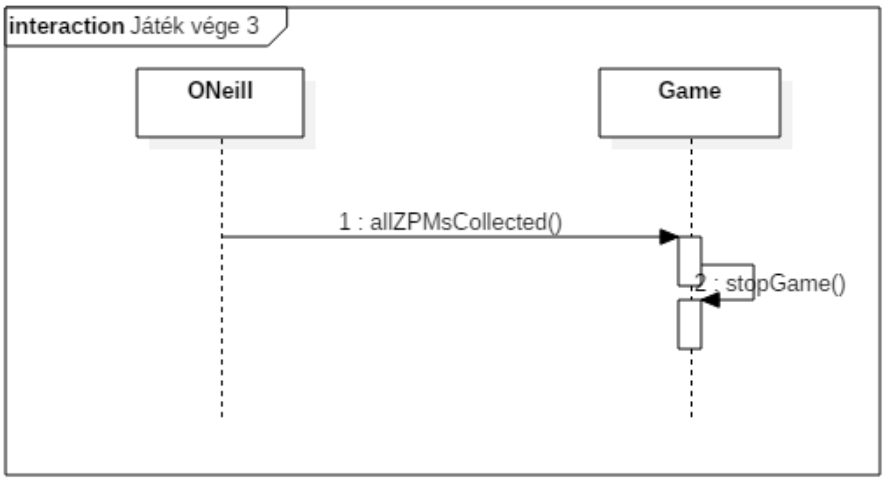

interaction Elkezdődik az idő
Game
FlowOfTime
1 : startGame()
2 : start()

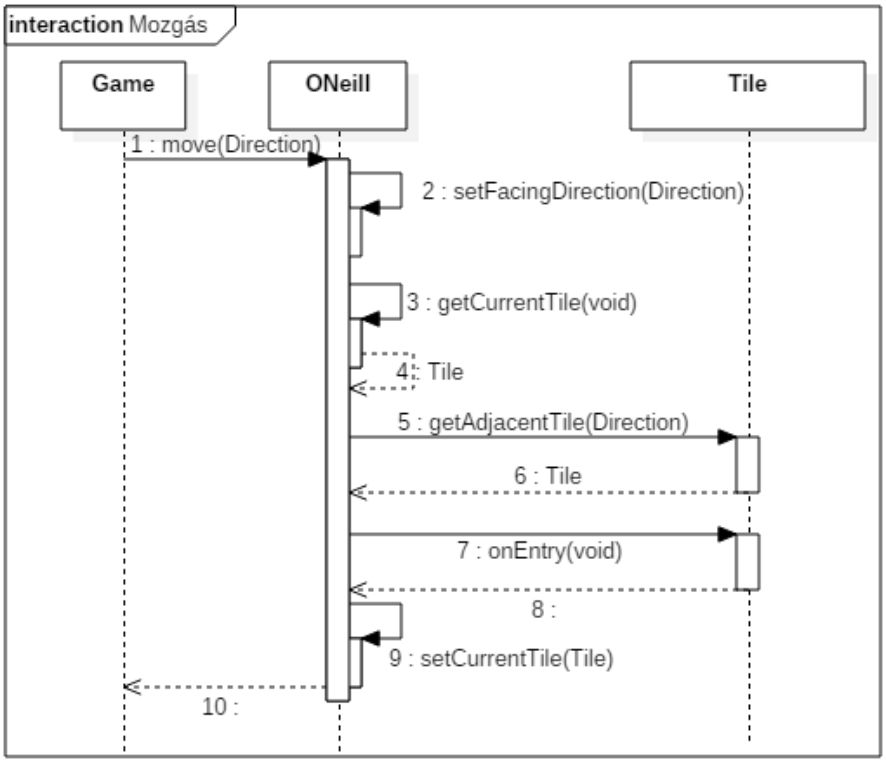

interaction Forgás
game: Game
oneill: ONeill
1 : setFacingDirection(direction: Direction)

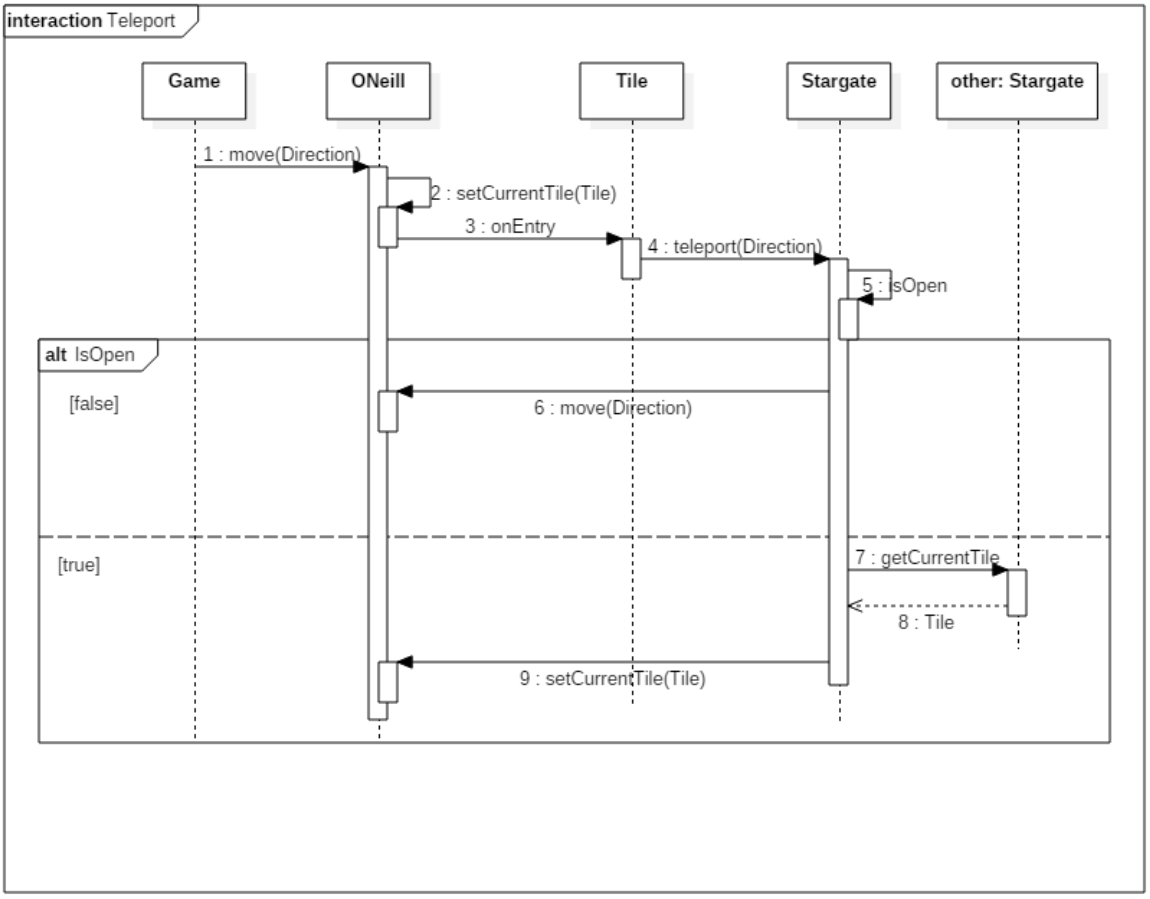

interaction Játék kezdete
Game
«create»
1 :
Dungeon
2 : buildDungeon(File)
3 : Field
«create»
4 :
ONeill
5 : setCurrentTile(Field)
«create»
6 :
FlowOfTime
7 : start()
8 : schedule(Task, gametime)

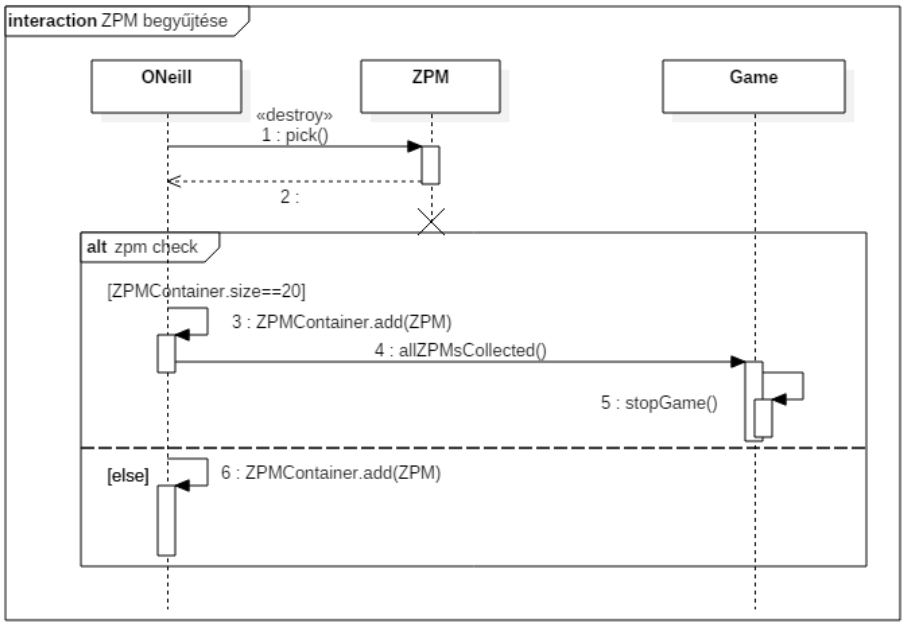

interaction Játék vége 1
ONeill
Gap
1 : onEntry(void)
2 : destroy(void)
«destroy»


interaction Játék vége 2
FlowOfTime
ONeill
1 : destroy(void)
«destroy»

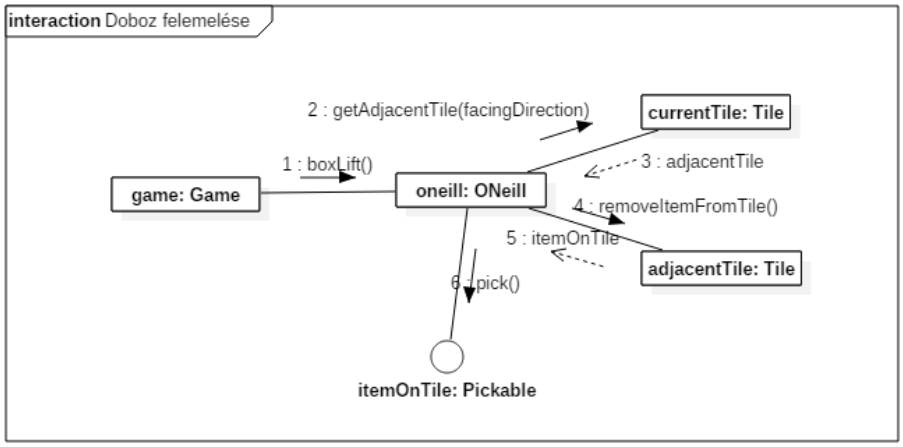

interaction Játék vége 3
ONeill
Game
1 : allZPMsCollected()
2 : stopGame()

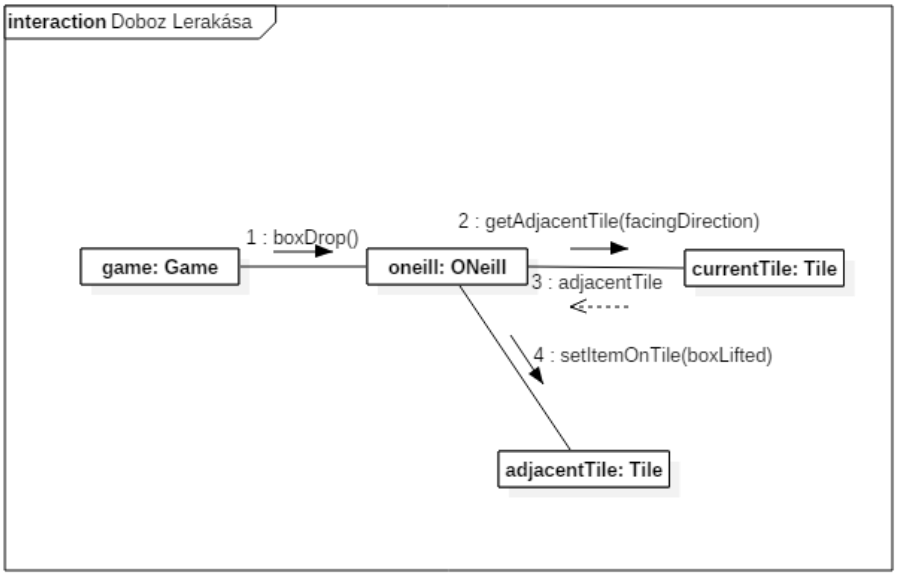

interaction Mozgás
Game
ONeill
Tile
1 : move(Direction)
2 : setFacingDirection(Direction)
3 : getCurrentTile(void)
4 : Tile
5 : getAdjacentTile(Direction)
6 : Tile
7 : onEntry(void)
8 :
9 : setCurrentTile(Tile)
10 :

