

Szoftver Labor 4 – házi feladat

11 – lvl5_cmm_paladins_

Konzulens:

Demény Fruzsina Gyöngyi

Csapattagok:

Graics Bence
Herold Kristóf
Sallai Gyula
Verbőczy Kristóf

AAKOUJ grbeni@gmail.com
L3YEFG kristof.herold@me.com
H93K9J salla@sch.bme.hu
IVWHC4 verbocika@gmail.com

2015.05.14.

2. Követelmény, projekt, funkcionalitás

2.1. Bevezetés

2.1.2. Cél

Jelen dokumentum célja, hogy bemutassa a LVL5 CMM Paladins csapat által a Szoftver Labor 4 tárgy keretein belül megvalósítandó programot, annak részletes belső áttekintését, elkészítésének tervét.

2.1.3. Szakterület

A kialakítandó szoftver egy grafikus ügyességi játék, amely próbára teszi a felhasználó reakcióidejét, élelmességét, taktikai képességeit. Célja a jókedvű időtöltés, szórakoztatás biztosítása két ember számára.

2.1.4. Definíciók, rövidítések

Dropbox: felhő alapú tárhely szolgáltatás

Eclipse: Java fejlesztőkörnyezet

Git: elosztott verziókezelő szoftver

GitHub: git tárhely szolgáltatás

HSZK: Hallgatói Számítógép Központ

Latex: szövegjelölő nyelv

OpenAmeos: UML szerkesztő

Microsoft Word: grafikus szövegszerkesztő

UML: Unified Modeling Language, modellező szabvány

2.1.5. Hivatkozások

<https://www.iit.bme.hu/~szoftlab4/>

<https://www.iit.bme.hu/~szoftlab4/feladat.shtml>

<https://www.dropbox.com>

<https://www.github.com>

2.1.6. Összefoglalás

A dokumentum további részében a következőkről lesz szó: a szoftver magas szintű áttekintése, amelyben bemutatásra kerülnek a majdani program funkciói, a felhasználókkal szemben elvárt tulajdonságok és az elkészítendő szoftverre vonatkozó előírások. Az elkészítendő szoftverrel szemben támasztott követelmények, a későbbi megvalósításhoz kapcsolódó use-case-ek. Egy szótár, amely a dokumentumban használt nem egyértelmű kifejezések definíciót tartalmazza. Végül a projekt végrehajtásának lépései, az eredmények

határideje, az egyes feladatok elvégzéséért felelős csapattagok neve és munkaköre, a csapat által felhasznált (a csoportmunkát támogató) eszközök, technikák és technológiák.

2.2. Áttekintés

2.2.1. Általános áttekintés

Az alkalmazás egy egyszerű játékprogram, mely a felhasználó által adott bemenetekre reagálva jeleníti meg a képernyőn a játék jelenlegi helyzetét. Ennek figyelembevételével három fő komponenst kell megkülönböztetnünk: a felhasználói bemenetek kezelését, a játék saját logikáját, illetve a játék adott állásának grafikus reprezentációját létrehozó modult.

Milyen modulok, hogyan kapcsolódnak egymáshoz, hogyan reagálnak?

2.2.2. Funkciók

Az elkészítendő szoftver egy oldalnézetes ügyességi platformjáték. Az ilyen fajta játékok lényege, hogy a különböző akadályokat kikerülve, esetleges platformokon való ugrálással, ügyességünket és reakcióidónkat akaratlanul is fejleszтve vezessük végig a pályán irányítandó karakterünket. Ha platformjátékokra(vagy platformer) gondolunk, először a 'Donkey Kong' és a 'Super Mario' játékok juthatnak eszünkbe. Esetünkben a megvalósuló kivitelezés végül inkább a 'Super Mario'-val hozható asszociációba. A platformerek fő, nélkülözhetetlen eleme, mely az összes ilyen játékban megtalálható, az az ugrás gomb, ez esetünkben sem lesz másképp, a játék ezen gomb ügyes használatára helyezi a hangsúlyt.

A játék több játékost igényel, pontosan kettőt. A felhasználóknak, akiknek lehetőségük lesz éppen a játékkal játszani, mindannyian egy-egy robotot fognak irányítani, melyeket vezetve kell a pályán a lehető legnagyobb távot megtenniük. A robotokat előre és hátra lehet mozgatni, illetve ugrani lehet velük.

Tehát a játékosok célja az, hogy a lehető legmesszebb jussanak a pályán. Egyszerűnek tűnhet, viszont a játékosok dolgát nehezíti, hogy egy adott időkorlát idejéig játszhatnak csak, illetve az ügyességi akadályokkal is meg kell küzdeniük, melyek akár végzetesek is lehetnek. A játékosok nem látják előre a pálya egészét, hanem csak annak, az éppen megjeleníthető szélességű részét. Ha a két játékos olyan távolságra kerül egymástól, hogy nem férnének bele a megjeleníthető pályarészbe, akkor a lemaradó játékos kiesik.

Akadályok közé két dolgot sorolhatunk, a különböző ugrást igénylő akadályokat, illetve a foltokat. Az alábbi sorokban az ugrást igénylő akadályokról ejtünk szót, a foltokra később fog kitérni a dokumentáció.

Két különböző esetet vehetünk az ugrást igénylő akadályokhoz, amelyek mindegyike esetén a megfelelő ugrást elmulasztva a játékos robota végzetes károkat szenved, és nem tudja folytatni a versenyt. A játékos ekkor kiesik a játékból.

Először is lehetnek szakadékok a pályán, melynek két széle a talajjal van egy szinten, egy jól időzített ugrással sikerülhet a játékosnak kiküszöbölnie ezt az akadályt.

Másodszor beszélhetünk olyan ugrást megkövetelő akadályokról, amelyek egy nagyobb szakadék áthalását jelenetik. Ekkor a játékosnak ugrások egy sorozatát kell végrehajtania platformról - platformra.

A játékélményt tovább színező funkció lesz a foltok rendszere. Esetünkben nem egy kooperatív játékról beszélünk, hanem egy kompetitív játékról, ennek elengedhetetlen kellékei lesznek a foltok. Fontos, hogy az ellenfél játékos dolgát is nehezítsük, miközben saját maximumunkat próbáljuk teljesíteni, ezáltal mindenki játékos szórakozását tovább fokozzuk. A pályán alapból is el vannak helyezve foltok, és a játékosok is képesek lesznek ugráskor gombnyomásra elhelyezni olajfoltot, vagy ragacsfoltot. A játékosok új játék indítása esetén meghatározott számú foltokkal fognak rendelkezni, amelyek utánpótlására a pálya további részeiben lesz lehetőség.

Minden fajta folt a játékosokat korlátozza mozgásukban, de természetesen ezek a korlátozások nem tartanak örökké, csak egy meghatározott ideig (a hatások időben, vagy téren hatnak). A korlátozás akkor érinti a játékost, ha a robot érintkezett a folttal.

Foltokból két különböző fajtát valósítunk meg. Az olajfoltot, és a ragacsfoltot.

Olajfolt esetén a hatás csak addig tart, amíg a játékos az olajfolt területén tartózkodik, ekkor a játékos képtelen bármiféle irányváltoztatásra, vagy ugrásra. Tehát egy olajfolt esetén azt lehet mondani a való élet terminológiáját használva, hogy a robot megcsúszik.

Ragacsfolt esetén a hatás meghatározott ideig tart, ezalatt a játékos csak fele akkora sebességgel tud haladni, hiszen a robot kerelei lassabban tudnak csak forogni a ragacsos felület miatt, ami felkerült a gumik felszínére.

Összegzésképpen tehát egy olyan több-játékos kompetitív platform-játekról van szó, amely a felhasználók ügyességi képességeit teszi próbára, egyaránt az ugrások megfelelő ütemezésével, illetve a foltok leghatékonyabb elhelyezésével, hogy a másik játékost tudják hátrálhatni a játékban, hogy a fényes dicsőséghez közelebb kerülhessenek.

2.2.3. Felhasználók

A program korhatár nélkül használható bárki által, aki a billentyűzetet, és az ablakos felületeket is készségesen tudja használni. A felhasználó logikai készsége által és a játék egyszerű mivolta miatt az irányítás is hamar elsajátítható.

2.2.4. Korlátozások

A szoftvernek futnia kell Windows 7 operációs rendszeren. A programnak le kell fordulnia Java 1.7-es fejlesztőkörnyezetben, és futtathatónak kell lennie. Továbbá a Java standard könyvtárkészletet kell használnia, nem használhat más csomagokat.

2.2.5. Feltételezések, kapcsolatok

A dokumentumban használt anyagok:

- Feladat kiírás: <https://www.iit.bme.hu/~szoftlab4/feladat.shtml>
- Tárgyhonlap: <https://www.iit.bme.hu/~szoftlab4/>

2.3. Követelmények

2.3.1. Funkcionális követelmények

Azon.	Leírás	Ellenőrzés	Prioritás	Forrás	Use-case	Komment
1.01	Játékot el lehessen indítani	Bemutatás /Kiértékelés	Alapvető	Megrendelő	Új játék indítása	
1.02	Játékos mozgathassa a robotot	Bemutatás /Kiértékelés	Alapvető	Megrendelő	Robot mozgatása	gurulás és ugrás
1.03	Robot leesése a platformról, a játékból való kizárást eredményezze	Bemutatás /Kiértékelés	Alapvető	Megrendelő		robot kiugrik a pályáról
1.04	Játékos tehessen le foltokat	Bemutatás /Kiértékelés	Alapvető	Megrendelő	Foltok lerakása	ragacs, olaj
1.05	Robot mozgását befolyásolja a folt	Bemutatás /Kiértékelés	Alapvető	Megrendelő	Belelépés foltba	ragacs, olaj
1.06	Játékos visszatölthesse a folt készleteit	Bemutatás /Kiértékelés	Opcionális	Csapat	Folt készlet feltöltése	
1.07	A játékot újra lehessen kezdeni	Bemutatás /Kiértékelés	Opcionális	Csapat	Újrakezdés	
1.08	A játékból ki lehessen lépni	Bemutatás	Alapvető	Csapat	Kilépés	

2.3.2. Erőforrásokkal kapcsolatos követelmények

Azon.	Leírás	Ellenőrzés	Prioritás	Forrás	Komment
2.01	Java JDK	Kézileg (Eclipse vagy parancssoron keresztül)	Alapvető	Megrendelő	Fejlesztéshez szükséges
2.02	HSZK-ban számítógép	Bemutatás	Alapvető	Megrendelő	Ezekben a számítógépeken mutatjuk be a programot
2.03	Monitor	Triviális	Alapvető	Csapat	Használathoz szükséges
2.04	Billentyűzet	Triviális	Alapvető	Csapat	Használathoz szükséges
2.05	Eclipse LUNA	Triviális	Alapvető	Csapat	Fejlesztéshez szükséges
2.06	Git - remote repositoryval	Parancssoron keresztül	Alapvető	Csapat	Verziókezeléshez
2.07	Dropbox	Webes felület	Fontos	Csapat	Dokumentumok megosztása és tárolása
2.08	Microsoft Office Word	Triviális	Opcionális	Csapat	Dokumentáláshoz
2.09	OpenAmeos	Triviális	Opcionális	Csapat	UML diagramok készítéséhez

2.3.3. Átadással kapcsolatos követelmények

Azon.	Leírás	Ellenőrzés	Prioritás	Forrás	Komment

3.01	Futnia kell a programnak a HSZK számítógépein.	Bemutatás	Alapvető	Megrendelő	A végső határidő előtt lesz lehetőségünk tesztelni az ottani gépeken.
3.02	Java 1.7	Kézileg (Eclipse vagy parancssoron keresztül)	Alapvető	Megrendelő	Használathoz szükséges
3.03	Windows 7	parancssor	Alapvető	Megrendelő	Használathoz szükséges
3.04	Internet - hozzáférés vagy USB	parancssor	Alapvető	Csapat	Forrásfájlok átviteléhez a HSZK-s számítógépre

2.3.4. Egyéb nem funkcionális követelmények

Azon.	Leírás	Ellenőrzés	Prioritás	Forrás	Komment
4.01	A billentyűzet használatának képessége.	nincs	Alapvető	csapat	
4.02	Alapvető angol tudás	nincs	Opcionális	csapat	
4.03	Képesség a monitoron történő események gyors értelmezésére és reakcióra	nincs	Alapvető	csapat	

2.4. Lényeges use-case-ek

2.6.1. Use-case leírások

Use-case neve	Új játék indítása
Rövid leírás	A játékos elindítja a játékot.
Aktorok	Játékos
Forgatókönyv	A felhasználó rákattint a játékra. Betöltődik a pálya. A robotok kezdőállapotba kerülnek. A foltkészletek alapértékre állítódnak.

Use-case neve	Játék bezárása
Rövid leírás	A játékos kilép a játékból.
Aktorok	Játékos
Forgatókönyv	A játékos megnyomja az ablak jobb felső sarkában található bezárás gombot.

Use-case neve	Játék újraindítása
Rövid leírás	Játékmenet újraindul.
Aktorok	Játékos
Forgatókönyv	Robotok visszakerülnek a kezdőpozíciójukba, az óra is alaphelyzetbe áll. A játékosok foltkészletei is visszatöltődnek.

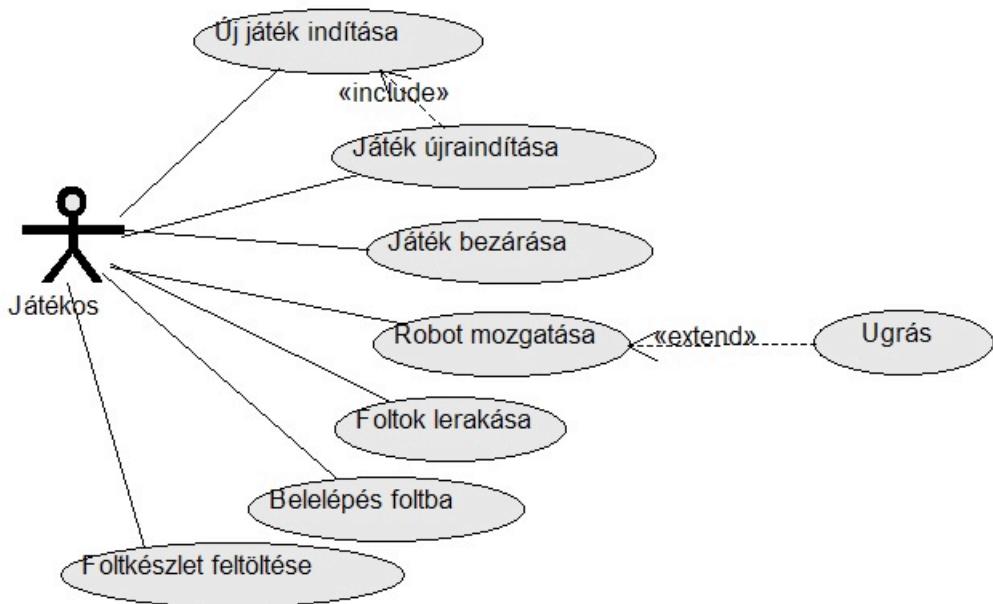
Use-case neve	Robot mozgatása
Rövid leírás	A játékos irányítja a robotot.
Aktorok	Játékos
Forgatókönyv	Játékos irányíthatja a robotot a megfelelő nyomógombok leütésével, és nyomvatartásával.

Use-case neve	Foltok lerakása
Rövid leírás	Játékos elhelyez egy foltot a pályán.
Aktorok	Játékos
Forgatókönyv	A játékos a megfelelő nyomógomb lenyomásával (ha készletei megengedik), egy ragacs- vagy olajfoltot rak a pálya egy részére.

Use-case neve	Foltkészlet feltöltése
Rövid leírás	A játékos folt-töltényt vesz fel.
Aktorok	Játékos
Forgatókönyv	A játékos a foltkészlet feltöltéséért felelős objektum érintésével felveszi a folt-töltényeket.

Use-case neve	Belelépés foltba
Rövid leírás	A játékos belelép egy foltba.
Aktorok	Játékos
Forgatókönyv	A játékos érintkezik egy, a talajra lehelyezett folttal, amely csökkenti a sebességét.

2.6.2. Use-case diagram



2.5. Szótár

akadály: olyan objektum, amelyet a játékosnak ki kell kerülnie

folt: azon objektum, melyet a robotok képesek lerakni, és ha erre rámennék, akkor az befolyásolja a mozgásukat (attól függően, hogy hogyan, lehet olajfolt és ragacsfolt)

időkorlát: a játékosoknak megadott idő áll rendelkezésükre az egymással való versengésre

karakter: lásd robot

kiesik: a játékos kiesés után elveszíti a játékot

kompetitív: versenyszerű

kooperatív: együttműködő

megcsúszik: a játékos nem képes megváltoztatni a robot mozgás állapotát

megjeleníthető szélességű: a látható játéktér mérete egyezik az ablak méretével

olajfolt: a robot ezen megcsúszik

oldalnézet: a játéktér profilból látszik

pálya: a játéktér, melyen a robotok gigászi küzdelmet vívnak

powerup: olyan objektum, melyet felvéve a robot képes lerakni foltot

platform: olyan objektum, melyen a robotok mozognak

platformjáték: olyan játék amely platformokat használ és oldalnézetes

ragacsfolt: olyan folt, amely megfelezi a robot sebességét

robot: a játékost szimbolizáló objektum a pályán

szakadék: a pálya azon része, melyen nincsen platform; ha a robot belemegy akkor kiesik

táv: az időkorlát leteltéig megtett távolság

ugrás gomb: ezzel lehet ugrani

végzetes: amikor kiesik a robot

2.6. Projekt terv

A feladatot egy négy fős csapat fogja végrehajtani. A csapat beosztása során igyekeztünk kiegyensúlyozottan elosztani a szerepeket, továbbá próbáltuk a specializációkat kerülni és maximalizálni a közös munkát – a legtöbb feladatot közös munkával, legalább párban kívánjuk megoldani. Nyilván különböző problémák különböző képességeket igényelnek, így a beosztás létrehozásakor fontos volt az adott ember érdeklődési köre.

Név	Feladatak
Graics Bence	Dokumentáció, Modellezés
Herold Kristóf	Szervezés(csapatvezető), Kód
Sallai Gyula	Kód, Dokumentáció
Verbőczy Kristóf	Kód, Dokumentáció

2.6.1. Verziókezelés

A csapat kommunikációjához és koordinált munkájához feltétlenül szükséges az egyes változtatások rendszerezett menedzselése – erre nyújt megoldást a git verziókövető rendszer, melynek használatával képesek leszünk minden projekt kód bázisának legfrissebb változatát megosztani egymással. Az ehhez szükséges tárhelyet a GitHub biztosítja.

Mivel a Latex kezelését nem mindenki ismeri a csapatból, így minden képp szükséges Word-dokumentum alapú fájlokat is kezelnünk. Bár felmerült, hogy ezeket a fájlokat is git

segítségével oldjuk meg, a gitben kényelmetlen a nem szöveges fájlok kezelése. Ennek tükrében úgy döntöttünk, hogy a dokumentációhoz tartozó fájlokat Dropbox segítségével fogjuk tárolni és megosztani egymással.

2.6.2. Kommunikáció

A csapat kommunikációja lokalitási okokból nagyon könnyen megoldható személyes kommunikációval. Bár napi szinten is könnyen tudunk e célból találkozni, tervezünk heti egyszer egy komolyabb, fix időpontú beszélgetést is tartani. A fontosabb és rögzítendő társalgásokat e-mailen keresztül végezzük majd, feljegyzésekhez pedig a GitHub wiki szolgáltatását fogjuk használni.

2.6.3. Használt segédprogramok

A forráskód szerkesztéséhez az Eclipse fejlesztőkörnyezetet tervezük használni, a dokumentáció Microsoft Word segítségével fog készülni, az UML diagramok pedig az OpenAmeos nevű modellező eszközzel.

2.6.4. Ütemterv, határidők

A projekt ütemterve azonosul a kiadott határidőkkel:

Hatóidő	Feladatok	Ellenőrzés
febr. 23.	Követelmény, projekt, funkcionálitás	beadás
márc. 2.	Analízis modell kidolgozása 1.	beadás
márc. 9.	Analízis modell kidolgozása 2.	beadás
márc. 16.	Szkeleton tervezése	beadás
márc. 23.	Szkeleton	bemutatás
márc. 30.	Prototípus koncepciója	beadás
ápr. 7.	Részletes tervezés	beadás
ápr. 20.	Prototípus	bemutatás
ápr. 27.	Grafikus felület specifikációja	beadás
máj. 11.	Grafikus változat	bemutatás
máj. 15.	Összefoglalás	beadás

2.7. Napló

Kezdet	Időtartam	Résztvevők	Leírás
2015.02.17. 20:30	1,5 óra	Sallai Herold Verbőczy Graics	Értekezlet. Elemezzük a kiadott feladatot. Felmerülnek a kezdeti problémák. (Milyen nézetű legyen a játék, ugrás implementációja.) Kigyűjtjük és lejegyezzük ezen kérdéses pontokat, hogy másnap beszélhessünk róluk a konzulensünkkel. Sallai vezet a verziókezelés alapjaiba.
2015.02.18 20:20	1,5 óra	Sallai Herold Verbőczy Graics	Értekezlet. Véglegesítjük a projekttel kapcsolatos elképzéseinket. (Játék nézete, ugrás implementációja.) Az aznapi konzultáció alapján ábeszéljük az első dokumentum anyagát, kiosztjuk az egyes feladatokat: Graics csinálja a 2.1-es pont egészét és a 2.3.1-es pontot. Herold elkészíti 2.2.2-es pontot és a 2.3.2-es pontot. Sallai elkészíti a 2.2.1-es pontot, a 2.3.3-as pontot és a 2.6-os pontot. Verbőczy elkészíti a 2.2.3, 2.2.4, 2.2.5, 2.3.4-es pontokat. A Use-case eseteket közösen fogjuk megoldani. A szótárban minden elkészíti a saját munkájában szereplő kifejezéseket, majd ha mindenki megvan, összedolgozzuk. Döntünk a projekt elkészítésekor használt technológiáról. (Iسد.: 2.6-os pont.)
2015.02.19 15:20	1,5 óra	Graics	Tevékenység: Graics elkészíti a dokumentáció 2.1-es Bevezetés pontját. Leírja az értekezletek menetét és eredményeit.