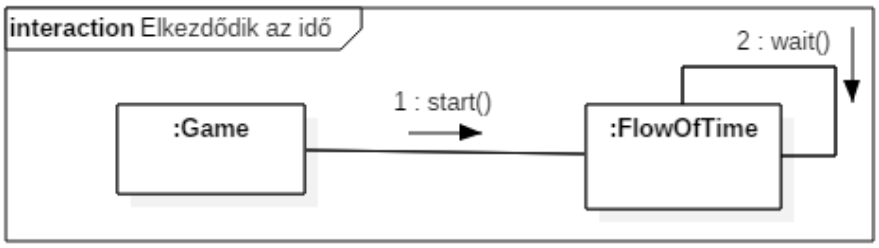

interaction Teleport
Game
ONeill
Tile
Stargate
other: Stargate
1 : move(Direction)
2 : setCurrentTile(Tile)
3 : onEntry
4 : teleport(Direction)
5 : isOpen
alt IsOpen
[false]
6 : move(Direction)
[true]
7 : getCurrentTile
8 : Tile
9 : setCurrentTile(Tile)

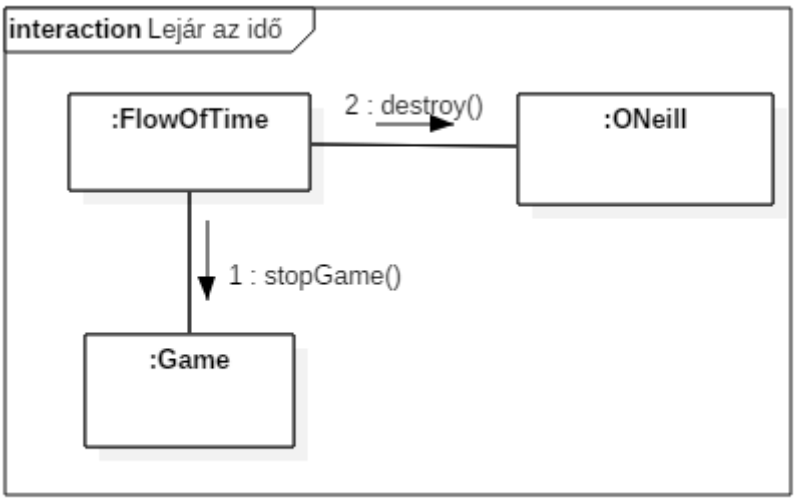


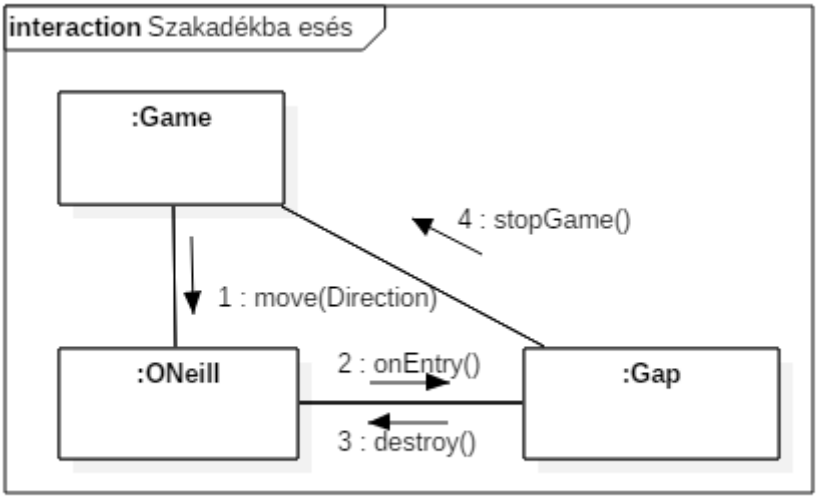
5.4 Kommunikációs diagramok

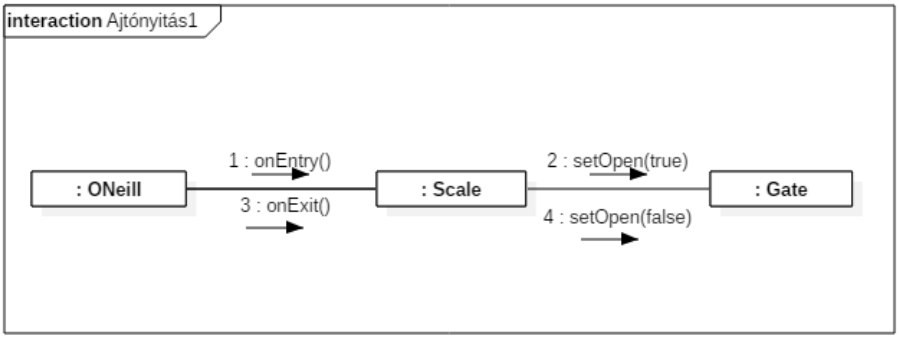


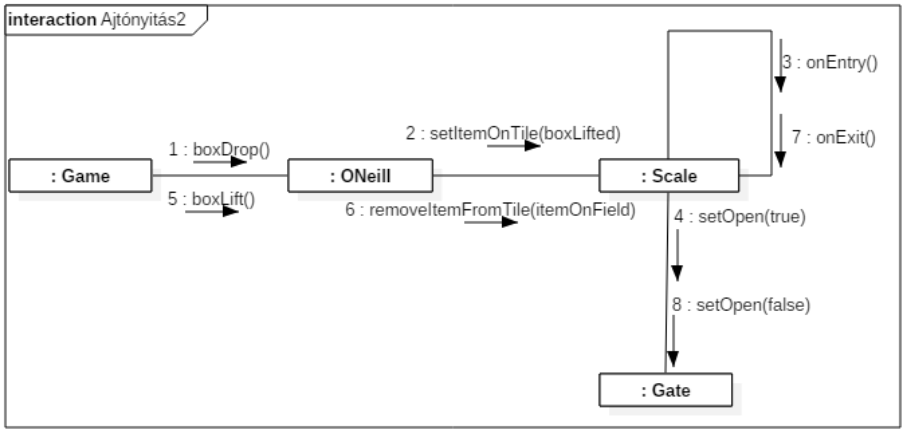

interaction Doboz Lerakása
1 : boxDrop()
2 : getAdjacentTile(facingDirection)
game: Game
oneill: ONeill
currentTile: Tile
3 : adjacentTile
4 : setItemOnTile(boxLifted)
adjacentTile: Tile

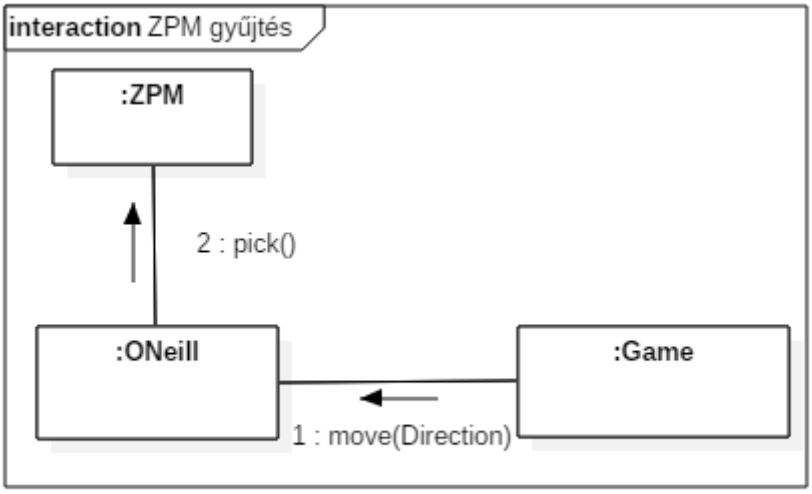

interaction Elkezdődik az idő
2 : wait()
1 : start()
:Game
:FlowOfTime


interaction Lejár az idő
:FlowOfTime
2 : destroy()
:ONeill
1 : stopGame()
:Game

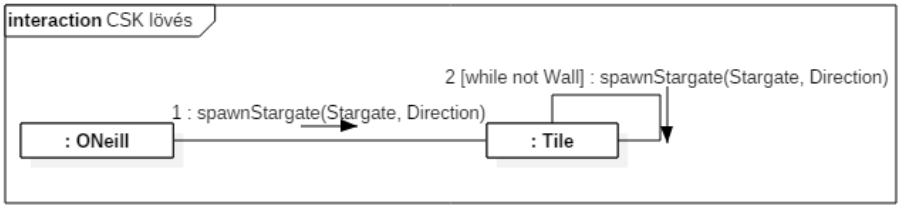

interaction Szakadékba esés
:Game
4 : stopGame()
1 : move(Direction)
:ONeill
2 : onEntry()
:Gap
3 : destroy()

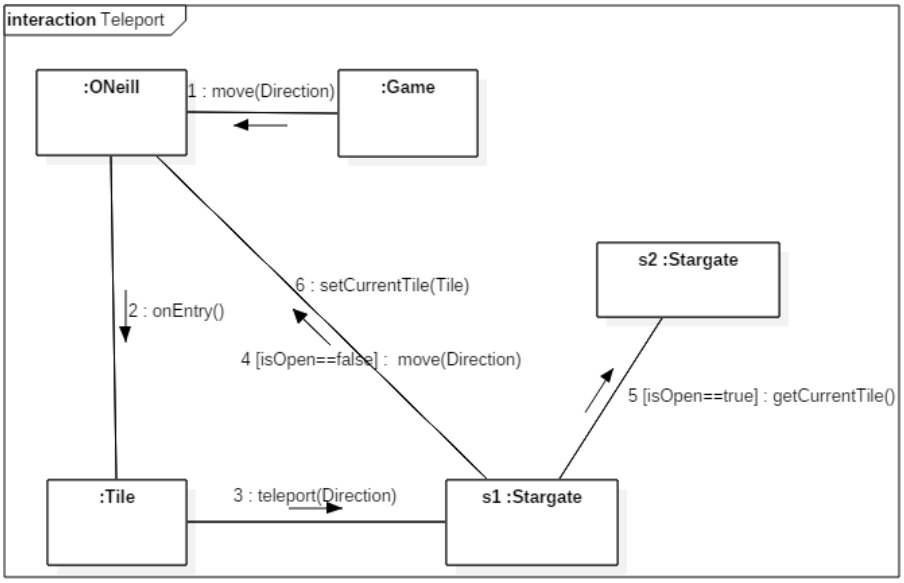

interaction Ajtónyitás1
: ONeill
1 : onEntry()
3 : onExit()
: Scale
2 : setOpen(true)
4 : setOpen(false)
: Gate


interaction Ajtónyitás2
: Game
1 : boxDrop()
5 : boxLift()
: ONeill
2 : setItemOnTile(boxLifted)
6 : removeItemFromTile(itemOnField)
: Scale
3 : onEntry()
7 : onExit()
4 : setOpen(true)
8 : setOpen(false)
: Gate


interaction ZPM gyűjtés
:ZPM
2 : pick()
:ONeill
:Game
1 : move(Direction)


interaction Összes ZPM begyűjtése
2 : stopGame()
:ONeill
1 : allZPMsCollected()
:Game


interaction CSK lövés
2 [while not Wall] : spawnStargate(Stargate, Direction)
1 : spawnStargate(Stargate, Direction)
: ONeill
: Tile


interaction Teleport
:ONeill
1 : move(Direction)
:Game
s2 :Stargate
6 : setCurrentTile(Tile)
2 : onEntry()
4 [isOpen==false] : move(Direction)
5 [isOpen==true] : getCurrentTile()
:Tile
3 : teleport(Direction)
s1 :Stargate

5.5 Napló

Kezdet	Időtartam	Résztvevők	Leírás
2016. 03. 16. 16:15	60 perc	Bokros Jáhn Hegedüs Siket Tóth Varga	Értekezlet. A konzultáción hallottak ismertetése, a heti feladat értelmezése.
2016. 03. 19. 18:34	70 perc	Bokros Jáhn Tóth Varga	Értekezlet. A feladat részleteinek értelmezése, az egyes feladatok szétosztása.
2016.03.20. 18:30	4 óra	Bokros	Tevékenység. A szkeleton kezelői felületének terve, dialógusok kidolgozása és dokumentálása.
2016.03.20. 18:00	2,5 óra	Hegedüs	Tevékenység. 5.1-es fejezet kidolgozása.
2016.03.20. 19:00	2 óra	Tóth	Tevékenység. Tóth beviszi a dokumentumba az összes elkészült szekvenciadiagramot, majd lektorálja azt.
2016. 03. 20. 11:00	2 óra	Varga	Tevékenység: Varga elkészíti a kommunikációs diagramok felét..
2016. 03. 20. 21:00	1,5 óra	Jáhn	Tevékenység. Jáhn elkészíti a kommunikációs diagramok egy részét.
2016. 03. 21. 01:30	1 óra	Siket	Tevékenyég. Siket elkészíti a kommunikációs diagramok egy részét.
2016. 03. 21. 2:40	30 perc	Siket	Tevékenység. Siket lektorálja a dokumentum eddig elkészített részeit.

6. Szkeleton beadás

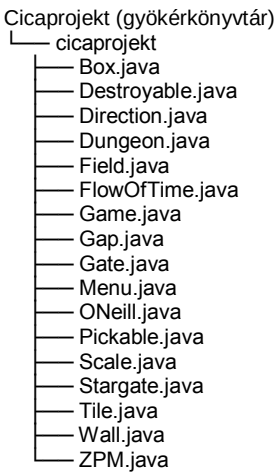
6.1 Fordítási és futtatási útmutató

6.1.1 Fájllista

Fájl neve	Méret	Keletkezés ideje	Tartalom
Box.java	700 bájt	2016.03.28.	Box osztály.
Destroyable.java	84 bájt	2016.03.28.	Destroyable osztály.
Direction.java	80 bájt	2016.03.28.	Direction osztály.
Dungeon.java	479 bájt	2016.03.28.	Dungeon osztály.
Field.java	1.18 KB	2016.03.28.	Field osztály.
FlowOfTime.java	489 bájt	2016.03.28.	FlowOfTime osztály.
Game.java	1.13 KB	2016.03.28.	Game osztály.
Gap.java	1.16 KB	2016.03.28.	Gap osztály.
Gate.java	1.84 KB	2016.03.28.	Gate osztály.
ONeill.java	4.87 KB	2016.03.28.	O'Neill oszály.
Pickable.java	98 bájt	2016.03.28.	Pickable interface.
Scale.java	1.55 KB	2016.03.28.	Scale osztály.
Stargate.java	2.06 KB	2016.03.28.	Stargate osztály.
Tile.java	1.83 KB	2016.03.28.	Tile osztály.
Wall.java	1.46 KB	2016.03.28.	Wall osztály.
ZPM.java	677 bájt	2016.03.28.	ZPM osztály.

6.1.2 Fordítás

A projekt könyvtárszerkezete a következő:



A szoftver fordítását a forráskódok gyökérkönytárába lépve lehet elvégezni. A fordításhoz a *javac cicaprojekt/** parancsot kell kiadni, aminek hatására létrejönnek a futtatható .class állományok.

Futtatás

A fordítást követően a .class állományok futtatása a *java cicaprojekt.Menu* parancs kiadásával történik (természetesen ezt is a forráskódok gyökérkönyvtárából). Ez után a parancssorból lehet használni a szoftvert.

6.2 Értékelés

Tag neve	Munka százalékban
Bokros Bálint Botond	17
Hegedüs Fanni	16
Jáhn Erik	17
Siket Melinda Tekla	16
Tóth Kristóf	17
Varga Péter	16

6.3 Napló

Kezdet	Időtartam	Résztvevők	Leírás
2016.03.28. 00:00	2 óra	Jáhn	Tevékenység. Jáhn elkészíti a szkeleton rá kirótt részét.
2016.03.28. 18:00	1 óra	Tóth Hegedűs Bokros	Értekezlet. A résztvevők megbeszélik a további teendőket
2016.03.28. 19:30	30 perc	Bokros	Tevékenység: Bokros helyreállítja a repot.
2016.03.28. 23:00	4 óra	Bokros Tóth	Tevékenység: Bokros és Tóth lekódolják a szkeleton osztályainak rájuk kirótt részét, lektorálják a többiek munkáját, illetve javításokat végeznek azon. Eközben egyeztetnek, hogy a kód konzisztens maradjon.
2016.03.28. 23:30	2 óra	Hegedüs	Tevékenység: Hegedüs lekódolja a szkeleton osztályainak rá kirótt részét.
2016.03.29.	1,5 óra	Bokros	Tevékenység. Bokros kiegészíti a programot ott, ahol az még hiányos.
2016.03.29. 11:00	1 óra	Tóth Jáhn Bokros	Tevékenység. A szkeleton hibáinak javítása.

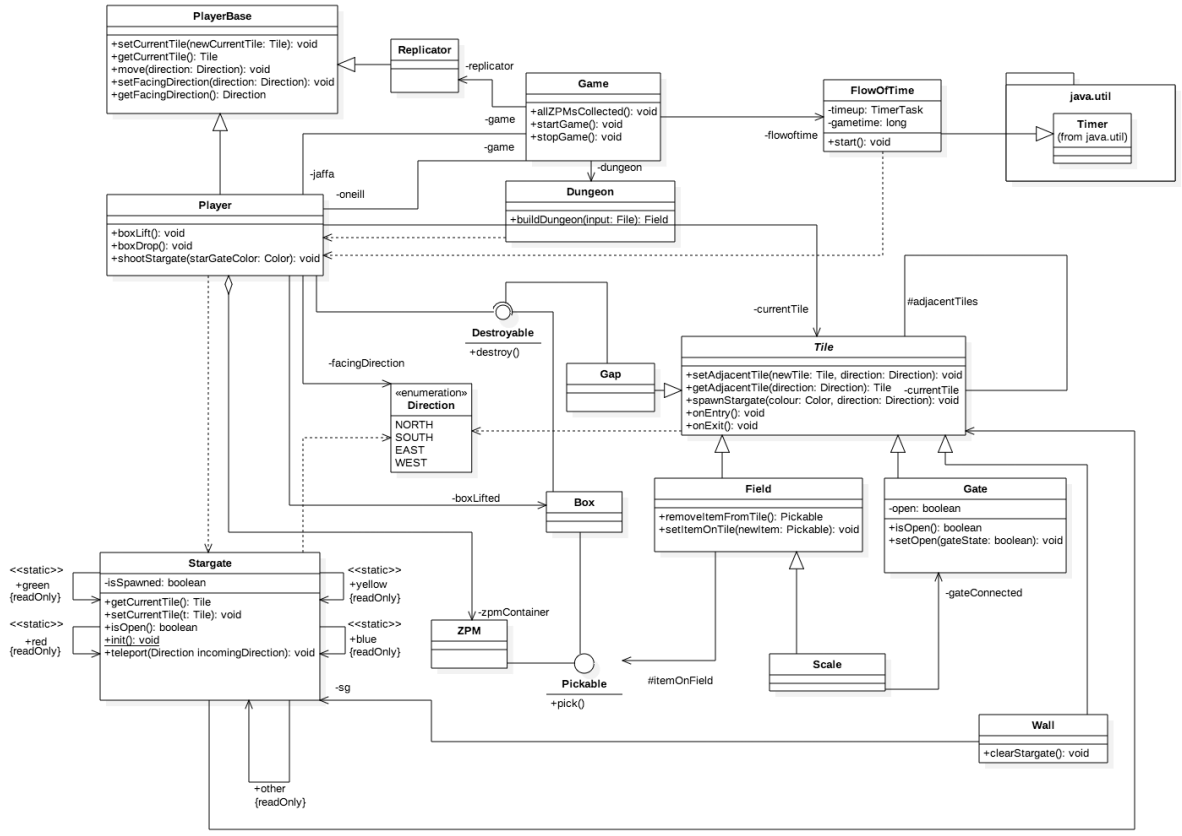
7. Prototípus koncepciója

7.0 Követelmények változása

A szoftver specifikációjában változások léptek életbe, ezeket a következőkben fejtjük ki:

1. **ZPM mechanizmus**
   A ZPM modulok felvétele olyan módon változott, hogy mostantól minden második felvett ZPM után a pályán valahol egy új keletkezik. Természetesen ez csak olyan helyre történhet, ahol az eddigiek szerint lehet ZPM, illetve ez a ciklus egyszer nyilván véget ér, mivel ($Z + \sum_{\lfloor \frac{Z}{2} \rfloor}(1) - \sum_{ZPM\ felvételek\ száma}(1)$), – ahol Z a pályán lévő ZPM modulok száma – triviálisan 0, amennyiben elég nagy a *ZPM felvételek száma*: ez a játék végére persze teljesül.
2. **Replikátor**
   Mostantól minden pályán – véletlenszerű pozícióból indulva – egy replikátor mozog. Mozgásának sebessége és irány előre nem definiált: Úgy lesz megválasztva, hogy a játékos számára élvezhető legyen. A replikátor csak olyan mezőkön tartózkodhat, ahol O'Neill ezredes is, ugyanakkor semmilyen a virtuális világban lévő objektummal nem konfrontálódik, így például ráléphet egy mezőre, ahol egy doboz van. Ha a replikátort eltalálja egy csillagkapu, akkor eltűnik. Amennyiben egy szakadékra lép (*Gap* objektum), akkor szintén eltűnik, illetve a szakadék sima, járható mezővé alakul (*Field* objektum).
3. **Jaffa**
   A játékban mostantól O'Neill ellenfele egy Jaffa, akinek a célja megegyezik az ezredesével: összegyűjteni az összes ZPM-et. Ő szintén tud csillagkapukat lőni, melyek a színüket leszámítva – melyek a piros illetve a zöld – teljesen megegyeznek O'Neill-évél. Az, hogy kinek a csillagkapujába ki léphet be előre nem definiált, ez a későbbiekben változhat. A Jaffát egy második játékos irányítja valamilyen módon, így a replikátortól eltérően nem „magától" cselekszik.
4. **Objektumok súlya**
   A dobozoknak, az ezredesnek, illetve a Jaffának mostantól súlya is van. Egy mérleg – amire mostantól stack szerűen több dolog is felpakolható és levehető – csak akkor nyílik ki, ha a rajta lévő objektumok összsúlya legalább annyi, mint egy minden mérleghez előre hozzárendelt súlyhatár.

Változások integrálása a projektbe:
Az új funkciók a projekbe való beépítéséhez szükséges néhány apróbb változtatást eszközölni az objektummodellen. A megváltozott követelményeket ezen változások nélkül is tudta volna teljesíteni a projekt, ezt akkor fontoltuk volna meg, hogyha már nagy mennyiségű kód meg lenne írva – ami persze kevésbé tűrné a változó modellt –, de mivel ez a helyzet nem áll fenn, probléma nélkül vihetünk be változtatásokat (egy ilyen kis projekt esetében ha a teljes kód meg lenne írva, egy kultúrált IDE akkor is percek alatt átvinne egy ilyen szintű refactort).
Válzotások az objektummodellben:
A játékos által irányított karakternek megfeleltethető osztály eddig az *ONeill* volt. Mivel mostmár nem csak az ezredes irányítható a játékban, így ez az osztály gyakorlatilag ugyan az, mint eddig volt, csak *Player* néven (hogy többetmondó legyen a neve). Az ezredes és a Jaffa ennek példányai. Ami egy fokkal komolyabb változás, hogy a *Player* osztály alapvető funkcióit mostantól egy *PlayerBase* nevű ősosztály valósítja meg. Alapvető funkcióknak ebben az esetben minden a pályán való mozgáshoz kapcsolható metódust és attribútumot tekintünk. Ez azért jó, mert a megváltozott követelményekben szereplő replikátornak is szüksége van az alapvető mozgás képességére, így az ezt modellező új *Replicator* osztály a *PlayerBase* leszármazottja, hogy elkerüljünk mindenféle kódduplikációt. Az új osztálydiagram (az összes változás a bal felső sarokban látható):



7.1 Prototípus interface-definíciója

7.1.1 Az interfész általános leírása

Az interfész sztenderd bemenetről fogad parancsokat, és a sztenderd kimenetre írja azt, ami arra érdemesnek ítéltetik, hogy kiírásra kerüljön. Ennek köszönhetően terminálból vezérelhető, valamint megoldható az automatizált tesztelés, amennyiben egy előre elkészített

bemeneti fájlt eresztünk az inputra, majd az ennek hatására keletkező outputot egy másik fájlba irányítjuk. Az így létrejött fájl utána segédprogramok (pl. diff) segítségével összevethető az adott bemenetre elvárt kimeneti fájllal. Egy teszt akkor sikeres, ha nincs különbség a két fájl között (whitespace elfogadható).