2015.02.19 15:30	1,5 óra	Verbőczy	Tevékenység: Verbőczy elkészíti a dokumentáció 2.2.3, 2.2.4, 2.2.5, 2.3.1, 2.3.4 pontjait.
2015.02.20. 18:03	2 óra	Herold	Tevékenység: Herold elkészíti a dokumentáció 2.2.2-es pontját, kb. 4000 karakterben ismerteti a játék funkcióit.
2015.02.20. 21:00	0,5 óra	Herold	Tevékenység: Herold elkészíti a dokumentáció 2.3.2-es pontját (Erőforrásokkal kapcsolatos követelmények).
2015.02.20. 21:00	0,5 óra	Graics	Tevékenység: Graics elkészíti a dokumentáció 2.3.1-es pontját (Funkciókkal kapcsolatos követelmények).
2015.02.21. 18:24	1 óra	Sallai	Tevékenység: Sallai elkészíti a dokumentáció 2.2.1-es pontját, illetve a 2.6-os pontot és annak alpontjait.
2015.02.22. 19:20	0,5 óra	Graics Herold Sallai Verbőczy	Tevékenység: A csapat megvitatja a use-case-eket, közösen kitöltik a dokumentáció megfelelő táblázatait.

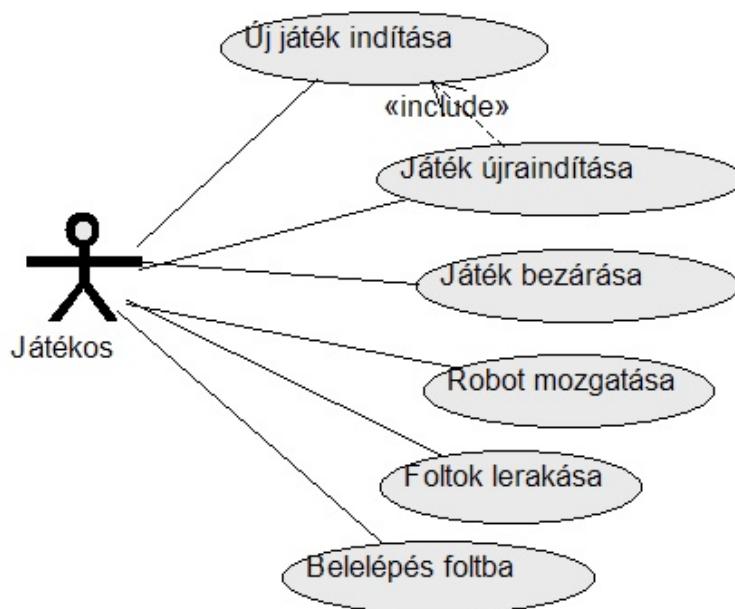
2015.02.22. 19:50	0,25 óra	Graics	Tevékenység: Graics elkészíti a Use-case diagramot az OpenAmeos nevezetű programmal.
2015.02.22. 20:30	0,5 óra	Herold Sallai Verbőczy	Tevékenység: Szótár tartalmának megbeszélése, illetve dokumentálása.
2015.02.22. 21:00	0,75 óra	Herold	Tevékenység: Dokumentáció összeszerkesztése, ellenőrzése.
2015.02.22. 22.00	0,5 óra	Graics Verbőczy	Tevékenység: Dokumentáció ellenőrzése.

0. Funkcióbéli változtatások

A csapat közös megegyezés után az előző dokumentációban felvázolt funkcionálisok változtatása mellett döntött.

Az eredeti oldalnézetes, valós idejű platformer játék elgondolásról inkább egy felülnézetes körökre osztott játék koncepciójára állunk át. A robotok csak ugrálva lesznek képesek mozogni. A körökben a játékosoknak meg kell adniuk egy irányt, hogy merre szeretnének ugrálni a robotokkal. Ekkor lesz lehetőség arra is, hogy a játékosok, mielőtt elmozognak egy adott mezőről, foltokat helyezzenek a pályára. A robotok nyolc különböző irányba lesznek képesek elmozdulni. A cél továbbra is az, hogy a megadott időkorlát ideje alatt a lehető legmesszebb jussanak.

A pálya négyzetekre osztott, a robotok ezeken ugrálhatnak, illetve helyezhetnek el foltokat. A játék elindítása esetén a pálya már alapból tartalmazhat láthatatlan, illetve látható foltokat. Akadályok közül kikerültek azon fajta szakadékok, melyek a platformokon való ugrálást igényelték volna. Viszont a játék továbbra is tartalmazni fog kisebb szakadékokat, melyeket a játékos átugorhat. A játékból kikerült a Power Up funkció, tehát a játékosnak nem lesz lehetősége a folt készleteinek feltöltésére. Ezek után a use-case diagram a következő lett:



Továbbá módosítottunk az eszköztárunkon is: egyes UML diagramok rajzolására az Eclipse platform Papyrus UML rajzoló pluginját is elkezdtük használni.

3. Analízis modell kidolgozása

3.1 *Objektum katalógus*

3.1.1 Cell

A négyzetrácsos pálya egy-egy négyzetét jelenti egy Cell. Egy cellán tartózkodhat legfeljebb egy játékos, illetve legfeljebb egy folt.

3.1.2 Game

A játék logikáját tartalmazó osztály. Itt tartjuk számon a játékosokat, a térképet, itt kezeljük a kapott bemeneteket.

3.1.3 Map

Itt tárolunk minden olyan információt, amely a térképpel kapcsolatos, azaz itt tároljuk a Cellekből álló kétdimenziós tömbünket is, illetve az adott pályára vonatkozó maximális játékidőt.

3.1.4 Player

A játékosokat megtestesítő objektum. minden játékos egy-egy Playernek feleltethető meg. minden játékos pontosan egy cellán tartózkodik.

3.1.5 (Glue)Stain, (Oil)Stain

A foltokat megvalósító osztályok. minden folt szerepelhet pontosan egy cellán. Ha egy játékos olyan cellára lép, amelyen folt is van, a folt valamilyen negatív hatást fejt ki a játékosra.

3.2 Statikus struktúra diagramok

3.3 Osztályok leírása

3.3.1 Cell

- Felelősség

A játéktér egy celláját megtestesítő osztály. Egy cellán legfeljebb egy objektum helyezkedhet el. Amennyiben egy játékos a cellára lép, a cella erről értesíti a rajta található objektumokat.

- Ősosztályok

nincs

- Interfészek

EventSubscriber

- Attribútumok

- **objects**: A cellán elhelyezkedő objektumok listája
- **currentPlayer**: Az aktuálisan cellán álló játékos.
- **type**: A cella típusa, lehet érvényes (VALID), vagy érvénytelen (INVALID). Amennyiben egy játékos érvénytelen cellára lép, azonnal elveszíti a játékot.

- Metódusok

- **void interact()**: Jelzi a cella összes objektumának, hogy p játékos a cellára lépett. Ezen metódus segítségével tudják a cellán lévő foltok kifejteni hatásukat.

3.3.2 EventSubscriber

- **Felelősség**

Minden játékelemnek ezt az interfészt kell implementálnia. Metódusai segítségével tudunk a Game osztály eseményeire reagálni, illetve szükség esetén kirajzolni az entitást.

- **Ősosztályok**

nincs

- **Metódusok**

- **void onTurnStart()**: A kör eleje eseményt kezelő metódus
- **void onTurnEnd()**: A kör vége eseményt kezelő metódus
- **void draw()**: Amennyiben újra kell rajzolnunk a pályát, ezzel a metódussal rajzolhatjuk ki az adott entitást.

3.3.3 Game

- **Felelősség**

A rendszer központi osztálya. Nyilvántartja a játékosokat és a játékelemeket. Az eseményközpontú rendszer hírforrása, minden regisztrált játékelem ide iratkozik fel.

- **Ősosztályok**

nincs

- **Interfészek**

nincs

- **Attribútumok**

- **map**: A játéktér tárolására.
- **players**: A játékosok listája
- **turnCount**: Az éppen aktuális kör száma

- **Metódusok**

- **void start()**: Jelzi a játék indulását, hatására felépül a pálya és megkezdődik az első kör. A játék ebben az állapotban marad újraindításig és kilépésig.
- **void reset()**: Újraindítja az aktuális játékot. minden játékos visszakerül a kezdőpozícióba, foltkészleteik feltöltődnek.
- **void quit()**: Befejezi a játékot és kilép a programból.
- **void endTurn()**: Jelzi a kör végét az összes feliratkozott entitásnak.
- **void beginTurn()**: Jelzi a kör elejét az összes feliratkozott entitásnak.

3.3.4 Map

- **Felelősség**

A játékeret megtestesítő osztály. Ő tartalmazza a játéktér celláit és az adott játéktérhez kapcsolódó információkat (pl. a maximális körök számát). minden megkapott eseményt továbbad a benne tárolt celláknak.

- **Ősosztályok**

nincs

- **Interfészek**

EventSubscriber

- **Attribútumok**

- **cells**: Egy kétdimenziós adatszerkezet, amely a játék celláit tartalmazza.
- **maxTurns**: A maximális körök száma

- **Metódusok**

nincs nem öröklött metódusa

3.3.5 Object

- **Felelősség**

Egy nem játékos objektum, amely interakcióba léphet egy játékossal. Ezek lehetnek például a játékosra ható effektek. Ez az osztály egy absztrakt ősosztály.

- **Ősosztályok**

nincs

- **Interfészek**

EventSubscriber

- **Attribútumok**

nincs

- **Metódusok**

- **interact(Player player)**: Alkalmazza az adott hatást a player játékosra.

3.3.6 Player

- **Felelősség**

Egy-egy játékos karaktert megtestesítő objektum. minden körben az éppen aktuális játékost tudjuk a bemenetek segítségével manipulálni.

- **Ősosztályok**

nincs

- **Interfészek**

EventSubscriber

- **Attribútumok**

- **index**: A játékos azonosítója. Egyedinek kell lennie.

- **currentCell**: Az a cella, ahol a játékos éppen áll.

- **storedStains**: Egy kulcs-érték alapú adatszerkezet, amelyben nyilvántartjuk, hogy egy adott típusú foltból hány darab áll a játékos rendelkezésére.

- **speed**: Az adott játékos sebessége.

- **canChangeDirection**: Jelzi, hogy egy játékos képes-e irányt változtatni (ez alapértelmezésben igaz, de pl. olajfoltra lépés esetén nem).

- **Metódusok**

- **void move(Cell cell)**: Átlépteti a játékost egy másik cellára.

- **void putStain(String stainType)**: Elhelyez egy foltot az aktuális cellán.

3.3.7 Stain

- **Felelősség**

A specifikációban szereplő foltok absztrakt ősosztálya. Önálló szerepe nincs, csupán egy későbbi esetleges bővíthetőség miatt van jelen.

- **Ősosztályok**

Object

- **Interfészek**

EventSubscriber

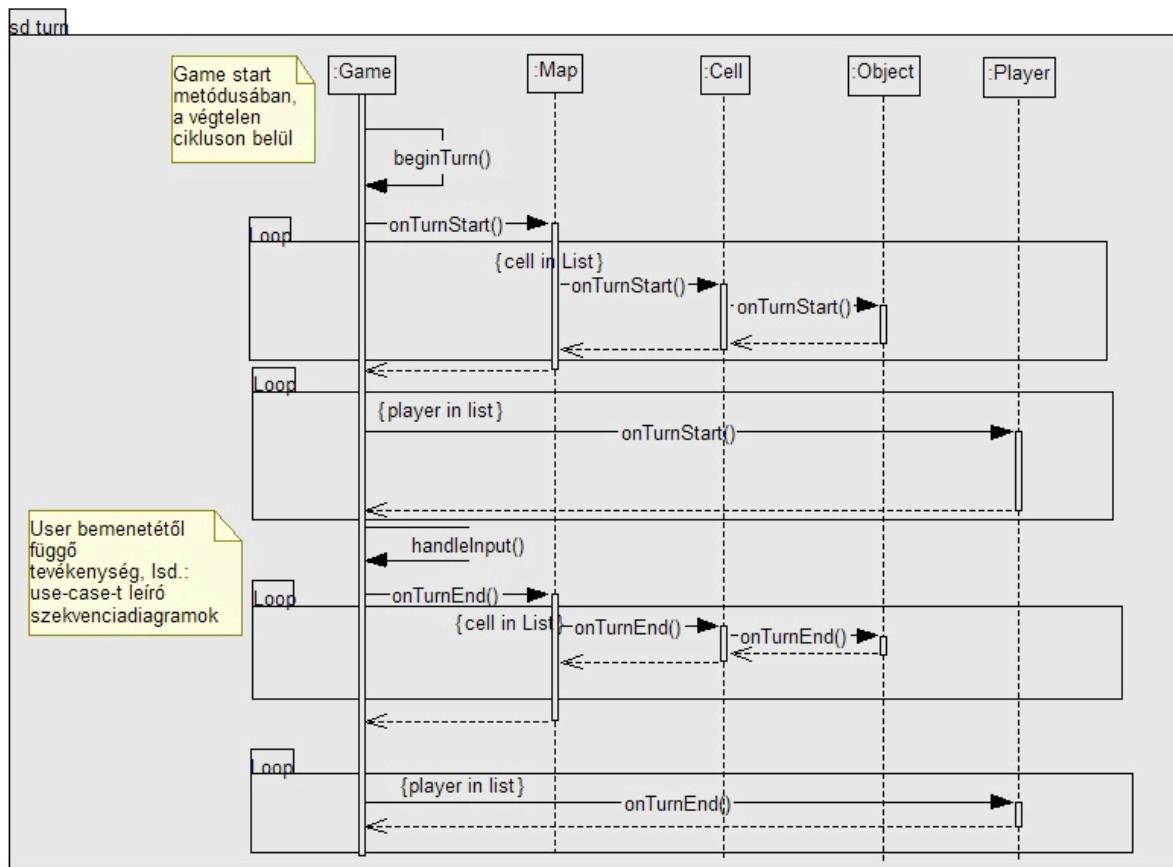
- **Attribútumok**

nincs

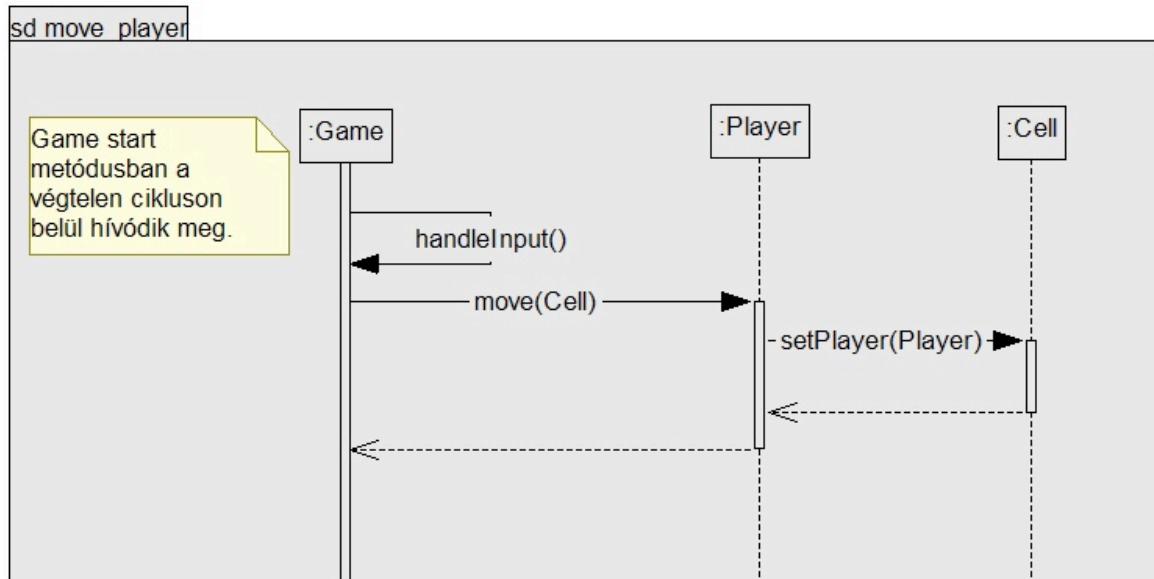
- **Metódusok**

nincs

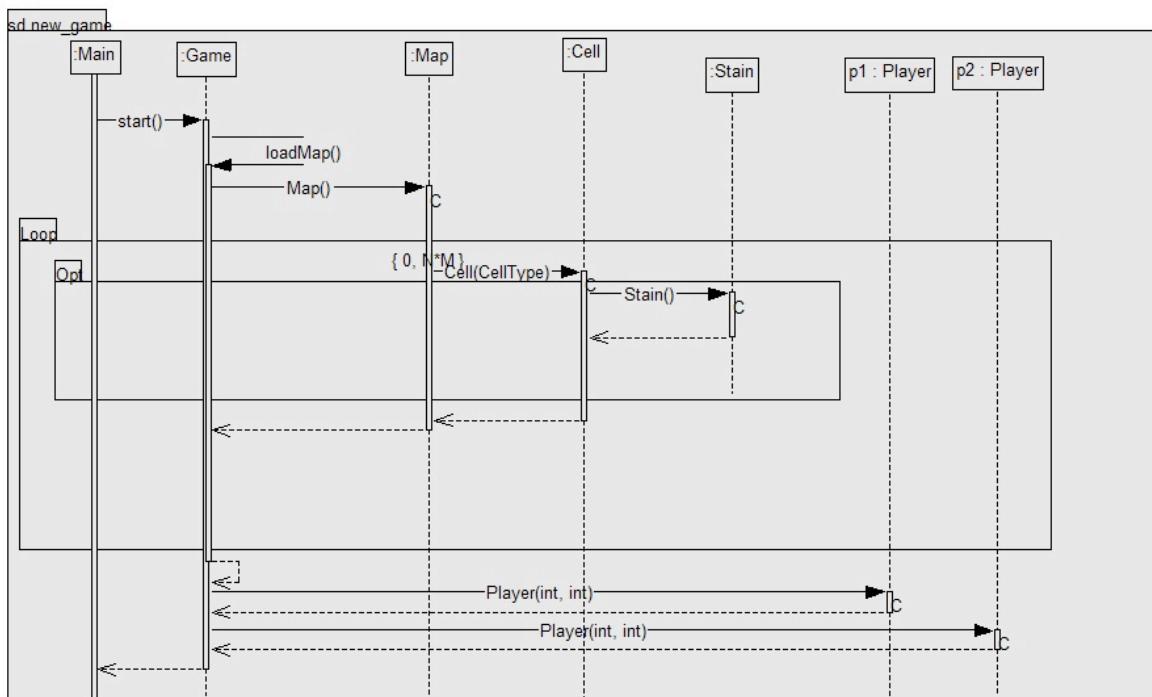
3.4 Szekvencia diagramok



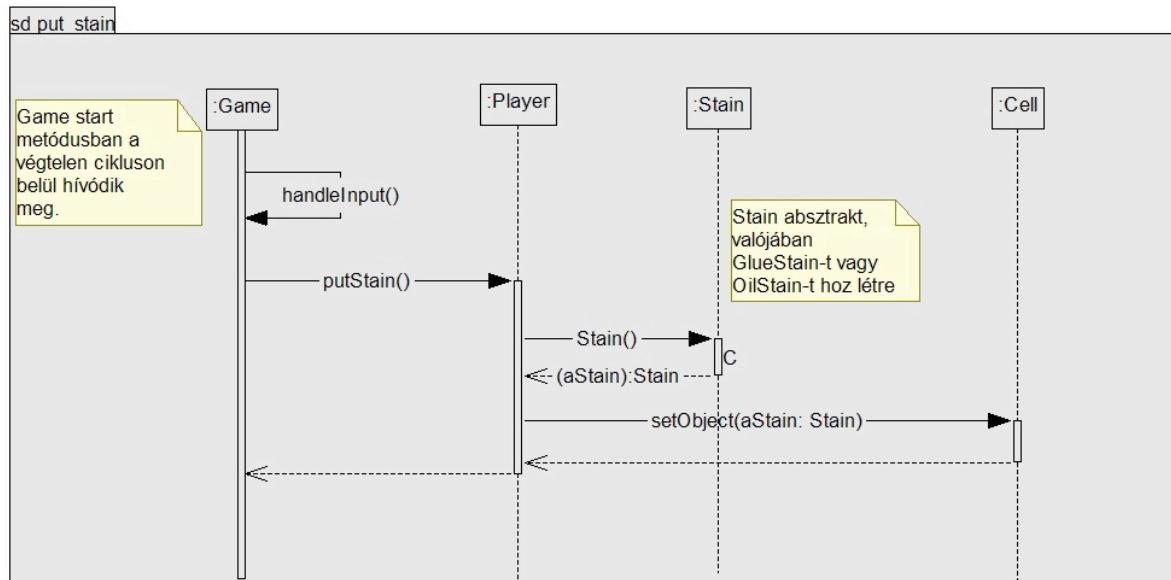
A robot mozgatását bemutató szekvenciadiagram



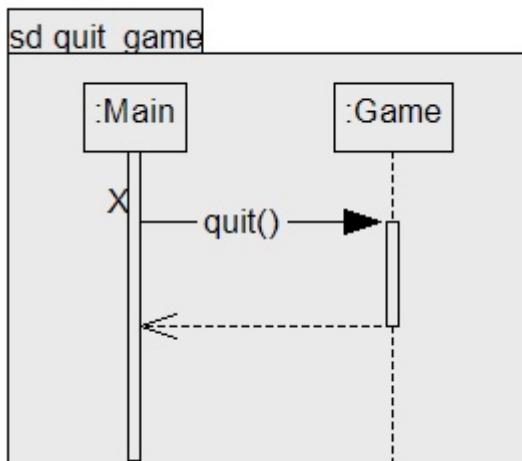
A játék kezdetét, a pálya létrehozását bemutató szekvenciadiagram.