7.1.2 Bemeneti nyelv

A bemeneti nyelv két részből áll, első része a parancs, ezt követheti egy szóköz és egy kiegészítő utasítás, amely az opcióknál, szögletes zárójelek között található. A szögletes zárójelek közötti utasítások közül, amennyiben a | jellel vannak elválasztva, csak az egyik használandó, a felhasználás céljának megfelelően. Amennyiben az opcióknál csak egy – jel található, úgy a parancshoz nem tartozik kiegészítő utasítás. Nemlétező parancsparaméter esetén a kimeneten a "Hiba: Nemlétező parancs!" szöveg jelenik meg.

boxDrop

Leírás: Az O'Neill/Jaffa kezében lévő doboz eldobása az előtte található mezőre
Opciók: O'Neill/Jaffa – [O|J]

boxLift

Leírás: Az O'Neill/Jaffa felveszi az előtte lévő mezőről a tetején lévő dobozt, amennyiben van ilyen
Opciók: O'Neill/Jaffa – [O|J]

loadMap

Leírás: térkép betöltése
Opciók: a betöltendő pálya térképe

listAdjacentFields

Leírás: dobozok listázása
Opciók: O'Neill/Jaffa – [O|J]

listBoxes

Leírás: dobozok listázása
Opciók: –

listStargates

Leírás: csillagkapuk listázása
Opciók: –

listStargates

Leírás: csillagkapuk listázása
Opciók: –

listZPMs

Leírás: ZPM-ek listázása
Opciók: –

move

Leírás: lépés az aktuális irányba
Opciók: O'Neill/Jaffa – [O|J]

rotate
Leírás: O'Neill/Jaffa forgatása jobbra/balra
Opciók: O'Neill/Jaffa; balra/jobbra – [O|J][L|R]

shootONeillsGun
Leírás: Lövés O'Neill fegyverével az aktuális irányba
Opciók: kék/sárga – [B|Y]

shootJaffasGun
Leírás: Lövés O'Neill fegyverével az aktuális irányba
Opciók: piros/zöld – [R|G]

Mappafájl nyelvtana
A pályát descartes-i koordinátarendszerben írjuk le, ahol X és Y a koordináták és 0;0 a bal felső sarok.
A pályaleíró formátuma: pontosvesszővel tagolt CSV fájl, amelynek első sora így néz ki:
`Width;Length;ONeillX;ONeillY;JaffaX;JaffaY;ReplicatorX;ReplicatorY`
ahol az első két paraméter pálya szélességét és magasságát tárolja, a többi pedig az azon mozgó entitások pozícióját értelemszerűen.
A további sorok az alábbi formátum szerint épülnek fel:
`X;Y;TileType;[0|1];(\d;)*`
Ahol X és Y a mező koordinátái, TileType a mező típusa, [0|1], jelöli hogy található-e ZPM a mezőn, majd a további számok a mezőn található dobozok súlyát írják le, a stackben lentről felfelé haladva. Természetesen utóbbi attribútumok nem minden mezőnél szerepelnek. Scale mezőn a játék indulásakor nem lehet doboz, így itt a ZPM attribútum után a mérleg küszöbértéke található.

7.1.3 Kimeneti nyelv

A különböző kimenetek az alábbi kategóriákba sorolhatók, a kategóriák mögött a kimenetek formátuma van definiálva:
<égtáj>: NORTH, EAST, WEST, SOUTH
<mező típusa>: FIELD, WALL, SCALE, GATE, GAP
<ZPM>: 0, 1
<dobozok>: pozitív egész számok, vesszővel elválasztva, ahol az első a legalsó doboz tömege, az utolsó pedig a legfelső doboz tömege
<mező koordinátái> x,y; ahol az x és az y pozitív egész, a pálya szélességénél illetve magasságánál kisebb számok
<doboz tömege>: valamilyen pozitív egész szám
<csillagkapu színe>: BLUE, YELLOW, RED, GREEN

Kimenetet csak az alábbi parancsok adnak:
listAdjacentFields
Kimenet: <égtáj> <típus> <ZPM> <dobozok>
Megjegyzés: északról indulva óramutató járása szerint

listBoxes
Kimenet: <mező típusa> <mező koordinátái> <doboz tömege>
Megjegyzés:

listStargates

Kimenet: <mező típusa> <mező koordinátái> <csillagkapu színe>
Megjegyzés: mind a négy

listZPMs

Kimenet: <mező típusa> <mező koordinátái>
Megjegyzés:

7.2 Összes részletes use-case

Use-case neve	**boxDrop**
Rövid leírás	A doboz eldobása.
Aktorok	Tesztelő
Forgatókönyv	O'Neill / Jaffa eldobja a dobozt az előtte található mezőre.

Use-case neve	**boxLift**
Rövid leírás	A doboz felvétele.
Aktorok	Tesztelő
Forgatókönyv	O'Neill / Jaffa felveszi a legfelső dobozt az előtte található mezőről (ha van ilyen).

Use-case neve	**loadMap**
Rövid leírás	Pálya betöltése.
Aktorok	Dungeon
Forgatókönyv	Betölti a pályát.

Use-case neve	**listAdjacentFields**
Rövid leírás	Szomszédos mezők listázása.
Aktorok	Tesztelő
Forgatókönyv	Kilistázza a játékos melletti mezőket.

Use-case neve	**listBoxes**
Rövid leírás	Dobozok listázása.
Aktorok	Tesztelő
Forgatókönyv	Kiírja a játékban lévő dobozokat a kimenetre.

Use-case neve	**listStargates**
Rövid leírás	Csillagkapuk listázása.
Aktorok	Tesztelő
Forgatókönyv	Kiírja a játékban szereplő csillagkapukat a kimenetre.

Use-case neve	**listZPMs**
Rövid leírás	ZPM-ek listázása.
Aktorok	Tesztelő
Forgatókönyv	Kiírja a játékban szereplő ZPM-eket a kimenetre.

Use-case neve	**move**
Rövid leírás	Lépés az aktuális irányba.
Aktorok	Tesztelő
Forgatókönyv	O'Neill / Jaffa lép egyet az aktuális irányba.

Use-case neve	**rotate**
Rövid leírás	Fordulás.
Aktorok	Tesztelő
Forgatókönyv	O'Neill / Jaffa forgatása jobbra vagy balra.

Use-case neve	**shootONeillsGun**
Rövid leírás	Lövés O'Neill fegyverével.
Aktorok	Tesztelő
Forgatókönyv	O'Neill fegyvere kilő egy sárga vagy kék csillagkaput az adott irányba.

Use-case neve	**shootJaffasGun**
Rövid leírás	Lövés Jaffa fegyverével.
Aktorok	Tesztelő
Forgatókönyv	Jaffa fegyvere kilő egy zöld vagy piros csillagkaput az adott irányba.

7.3 Tesztelési terv

A tesztek között lehetnek kis átfedések, illetve sok teszteset nem atomi, azaz több "apró" tesztesetet magába foglalhat.

Teszt-eset neve	**O'Neill/Jaffa mozgatása**
Rövid leírás	A szereplők irányítása.
Teszt célja	A pálya bejárhatóságának tesztelése, és az egyes pályaelemek tulajdonságainak tesztelése. (Például, hogy át tudunk-e menni a falon.)

Teszt-eset neve	**Csillagkapu lövés és teleport**
Rövid leírás	O'Neill/Jaffa fegyverének használata.
Teszt célja	A szereplők megfelelően tudnak-e csillagkapukat kilőni és azokon keresztül mozogni. Ajtóra ellenőrizni, hogy nem lehet csillagkaput lőni.

Teszt-eset neve	**Mérleg használat**
Rövid leírás	Ha a mérlegre elegendő súly kerül, kinyílik a hozzá tartozó ajtó.
Teszt célja	Megvizsgálja, hogy ha a mérlegre a súlyhatárának megfelelő súly kerül, legyen az akár O'Neill, Jaffa vagy doboz(ok), akkor a hozzá tartozó ajtó kinyílik-e.

Teszt-eset neve	**Szakadékba esés**

Rövid leírás	O'Neill/Jaffa beleesik egy szakadékba és véget ér a játék.
Teszt célja	Tényleg befejeződik-e a játék, és veszít-e a játékos.

Teszt-eset neve	**Összes ZPM begyűjtése**
Rövid leírás	O'Neill/Jaffa begyűjt minden ZPM-et.
Teszt célja	Tényleg befejeződik-e a játék, és nyer-e a játékos.

Teszt-eset neve	**Lejár az idő**
Rövid leírás	Lejár az idő és meghal a játékos.
Teszt célja	Tényleg befejeződik-e a játék, és veszít-e a játékos.

Teszt-eset neve	**Forgás**
Rövid leírás	O'Neill/Jaffa forgatása
Teszt célja	Lehet-e minden irányba fordulni, attól függetlenül, hogy hol vagyunk és merre kezdtük.

Teszt-eset neve	**Dobozok mozgatása**
Rövid leírás	A dobozok emelése, lerakása, mozgatása
Teszt célja	Meggyőződni arról, hogy szomszédos mezőről fel tudunk-e venni dobozt, majd azt egy pár mezővel arrébb cipelve, le tudjuk-e rakni. Mérlegre szintén ki kell próbálni. Ellenőrizni kell, hogy több doboz lehet-e egyszerre a játékosnál.

Teszt-eset neve	**Replikátor helyzete**
Rövid leírás	A replikátor más objektumokkal áll egy mezőn.
Teszt célja	Annak ellenőrzése, hogy nem dob hibát a program, ha a replikátor azonos mezőre áll egy dobozzal/mérleggel/játékossal.

Teszt-eset neve	**Replikátor a szakadékban**
Rövid leírás	A replikátor beleesik egy szakadékba.
Teszt célja	Megbizonyosodni arról, hogy ilyenkor a replikátor eltűnik, és "betemeti" a szakadékot.

Teszt-eset neve	**Extra ZPM-ek**
Rövid leírás	Újonnan létrejön egy ZPM 2 begyűjtése után.
Teszt célja	Tényleg létrejön-e egy új ZPM kettő begyűjtése után.

7.4 Tesztelést támogató segéd- és fordítóprogramok specifikálása

A tesztelést egy Tester nevű osztállyal végezzük, amely az egyes teszteseteknél a program által generált kimenetet hasonlítja össze az elvárttal.
A teszteléshez használt keretrendszer: **JUnit 4.11**

7.5 Napló

Kezdet	Időtartam	Résztvevők	Leírás
2016.04.01. 14:00	1 óra	Tóth Bokros Hegedűs	Értekezlet. Tóth, Bokros és Hegedűs megbeszélik a teendőket és elosztják a feladatokat.
2016.04.02. 23:00	3 óra	Tóth	Tevékenység. Tóth elkészíti a módosult osztálydiagramot és megírja a dokumentum rá kirótt részét, illetve megbizonyosodik a tervek helyességéről.
2016.04.03. 17:00	1,5 óra	Bokros	Tevékenység. Bokros elkezdi a dokumentum rá kirótt részének megírását. (7.1. fejezet)
2016.04.03. 20:00	1 óra	Bokros	Tevékenység. Bokros befejezi a 7.1 fejezetet, lektorálja Tóth munkáját, illetve a dokumentum formázását.
2016.04.03. 21:00	1 óra	Hegedüs	Tevékenység. Hegedüs megcsinálja a feladat rá kirótt részét (7.2 egész és 7.3 fele)
2016.04.03. 22:00	1 óra	Tóth	Tevékenység. Tóth lektorálja a dokumentumot.
2016.04.03. 22:00	1,5 óra	Varga	Tevékenység. Varga elkészíti a 7.3-as felét, majd a 7.4-est,

			végül lektorálja a dokumentumot.

8. Részletes tervek

8.1 A tesztek részletes tervei, leírásuk a teszt nyelvén -

Az elvárt kimenetnél csak az output azon sorai szerepelnek, amelyek érdekesek a teszt szempontjából.

8.1.1 O'Neill/Jaffa mozgatása

- **Leírás**

A teszteset azt ellenőrzi, hogy a játék irányítható szereplői tudnak-e a pályán mozogni. Mivel a megvalósítás azonos minden szereplőre, ez csak ONeill esetében történik meg. Ezzel a feltételezéssel a többi teszteset is él.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Mozgás
Nem úgy mozognak a szereplők, ahogy kéne.

- **Bemenet**

```
loadMap moveTest

listPlayers

move O

listPlayers
```

- **Elvárt kimenet**

```
ONeill: 1,1
ONeill: 1,2
```

8.1.2 Csillagkapu lövés és teleport

- **Leírás**

A teszteset azt ellenőrzi, hogy a csillagkapuk kilőhetőek-e, illetve valóban megnyílik-e köztük a hipertér.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

az Csillagkapuk működése.
Nem működnek a csillagkapuk.

- **Bemenet**

```
loadMap testStargates

shootONeillsGun B

rotate O L

shootONeillsGun Y

listStargates

move //akár többször, amíg eljut a csillagkapuig, és belelép

listPlayers
```

- **Elvárt kimenet**

```
BlueStargate: 5, 10

YellowStargate: 10, 5

ONeill: 5, 9
```

8.1.3 Mérleg használat

- **Leírás**

  A teszteset azt ellenőrzi, hogy a megfelelő súly egy adott mérlegre helyezése kinyitja-e a hozzá tartozó ajtót.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

  Mérlegek/ajtók működése

  Az ajtó nem nyílt ki.

- **Bemenet**

```
loadMap testScalesAndGates

boxLift O

rotate L

boxDrop O
```

- **Elvárt kimenet**

```
door open
```

8.1.4 Szakadékba esés

- **Leírás**

  Azt ellenőrzi, hogy ONeill / a Jaffa meghal-e amennyiben leesik egy *Gap* típusú mezőre lép.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

  Szakadékok.

  A karakter nem halt meg.

- **Bemenet**

```
loadMap gapTest

move O
```

- **Elvárt kimenet**

```
ONeill has fallen
```

8.1.5 Összes ZPM begyűjtése

- **Leírás**

  Ellenőrzi, hogy a játékot valóban megnyeri-e az a karakter, aki teljesítette a játék megnyerésére vonatkozó ZPM kritériumokat

- **Ellenőrzött funkcionalitás, várható hibahelyek**

  Játék megnyerése.

  A játékot nem nyerte meg a karakter, pedig összegyűjtött elég ZPM-et.

- **Bemenet**

```
loadMap ZPMTest

move O
```

- **Elvárt kimenet**

```
ONeill won
```

8.1.6 Lejár az idő

- **Leírás**

Ellenőrzi, hogy a játék tényleg véget ér-e, amennyiben lejárt az idő.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Játék vége.

Az idő lejárt, de a játék még mindig nem ért véget.

- **Bemenet**

```
loadMap timeUpTest
```

- **Elvárt kimenet**

```
time is up
```

8.1.7 Forgás

- **Leírás**

Ellenőrzi, hogy a szereplő mozog-e, amennyiben erre utasítják.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Forgás
Nem forog

- **Bemenet**

```
Rotate O L

Rotate O R
```

- **Elvárt kimenet**

```
ONeill's facing direction is North
ONeill's facing direction is West
ONeill's facing direction is North
```

8.1.8 Dobozok mozgatása

- **Leírás**

Ellenőrzi, hogy a dobozok mozgathatóak, azaz felemelhetőek és letehetőek-e.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Dobozok felemelése és lerakása
Nem lehet a dobozt felemelni, lerakni

- **Bemenet**

```
LoadMap testBoxes
boxLift
boxDrop
```

- **Elvárt kimenet**

```
box3 lifted
box3 dropped
```

8.1.9 Replikátor helyzete

- **Leírás**

Annak ellenőrzése, hogy nem dob hibát a program, ha a replikátor azonos mezőre áll egy dobozzal/mérleggel/játékossal. A replikátor ONeill előtt kerül elhelyezésre a teszttérképen, így csupán előre kell lépnie egyet.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Replikátor és más egy mezőn állása
Egy mezőre kerülnek olyan entitások, akiknek nem kéne.

- **Bemenet**

```
Loadmap testReplicatorPosition
move O
```

- **Elvárt kimenet**

```
*semmi*
```

8.1.10 Replikátor a szakadékban

- **Leírás**

Megbizonyosodni arról, hogy ilyenkor a replikátor eltűnik, és "betemeti" a szakadékot.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Eltűnik a replikátor, létrejön az új mező.
Nem tűnik el a replikátor, nem jön létre az új mező.

- **Bemenet**

loadMap testReplicatorInGap

- **Elvárt kimenet**

8.1.11 Extra ZPM-ek

- **Leírás**

Tényleg létrejön-e egy új ZPM kettő begyűjtése után. A teszt során az ONeill előtt lévő két mezőn két ZPM kerül elhelyezésre, így két lépéssel be tudja őket gyűjteni.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Létrejön.
Nem jön létre.

- **Bemenet**

```
loadMap testExtraZPMs

listZPMs

move O

move O

listZPMs
```

- **Elvárt kimenet**

```
*n db ZPM listája*
*n-1 db ZPM listája*
```

8.2 A tesztelést támogató programok tervei

A tesztprogram megadott bemenet(ek)re kapott eredményeket hasonlítja össze a várt kimenettel. Ahhoz, hogy a program kimeneteit ellenőrizzük, a Game osztálynak egy public static stream attribútumot adunk, amire a program kimeneteit küldjük ki. Ezekkel hasonlítjuk össze az elvárt kimenete(ke)t. Az osztály csak azokat a teszteseteket futtatja le, amik a "@Test" annotációval lettek ellátva.

Tester osztály:

- **Felelősség:**

Az alkalmazás funkcióinak helyes működésének ellenőrzése.

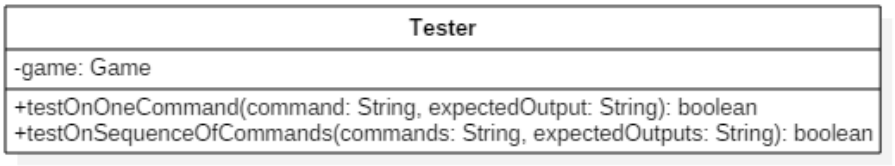
- **Attribútumok:**

Game game: A játékmenet futtatásáért felelős objektum.

- **Metódusok:**

boolean testOnOneCommand(String command, String expectedOutput): Tesztelés egy parancsú bemenetre. Ha a kimenet megegyezik az elvárt kimenettel, *true*, ellenkező esetben *false* értékkel tér vissza.

boolean testOnSequenceOfCommands(String[] commands, String[] expectedOutputs): Tesztelés parancsok sorára. Ha a kimenet megegyezik az elvárt kimenettel, *true*, ellenkező esetben *false* értékkel tér vissza.



8.3 Napló

Kezdet	Időtartam	Résztvevők	Leírás
2016.04.06 15:00	1 óra	Bokros Hegedüs Jáhn Siket Tóth Varga	Értekezlet. A módosítások átbeszélése, a heti feladatok szétosztása.
2016.04.10. 21:00	2 óra	Jáhn	Tevékenység: Jáhn megtervezi a tesztelést végrehajtó osztályt.
2016.04.10 21:20	2 óra	Bokros	Tevékenység. Bokros elkészíti a tesztesetek rá eső hányadát, és egyeztet csapattársaival.
2016.04.10. 22:00	1,5 óra	Siket	Tevékenység: Siket kitölti a 8.1-es pontot a szükséges javításokkal.
2016.04.10 22:15	30 perc	Tóth	Tevékenység. Tóth elkészíti a tesztesetek rá eső hányadát.
2016.04.10. 21:00	3 óra	Varga	Tevékenység. Varga elkészíti az osztályleírásokat.

10. Prototípus beadása

10.1 Fordítási és futtatási útmutató

10.1.1 Fájllista

Fájl neve	Méret	Keletkezés ideje	Tartalom
Box.java		2016.03.28.	Box osztály implementációja
Destroyable.java		2016.03.28.	Destroyable osztály implementációja
Direction.java		2016.03.28.	Direction osztály implementációja
Dungeon.java		2016.03.28.	Dungeon osztály implementációja
Field.java		2016.03.28.	Field osztály implementációja
FlowOfTime.java		2016.03.28.	FlowOfTime osztály implementációja
Game.java		2016.03.28.	Game osztály implementációja
Gap.java		2016.03.28.	Gap osztály implementációja
Gate.java		2016.03.28.	Gate osztály implementációja
Measurable.java		2016.04.20.	Measurable osztály implementációja
Menu.java		2016.04.20.	Menu osztály implementációja
Pickable.java		2016.03.28.	Pickable interface implementációja
Player.java		2016.04.20.	Player osztály implementációja
PlayerBase.java		2016.04.20.	PlayerBase osztály implementációja
Scale.java		2016.03.28.	Scale osztály implementációja
Stargate.java		2016.03.28.	Stargate osztály implementációja
Tile.java		2016.03.28.	Tile osztály implementációja
Wall.java		2016.03.28.	Wall osztály implementációja
ZPM.java		2016.03.28.	ZPM osztály implementációja

10.1.2 Fordítás

Cicaprojekt (gyökérkönyvtár)
- cicaprojekt
  - Box.java
  - Destroyable.java
  - Direction.java
  - Dungeon.java
  - Field.java
  - FlowOfTime.java
  - Game.java
  - Gap.java
  - Gate.java
  - Measurable.java
  - Menu.java
  - Pickable.java
  - Player.java
  - PlayerBase.java
  - Scale.java
  - Stargate.java
  - Tester.java
  - Tile.java

```
    ├── Wall.java
    └── ZPM.java
├── exampledungeon.dungeon
├── gapTest.dungeon
├── moveTest.dungeon
├── rotate.dungeon
├── testBoxes.dungeon
├── testReplicatorInGap.dungeon
├── testReplicatorPosition.dungeon
├── testScalesAndGates.dungeon
├── testStargates.dungeon
├── ZPMTest.dungeon
└── timeUpTest.dungeon
```

A szoftver fordítását a forráskódok gyökérkönytárába lépve lehet elvégezni. A fordításhoz a javac cicaprojekt/* parancsot kell kiadni, aminek hatására létrejönnek a futtatható .class állományok.

10.1.3 Futtatás

A fordítást követően a .class állományok futtatása a *java cicaprojekt.Menu* parancs kiadásával történik (természetesen ezt is a forráskódok gyökérkönyvtárából). Ez után a parancssorból lehet használni a szoftvert.

10.2 Tesztek jegyzőkönyvei

10.2.1 moveTest

Tesztelő neve	Bokros Bálint
Teszt időpontja	2016.04.25.

10.2.2 testStargates

Tesztelő neve	Tóth Kristóf
Teszt időpontja	2016.04.25.

10.2.3 testScalesAndGates

Tesztelő neve	Hegedüs Fanni
Teszt időpontja	2016.04.25.

10.2.4 gapTest

Tesztelő neve	Varga Péter
Teszt időpontja	2016.04.25.

10.2.5 ZPMTest

Tesztelő neve	Jáhn Erik
Teszt időpontja	2016.04.25.

10.2.6 timeUpTest

Tesztelő neve	Siket Melinda
Teszt időpontja	2016.04.25.

10.2.7 move

Tesztelő neve	Bokros Bálint
Teszt időpontja	2016.04.25.

10.2.8 testBoxes

Tesztelő neve	Tóth Kristóf
Teszt időpontja	2016.04.25.

10.2.9 testReplicatorPosition

Tesztelő neve	Hegedüs Fanni
Teszt időpontja	2016.04.25.

10.2.10 testReplicatorInGap

Tesztelő neve	Varga Péter
Teszt időpontja	2016.04.25.