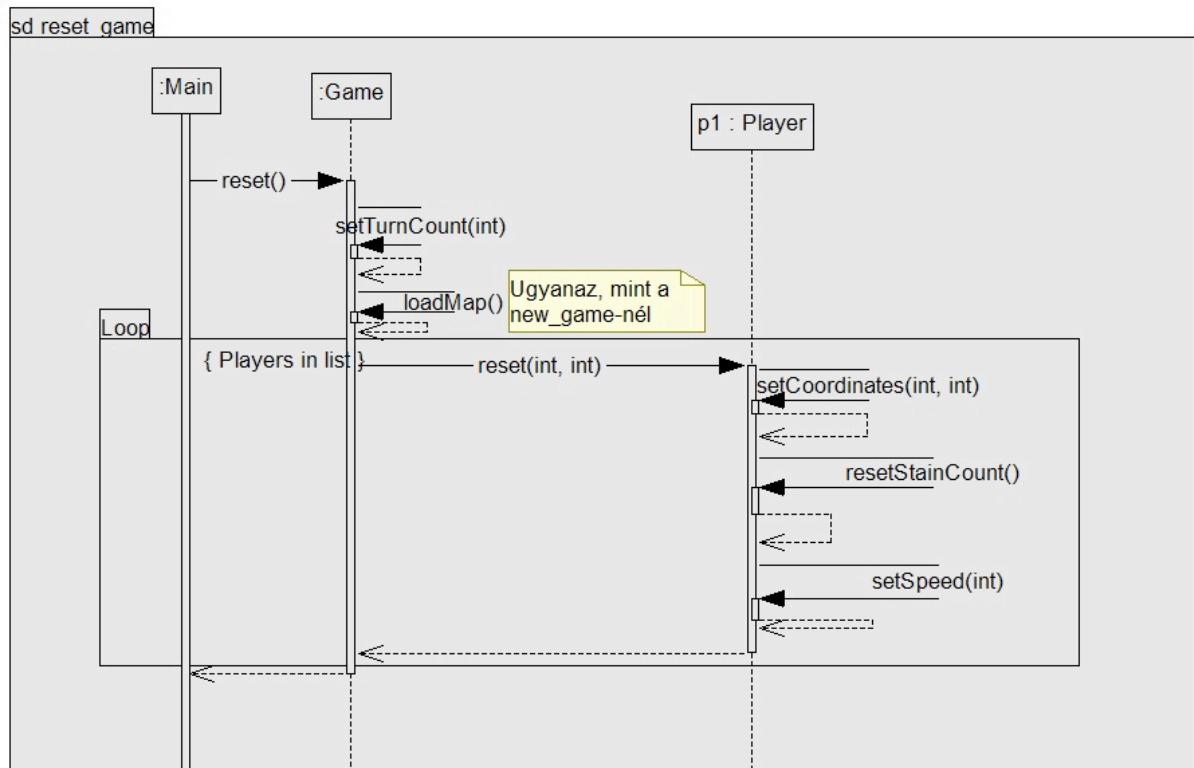
Egy folt letételelét bemutató szekvenciadiagram



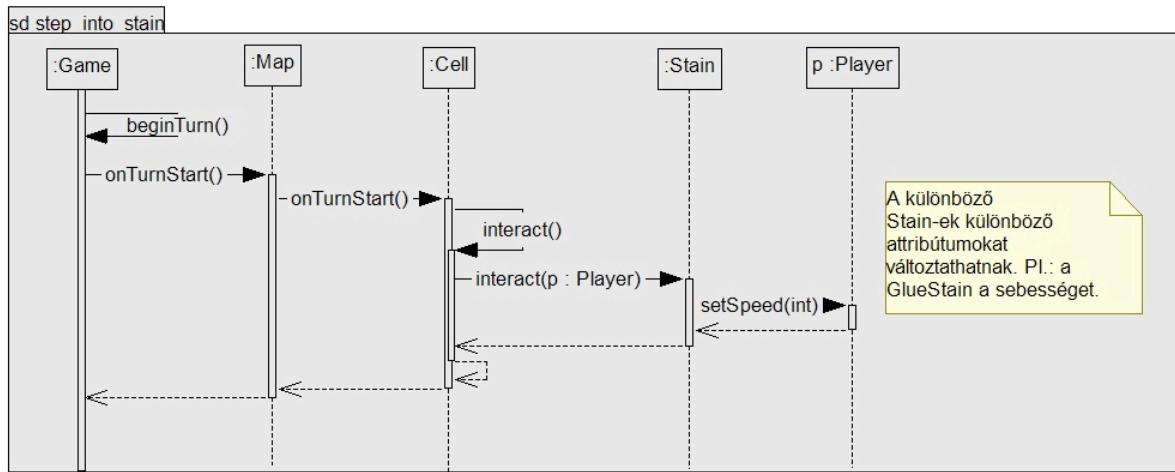
A programból való kilépést bemutató szekvenciadiagram



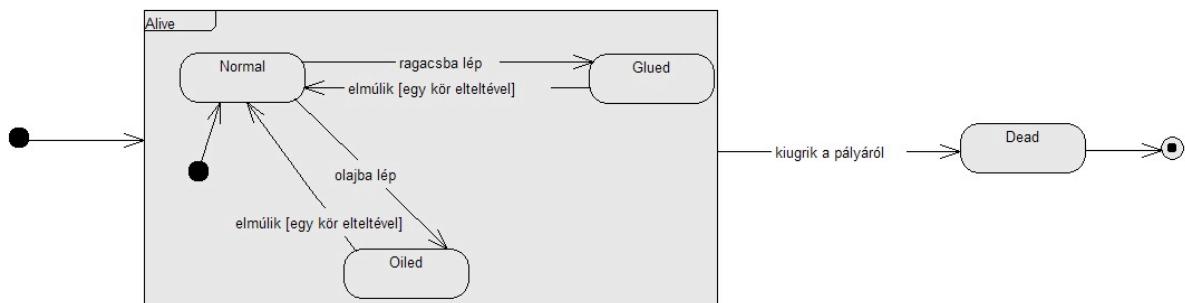
A játék újraindítását bemutató szekvenciadiagram



A foltba lépést bemutató szekvenciadiagram



3.5 State-chartok



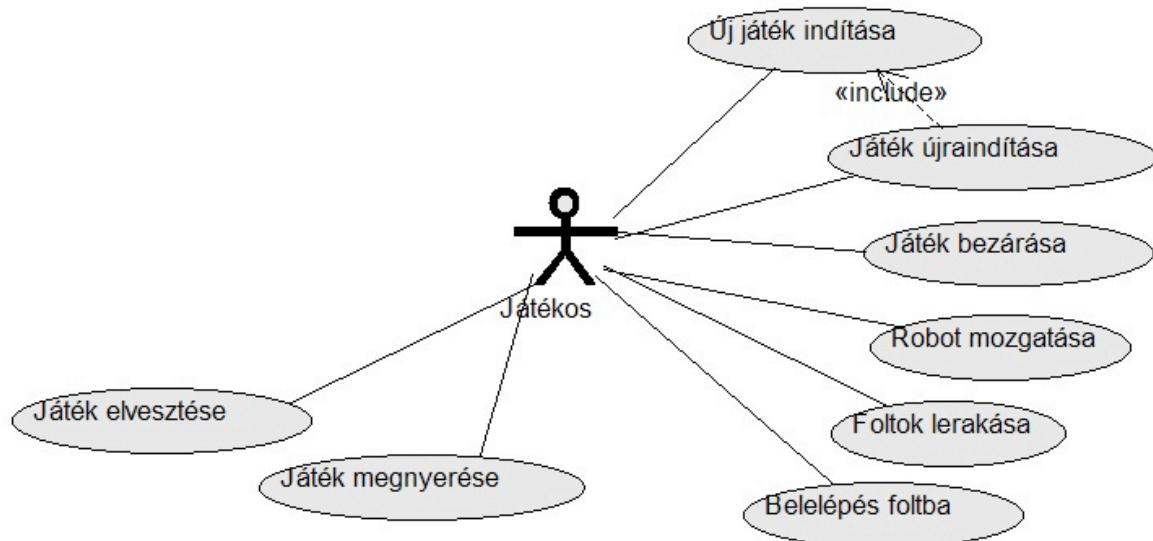
3.6 Napló

Kezdet	Időtartam	Résznevők	Leírás
2015.02.25. 20:00	0,7 óra	Graics Herold Sallai Verbőczy	Értekezlet. Döntés: Szemléletváltás a játék nézetét és az idő folyását illetően: felső nézetes, körökre osztott játékot fogunk készíteni. Herold elkészíti a váltáshoz szükséges dokumentumokat. Sallai ismerteti az ötletét az osztálydiagrammal kapcsolatban. Többiek javaslatokat tesznek.
2015.02.28. 16:00	0,5 óra	Herold	Tevékenység: Herold dokumentálta, hogy milyen változtatásokat eszközölt a csapat a funkciókban.
2015.02.28. 19:00	2 óra	Graics Herold Sallai Verbőczy	Online megbeszélés: Osztálydiagram tervének elkészítése. A játék menetének pontosítása.
2015.02.28. 21:00	0,5 óra	Verbőczy	State-chart elkészítése.
2015.02.28. 21:00	5 óra	Sallai	Az osztálydiagram, az objektumkatalógus és az

osztálykatalógus kidolgozása			
2015.02.28. 17:00	2,5 óra	Graics	A use-case-ekhez tartozó szekvenciadiagramok elkészítése. Graics: Robot mozgatása. Foltok lerakása, Belelépés foltba.
2015.02.28. 17:00	2,5 óra	Herold	A use-case-ekhez tartozó szekvenciadiagramok elkészítése. Herold: Új játék indítása, Játék újraindítása, Játék bezárása.
2015.02.28. 20:00	2 óra	Graics	Egy kört bemutató szekvenciadiagram elkészítése. Egyedi munkák leellenőrzése, átolvasása.
2015.02.28. 21:00	2 óra	Verbőczy	Szekvenciadiagramok ellenőrzése. Dokumentum megszerkesztése.

0. Use-case változások

A use-case-ekhez felkerültek a játék megnyerése és a játék elvesztése esetek.



4. Analízis modell kidolgozása

4.1 Objektum katalógus

4.1.1 Cell

A négyzetrácsos pálya egy-egy négyzetét jelenti egy Cell. Egy cellán tartózkodhat legfeljebb egy játékos, illetve legfeljebb egy folt.

4.1.2 Game

A játék logikáját tartalmazó osztály. Itt tartjuk számon a játékosokat, a térképet, illetve a maximális körök számát, továbbá itt kezeljük a kapott bemeneteket.

4.1.3 Map

Itt tárolunk minden olyan információt, amely a térképpel kapcsolatos, azaz itt tároljuk a Cell-ekből álló kétdimenziós tömbünket is.

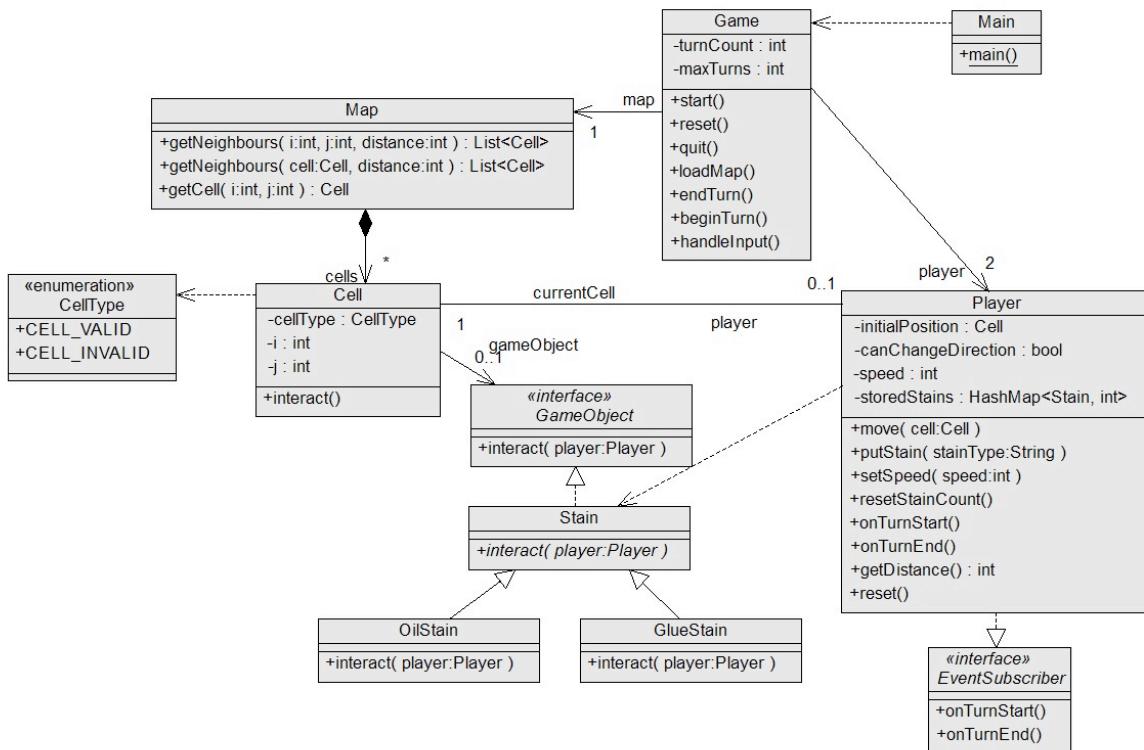
4.1.4 Player

A játékosokat megtestesítő objektum. minden játékos egy-egy Playernek feleltethető meg. minden játékos pontosan egy cellán tartózkodik.

4.1.5 (Glue)Stain, (Oil)Stain

A foltokat megvalósító osztályok. minden folt szerepelhet pontosan egy cellán. Ha egy játékos olyan cellára lép, amelyen folt is van, a folt valamilyen negatív hatást fejt ki a játékosra.

4.2 Statikus struktúra diagramok



4.3 Osztályok leírása

4.3.1 Cell

- **Felelősség**

A játéktér egy celláját megtestesítő osztály. Egy cellán tartózkodhat egyszerre legfeljebb egy játékos és legfeljebb egy folt (azaz léteznek üres cellák is). Amennyiben egy játékos a cellára lép, a cellán tartózkodó folt interakcióba lép a játékossal, tehát kifejteti hatását a játékosra a következő kör elején.

- **Attribútumok**

- **gameObject:** A cellán elhelyezkedő objektumok(folt (logikai-)vagy játékos) listája
- **player:** Az aktuális cellán álló játékos.
- **type:** A cella típusa, lehet érvényes (VALID), vagy érvénytelen (INVALID). Amennyiben egy játékos érvénytelen cellára lép, azonnal elveszíti a játékot. Érvénytelen cellának számítanak a szakadékok, illetve a pálya szélén túl elhelyezkedő cellák. Érvényes cellának számít minden más egyéb cella, amely nem érvénytelen.
- **i, j:** Az adott cellának a koordinátái a mátrixban(a Map osztályban).

- **Metódusok**

- **void interact()**: Ha a játékos az adott cellára lépett, ezen metódus hívódik meg. A metódus pedig a cellán elhelyezkedő gameObjectnek hívja meg a megfelelő metódusát, amely a paraméterként átvett Player objektumra kifejti a hatását. Ha a cellán nincsen folt, természetesen nem történik semmi.

4.3.2 EventSubscriber

- **Felelősség**

Metódusai segítségével tudunk a Game osztály eseményeire reagálni. A későbbi bővíthetőség szem előtt tartása tette indokolttá az interfész bevezetését.

- **Metódusok**

- **void onTurnStart()**: A kör eleje eseményt kezelő metódus.
- **void onTurnEnd()**: A kör vége eseményt kezelő metódus.

4.3.3 Game

- **Felelősség**

A rendszer központi osztálya. Nyilvántartja a játékosokat, a térképet, és a további játékelemeket. Ő felelős a játék vezérléséért (játék indítás, újraindítás, kilépés). Továbbá felelős a játék mechanikájáért, tehát vezérli a körök lejátszását (kör indítása, befejezése, input kezelése).

- **Attribútumok**

- **map**: A játéktér tárolására.
- **players**: A játékosok listája.
- **turnCount**: Az éppen aktuális kör száma.
- **maxTurns**: A maximális körök száma, egy konstans érték.

- **Metódusok**

- **void start()**: Jelzi a játék indulását, hatására felépül a pálya, játékosok létrejönnek, és megkezdődik az első kör. A játék ebben az állapotban marad újraindításig és kilépésig.
- **void reset()**: Újraindítja az aktuális játékot. minden játékos visszakerül a kezdőpozícióba, foltkészleteik feltöltődnek. A turnCount nulla értéket kap.
- **void quit()**: Befejezi a játékot és kilép a programból.
- **void endTurn()**: Jelzi a kör végét az összes feliratkozott játékosnak. Tehát meghívja a játékosok *onTurnEnd()* metódusát.
- **void beginTurn()**: Jelzi a kör elejét az összes feliratkozott játékosnak. Tehát meghívja a játékosok *onTurnStart()* metódusát.
- **void handleInput()**: Kezeli a kapott bemenetet.

4.3.4 Map

- **Felelősség**

A játékteret megtestesítő osztály. A játéktér celláit egy kétdimenziós tömbben(mátrix) tárolja, illetve innen kérhetők le egy-egy cella szomszédai.

- **Attribútumok**

- **cells**: Egy kétdimenziós adatszerkezet(mátrix), amely a játék celláit tartalmazza.

- **Metódusok**

- **Cell getCell(int i, int j)**: Visszatér a paraméterben megadott indexek által meghatározott cellával .
- **List<Cell> getNeighbours(Cell cell, int distance)**: Paraméterként átvesz egy cellát és visszatér egy cellákat tartalmazó listával, amely azokat a cellákat tartalmazza,

amelyekre el lehet jutni a paraméterben átvett celláról. Átvesz egy egész számot, amely azt határozza meg, hogy milyen messzire lehet eljutni az adott celláról.

- **List<Cell> getNeighbours(int i, int j, int distance):**

Működése hasonló a fentihez, csak cella helyett koordinátákat(mátrix indexet) vesz át.

4.3.5 GameObject

- **Felelősség**

Egy nem játékos objektum, amely interakcióba léphet egy játékossal. Ezek lehetnek például a játékosra ható effektek. A bővíthetőség kedvéért alkalmazzuk, arra az esetre, ha esetleg a foltokon kívül egyéb nem játékos objektum is szükséges lenne.

- **Metódusok**
- **interact(Player player):** Alkalmazza az adott hatást a player játékosra.

4.3.6 Player

- **Felelősség**

Játékosokat megtettesítő osztály. minden körben az éppen aktuális játékosot tudjuk a bemenetek segítségével manipulálni, illetve rá jellemző információkat lekérdezni. Például, hogy melyik cellán tartózkodik, vagy hogy mekkora a sebessége.

- **Interfészek**

EventSubscriber

- **Attribútumok**
- **currentCell:** Az a cella, ahol a játékos éppen áll.
- **initialPosition:** Egy cella objektum, amelyen a játékos a játék elindításakor áll.
- **storedStains:** Egy kulcs-érték alapú adatszerkezet, amelyben nyilvántartjuk, hogy egy adott típusú foltból hány darab áll a játékos rendelkezésére.
- **speed:** Az adott játékos sebessége.
- **canChangeDirection:** Jelzi, hogy egy játékos képes-e irányt változtatni (ez alapértelmezésben igaz, de pl. olajfoltra lépés esetén nem).
- **Metódusok**
- **void move(Cell cell):** Átlépteti a játékosot a paraméterként átvett cellára. Tehát átállítja a referenciát(currentCell).
- **void putStain(String stainType):** Elhelyez egy foltot az aktuális cellán, ha a foltkészlete megengedi. Mindig lehet le a foltot, ha valid cellán tartózkodik, hiszen ha egy cellán elhelyezkedő foltra rálép, akkor az megszűnik. (Természetesen a cella nem lehet invalid.)
- **void onTurnStart():** Reagál a kör eleje eseményre. Meghívja annak a cellának az *interact()* metódusát, amelyiken tartózkodik.
- **void onTurnEnd():** Reagál a kör vége eseményre. A *player* megfelelő attribútumait alaphelyzetbe állítja (például: sebesség).
- **void resetStainCount():** A *player* foltkészletének alapértelmezettre állítása.
- **void reset():** A játékos egyes attribútumainak alapértékre állítása a játék újraindítása esetén. Itt állítjuk vissza a sebességét, illetve hogy tud-e irányt váltani.
- **int getDistance():** A currentCell és initialPosition közti táv. Játék megnyerése esetén a játékosok megtett távját hasonlítjuk össze.

4.3.7 Stain

- **Felelősség**

A specifikációban szereplő foltok absztrakt ősosztálya. Önálló szerepe nincs, csupán egy későbbi esetleges bővíthetőség miatt van jelen.

- **Interfészek**

GameObject

- **Metódusok**
- **interact(Player player)**: Abszrakt metódus, amelyet a leszármazott osztályok felüldefiniálnak.

4.3.8 GlueStain

- **Felelősség**

A ragacsfoltokat megtestesítő osztály. Erre lépve a játékos sebessége megfeleződik.

- **Ősosztályok**

Stain

- **Interfészek**

GameObject

- **Metódusok**

- **interact(Player player)**: Megfelezi a *player* játékos sebességét. Tehát meghívja a *player* *setSpeed(..)* metódusát, amellyel beállítja az attribútumot a megfelelő értékre.

3.3.9 OilStain

- **Felelősségek**

Az olajfoltokat megtestersítő osztály. Erre lépve a játékos az adott körben nem képes irányítani a mozgását.

- **Ősosztályok**

Stain

- **Interfészek**

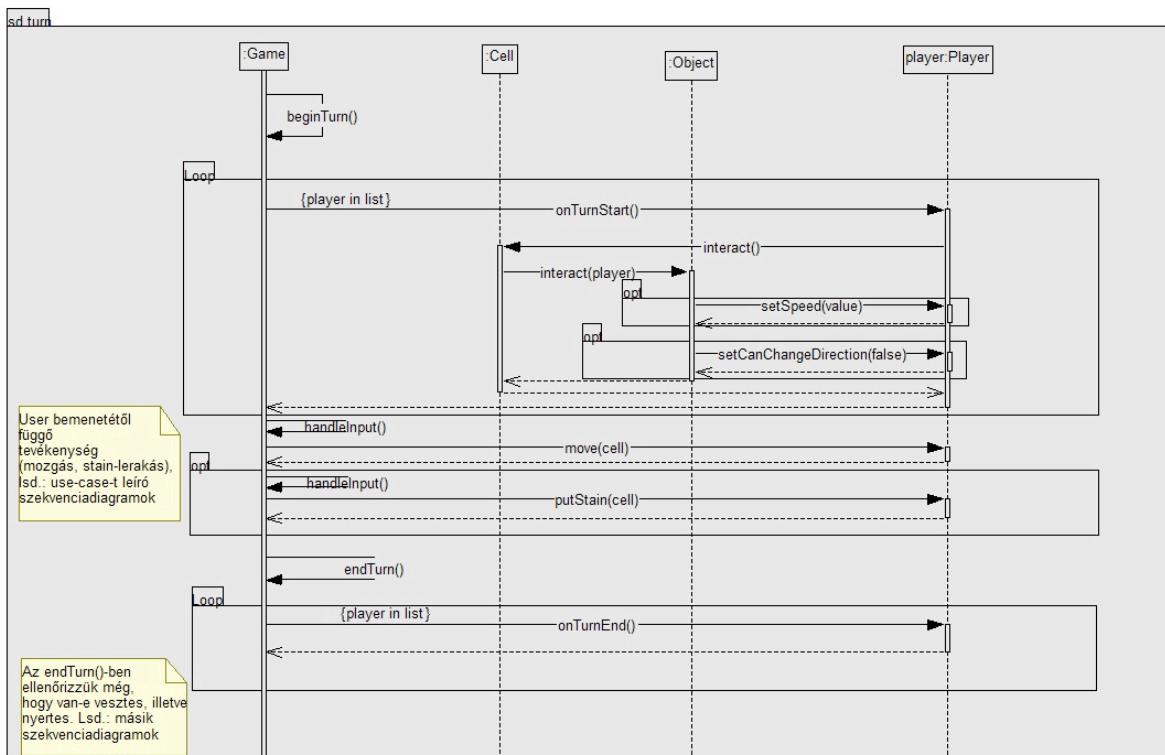
GameObject

- **Metódusok**

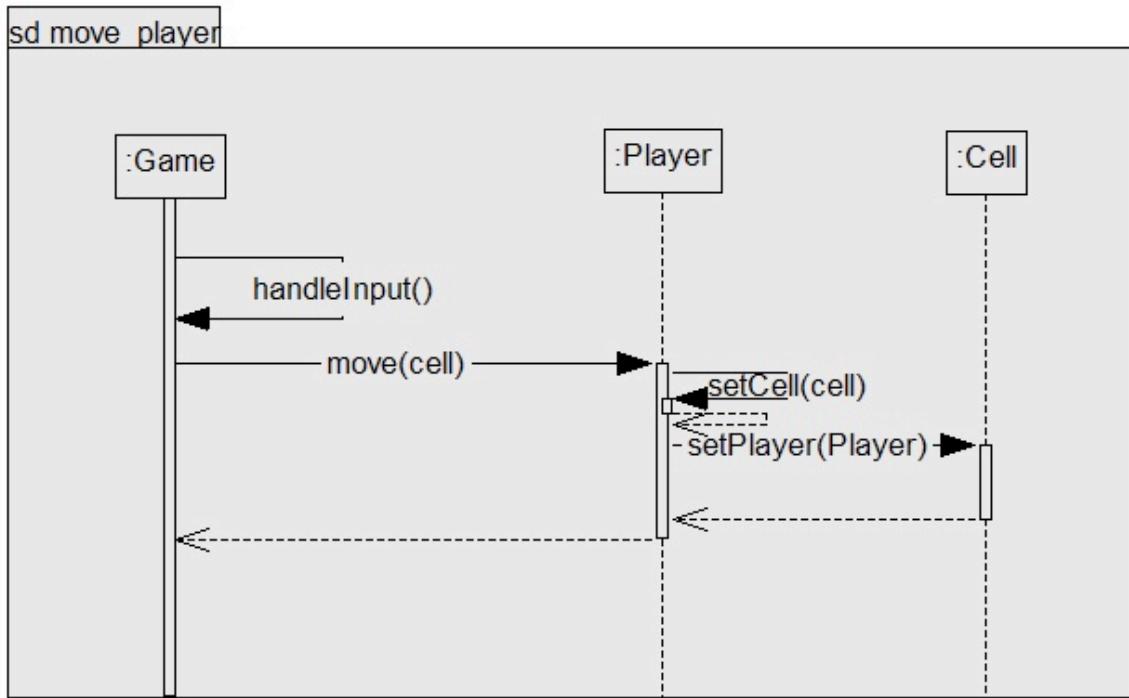
- **interact(Player player)**: Elhelyezi a *player* játékosot egy olyan véletlenszerű cellán, mely elérhető a jelenlegi pozíciójából, továbbá letiltja a játékos mozgását egy körre.

4.4 Szekvencia diagramok

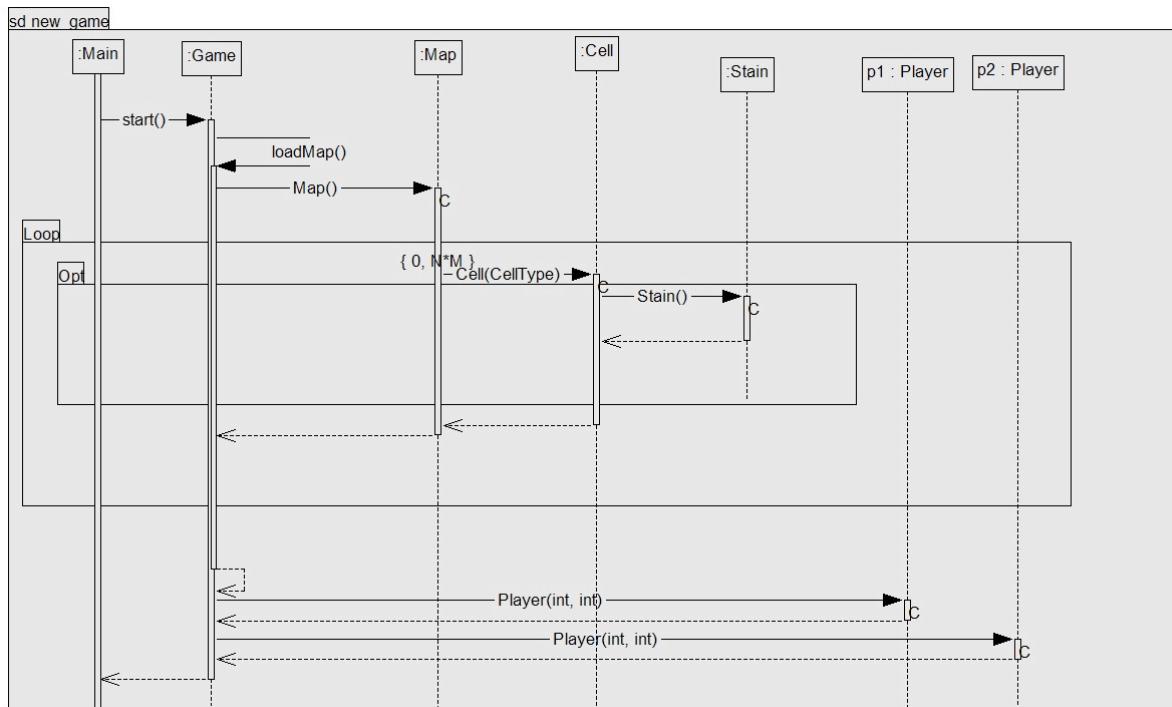
Egy kört bemutató szekvenciadiagram



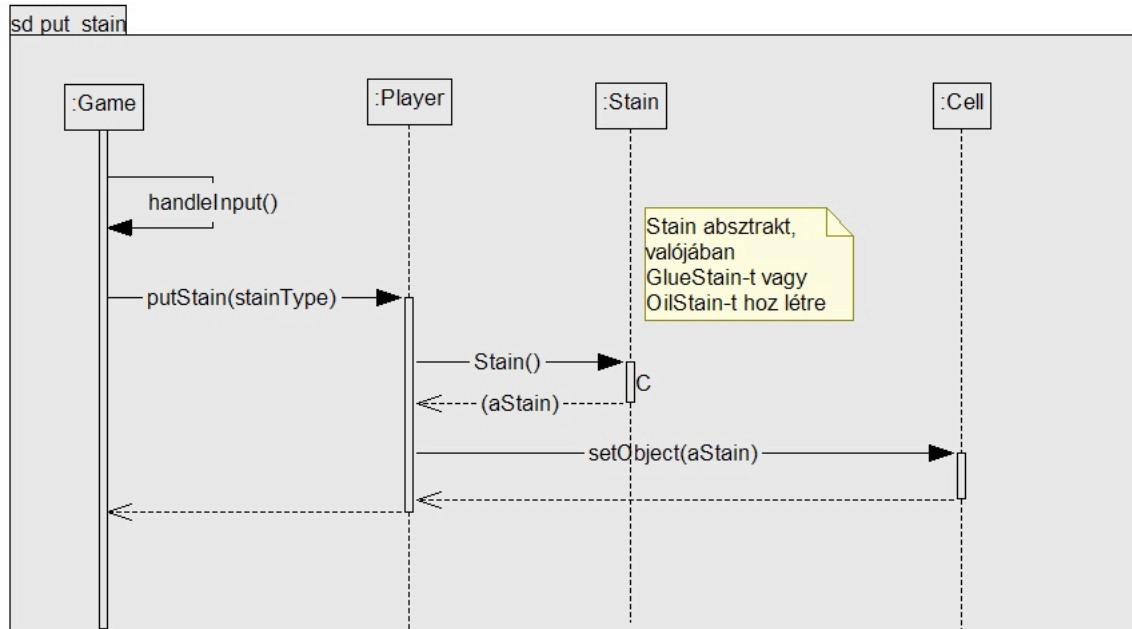
A robot mozgatását bemutató szekvenciadiagram



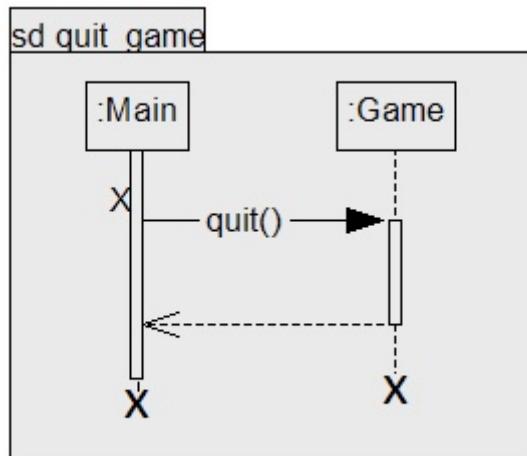
A játék kezdetét, a pálya létrehozását bemutató szekvenciadiagram.



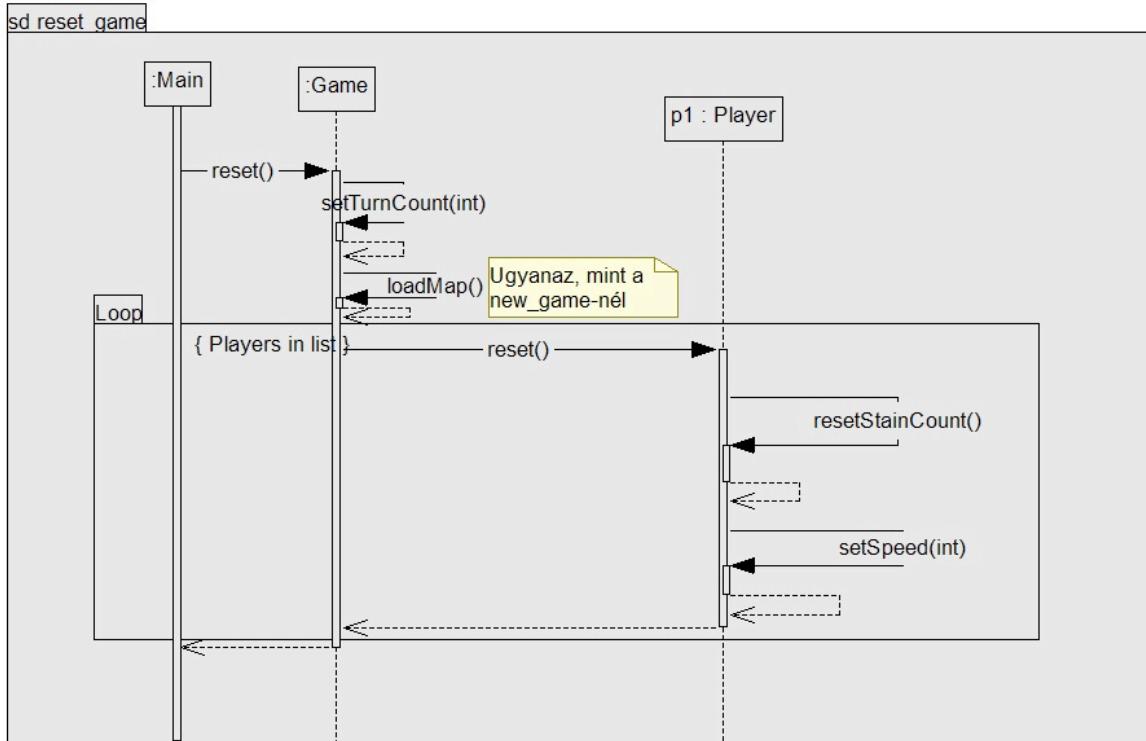
Egy folt letételét bemutató szekvenciadiagram



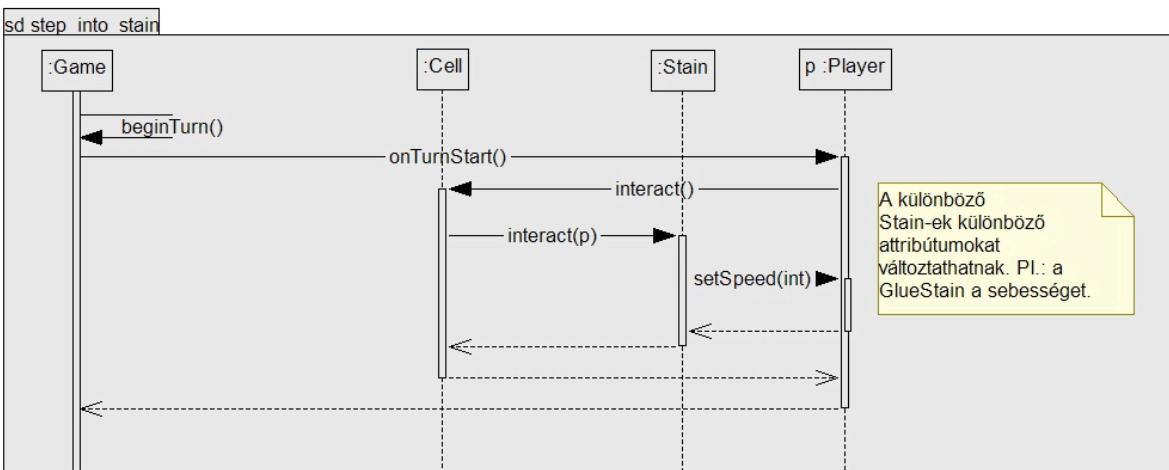
A programból való kilépést bemutató szekvenciadiagram



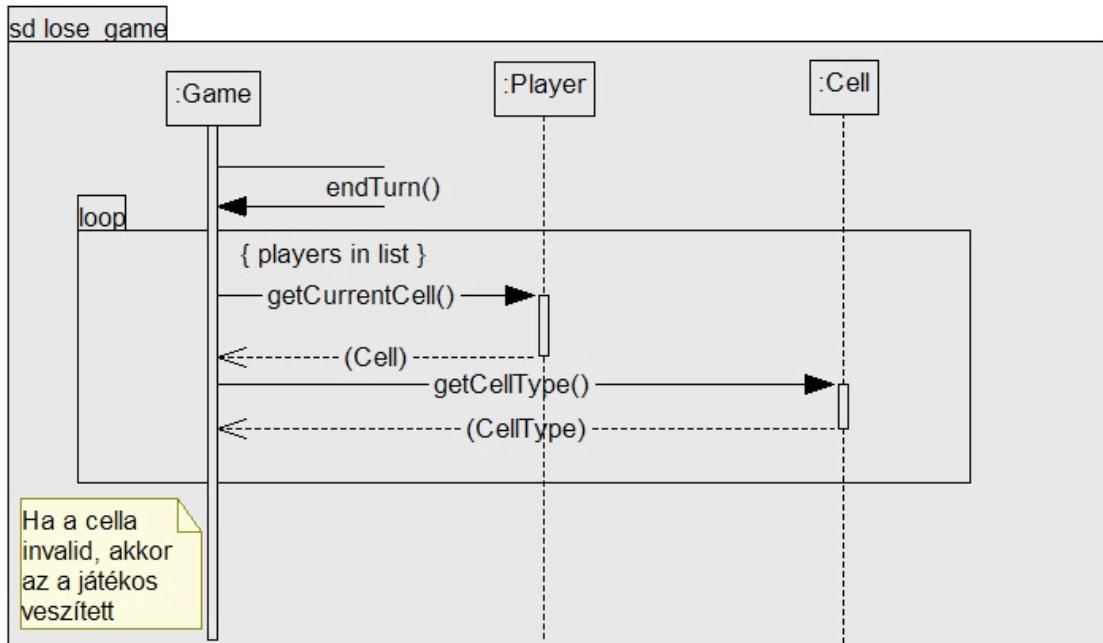
A játék újraindítását bemutató szekvenciadiagram



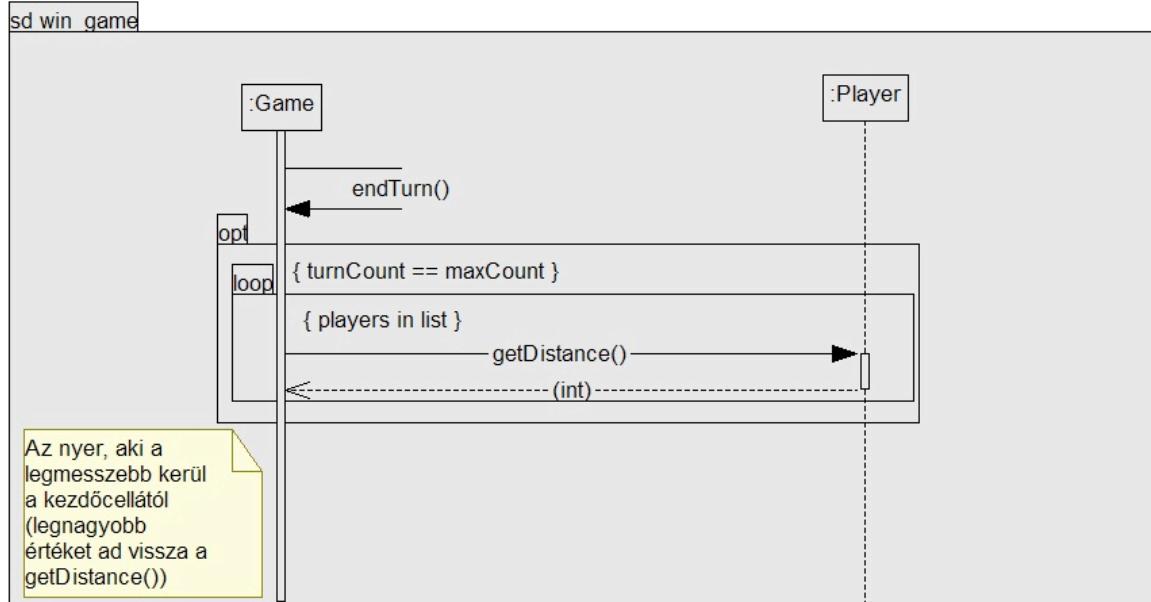
A foltba lépést bemutató szekvenciadiagram



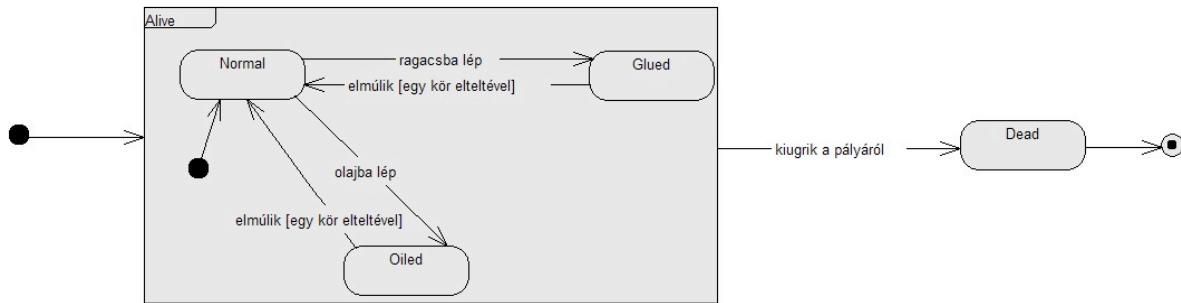
A játék elvesztését bemutató szekvenciadiagram



A játék megnyerését bemutató szekvenciadiagram



4.5 State-chartok



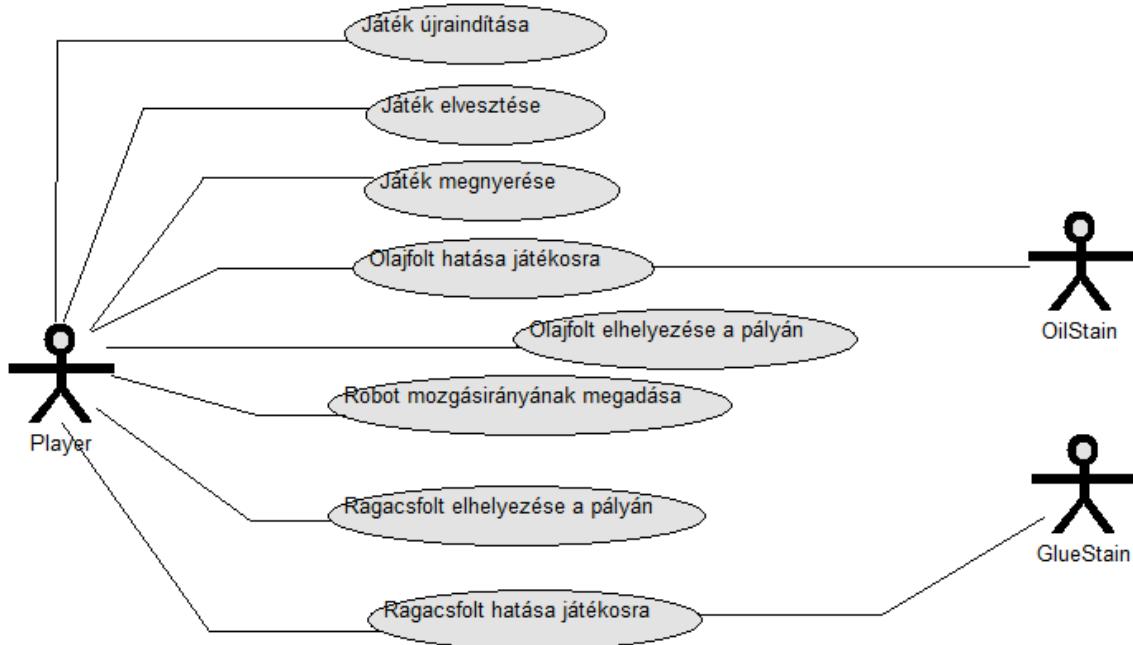
4.6 Napló

Kezdet	Időtartam	Résznevők	Leírás
2015.03.07. 11:00	1,5 óra	Graics Herold Sallai Verbőczy	Értekezlet. Előző dokumentációban lévő hibák elemzése. A modell egyes részeinek pontosítása. Feladatak kiosztása. Graics: osztálydiagram és szekvenciadiagramok kijavítása. Herold: Objektum-leírások pontosítása. Sallai: Objektum-leírások pontosítása. Verbőczy: Szekvenciadiagramok javítása.
2015.03.07 12:00	0,5 óra	Graics	Osztálydiagram elkészítése OpenAmeosban.
2015.03.07 12:00	0,5 óra	Verbőczy	Szekvenciadiagramokban pontatlanságok kigyűjtése, leírása.
2015.03.08 14:00	0,5 óra	Graics Herold Sallai Verbőczy	Értekezlet. Modell további módosítása. Ötletek az inkonzisztencia javítására.
2015.03.08 15:00	1 óra	Herold	Tevékenység: Objektum-leírások pontosítása, kiegészítése: Cell, Game, Map
2015.03.08 15:00	1 óra	Sallai	Tevékenység: Objektum-leírások pontosítása, kiegészítése: Player, GameObject, Stain, GlueStain, OilStain, EventSubscriber
2015.03.08 15:00	1,5 óra	Graics	Tevékenység: Szekvenciadiagramok kijavítása. Játék-nyerés, játék-vesztés szekvenciadiagramok elkészítése.
2015.03.08 15:00	1 óra	Verbőczy	Tevékenység: Szekvenciadiagramok kijavítása.