10.3 Értékelés

Tag neve	Munka százalékban
Bokros Bálint Botond	24 %
Hegedüs Fanni	13 %
Jáhn Erik	19 %
Siket Melinda Tekla	9 %
Tóth Kristóf	24 %
Varga Péter	11 %

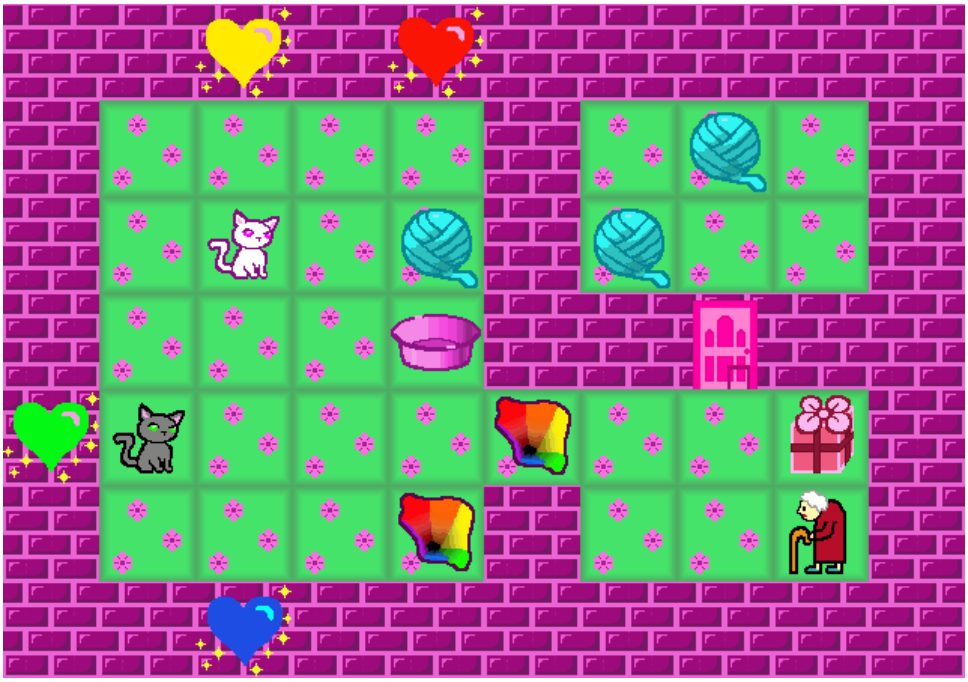
10.4 Napló

Kezdet	Időtartam	Résztvevők	Leírás
2016.04.20. 15:00	1 óra	Bokros Jáhn Tóth Hegedüs Siket Varga	Értekezlet. Heti feladatok szétosztása.
2016.04.24. 17:00	5,5 óra	Jáhn	Tevékenység: Jáhn implementálja a Tile osztály leszármazottjait.
2016.04.24. 17:00	4 óra	Bokros	Tevékenység: Bokros implementálja a rá osztott feladatot.
2016.04.24. 17:00	4 óra	Tóth	Tevékenység: Tóth implementálja a rá osztott feladatot.
2016.04.23. 12:00	3 óra	Tóth	Tevékenység: Tóth implementálja a rá osztott feladatot.
2016.04.19. 9:30	1 óra	Bokros	Tevékenység: Bokros implementálja a rá osztott feladatot.
2016.04.24. 18:00	1 óra	Varga	Tevékenység: Varga implementálja a rá osztott feladatot.
2016.04.25. 18:00	8 óra	Bokros	Tevékenység: Bokros implementálja a rá osztott feladatot.
2016.04.25. 18:00	8 óra	Tóth	Tevékenység: Tóth implementálja a rá osztott feladatot.
2016.04.25. 18:00	7 óra	Hegedüs	Tevékenység: Hegedüs implementálja a rá osztott feladatot. Továbbá megírja a dokumentációt.
2016.04.25. 18:00	8 óra	Jáhn	Tevékenység: Jáhn implementálja a rá osztott feladatot.

11. Grafikus felület specifikációja

11.1 A grafikus interfész

Menü:



- Fehér és fekete macska ikon: A fehér macska O'Neill ezredes, a fekete macska pedig Jaffa.
- Téglafal ikon: Azt a falat (Wall) reprezentálja, melyre lehet csillagkaput lőni.
- Sárga, kék, zöld, piros szívecske csillagkapu ikon: O'Neill és Jaffa által kilőtt csillagkapuk. (YellowStargate, BlueStargate, GreenStargate, RedStargate)
- Szivárvány szakadék ikon: A szakadékot (Gap) jelenti.
- Néni ikon: A replikátor (Replicator) testet öltése.
- Gombolyag ikon: A ZPM megjelenítése.

- Cicakosár ikon: A mérleget (Scale) reprezentálja.
- Ajtó ikon: Az ajtó (Gate) megjelenítése, ha a cicakosárba elegendő súly kerül, akkor ez kinyílik.
- Ajándékdoboz ikon: A doboz (Box) megjelenítése.
- Virágos föld ikon: A talaj (Field) reprezentálása.

11.2 A grafikus rendszer architektúrája

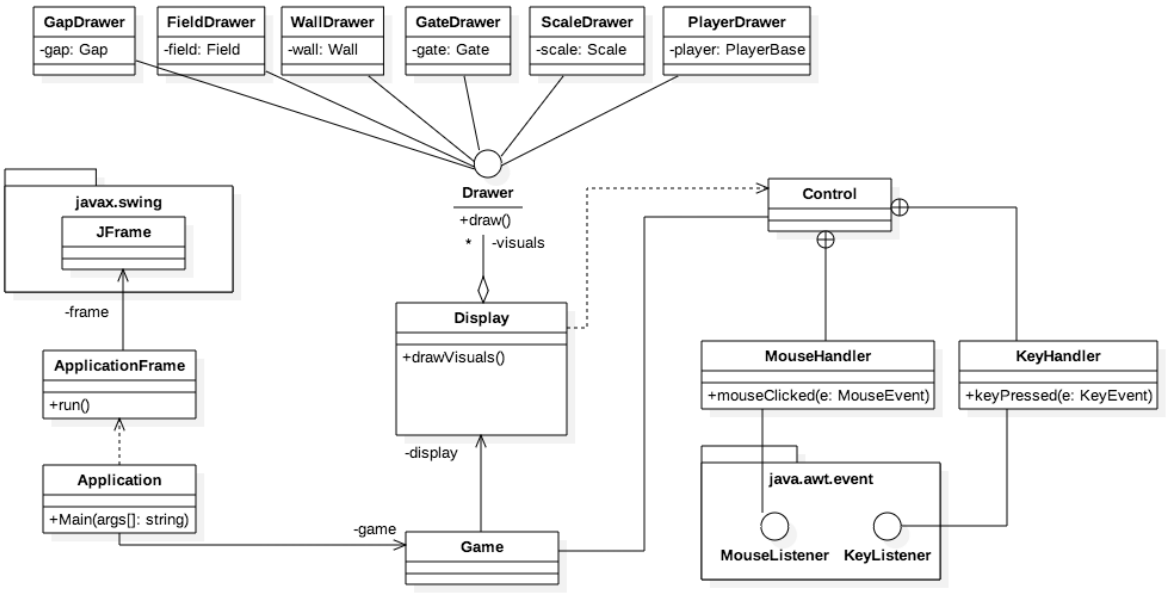
11.2.1 A felület működési elve

A grafikai felület architektúrájának megtervezésekor az MVC tervezési mintát követtük: ennek keretein belül a már korábban megtervezett osztályok tartoznak a Model komponensbe, míg az újonnan tervezettek a View-ba és a Controller-be. A View-ba tartozik az *ApplicationFrame*, a *Display* és a különböző *Drawer* interfészt megvalósító osztályok, míg a Controllerbe a *Control* osztály és a benne definiált nested eseménykezelő osztályok tartoznak.
A *Control* osztály működése esemény alapú: feliratkozik különböző eseményekre a grafikus felületen, majd eszközöli az azoknak megfelelő változásokat a *Game*-en. Az alkalmazás fő belépési pontja az *Application.Main* metódus. Itt jön létre az alkalmazás fő ablaka, az *ApplicationFrame* (pontosabban ennek a *JFrame* attribútuma). A kirajzolás push elven működik: a *Game* osztály értesíti a változásokról a *Display*-t, amely ezeknek megfelelően az újrarajzolást intézi. A *Display* osztály alapját egy *Drawer* interfészt megvalósító objektumokból képzett heterogén kollekció adja. Újrarajzoláskor a *Display* egy előre nem specifikált metódusa végigiterál az itt tárolt objektumokon és egyesével kirajzolja azokat.

Az MVC minta biztosítja, hogy a szoftver megjelenítési funkciói elválasztódnak a belső logikától, ill. a felhasználói bemenetet kezelő részektől, így a komponensek szükség esetén könnyen cserélhetők, variálhatók (első sorban itt nyilván arra gondolunk, hogy mondjuk valaki szeretne később egy módosított megjelenést készíteni a játékhoz, akár úgy, hogy leszármaztat valamit a *Display* osztályból, vagy komponensként felhasználja azt egy új *Display* variánshoz, vagy akár úgy, hogy teljesen átírja azt).

11.2.2 A felület osztály-struktúrája

Az alkalmazás grafikus részének architektúrális képe:



11.3 A grafikus objektumok felsorolása

11.3.1 Application

- **Felelősség**

Az alkalmazást megvalósító osztály, itt van a program belépési pontja.

- **Ősosztályok**

Nem származik le más osztályokból.

- **Interfészek**

Nem valósít meg interfészt.

- **Attribútumok**
  - **-Game game:** a játékot megtestesítő objektum referenciája.

- **Metódusok**
  - **+<u>void Main(string[] args)</u>**: az egész alkalmazás belépési pontja. Itt történik az ApplicationFrame inicializálása, ami lényegében egy *JFrame*, és ez az alkalmazás fő ablaka.

11.3.2 ApplicationFrame

- **Felelősség**

A grafikus részei a programnak ebben vannak megvalósítva, ez képezi a gerincét a GUI-nak. Az alkalmazás ablak (egy JFrame) is itt található.

- **Ősosztályok**

Nem származik le más osztályokból.

- **Interfészek**

Nem valósít meg interfészt.

- **Attribútumok**
  - **-JFrame frame**: a játékablak objektum.

- **Metódusok**

Nincsenek metódusok.

11.3.3 Control

- **Felelősség**

Az eseménykezelésért felelős osztály, konkrétan a felhasználói akciókat, mint a billentyűparancsok és az egérkattintások, kezeli le. Ezek alapján aztán átvezeti a változásokat a business logikába. Működése eseményalapú, feliratkozik az eseményekre a grafikus felületen, majd a *Game*-ben eszközöli a változásokat. Tartalmaz nested osztályokat (*MouseHandler, KeyHandler*), melyek az eseménykezelésért felelnek az osztályon belül.

- **Ősosztályok**

Nem származik le más osztályból.

- **Interfészek**

Nem valósít meg interfészt.

- **Attribútumok**

Nincsenek attribútumok.

- **Metódusok**
- **+run():** a főablak létrehozásáért felelő metódus.

11.3.4 Display

- **Felelősség**

Felelős egy Game és az ahhoz tartozó minden további játékbeli objektum kirajzolásáért.

- **Ősosztályok**

Nem származik le más osztályból.

- **Interfészek**

Nem valósít meg interfészt.

- **Attribútumok**
  - **-List<Drawer> visuals:** a játékbeli kirajzolandó objektumok kirajzolóját tartalmazó lista.
  - **-Control control**: az eseménykezelő egy példánya, hogy a kijelzővel összeköttetésben álljon.

- **Metódusok**
- **+drawVisuals():** meghívja az egyes kirajzolható objektumok rajzolófüggvényeit.

11.3.5 Drawer

- **Felelősség**

A kirajzoló interfész, amit az objektumok kirajzoló osztályai implementálnak.

- **Ősosztályok**

Interfész lévén nem származik le más osztályokból.

- **Interfészek**

Interfész lévén nem valósít meg interfészt.

- **Attribútumok**

Nincsenek attribútumok.

- **Metódusok**
  - ***+void draw()***: absztrakt kirajzoló osztály, amit osztályonként máshogy valósítunk meg, az egyes objektumok képernyőre rajzolásáért felel. Ez úgy nyilvánul meg, hogy betölti az objektumokhoz tartozó képeket, és elhelyezi azokat a képernyőn.

11.3.6 FieldDrawer

- **Felelősség**

A pályán lévő mezőket (Field típusú mező) kirajzoló osztály.

- **Ősosztályok**

Nem származik le más osztályból.

- **Interfészek**

A *Drawer* interfészt valósítja meg.

- **Attribútumok**
  - **-Field field**: az az objektum, amit a metódusnak meg kell rajzolnia.
- **Metódusok**
  - **+void draw()**: az osztály kirajzoló metódusa, ami a *Drawer* interfész *draw()* metódusát override-olja.

11.3.7 Game

- **Felelősség**

Ez az osztály a korábbi dokumentációkban már le volt írva, a változás az előzményekhez képest annyi, hogy kap egy *Display* és egy *Control* típusú attribútumot, hogy a grafikus megjelenítést is tudja kezelni. Szerepe tehát annyival bővül, hogy ha a játék során valamilyen

változás lép fel (pl. Az egyik játékos elmozdul), akkor újrarajzoltatja a Display-jel a képernyőt.

- **Ősosztályok**

Nem származik le más osztályból.

- **Interfészek**

Nem valósít meg interfészt.

- **Attribútumok**

Korábban már kifejtésre kerültek a lentieken kívül.

  - **-Display display:** a kirajzolásért felelős objektum, a Game ezen keresztül tudja frissíteni a képernyőt.
  - **-Control control**: a felhasználói interakcióért felelős objektum, a Game ezen keresztül tudja például a billentyűparancsokat lekezelni.

- **Metódusok**

Korábban már kifejtésre kerültek.

11.3.8 GapDrawer

- **Felelősség**

A pályán található szakadékok (Gap típusú mezők) kirajzolásáért felelős osztály.

- **Ősosztályok**

Nem származik le más osztályból.

- **Interfészek**

A *Drawer* interfészt valósítja meg.

- **Attribútumok**
  - **-Gap gap**: az az objektum, amit a metódusnak meg kell rajzolnia.
- **Metódusok**
  - **+void draw()**: az osztály kirajzoló metódusa, ami a *Drawer* interfész *draw()* metódusát override-olja.

11.3.9 KeyHandler

- **Felelősség**

A billentyűzet és az azon keresztül történő felhasználói beavatkozások kezelését megvalósító osztály, mely a Control osztályon keresztül kapcsolatban áll a Game osztállyal.

- **Ősosztályok**

Nem származik le más osztályból.

- **Interfészek**

A *KeyListener* interfészt valósítja meg.

- **Attribútumok**

Nincsenek attribútumok.

- **Metódusok**
- **+keyPressed(KeyEvent e):** billentyű lenyomására hívódik meg, ezt az eseményt kezeli le.

11.3.10 MouseHandler

- **Felelősség**

Az egér és az azon keresztül történő felhasználói beavatkozások kezelését megvalósító osztály, mely a Control osztályon keresztül kapcsolatban áll a Game osztállyal.

- **Ősosztályok**

Nem származik le más osztályból.

- **Interfészek**

A *MouseListener* interfészt valósítja meg.

- **Attribútumok**

Nincsenek attribútumok.

- **Metódusok**
- **+mouseClicked(MouseEvent e):** egérgomb lenyomására hívódik meg, ezt az eseményt kezeli le.

11.3.11 PlayerDrawer

- **Felelősség**

A játékban szereplő játékosok (O'Neill és Jaffa) kirajzolásáért felelős osztály.

- **Ősosztályok**

Nem származik le más osztályból.

- **Interfészek**

A *Drawer* interfészt valósítja meg.

- **Attribútumok**
  - **-PlayerBase player**: a játékos objektum, amit meg kell rajzolnia a metódusnak.
- **Metódusok**
  - **+void draw()**: az osztály kirajzoló metódusa, ami a *Drawer* interfész *draw()* metódusát override-olja.

11.3.12 ScaleDrawer

- **Felelősség**

A pályán található mérlegek (*Scale*) kirajzolásáért felelős osztály.

- **Ősosztályok**

Nem származik le más osztályból.

- **Interfészek**

A *Drawer* interfészt valósítja meg.

- **Attribútumok**
  - **-Scale scale:** a mérleg objektum, amit ki kell rajzolnia.

- **Metódusok**
  - **+void draw()**: az osztály kirajzoló metódusa, ami a *Drawer* interfész *draw()* metódusát override-olja.

11.3.13 WallDrawer

- **Felelősség**

A pályán található falak (Wall típusú mezők) kirajzolásáért felelős osztály.

- **Ősosztályok**

Nem származik le más osztályból.

- **Interfészek**

A *Drawer* interfészt valósítja meg.

- **Attribútumok**
  - **-Wall wall**: a falelem, amit kirajzol.

- **Metódusok**
  - **+void draw()**: az osztály kirajzoló metódusa, ami a *Drawer* interfész *draw()* metódusát override-olja.

11.3.14 GateDrawer

- **Felelősség**

A pályán található kapukat (*Gate* típusú mezők) kirajzoló osztály.

- **Ősosztályok**

Nem származik le más osztályból.

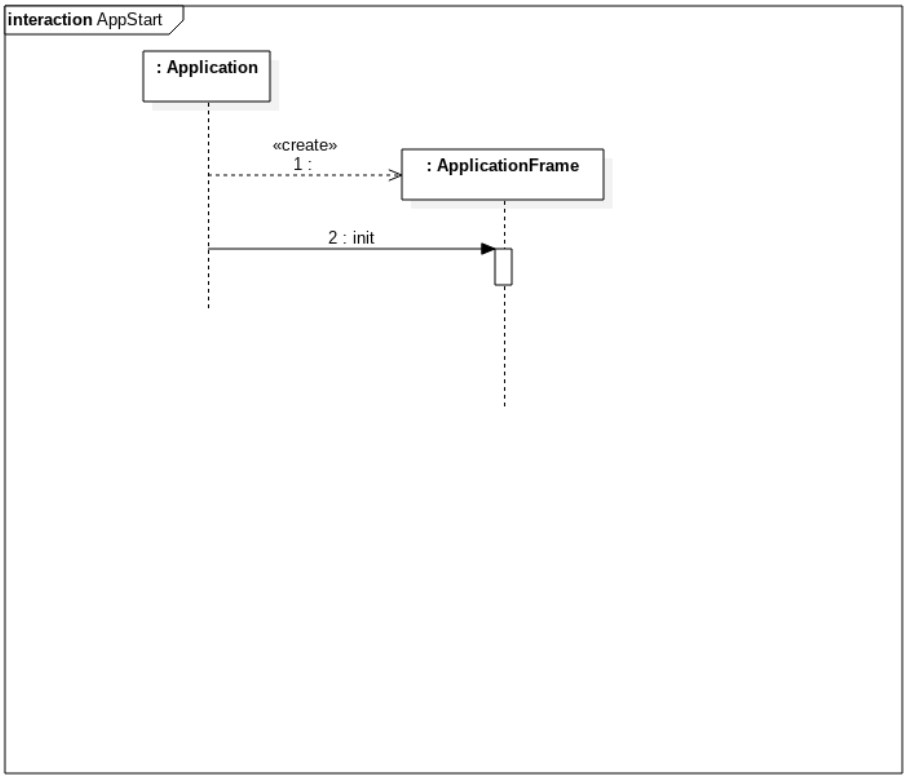
- **Interfészek**

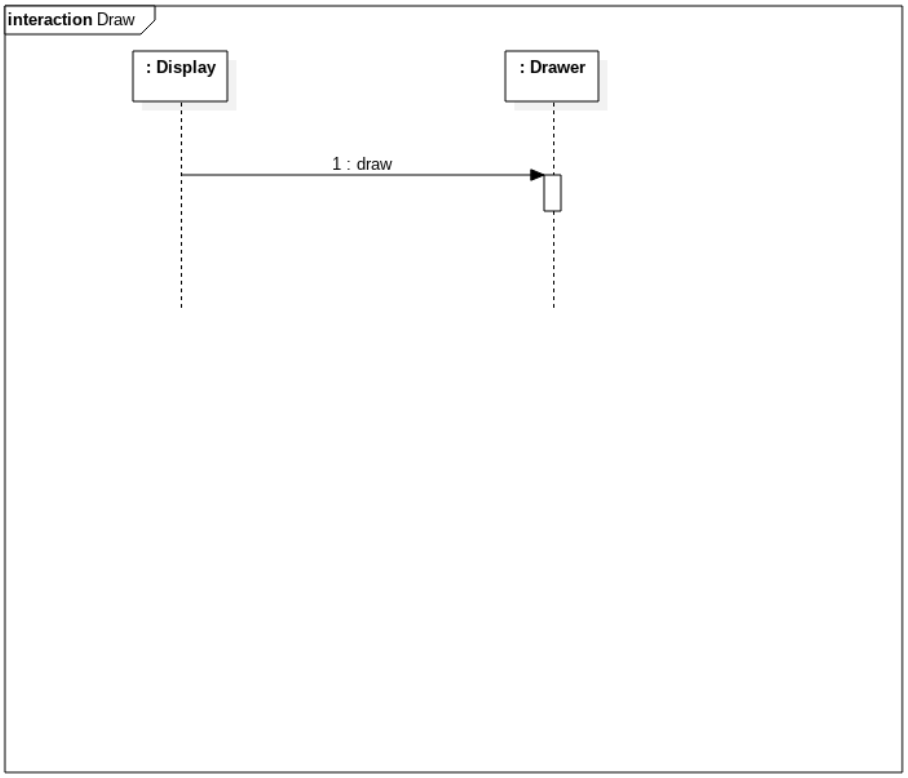
A *Drawer* interfészt valósítja meg.

- **Attribútumok**
  - **-Gate gate**: a kapu objektum, amit ki kell rajzolnia.

- **Metódusok**
  - **+void draw()**: az osztály kirajzoló metódusa, ami a *Drawer* interfész *draw()* metódusát override-olja.

11.4 Kapcsolat az alkalmazói rendszerrel




interaction Draw
: Display
: Drawer
1 : draw

11.5 Napló

Kezdet	Időtartam	Résztvevők	Leírás
2016.04.30. 19:00	30 perc	Bokros Tóth Jáhn Varga Hegedűs Siket	Értekezlet. A feldatok felosztása egymás között.
2016.04.30. 19:30	30 perc	Jáhn	Tevékenység: Jáhn elkészíti a program menüjének design-ját.
2016.04.30. 21:00	2 óra	Varga	Tevékenység: Varga elkészíti az osztályok leírását (11.3)
2016.04.30. 21:00	1 óra	Siket	Tevékenység: Siket elkészíti a 11.2.1-es pontot.
2016.04.30. 21:30	2 óra	Tóth	Tevékenység: Tóth megtervezi a szoftver grafikus részének architektúráját.
2016.04.30. 22:00	30 perc	Siket	Tevékenység: Siket lektorálja az eddig leírtakat.
2016.05.01. 22:00	30 perc	Bokros	Tevékenység: Bokros elkészíti a szekvenciadiagramokat
2016.05.01. 12:00	1 óra	Tóth	Tevékenység: Tóth elkészíti a grafikus rész osztálydiagramját.
2016.05.01. 18:00	1 óra	Tóth	Tevékenység: Tóth lektorálja, ill. kiegészíti a többiek munkáját.
2015.04.30. 20:00	3,5 óra	Hegedüs	Tevékenység: Hegedüs rajzolja a karaktereket.
2015.05.01. 19:00	3,5 óra	Hegedüs	Tevékenység: Hegedüs rajzolja a mezőket.