5. Szkeleton tervezése

5.1 A szkeleton modell valóságos use-case-ai

5.1.1 Use-case diagram



5.1.2 Use-case leírások

Use-case neve	Játék újraindítása
Rövid leírás	A felhasználó újraindítja a játékot.
Aktorok	Player
Forgatókönyv	A megfelelő parancs beütése esetén a játék visszakerül a kezdeti állapotába. A program minden alapértékre állít, tehát a játékosok visszakerülnek az <i>initialPosition</i> -ben tárolt cellára, <i>sebességük</i> és az <i>irány megadásának a lehetősége</i> (bool érték) is az alapértékre állítódik, a foltkészlet újboli feltöltése is megtörténik ekkor.

Use-case neve	Játék megnyerése
Rövid leírás	A játék megnyerése.
Aktorok	Player
Forgatókönyv	A játékos megnyeri a játékot, ha a körök letelte után nagyobb távolságot tett meg, mint a másik játékos, vagy a másik játékos veszített.

Use-case neve	Játék elvesztése
Rövid leírás	A játék véget ér az adott játékos számára.
Aktorok	Player
Forgatókönyv	A játékos elveszíti a játékot, ha invalid cellára ugrik. Ez azt jelenti, hogy tovább nem folytathatják a játékot, nem ugrálhatnak tovább, hiszen a másik játékos nyert.

Use-case neve	Olajfolt hatása játékosra
Rövid leírás	Olajfolt kifejtő hatását a játékosra.
Aktorok	Player, OilStain
Forgatókönyv	A játékos minden kör elején megkéri a cellát, amelyen tartózkodik, hogy lépjene vele interakcióba. A cella pedig megkéri a rajta lévő olajfoltot, hogy fejtse ki hatását a cellán tartózkodó játékosra. Az olajfolt letiltja a játékos lehetőségét arra, hogy irányt változtathasson a következő körben.

Use-case neve	Olajfolt elhelyezése a pályán
Rövid leírás	A felhasználó elhelyezi egy olajfoltot a pályán.
Aktorok	Player
Forgatókönyv	A megfelelő parancs hatására a játékos elhelyezhet egy olajfoltot a pályán, arra a cellára, amelyiken éppen tartózkodik.

Use-case neve	Ragacsfolt hatása játékosra
Rövid leírás	Ragacsfolt kifejtő hatását a játékosra.
Aktorok	Player, GlueStain
Forgatókönyv	A játékos minden kör elején megkéri a cellát, amelyen tartózkodik, hogy lépjene vele interakcióba. A cella pedig megkéri a rajta lévő ragacsfoltot, hogy fejtse ki hatását a cellán tartózkodó játékosra. A ragacsfolt megfelezi a játékos sebességét.

Use-case neve	Ragacsfolt elhelyezése a pályán
Rövid leírás	A felhasználó elhelyezi egy ragacsfoltot a pályán.
Aktorok	Player
Forgatókönyv	A megfelelő parancs hatására a játékos elhelyezhet egy ragacsfoltot a pályán, arra a cellára, amelyiken éppen tartózkodik.

5.2 A szkeleton kezelői felületének terve, dialógusok

A szkeleton a parancssorból lesz használható, menüvezérelt módon. A felhasználó adott parancsokkal tudja kezelni a programot. Az indítás után betöltődik a pálya; a program várja, hogy az első játékos megadja, hova akar lépni, és kíván-e lerakni foltot, és ha igen akkor milyet.

A menü felépítése:

- 1. Kör eleje**
 - 1.1. Milyen folton áll az első játékos? (O/G/N)
 - 1.2. Vesztett-e az első játékos? (Y/N)
 - 1.3. Milyen folton áll az második játékos? (O/G/N)
 - 1.4. Vesztett-e az második játékos? (Y/N)
 - 1.5. Lejártak a körök? (Y/N)
- 2. Első játékos lép**
 - 2.1. Hova lép? (W/NW/N/NE/E/SE/S/SW)
 - 2.2. Mit rak le? (O/G/N)
- 3. Második játékos lép**
 - 3.1. Hova lép? (W/NW/N/NE/E/SE/S/SW)
 - 3.2. Mit rak le? (O/G/N)
- 4. Kör vége**
 - 4.1. Újraindítja a játékot? (Y/N)

Dölten szedtük azokat a parancsokat, amik automatikusan elvégződnek. A kérdések végén megadtuk a lehetséges válaszokat. A rövidítésekhez jelmagyarázatot is készítettünk, amely alább látható.

Jelmagyarázat:

- (O/G/N) - (Oil/Glue/None)
- (Y/N) - (Yes/No)
- (W/NW/N/NE/E/SE/S/SW) -
(West/NorthWest/North/NorthEast/East/SouthEast/South/SoutWest)

5.3 Szekvencia diagramok a belső működésre

Megjegyzés: Mint már korábban említve lett, a pályát nem a játékos vezérlésével hozzuk létre, hanem előre létrehozott pályával dolgozunk. Ezen pályát egy *txt* formátumú fájlban tároljuk az alábbi módon:

‘0’ - invalid cella

‘1’ - valid cella, folt nélkül

‘2’ - valid cella, olajfolttal

‘3’ - valid cella, ragacsfolttal

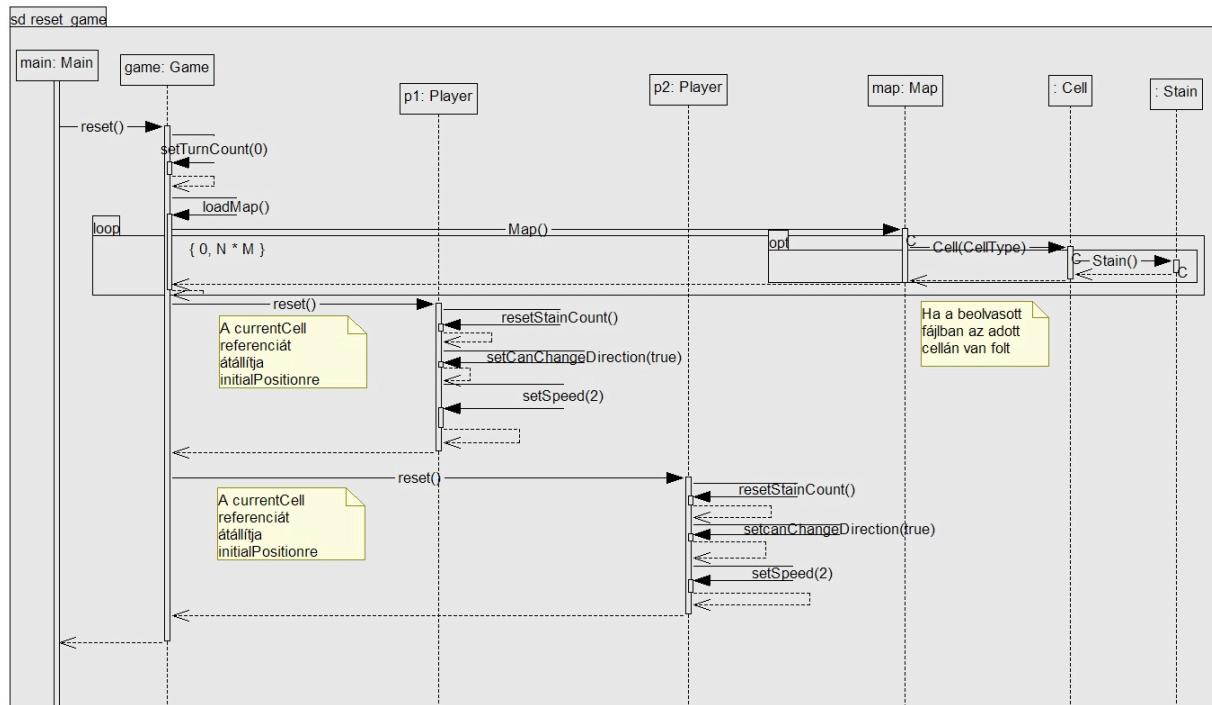
‘4’ - első játékos kezdeti pozíciója

‘5’ - második játékos kezdeti pozíciója

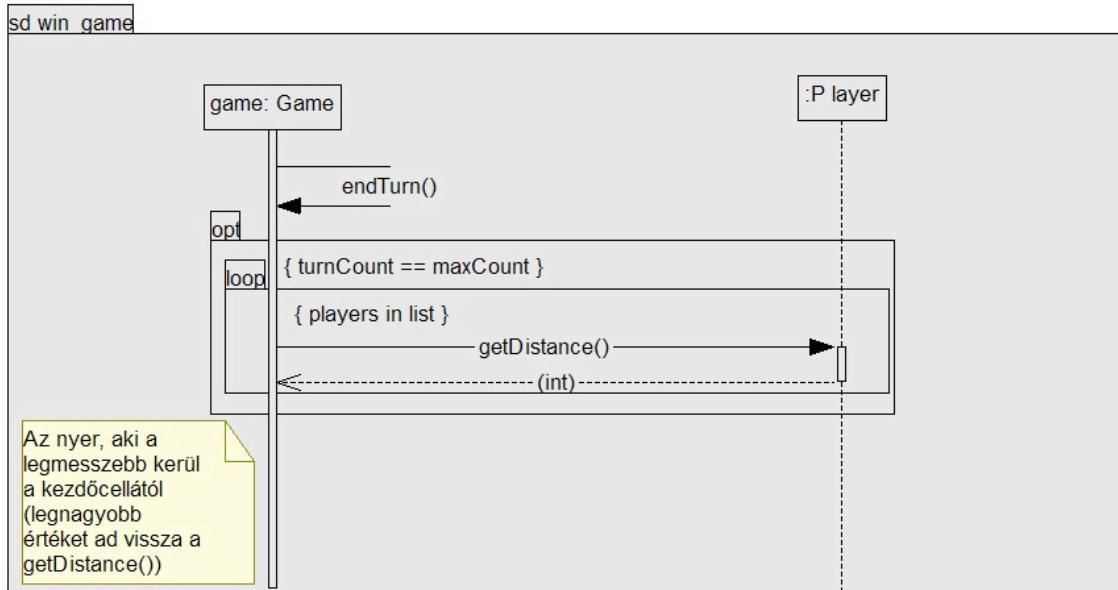
Például:

0	0	0	0	0	0	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	1	0	0	1	3	0
0	1	0	0	1	1	0
0	1	1	1	1	1	0
0	1	1	1	3	1	0
0	2	1	1	1	1	0
0	1	1	2	1	1	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	1	1	1	3	1	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	1	4	5	1	1	0
0	0	0	0	0	0	0

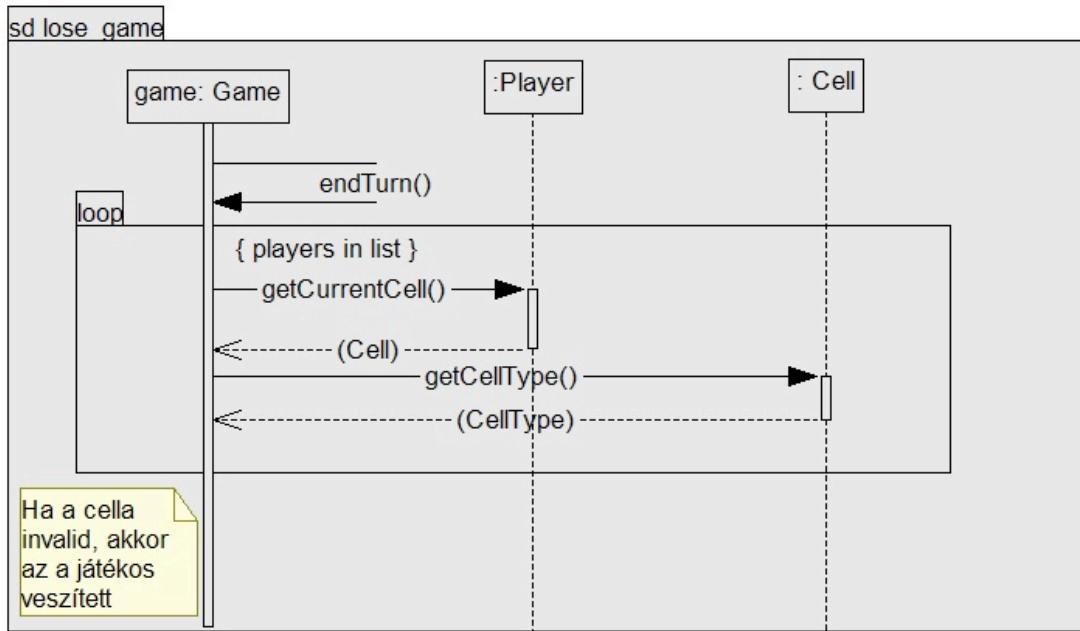
A játék újraindítását bemutató szekvencia-diagram.



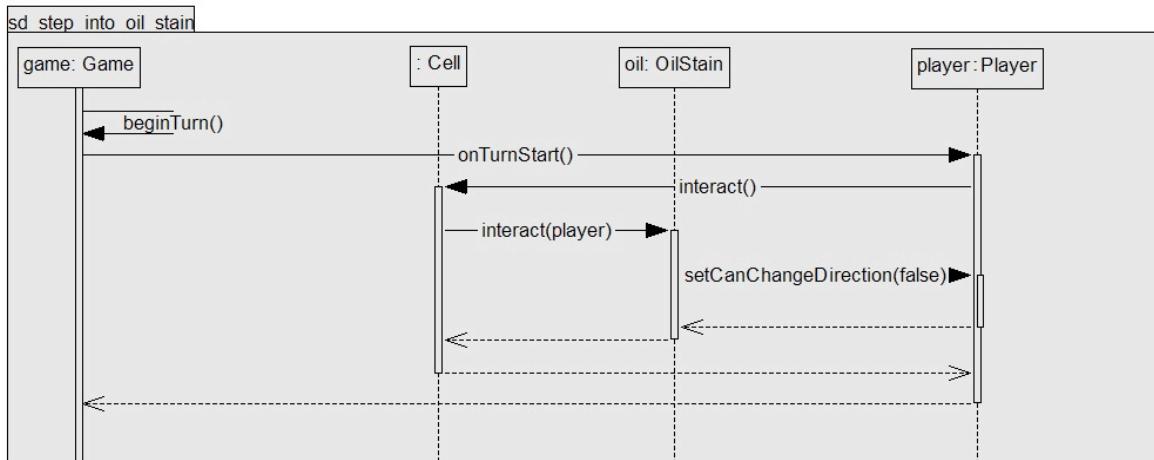
Játék megnyerését bemutató diagram.



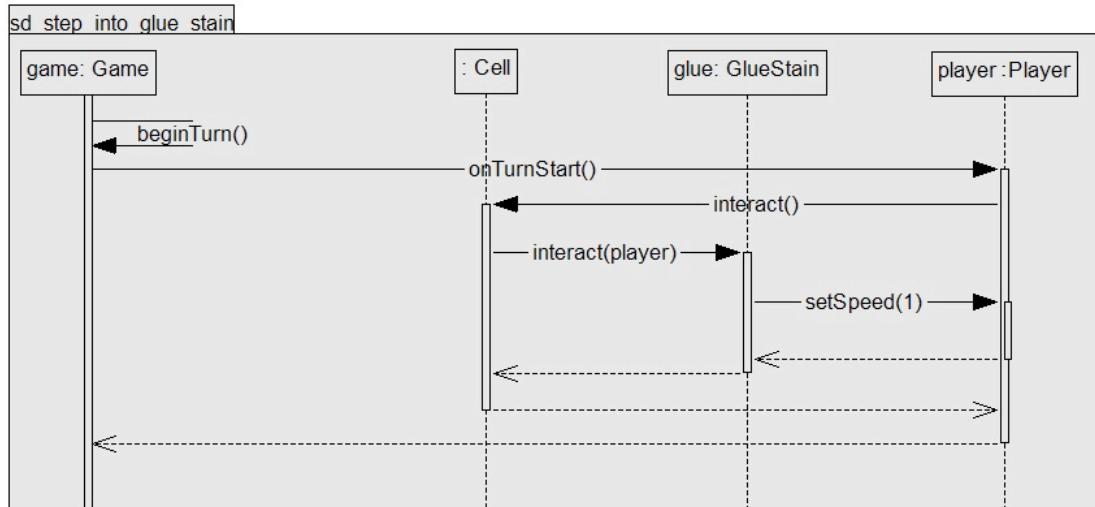
Játék elvesztését bemutató diagram.



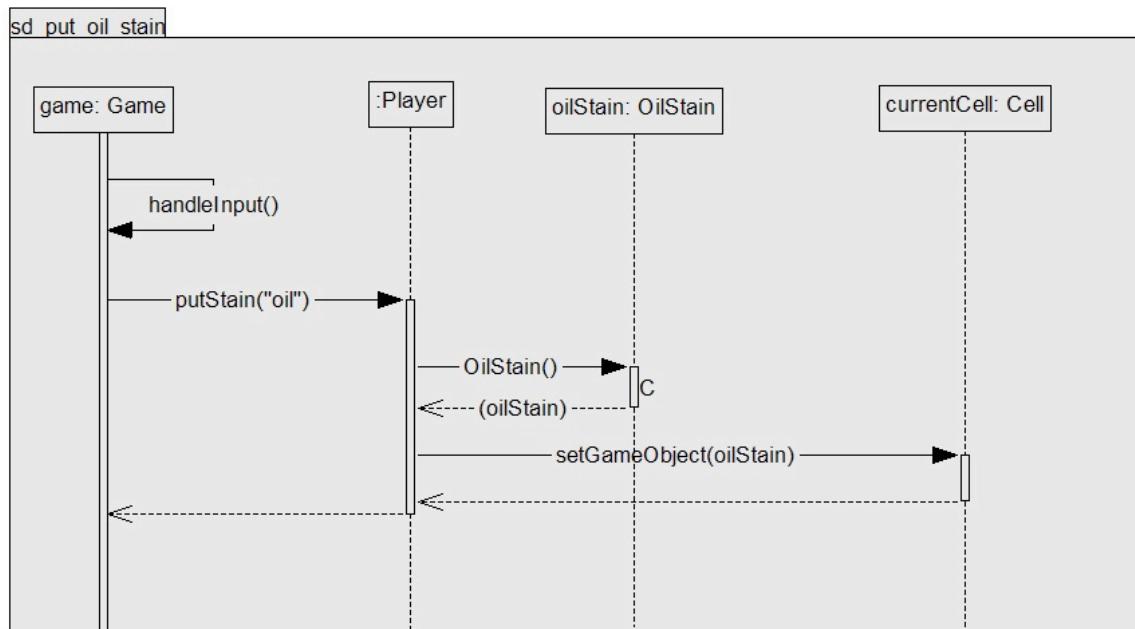
Olajfoltba lépést bemutató diagram, tehát az olajfolt hatása a játékosra.



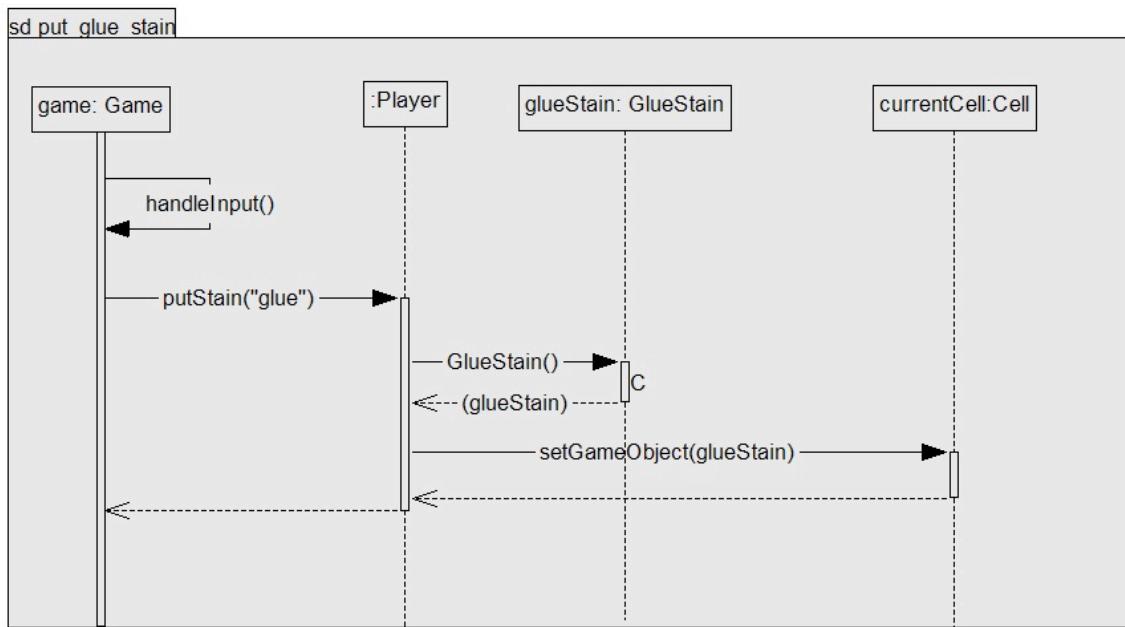
Ragacsfoltba lépést bemutató diagram, tehát a ragacsfolt hatása a játékosra.