13. Grafikus változat beadása

13.1 Fordítási és futtatási útmutató

13.1.1 Fájllista

Fájl neve	Méret	Keletkezés ideje	Tartalom
cicaprojekt\AbstractDrawer.java	1446 B	2016. 05. 09. 17:21	A kirajzoló osztályok absztrakt ősosztálya.
cicaprojekt\Application.java	588 B	2016. 05. 09. 13:34	Az alkalmazást megvalósító osztály.
cicaprojekt\ApplicationFrame.java	7736 B	2016. 05. 09. 13:19	Az MVC modell View részéért felelős osztály.
cicaprojekt\Box.java	776 B	2016. 05. 10. 16:57	A dobozokat megvalósító osztály.
cicaprojekt\Control.java	3640 B	2016. 05. 09. 11:43	A billentyűzetet kezelő osztály.
cicaprojekt\Destroyable.java	242 B	2016. 04. 21. 15:56	A megsemmisítésért felelős interfész.
cicaprojekt\Direction.java	1018 B	2016. 04. 22. 17:45	Az irányokat nyilvántartó, invertáló és randomizáló osztály.
cicaprojekt\Display.java	3492 B	2016. 05. 09. 13:34	Az MVC modell View részéért felelős osztály, mely a pályát jeleníti meg.
cicaprojekt\Drawer.java	730 B	2016. 05. 08. 12:23	Az objektumok kirajzolásáért felelős interfész.
cicaprojekt\Dungeon.java	8830 B	2016. 05. 09. 19:56	A pályák beolvasásáért, létrehozásáért és kirajzoltatásáért felelős

			osztály
cicaprojekt\Field.java	1703 B	2016. 05. 10. 00:21	A "semleges" mezőket megvalósító osztály.
cicaprojekt\FieldDrawer.java	1314 B	2016. 05. 10. 12:45	A mezők kirajzolásáért felelős osztály.
cicaprojekt\FlowOfTime.java	1548 B	2016. 04. 21. 21:22	Az idő méréséért és a Replicator mozgatásáért felelős osztály.
cicaprojekt\Game.java	14287 B	2016. 04. 21. 20:23	A játék logikáját összefogó osztály.
cicaprojekt\GameoverCause.java	214 B	2016. 04. 21. 11:41	A lehetséges befejeződéseket tartalmazó enum.
cicaprojekt\Gap.java	1745 B	2016. 04. 21. 20:23	A szakadékokat megvalósító osztály.
cicaprojekt\GapDrawer.java	942 B	2016. 04. 21. 11:41	A szakadékok kirajzolásáért felelős osztály.
cicaprojekt\Gate.java	3032 B	2016. 04. 21. 23:38	Az ajtókat megvalósító osztály.
cicaprojekt\GateDrawer.java	1576 B	2016. 05. 09. 11:41	Az ajtókat megrajzoló osztály.
cicaprojekt\ImageStorage.java	1245 B	2016. 05. 14. 22:47	Képek számontartásár a szolgáló osztály.
cicaprojekt\Measurable.java	286 B	2016. 04. 21. 20:22	Az objektumok súlyát megvalósító interfész.
cicaprojekt\Pickable.java	249 B	2016. 04. 21. 20:22	Egy objektum felszedését implementáló interfész.
cicaprojekt\Player.java	3319 B	2016. 04. 21. 20:22	Konkrétan O'Neill-t és Jaffát

			megvalósító osztály.
cicaprojekt\PlayerBase.java	4536 B	2016. 04. 21. 20:22	O'Neill, Jaffa és a Replicator közös metódusait és attribútumait megvalósító osztály.
cicaprojekt\PlayerBaseDrawer.java	2088 B	2016. 05. 09. 11:41	AReplicator kirajzolásáért felelős osztály.
cicaprojekt\PlayerDrawer.java	3938 B	2016. 05. 09. 11:41	O'Neill és Jaffa kirajzolásáért felelős osztály.
cicaprojekt\Scale.java	3833 B	2016. 04. 21. 20:23	A mérlegeket megvalósító osztály.
cicaprojekt\ScaleDrawer.java	1522 B	2016. 05. 09. 11:41	A mérlegek megrajzolásáért felelős osztály.
cicaprojekt\Stargate.java	5389 B	2016. 04. 21. 20:23	A csillagkapukat megvalósító osztály.
cicaprojekt\Tile.java	5920 B	2016. 04. 21. 20:23	A pályán lévő mezők ősosztálya.
cicaprojekt\Wall.java	3021 B	2016. 04. 21. 20:23	A falakat megvalósító osztály.
cicaprojekt\WallDrawer.java	1863 B	2016. 05. 09. 11:41	A falak megrajzolásáért felelős osztály.
cicaprojekt\ZPM.java	547 B	2016. 04. 21. 20:23	A ZPM-eket megvalósító osztály.
Anubis.png	103984 B	2016. 05. 10. 12:05	A játék végén Anubis-t ábrázoló kép.
BlueStargate.png	23075 B	2016. 05. 10. 14:13	A kék színű csillagkaput ikonja.
Box.png	20379 B	2016. 05. 10. 13:43	A doboz ikonja.
Deal_with_iit.png	85385 B	2016. 05. 10. 12:08	A játék megnyerésekor megjelenő kép.

Empty.png	17791 B	2016. 05. 10. 12:12	Ürességet jelző ikon, teszteléshez kellett.
Field.png	19588 B	2016. 05 10. 12:23	Az üres mező ikonja.
Gap.png	21132 B	2016. 05. 10. 12: 43	A szakadék ikonja.
Gate.png	20078 B	2016. 05. 10. 12: 57	A zárt ajtó ikonja.
GreenStargate.png	23034 B	2016. 05. 10. 14:10	A zöld csillagkapu ikonja.
Jaffa_Down.png	20358 B	2016. 05. 10. 13:04	A Délnek néző Jaffa ikonja.
Jaffa_Down_Box.png	21696 B	2016. 05. 13. 14:56	A Délnek néző dobozt cipelő Jaffa ikonja.
Jaffa_Left.png	20434 B	2016. 05. 10. 13:09	A Nyugatnak néző Jaffa ikonja.
Jaffa_Left_Box.png	22021 B	2016. 05. 13. 14:52	A Nyugatnak néző dobozt cipelő Jaffa ikonja.
Jaffa_Right.png	20457 B	2016. 05. 10. 13:08	A Keletnek néző Jaffa ikonja.
Jaffa_Right_Box.png	22075 B	2016. 05. 13. 14:54	A Keletnek néző dobozt cipelő Jaffa ikonja.
Jaffa_Up.png	20231 B	2016. 05. 10. 13:16	AzÉszaknak néző Jaffa ikonja.
Jaffa_Up_Box.png	21795 B	2016. 05. 13. 14:51	Az Északnak néző dobozt cipelő Jaffa ikonja.
Menu.png	471558 B	2016. 05. 10. 13:02	A menü háttere.
ONeill_Down.png	20197 B	2016. 05. 10. 13:20	A Délnek néző O'Neill ikonja.
ONeill_Down_Box.png	21576 B	2016. 05. 13. 14:38	A Délnek néző O'Neill ikonja dobozzal.
ONeill_Left.png	20209 B	2016. 05. 10. 13:35	A Nyugatnak néző O'Neill ikonja.
ONeill_Left_Box.png	21761 B	2016. 05. 13. 14:40	A Nyugatnak

			néző O'Neill ikonja dobozzal.
ONeill_Right.png	20218 B	2016. 05. 10. 13:28	A Keletnek néző O'Neill ikonja.
ONeill_Right_Box.png	21890 B	2016. 05. 13. 14:34	A Keletnek néző O'Neill ikonja dobozzal.
ONeill_Up.png	20113 B	2016. 05. 10. 13:22	Az Északnak néző O'Neill ikonja.
ONeill_Up_Box.png	21654 B	2016. 05. 13. 14:32	Az Északnak néző O'Neill ikonjadobozzal.
OpenGate.png	20069 B	2016. 05. 10. 14:20	Nyitott ajtó ikonja.
OpenGate_Box.png	22846 B	2016. 05. 13. 14:21	Nyitott ajtó ikonja, a mezőre helyezett dobozzal.
RedStargate.png	23049 B	2016. 05. 10. 14:15	Vörös csillagkapu ikonja.
Replicator_Down.png	20307 B	2016. 05. 13. 14:07	A Délnek néző Replicator ikonja.
Replicator_Left.png	20199 B	2016. 05. 13. 14:10	A Nyugatnak néző Replicator ikonja.
Replicator_Right.png	20241 B	2016. 05 13. 14:09	A Keletnek néző Replicator ikonja.
Replicator_Up.png	20076 B	2016. 05. 13. 14:05	Az Északnak néző Replicator ikonja.
Scale.png	20348 B	2016. 05. 10. 14:54	A mérleg ikonja.
Scale_Box.png	22856 B	2016. 05. 13. 14:02	A dobozzal/dobozokkal terhelt mérleg ikonja.
Wall.png	19665 B	2016. 05. 10. 14:05	A fal ikonja.

YellowStargate.png	23059 B	2016. 05. 10. 14:19	A sárga csillagkapu ikonja.
ZPM.png	20985 B	2016. 05. 10. 14:28	A ZPM ikonja.
MapM1_cooperation.dungeon	229 B	2016. 05. 14. 14:43	Az egyik multiplayer pálya.
MapM2_live_to_win.dungeon	368 B	2016. 05. 14. 15:51	Az egyik multiplayer pálya.
MapM3_race.dungeon	146 B	2016. 05. 14. 16:32	Az egyik multiplayer pálya.
MapM4_welcome_to_the_dungeon.dungeon	328 B	2016. 05. 14. 17:01	Az egyik multiplayer pálya.
MapM5_theres_a_story_of_my_dungy.dungeon	218 B	2016. 05. 14. 18:07	Az egyik multiplayer pálya.
MapM6_bonus_market_shooting_range.dungeon	247 B	2016. 05. 14. 18:35	Az egyik multiplayer pálya.
MapM7_two_way_fun.dungeon	242 B	2016. 05. 14. 19:21	Az egyik multiplayer pálya.
MapM8_bridge_of_death.dungeon	242 B	2016. 05. 14. 20:12	Az egyik multiplayer pálya.
MapS1.dungeon	131 B	2016. 05. 14. 20:59	Az egyik singleplayer pálya
MapS2.dungeon	60 B	2016. 05. 14. 21:42	Az egyik singleplayer pálya
MapS3_insufficient_funds.dungeon	284 B	2016. 05. 14. 22:35	Az egyik singleplayer pálya

13.1.2 Fordítás és telepítés

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

```
javac cicaprojekt/*.java
```

13.1.3 Futtatás

A futtatáshoz pedig a fordítást követően a következőt:

```
java cicaprojekt.Application
```

13.2 Értékelés

Tag neve	Munka százalékban
Bokros Bálint Botond	22%
Hegedűs Fanni	11%
Tóth Kristóf	23%
Siket Melinda Tekla	11%
Jáhn Erik	19%
Varga Péter	14%

13.3 Napló

Kezdet	Időtartam	Résztvevők	Leírás
2016. 05. 04. 16:00	1 óra	Bokros Jáhn Tóth Siket Varga	Értekezlet.
2016. 05. 07. 12:00	8 óra	Bokros Jáhn Tóth	Tevékenység. Bokros, Jáhn és Tóth írja a programot.
2016. 05. 08. 16:00	4 óra	Jáhn	Tevékenység. Jáhn a grafikus megjelenítésen dolgozik.
2016. 05. 13 12:00	12 óra	Bokros Tóth	Tevékenység. Bokros és Tóth írja a programot.
2016. 05. 13. 12:00	4 óra	Jáhn Hegedűs Siket Varga	Tevékenység. Jáhn, Hegedűs, Siket és Varga írja a programot.
2016. 05. 14. 1:30	1,5 óra	Bokros	Tevékenység. Bokros a hazaúton rájön, hogy hogyan kell implementálni az ablakok átméretezését, és megteszi.
2016. 05. 14. 13:00	1 óra	Bokros	Tevékenység. Bokros a teljes kódbázist átnézi, és letisztázza.
2016. 05. 14. 14:00	5 óra	Jáhn	Tevékenység. Jáhn tesztel, bugokat javít, és hiányzó funkciókat valósít meg.
2016. 05. 04. 14:00	5 óra	Tóth	Tevékenység. Tóth bugokat javít, illetve optimalizálja az alkalmazást.
2016. 05. 10. 12:00	3 óra	Hegedüs	Tevékenység. Hegedüs megrajzol pár hiányzó grafikai elemet.

2016. 05. 13. 14:00	1 óra	Hegedüs	Tevékenység, Végleges grafikai elemek elkészítése.
2016. 05. 14. 14:00	10 óra	Varga	Tevékenység. Varga pályákat talál ki, és gyárt le, közben teszteli a programot és javítja a hibákat.
2016. 05. 15. 20:00	4 óra	Varga	Tevékenység. Varga teszteli a programot, a talált hibákat javítja.
2016. 05. 16. 13:00	8 óra	Siket	Tevékenység. Siket kommentezi a kódot és javítja a talált inkonzisztens részeket.
2016. 05. 16. 14:00	4 óra	Varga	Tevékenység. Varga elkészíti a fájllistát, kisebb módosításokat eszközöl.
2016. 05. 16. 20:00	1 óra	Varga	Tevékenység. Varga elkészíti a dokumentációt.