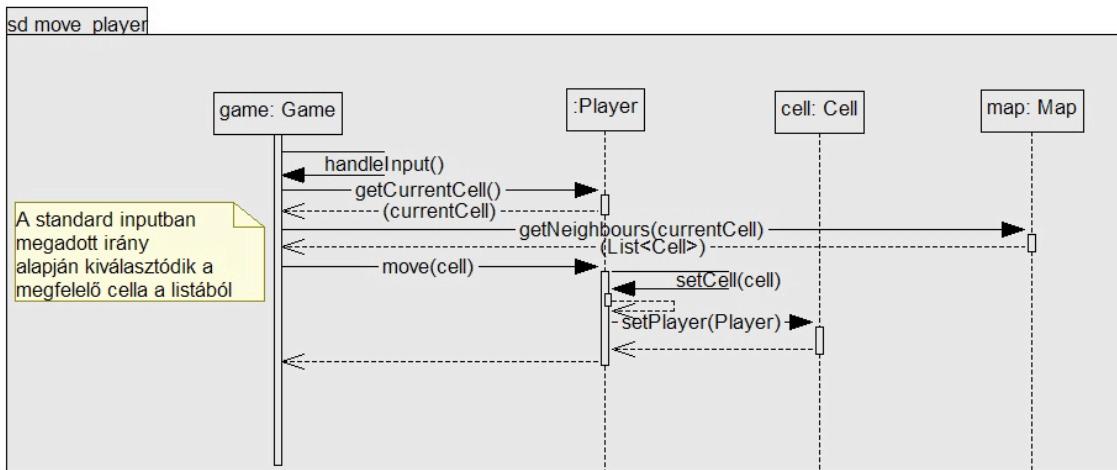
Olajfolt lerakását bemutató diagram.



Ragacsfolt elhelyezését bemutató diagram.

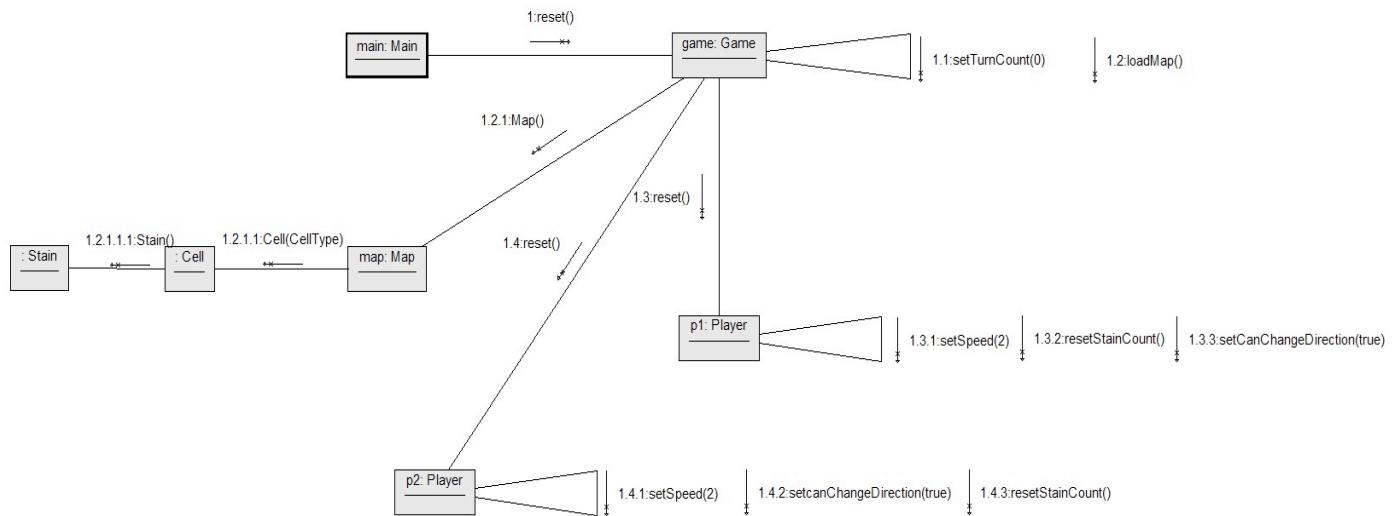


Mozgási irány megadását bemutató diagram.

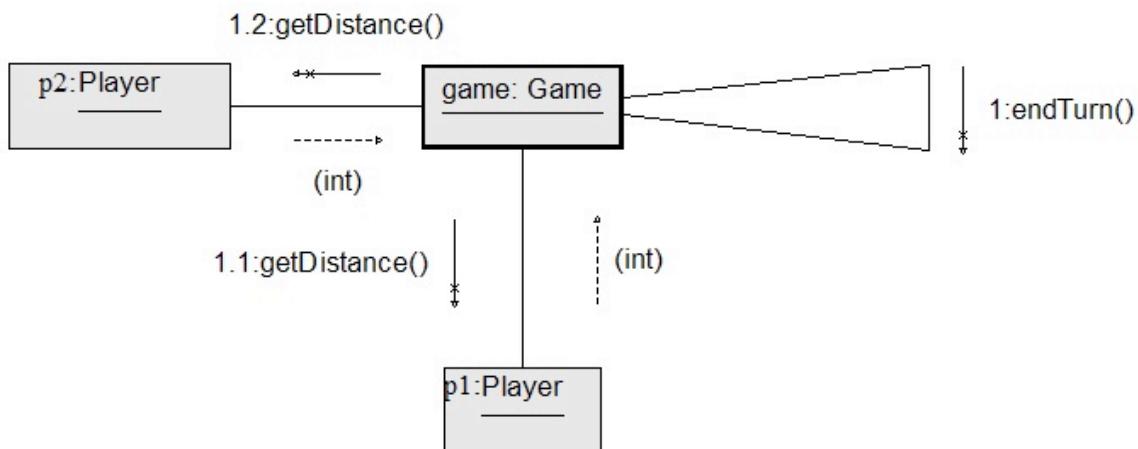


Kommunikációs diagramok

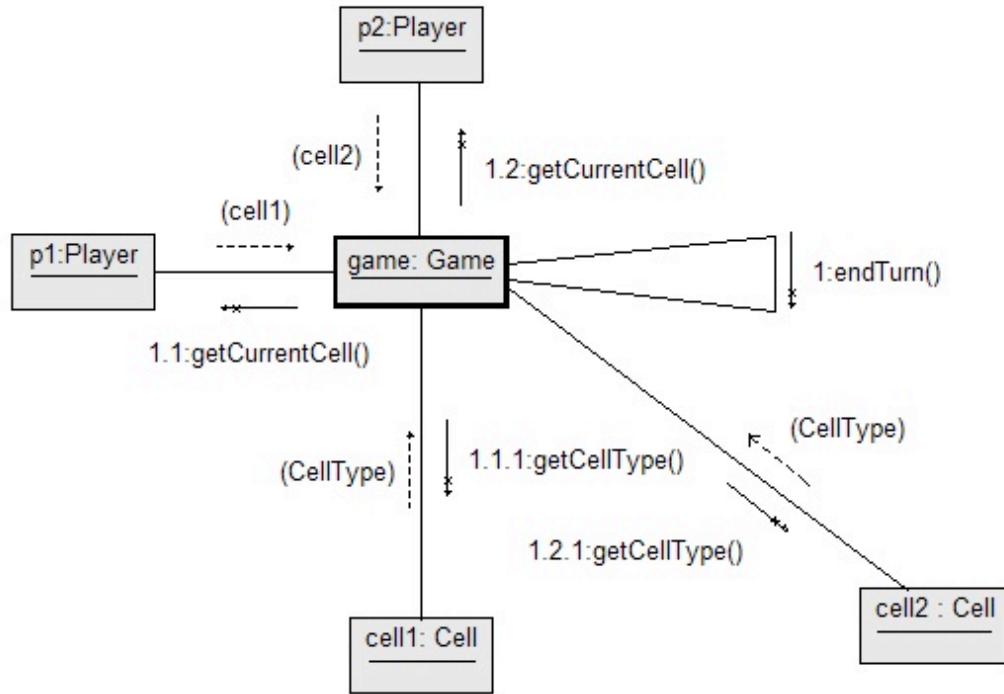
Játék újraindítását reprezentáló kommunikációs-diagram.



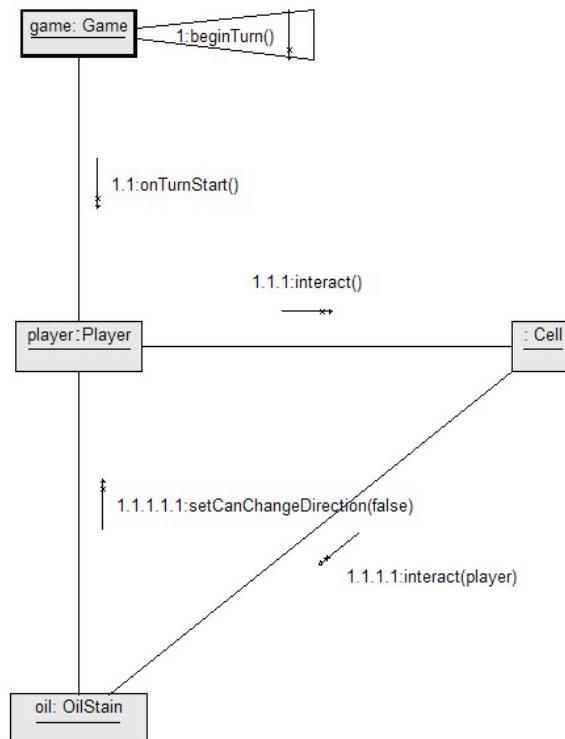
Játék megnyerését bemutató diagram.



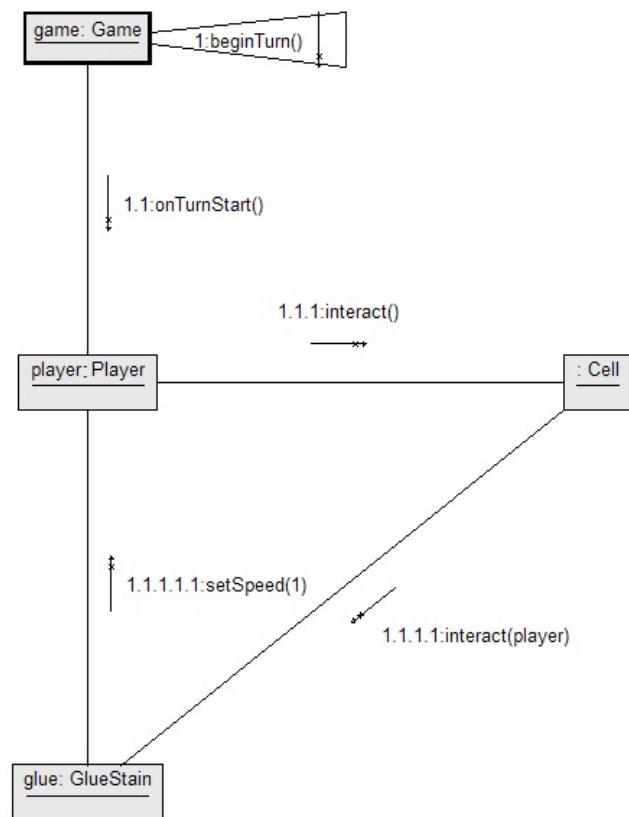
Játék elvesztését bemutató diagram.



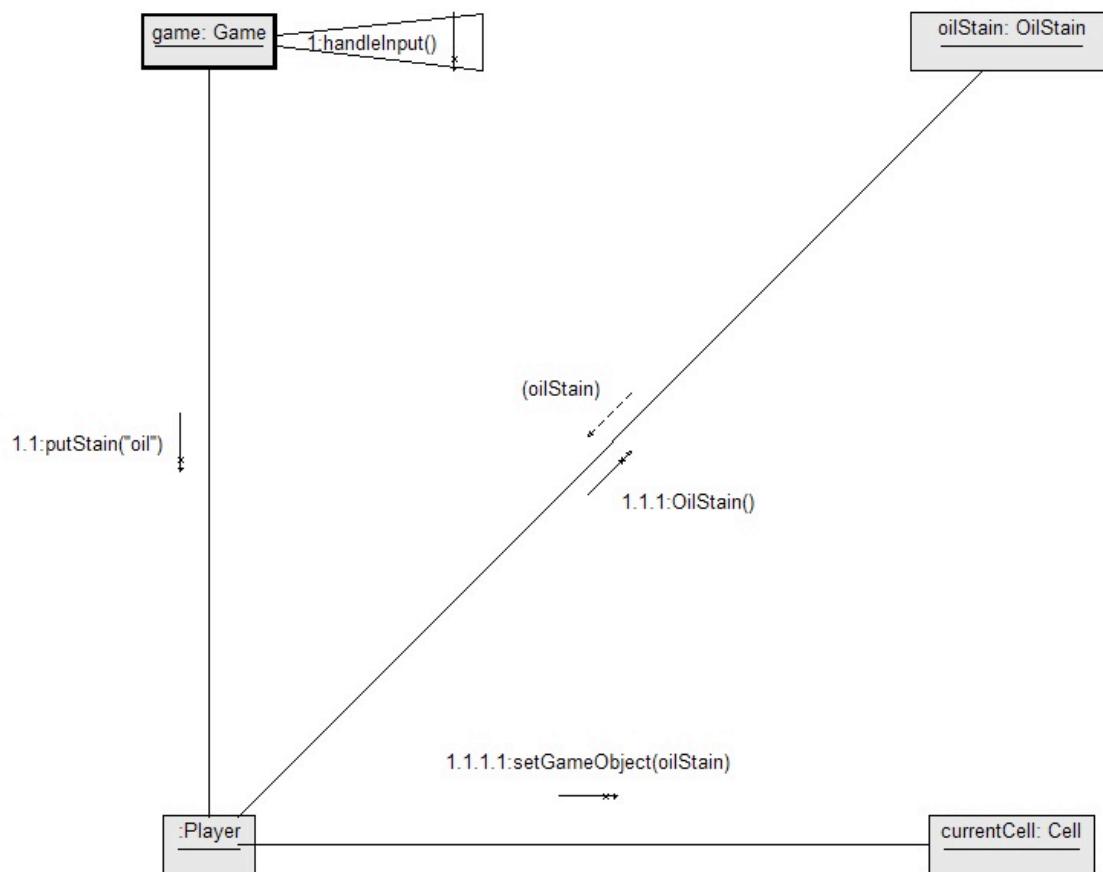
Olajfolt hatásának bemutatása a játékosra.



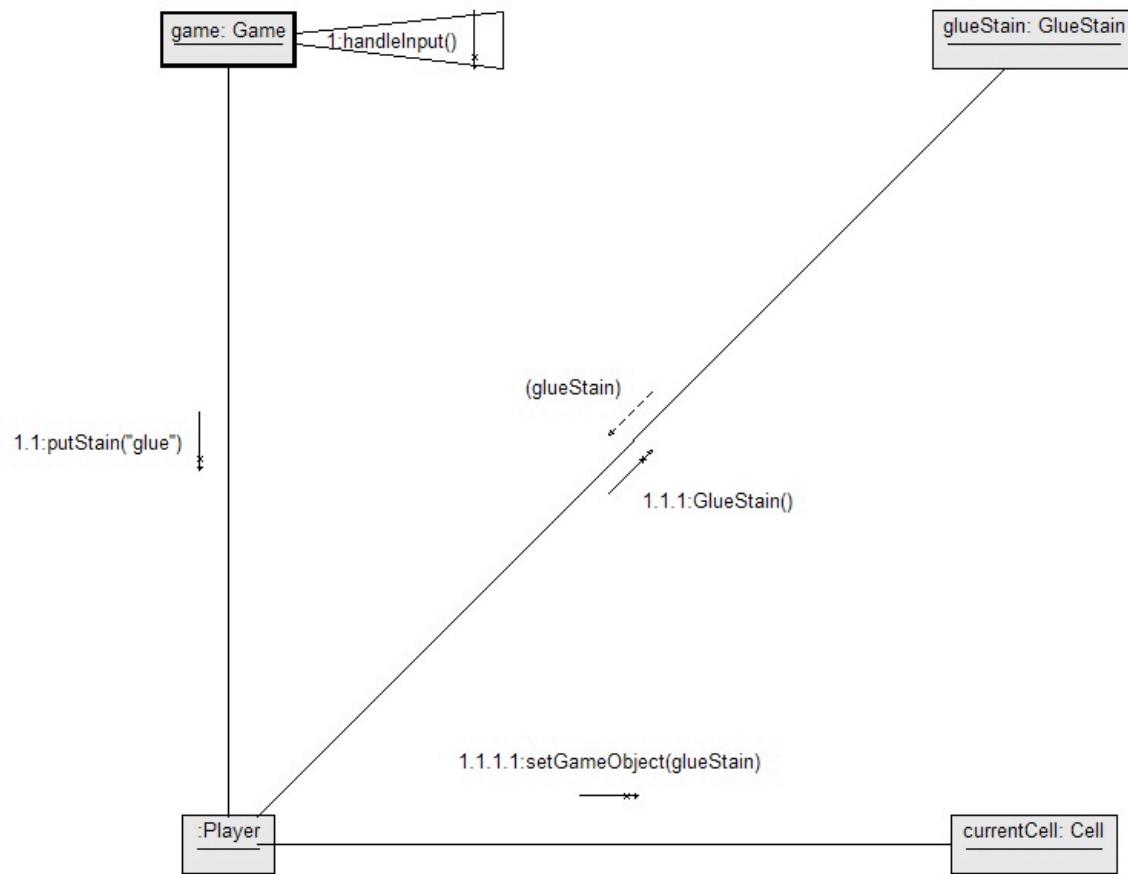
Ragacsfolt hatásának bemutatása a játékosra.



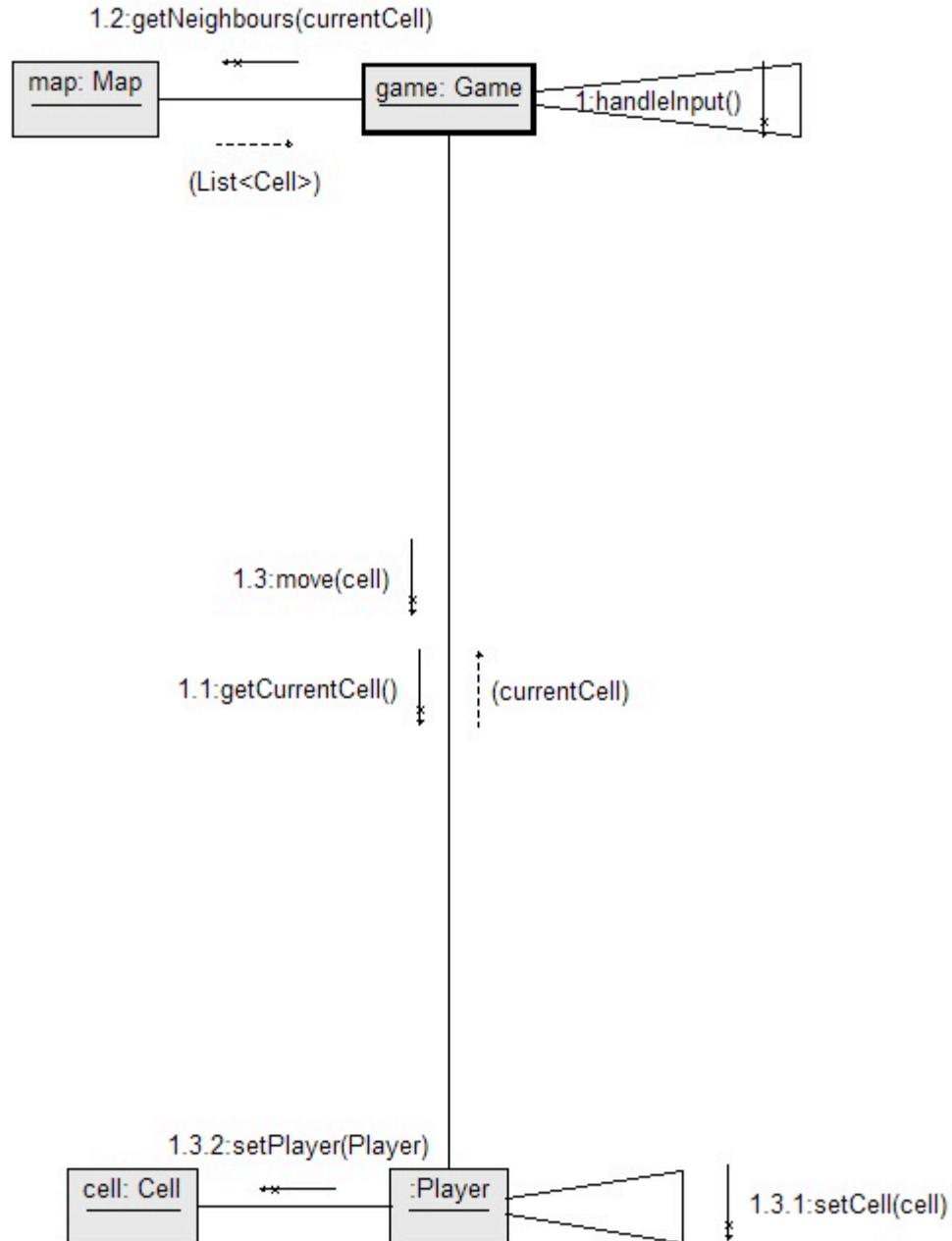
Olajfolt lerakását bemutató diagram.



Ragacsfolt lerakását bemutató diagram.



Mozgási irány megadását bemutató diagram.



5.4 Napló

Kezdet	Időtartam	Résztvevők	Leírás
2015.03.11. 18:00	0,5 óra	Graics Herold Sallai Verbőczy	Értekezlet. Valódi use-case-ek megvitatása. Problémák, nem egyértelmű részletek feljegyzése.
2015.03.12. 23:00	0,5 óra	Herold	Tevékenység: Szkeleton modell valóságos use-case diagramjának elkészítése.
2015.03.14. 18:50	1 óra	Herold	Tevékenység: Use-case leírások elkészítése.
2015.03.15. 21:05	1 óra	Sallai Verbőczy	A szkeleton kezelői felületének tervének elkészítése.
2015.03.15. 20:50	3,5 óra	Herold Graics	Szekvencia- és kommunikációs diagramok elkészítése.

6. Szkeleton beadás

6.1 Fordítási és futtatási útmutató

6.1.1 Fájllista

Fájl neve	Méret	Keletkezés ideje	Tartalom
Cell.java	2125 byte	2015.03.22. 23:40	Cella osztály.
CellType.java	69 byte	2015.03.20. 22:57	CellType enum.
Game.java	9876 byte	2015.03.23. 00:05	Game osztály.
GameObject.java	517 byte	2015.03.22. 23:40	GameObject interface.
GlueStain.java	410 byte	2015.03.22. 23:40	GlueStain osztály.
Logger.java	2662 byte	2015.03.20. 22:57	Logger osztály.
Main.java	193 byte	2015.03.22. 21:37	Main osztály.
Map.java	2658 byte	2015.03.22. 23:40	Map osztály.
OilStain.java	410 byte	2015.03.22. 23:40	OilStain osztály.
Player.java	6858 byte	2015.03.22. 23:40	Player osztály.
Stain.java	459 byte	2015.03.22. 23:40	Stain osztály.
map.txt	949 byte	2015.03.22. 21:12	A térkép fájl.

6.1.2 Fordítás

```
javac src/phoebe/*.java
```

6.1.3 Futtatás

```
cd src  
java phoebe.Main
```

6.2 Értékelés

Tag neve	Munka százalékban
Graics Bence	26,85%
Herold Kristóf	27,37%
Sallai Gyula	23,95%
Verbőczy Kristóf	21,83%

6.3 Napló

Kezdet	Időtartam	Résztvevők	Leírás
2015.03.21. 11:00	1 óra	Graics	Tevékenység: Graics implementálja és kommenteli a skeleton Map, Cell, és GameObject interfész megvalósító osztályokat.
2015.03.21. 16:10	2,5 óra	Sallai	Az osztálydiagram alapján a kód vázának létrehozása. Logger osztály implementálása.
2015.03.21. 22:00	1,5 óra	Verbőczy	Player osztály implementálása és kommentelése.
2015.03.21. 22:15	1,5 óra	Herold	Tevékenység: Game osztály metódusainak implementálása(kivéve start() és handleInput())
2015.03.22. 21:00	3 óra	Graics Herold Sallai Verbőczy	Az implementált osztályok egyberakása. Debuggolás. Fordítás. Dokumentáció elkészítése.

0. Változtatások

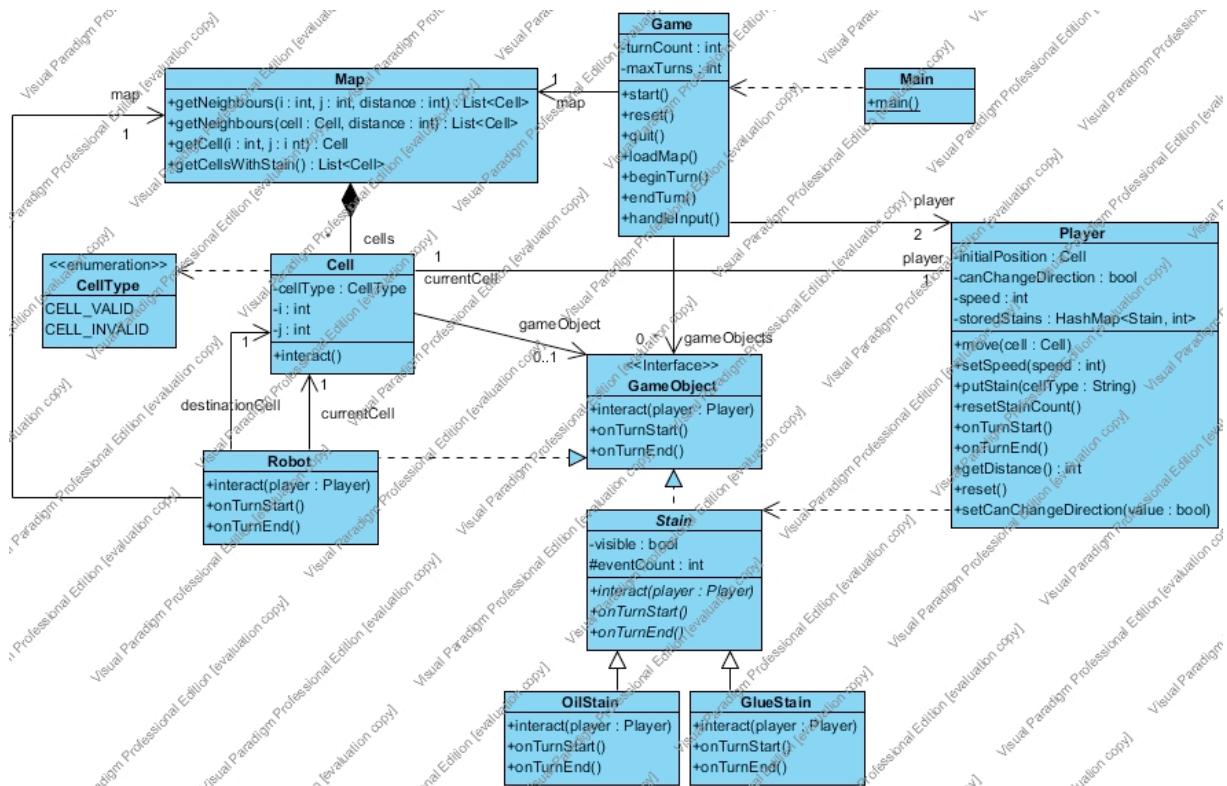
Változtatások a specifikációban:

- a ragacs eltűnik a pályáról, miután négy robot ráugrott (elkopik)
- egy meghatározott idő letelte után az olajfolt eltűnik a pályáról (felszárad)
- a pályára időnként keményen dolgozó kisrobotok jutnak be, akik szépen sorban feltakarítják a foltokat. A kisrobot egységnyi sebességgel halad, adott ideig (pl. két kör) takarítja a foltot. A folt feltakarítása után a legközelebbi folthoz indul. Ha egy robot ráugrik, akkor a kisrobot megsemmisül és olajfolt kerül a helyére; a ráugró robot nem szenved sérülést. Ha a kisrobot másik kisrobotnak vagy robotnak ütközik, irányt vált.
- a robotok képesek ütközni, ha azonos helyre érkeznek ugrásuk végén. Ilyenkor a gyorsabb robot összetöri (megsemmisíti) a lassabbat, és kettejük átlagsebességével halad tovább (vektorálag!).

Az új osztálydiagramot és szekvenciadiagramokat a Visual Paradigm nevű programmal készítettük el.

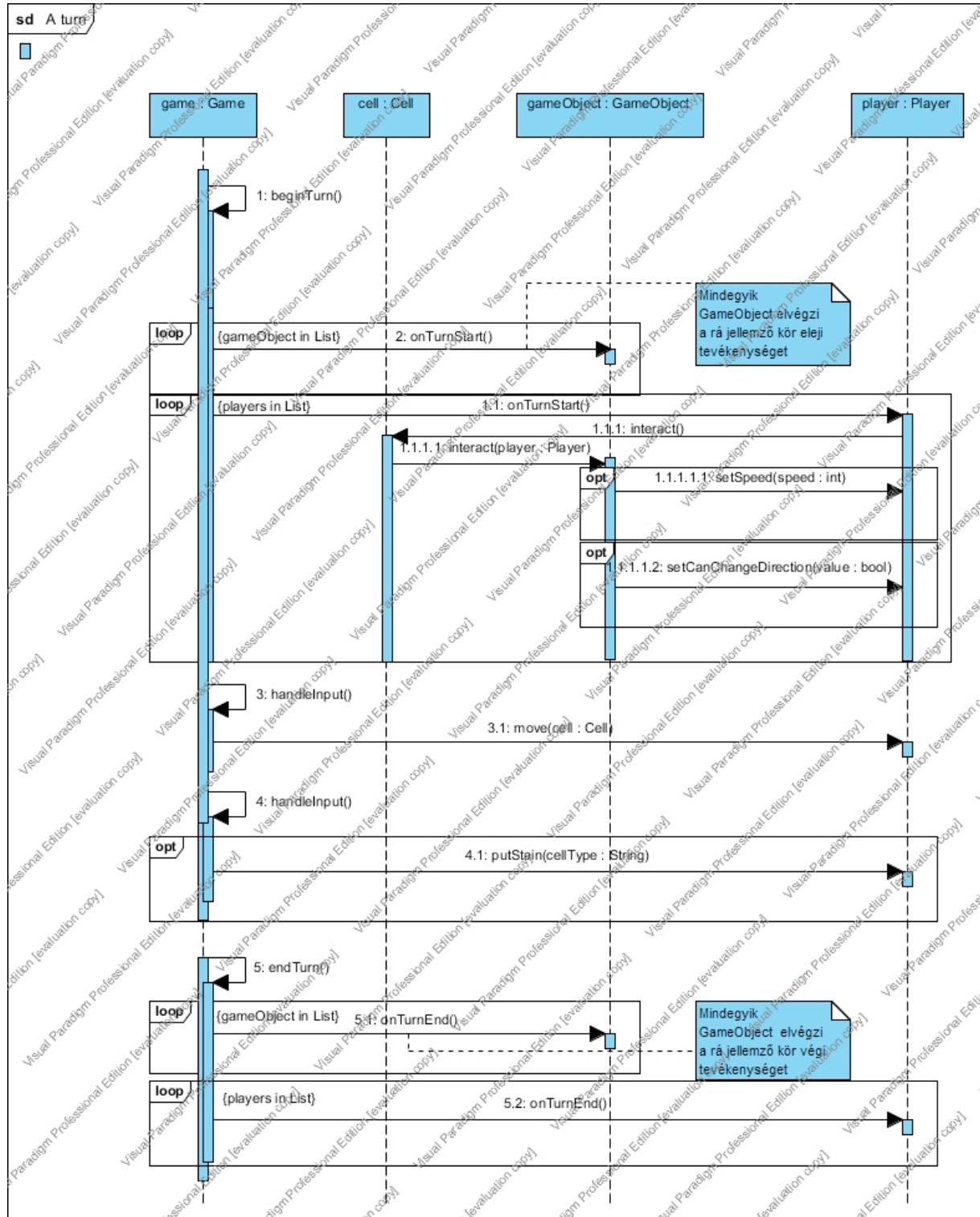
A megváltozott osztálydiagram

Bekerült egy új osztály, a Robot. Ő felelős a keményen dolgozó kisrobotok reprezentációjáért. Emellett a Game osztály asszociációba került a GameObject interféssel, hogy vezérelni tudja az összes viselkedéssel rendelkező játék-objektumot.

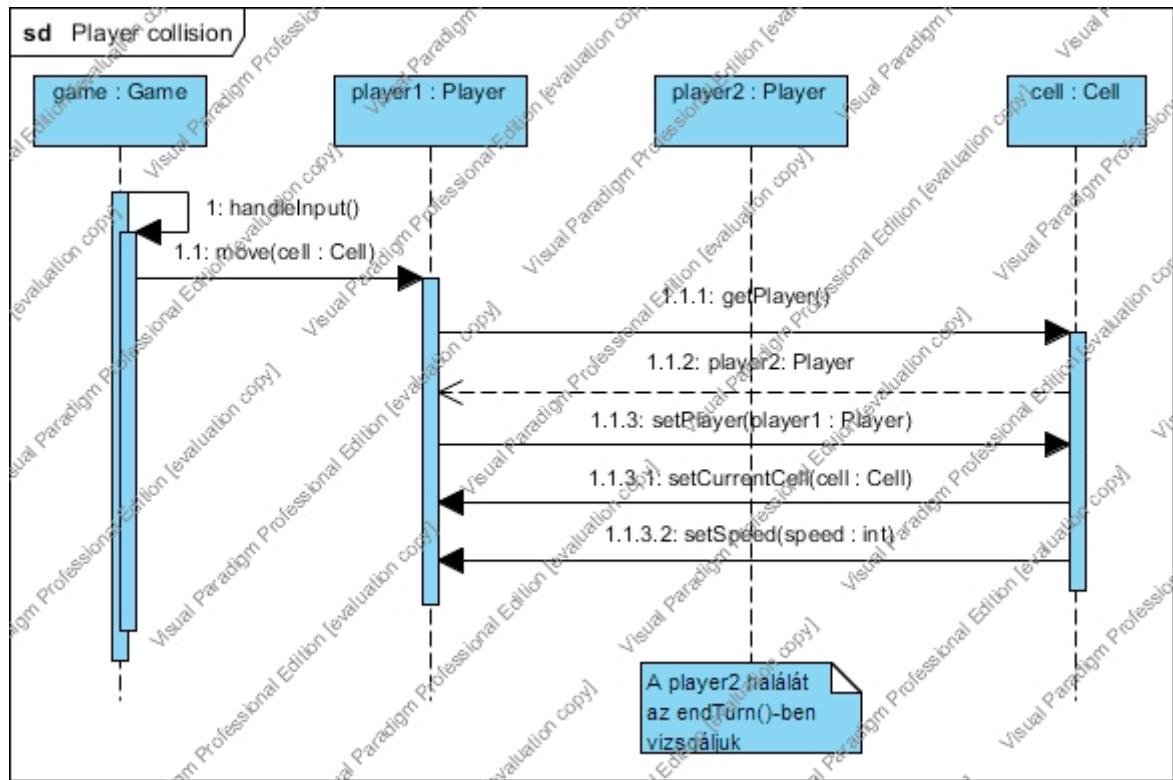


A megváltozott szekvenciadiagramok

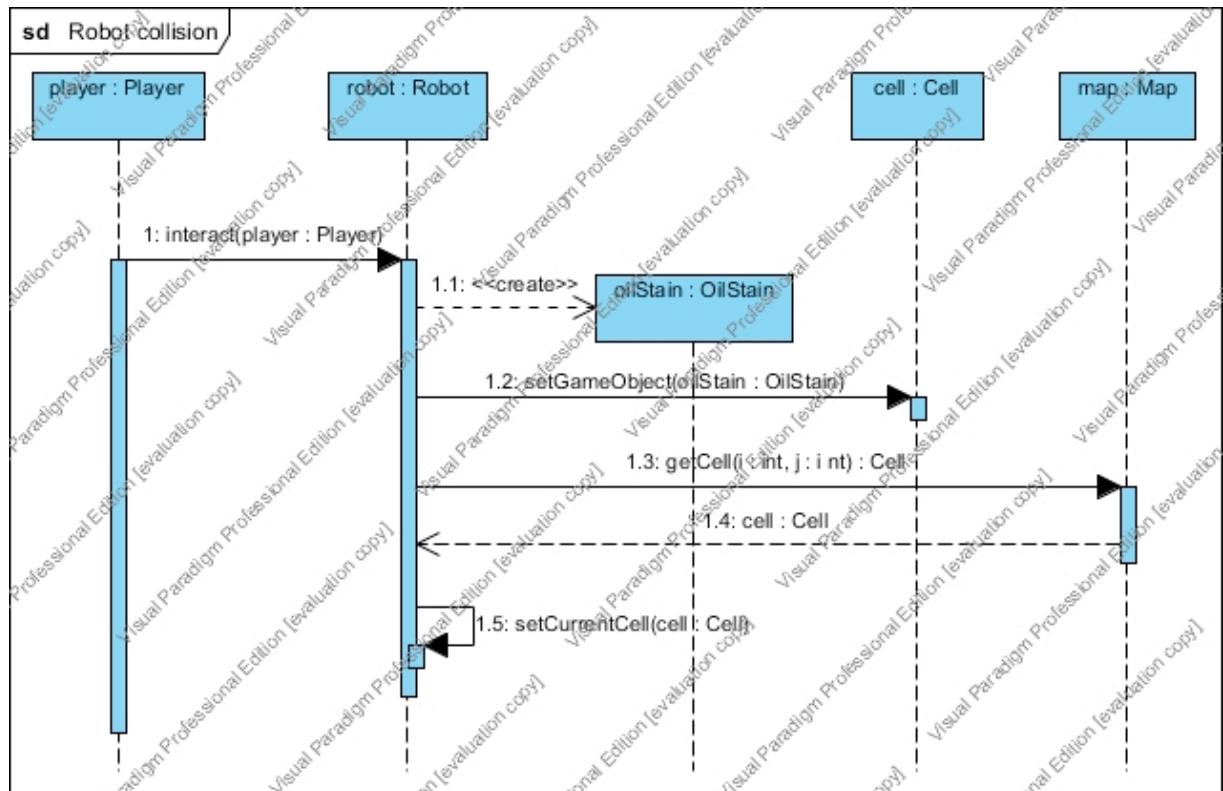
Az A turn szekvenciadiagramban a game objektum meghívja minden egyes gameObject onTurnStart és onTurnEnd metódusát az eddig is szereplő hívások mellett.



A Player collision szekvenciadiagram a játékosok robotjainak ütközését mutatja be.



A Robot collision szekvenciadiagram azt az esetet írja le, amikor egy játékos ráugrik a keményen dolgozó kisrobotra.



7. Prototípus koncepciója

7.1 Prototípus interface-definíciója

7.1.1 Az interfész általános leírása

Az általunk megvalósítandó interfész a szabványos bemenetről lesz képes parancsokat fogadni. Az esetleges kimeneteket szabványos kimenetre írja. A bemenetek fájlokban olvasását és a kimenetek fájlokba irányítását OS specifikus parancsokkal oldjuk meg. Az interfész parancssorból használható, de ezenfelül egy tesztelő segédprogramot is implementálunk (lásd.: 7.1.4-es pont).

7.1.2 Bemeneti nyelv

start <mapFileName>

Leírás: A játék elindítása a *mapFileName* fájlban található térképpel.

Opciók:

- *mapFileName*: A betöltendő pálya neve

reset

Leírás: A játék újraindítása.

quit

Leírás: Kilépés a játékból.

showmap

Leírás: A térkép kirajzolása.

move <direction>

Leírás: Mozgás egy adott irányba.

Opciók:

- *direction*: A választott irány (*N|NW|W|SW|S|SE|E|NE*)

put-stain <type>

Leírás: Egy olaj- vagy ragacsfolt lerakása.

Opciók:

- *type*: A folt típusa (O|G).

end-turn

Leírás: A kör vége jelzés. Hatására a másik játékos köre jön.

oil-random <setting>

Leírás: Az olajfolt véletlenszerű hatásának állítása.

Opciók:

- *setting*: A használandó beállítás (on|off)

hardworking-little-robot-random <setting>

Leírás: A takarító robot véletlenszerű hatásának beállítása.

Opciók:

- *setting*: A használandó beállítás (on|off)

robot-move <row> <col>

Leírás: A takarító robot mozgatása adott cellára. Csak akkor adható ki, ha kikapcsoltuk a robot véletlenszerűségét.

Opciók:

- *row*: Az új cella sor indexe (0-tól indexelünk).
- *col*: Az új cella oszlop indexe (0-tól indexelünk).

A pályát egy *txt* formátumú fájlban tároljuk az alábbi módon:

‘0’ - invalid cella

‘1’ - valid cella, folt nélkül

‘2’ - valid cella, olajfolttal

‘3’ - valid cella, ragacsfolttal

‘4’ - első játékos kezdeti pozíciója

‘5’ - második játékos kezdeti pozíciója

0	0	0	0	0	0	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	1	0	0	1	3	0
0	1	0	0	1	1	0
0	1	1	1	1	1	0
0	1	1	1	3	1	0
0	2	1	1	1	1	0
0	1	1	2	1	1	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	1	1	1	3	1	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	1	4	5	1	1	0
0	0	0	0	0	0	0

Például:

7.1.3 Kimeneti nyelv

Kimenetet az alábbi parancsok adnak:

move: Az irányváltás utáni pozíció

Player <sorszám>: <új pozíció>

showmap: A térkép jelenlegi állapota, az alábbi módon:

‘0’ - érvénytelen cella

‘1’ - érvényes cella

‘2’ - felfedezett ragacsfolt

‘3’ - felfedezett olajfolt

‘4’ - az első játékos jelenlegi pozíciója

‘5’ - a második játékos jelenlegi pozíciója

‘6’ - a takarító robot pozíciója

robot-move: Amennyiben rosszkor használtuk a parancsot, a felhasználó a következő hibaüzenetet kapja: “*Invalid command.*”

Továbbá még a következő események hatására jelenik meg kimenet:

Olajfoltra lépés: Amennyiben az olajfolt véletlenszerűsége be van kapcsolva, közöljük a játékos új pozíóját. Különben megjelenik a “Waiting for move command” üzenet, mely után a move parancsot kell kiadnia a felhasználónak.

A takarító robotra lépés: Amennyiben a takarító robot véletlenszerűsége be van kapcsolva, közöljük a robot új pozíóját. Különben megjelenik a “Waiting for robot-move command” üzenet

7.2 Összes részletes use-case

Use-case neve	A játék elindítása
Rövid leírás	A játék elindítása a megadott fájlban található térképpel
Aktorok	Felhasználó
Forgatókönyv	A felhasználó a megfelelő parancs segítségével elindítja a játékot, a parancs paramétereként megadott fájlból pedig betöljük a térképet.

Use-case neve	A játék újraindítása
Rövid leírás	A játék újraindítása, a kezdeti pozícióra való visszaállítással
Aktorok	Felhasználó
Forgatókönyv	A megfelelő parancs beütése esetén a játék visszakerül a kezdeti állapotába. A program minden alapértékre állít, tehát a játékosok visszakerülnek az <i>initialPosition</i> -ben tárolt cellára, <i>sebességük</i> és az <i>irány megadásának a lehetősége</i> (bool érték) is az alapértékre állítódik, a foltkészlet újboli feltöltése is megtörténik ekkor.

Use-case neve	Kilépés a játékból
Rövid leírás	A játék bezárása.
Aktorok	Felhasználó
Forgatókönyv	A felhasználó a megfelelő parancs segítségével bezárja a játékot, ekkor az alkalmazás kilép.

Use-case neve	A térkép kirajzolása
Rövid leírás	A térkép jelenlegi állapotának kirajzolása a szabványos kimenetre.
Aktorok	Felhasználó
Forgatókönyv	A felhasználó a megfelelő parancs segítségével kirajzolja a térképet, mely a szabványos kimeneten mátrixos formátumban jelenik meg, tehát sorokból és oszlopokból áll.

Use-case neve	Mozgás egy adott irányba
Rövid leírás	Az aktuális játékos mozgatása egy adott irányba. Amennyiben az új cellán olajfolt található, és ki van kapcsolva a véletlenszerűsége, akkor is ezt a parancsot kell kiadnia a felhasználónak.

Aktorok	Felhasználó
Forgatókönyv	A felhasználó a megfelelő parancs segítségével mozgatja a játékost, a parancs paramétereként megadott irány alapján. A felhasználó visszajelzést kap az új pozíóról.

Use-case neve	Folt lerakása
Rövid leírás	Egy olaj- vagy ragacsfolt elhelyezése a pályán.
Aktorok	Felhasználó
Forgatókönyv	A felhasználó a megfelelő parancs segítségével elhelyez egy foltot a jelenlegi pozíóján, a parancs paramétereként megadott típusú foltot (olaj vagy ragacs) teszi le.

Use-case neve	Kör vége jelzés
Rövid leírás	A kör vége jelzés hatására a másik játékos köre jön.
Aktorok	Felhasználó
Forgatókönyv	A felhasználó a megfelelő parancs segítségével jelzi a körének végét. Ezután a másik játékos mozoghat, illetve rakhatalmú foltot.

Use-case neve	Az olajfolt véletlen hatásának állítása
Rövid leírás	Az olajfolt véletlenszerű hatásának ki- illetve bekapcsolása.
Aktorok	Felhasználó
Forgatókönyv	A felhasználó a megfelelő parancs segítségével beállítja az olajfolt véletlen hatását a játékos, a parancs paramétereként megadott irány alapján. A felhasználó visszajelzést kap az új pozíóról.

Use-case neve	A takarító robot véletlen hatásának állítása
Rövid leírás	A takarító robot véletlenszerű hatásának ki- illetve bekapcsolása.
Aktorok	Felhasználó
Forgatókönyv	A felhasználó a megfelelő parancs segítségével beállítja a takarító robot véletlen hatását a játékos, a parancs paramétereként megadott irány alapján. A felhasználó visszajelzést kap az új pozíóról.

Use-case neve	A takarító robot mozgatása
Rövid leírás	A takarító robot mozgatása egy adott cellára. Csak kikapcsolt véletlenszerűség esetén használható.
Aktorok	Felhasználó
Forgatókönyv	A felhasználó a megfelelő parancs segítségével beállítja a takarító robot új pozícióját. Amennyiben a robot véletlenszerűsége ki van kapcsolva, hibaüzenetben értesítjük a felhasználót a tévedéséről.

7.3 Tesztelési terv

Teszt-eset neve	Idő lejár
Rövid leírás	Lejár az idő.
Teszt célja	A teszt azt az esetet vizsgálja, amikor azért van vége a játéknak, mert lejár az idő.

Teszt-eset neve	Mozgás
Rövid leírás	A játékos mozgatja a hozzá tartozó robotot.
Teszt célja	Teszteljük, hogy a robot a játékos által meghatározott cellára ugrik-e. (Az éppen aktuális robotnak kell odaugrania.)

Teszt-eset neve	Nagy robot ugrik kisrobotra
Rövid leírás	A játékos által irányított robot ráugrik a takarító robotra.
Teszt célja	Ezt ellenőrizzük, hogy ha a nagy robot ugrik rá a kis robotra, akkor a kis robot megsemmisül, a helyén olajfolt lesz. Továbbá egy új kis robot szabadul be a pályára.

Teszt-eset neve	Nagy robotok ütköznek
Rövid leírás	A két játékos által irányított robot közül az egyik arra a cellára ugrik, amelyen a másik játékos által irányított robot tartózkodik.
Teszt célja	Azt nézzük meg, hogy az ugró robot sebessége megváltozik-e, illetve megsemmisül-e a másik robot. (Ez egyben a játék végét is jelenti.)

Teszt-eset neve	Érvénytelen cellára ugrás
Rövid leírás	Az egyik játékos invalid cellára ugrik az általa irányított robottal.
Teszt célja	Azt teszteli, hogy ebben az esetben a játék véget ér-e. A kiugró a veszes, a másik pedig a győztes.

Teszt-eset neve	Olajfoltos cellára ugrás
Rövid leírás	Az egyik játékos olyan cellára ugrik, amelyen olajfolt található.
Teszt célja	Azt teszteli, hogy az adott <i>Player</i> lehetősége az irányváltoztatásra megszűnik-e. (<i>Player.canChangeDirection == false</i> ?)

Teszt-eset neve	Ragacsfoltos cellára ugrás
Rövid leírás	Az egyik játékos olyan cellára ugrik, amelyen ragacsfolt található.
Teszt célja	Azt teszteli, hogy az adott <i>Player</i> sebessége megfeleződik-e. (<i>Player.speed == 1 ?</i>)

Teszt-eset neve	Olajfolt lerakása
Rövid leírás	Az adott játékos olajfoltot rak le az adott cellára.
Teszt célja	Azt teszteljük, hogy miután a játékos a robotját arra utasította, hogy olajfoltot rakjon a cellára, az tényleg megjelent-e.

Teszt-eset neve	Ragacsfolt lerakása
Rövid leírás	Az adott játékos ragacsfoltot rak le az adott cellára.
Teszt célja	Azt teszteljük, hogy miután a játékos a robotját arra utasította, hogy ragacsfoltot rakjon a cellára, az tényleg megjelent-e.

Teszt-eset neve	Olajfolt felszáradása
Rövid leírás	Adott számú kör elteltével az olajfolt felszárad.
Teszt célja	Megvizsgáljuk, hogy adott számú kör elteltével, a celláról eltűnik-e az olajfolt.

Teszt-eset neve	Ragacsfolt elhasználódása
Rövid leírás	A ragacsfolt eltűnik, miután négyszer beleugrottak.
Teszt célja	Azt teszteljük, hogyha egy ragacsfoltba négyszer beleugrottak, akkor az eltűnik-e a celláról.

Teszt-eset neve	Kisrobot takarítása
Rövid leírás	A kisrobot eltakarítja a cellán lévő bármely foltot adott kör alatt.
Teszt célja	A robot egy olyan cellára érkezik, amelyen folt található. Megvizsgáljuk, hogy adott kör elteltével a cella üres(azaz nincs rajta folt) lesz-e.

Teszt-eset neve	Kisrobot célpont keresése
Rövid leírás	A kisrobot miután elvégezte a dolgát, új cellát választ.
Teszt célja	Megvizsgáljuk, hogy olyan cellát választ-e a kisrobot, amelyen folt található.

Teszt-eset neve	Kisrobot nagyrobotnak ütközik
Rövid leírás	A kisrobot ütközik egy nagyrobottal, és a kisrobot irányt

	vált.
Teszt célja	A kisrobot olyan cellára ugrik, amelyen nagyrobot tartózkodik. Azt teszteljük, hogy a kisrobot ekkor megváltoztatja-e az irányát.

7.4 Tesztelést támogató segéd- és fordítóprogramok specifikálása

A tesztelés támogatásához olyan segédprogramot készítünk, amely teszt bemeneteket és a hozzájuk tartozó várt kimeneteket tárolja. A segédprogram lefuttatja a megadott tesztet, és szöveges alapon összehasonlítja a kimenetet az elvárt kimenettel. Ha nem egyezik meg a kimenet az elvárt kimenettel, akkor nem sikerült a teszt, amennyiben viszont megegyezik, sikeresnek tekintjük a tesztet.

7.5 Napló

Kezdet	Időtartam	Résznevők	Leírás
2015.03.27. 16:00	0,5 óra	Graics Herold Sallai Verbőczy	Értekezlet: Módosítások átbeszélése. Előzetes tervezetek.
2015.03.27. 16:00	0,5 óra	Graics	Tevékenység: Graics elkezdi implementálni a változtatásra szoruló diagramokat.
2015.03.28. 18:30	1 óra	Graics Herold Sallai	Értekezlet: Módosítások pontosítása. Bemeneti, kimeneti nyelv formájáról döntés született.
2015.03.28. 18:30	1,75 óra	Graics	Tevékenység: Graics befejezi az osztálydiagramot, és elkészíti az új szekvenciadiagramo kat.
2015.03.29. 17:00	0,5 óra	Graics Herold Sallai Verbőczy	Értekezlet. Dokumentáció átbeszélése.
2015.03.29. 17:30	1 óra	Graics Verbőczy	Tevékenység: A tesztelési terv elkeszítése.
2015.03.29 17:30	1 óra	Herold Sallai	Tevékenység: Bemeneti, kimeneti nyelv, és use case leírások elkészítése.

0. Változtatások

0.1 Változtatások az osztálydiagramban:

- A Robot osztálynál bekerült egy int típusú id attribútum, amely az egyes kisrobotokat azonosítja
- A Stain osztály asszociációba került a Cell osztállyal, hogy egy folt adott esemény bekövetkezése után képes legyen eltűnni a celláról.

0.2 Változtatások a tesztesetekben:

“Kisrobot célpont keresése” teszteset bekerült a “Kisrobot takarítása” tesztesetbe, mivel úgy ítéltük meg, hogy ez a két funkció szorosan összefügg, és nincs értelme kettéválasztani.

0.3 Változtatások a teszt nyelvében:

Keményen dolgozó kisrobot mozgató parancsa: robot-move <ID><row><col>

- ID-ben megadható hogy melyik kisrobotot akarjuk mozgatni
- Kimenet: Hardworking-little-robot <ID>: New Position: Cell(<row>, <col>)

Cella állapotváltozásának kimenete:

- Cell(<row>, <col>): GameObject: <gameObject>

Játékos állapotának kimenete:

- Player <ID>: Cell(<row>, <col>) Speed: <spd> CanChangeDirection: <true/false>

Eltelt idő manipulálására bemeneti parancs: set-clock <T>

- T: eltelt körök száma

Játék kimenetelének a kimenete:

- Player 0: Distance - <distance>; Player 1: Distance - <distance>; Winner: <outcome>

8. Részletes tervezet

8.1 Osztályok és metódusok tervezet.

8.1.1 Cell

- **Felelősség**

A játéktér egy celláját megtestesítő osztály. Egy cellán tartózkodhat egy játékos vagy egy kisrobot, továbbá egy folt is. Amennyiben egy játékos olyan cellára ugrik, amelyen egy másik játékos tartózkodik, akkor az veszít, amelyiknek kisebb a sebessége (azonos sebesség esetén az győz aki ugrott). Ha olyan cellára ugrik, amelyen kisrobot található, akkor a kisrobot megsemmisül, és olajfolt lesz a helyén. Ha kisrobot ugrik olyan cellára, amelyen az egyik játékos, vagy egy másik kisrobot tartózkodik, akkor a kisrobot (amelyik ugrott) megváltoztatja irányát. Amennyiben egy játékos a cellára lép, a cellán tartózkodó folt interakcióba lép a játékossal, tehát kifejtíti hatását a játékosra a következő kör elején.

- **Attribútumok**

- **cellType:** A cella típusa, lehet érvényes (VALID), vagy érvénytelen (INVALID). Amennyiben egy játékos érvénytelen cellára lép, azonnal elveszíti a játékot. Érvénytelen cellának számítanak a szakadékok, illetve a pálya szélén túl elhelyezkedő cellák. Érvényes cellának számít minden más egyéb cella, amely nem érvénytelen. Láthatóság: - . Típus: CellType
- **gameObject:** A cellán elhelyezkedő objektumot reprezentáló attribútum, ez lehet folt, vagy kisrobot. Láthatóság: -. Típus: GameObject.
- **player:** Az aktuális cellán álló játékos. Láthatóság: -. Típus: Player.
- **i, j:** Az adott cellának a koordinátái a mátrixban(a Map osztályban). Láthatóság: - . Típus: int.

- **Metódusok**

- **void interact():** Ha a játékos az adott cellára lépett, ezen metódus hívódik meg. A metódus pedig a cellán elhelyezkedő gameObjectnek hívja meg a megfelelő metódusát(*interact(player: Player)*), ahol player paraméter az adott Cell objektum player attribútuma), amely a paraméterként átvett Player objektumra kifejtíti a hatását. Ha a cellán nincsen folt, természetesen nem történik semmi. Láthatóság: +.

8.1.2 Game

- **Felelősség**

A rendszer központi osztálya. Nyilvántartja a játékosokat, a térképet, és a további játékelemeket. Ő felelős a játék vezérléséért (játék indítás, újraindítás, kilépés). Továbbá

felelős a játék mechanikájáért, tehát vezérli a körök lejátszását (kör indítása, befejezése, input kezelése).

- **Attribútumok**

- **turnCount:** Az éppen aktuális kör száma.
Láthatóság: -. Típus: int.
- **maxTurns:** A maximális körök száma, egy konstans érték.
Láthatóság: -. Típus: int.
- **map:** A játéktér tárolására.
Láthatóság: -. Típus: Map.
- **gameObject:** A játéktéren elhelyezkedő objektumok(foltok, kisrobotok) listája.
Láthatóság: -. Típus: List<GameObject>.
- **players:** A játékosok listája.
Láthatóság: -. Típus: List<Player>.

- **Metódusok**

- **void start():** Jelzi a játék indulását, hatására felépül a pálya, a játékosok létrejönnek, és megkezdődik az első kör. A játék ebben az állapotban marad újraindításig és kilépésig. Láthatóság: +.
- **void start map.txt():** Újraindítja az aktuális játékot. minden játékos visszakerül a kezdőpozícióba, foltkészleteik feltöltődnek. A turnCount nulla értéket kap.
Láthatóság: +.
- **void quit():** Hatására befejeződik a játék, és a program kilép (*System.exit()*).
Láthatóság: +.
- **void beginTurn():** Jelzi a kör elejét az összes feliratkozott játékosnak, illetve GameObjectnek. Tehát meghívja a játékosok, és az összes GameObject *onTurnStart()* metódusát. Ez annyit jelent, hogy végigiterálunk minden objektumra.
Láthatóság: +.
- **void endTurn():** Jelzi a kör végét az összes feliratkozott játékosnak, illetve GameObjectnek. Tehát meghívja a játékosok, és az összes GameObject *onTurnEnd()* metódusát, a *beginTurn()* metódusban leírtaknak hasonlóan.
Láthatóság: +.
- **void handleInput():** Kezeli a kapott bemenetet. A megfelelő parancsok hatására a megfelelő szekvenciális események hajtódnak majd végre.
Láthatóság: +.

8.1.3 Map

- **Felelősség**

A játékteret megtestesítő osztály. A játéktér celláit egy kétdimenziós tömbben(mátrix) tárolja, illetve innen kérhetők le egy-egy cella szomszédai, és azok a cellák, amelyeken van folt.

- **Attribútumok**
- **cells:** Egy kétdimenziós adatszerkezet(mátrix), amely a játék celláit tartalmazza.
Láthatóság: -. Típus: Cell[][].
- **Metódusok**
- **Cell getCell(int i, int j):** Visszatér a paraméterben megadott indexek által meghatározott cellával .
Láthatóság: +.

- **List<Cell> getNeighbours(Cell cell, int distance):** Paraméterként átvesz egy cellát és visszatér egy cellákat tartalmazó listával, amely azokat a cellákat tartalmazza, amelyekre el lehet jutni a paraméterben átvett celláról. Átvesz egy egész számot, amely azt határozza meg, hogy milyen messzire lehet eljutni az adott celláról. Láthatóság: +.
- **List<Cell> getNeighbours(int i, int j, int distance):** Működése hasonló a fentihez, csak cella helyett koordinátákat(mátrix indexet) vesz át. Láthatóság: +.
- **List<Cell> getCellsWithStain():** Visszaadja azokat a cellákat, amelyeken folt található. Ezen metódus azért fontos, hogy a kisrobotok le tudják kérni, hogy merre találhatóak a foltok, amelyeket fel kell majd takarítaniuk. Láthatóság: +.

8.1.4 GameObject

- **Felelősség**

Egy interfész, azon osztályok valósítják meg, amelyek nem játékosok, és interakcióba léphetnek egy játékossal. Ezek a foltok, és a kisrobotok.

- **Metódusok**
- **void interact(Player player):** Alkalmazza az adott hatást a player játékosra. Láthatóság: +.
- **void onTurnStart():** Kör elején történő események megvalósításához. Láthatóság: +.
- **void onTurnEnd():** Kör végén történő események megvalósításához. Láthatóság: +.

8.1.5 Player

- **Felelősség**

Játékosokat megtestesítő osztály. minden körben az éppen aktuális játékosot tudjuk a bemenetek segítségével manipulálni, illetve rá jellemző információkat lekérdezni. Például, hogy melyik cellán tartózkodik, vagy hogy mekkora a sebessége.

- **Attribútumok**
- **currentCell:** Az a cella, ahol a játékos éppen áll. Láthatóság: -. Típus: Cell.
- **initialPosition:** Egy cella objektum, amelyen a játékos a játék elindításakor áll. A pálya betöltésekor állítjuk be a játékosra. Láthatóság: -. Típus: Cell.
- **storedStains:** Egy kulcs-érték alapú adatszerkezet, amelyben nyilvántartjuk, hogy egy adott típusú foltból hány darab áll a játékos rendelkezésére. Láthatóság: -. Típus: HashMap<String, Integer>.
- **speed:** A játékos sebessége. Esetünkben a játékos sebességét úgy definiáljuk, hogy egy körben mekkora távolságban elhelyezkedő cellákra tud eljutni. Tehát ha a sebessége 1, akkor a közvetlen mellette lévő 8 cellára juthat el, ha viszont 2, akkor a tőle 2 cellára elhelyezkedő 8 cellára juthat el. Láthatóság: -. Típus: int.
- **canChangeDirection:** Jelzi, hogy egy játékos képes-e irányt változtatni (ez alapértelmezésben igaz, de pl. olajfoltra lépés esetén nem). Láthatóság: -. Típus: bool.

- **Metódusok**

- **void move(Cell cell)**: Átlépteti a játékost a paraméterként átvett cellára. Tehát átállítja a referenciát(currentCell).
Láthatóság: +.
- **void putStain(String stainType)**: Elhelyez egy foltot az aktuális cellán, ha a foltkészlete megengedi. Mindig lehet le foltot, ha valid cellán tartózkodik, hiszen ha egy cellán elhelyezkedő foltra rálép, akkor az megszűnik. (Természetesen a cella nem lehet invalid.)
Láthatóság: +.
- **void onTurnStart()**: Reagál a kör eleje eseményre. Meghívja annak a cellának az *interact()* metódusát, amelyiken tartózkodik.
Láthatóság: +.
- **void onTurnEnd()**: Reagál a kör vége eseményre. A *player* megfelelő attribútumait alaphelyzetbe állítja (például: sebesség).
Láthatóság: +.
- **void start map.txtStainCount()**: A *player* foltkészletének alapértelmezettre állítása.
Láthatóság: +.
- **void start map.txt()**: A játékos egyes attribútumainak alapértékre állítása a játék újraindítása esetén. Itt állítjuk vissza a sebességét, illetve hogy tud-e irányt váltani. Valamint a játékost visszarakjuk az initialCell-ben tárolt cellára.
Láthatóság: +.
- **int getDistance()**: A currentCell és initialPosition közti táv. Játék megnyerése esetén a játékosok megtett távját hasonlítjuk össze. A távot manhattan-távolságként definiáltuk.
Láthatóság: +.
- **void setSpeed(int speed)**: Ez a metódus a paraméterében kapott értéket állítja be az adott játékos sebességének.
Láthatóság: +.
- **setCanChangeDirection(bool value)**: A canChangeDirction értékét állíthatjuk át egy bool érték segítségével ennek a metódusnak köszönhetően.
Láthatóság: +.

8.1.6 Stain

- **Felelősség**

Foltokat reprezentáló absztrakt osztály. OilStain és GlueStain ősosztálya. Definiálja a különböző foltok azonos tulajdonságait.

- **Interfészek**

GameObject

- **Attribútumok**
- **visible**: A folt láthatóságát adja meg.
Láthatóság: -. Típus: bool.
- **eventCount**: A foltknál számon kell tartanunk, hogy egyes események hányszor történtek meg eddig, erre a célra ezen attribútum felel meg.
Láthatóság: #. Típus: int.
- **Metódusok**
- **void interact(Player player)**: Absztrakt metódus, amelyet a leszármazott osztályok felüldefiniálnak. Láthatóság: +.
- **void onTurnStart()**: Absztrakt metódus, amelyet a leszármazott osztályok felüldefiniálnak. Láthatóság: +.

- **void onTurnEnd():** Abszrakt metódus, amelyet a leszármazott osztályok felüldefiniálnak. Láthatóság: +.

8.1.7 GlueStain

- **Felelősség**

A ragacsfoltokat megtestesítő osztály. Erre lépve a játékos sebessége megfeleződik.

- **Ősosztályok**

Stain

- **Interfészek**

GameObject

- **Attribútumok**
- **eventCount:** Ragacs folt esetén ezen attribútumban azt tároljuk, hogy hányszor léptek interakcióba a játékosok az adott ragacs folttal, adott számú interakció után a ragacs folt eltűnik a pályáról.
Láthatóság: #. Típus: int.

- **Metódusok**

- **interact(Player player):** Megfelezi a *player* játékos sebességét. Tehát meghívja a *player setSpeed(..)* metódusát, amellyel beállítja az attribútumot a megfelelő értékre. Továbbá eventCount attribútumát megnöveli eggyel. Miután elérte az eventCount a kritikus értéköt (4-et), a cellán, amelyen tartózkodott, átállítja a gameObject referenciáját nullra. (?)
Láthatóság: +.

- **void onTurnStart():** Ragacsfoltnak kör elején nincsen dolga.

Láhatóság: +.

- **void onTurnEnd():** Ragacsfoltnak kör elején nincsen dolga.

Láhatóság: +.

8.1.8 OilStain

- **Felelősség**

Az olajfoltokat megtestesítő osztály. Erre lépve a játékos az adott körben nem képes irányítani a mozgását.

- **Ősosztályok**

Stain

- **Interfészek**

GameObject

- **Attribútumok**
- **eventCount:** Olaj folt esetén ezen attribútumban azt tároljuk, hogy hány kör óta van az olaj folt a pályán. Hiszen az olaj folt meghatározott idő letelte után felszárad a pályáról.
Láthatóság: #. Típus: int.

- **Metódusok**

- **interact(Player player):** Elhelyezi a *player* játékest egy olyan véletlenszerű cellán, mely elérhető a jelenlegi pozíciójából, továbbá letiltja a játékos mozgását egy körre.
Láhatóság: +.

- **void onTurnStart():** Növeljük eventCount attribútumát. Láhatóság: +.

- **void onTurnEnd():** Ha az eventCount elérte kritikus értékét (adott számú kör), megszüntetjük az olaj foltot.

Láhatóság: +.

8.1.9 Robot

- **Felelősség**

A kisrobotokat megtestesítő osztály. A kisrobotok egyik foltról (miután azt feltakarították) mennek a hozzájuk legközelebb található folthoz.

- **Interfészek**

GameObject

- **Attribútumok**
- **currentCell:** A keményen dolgozó kisrobot aktuális pozícióját eltároló Cell objektum.
Láthatóság: -. Típus: Cell.
- **destinationCell:** A keményen dolgozó kisrobot cél celláját eltároló Cell objektum.
Láthatóság: -. Típus: Cell.
- **map:** A kisrobot ismeri az egész pályát, azért kell, hogy letudja kérni, hogy merre találhatóak a foltok a pályán.
Láthatóság: -, Típus: Map.
- **Metódusok**
- **interact(Player player):** Amikor a játékos ugyanarra a cellára ugrik, mint amin a kisrobot tartózkodik, interakcióba lépnek egymással, és ezen metódus meghívódik. Az adott cellára (ahol történt az interakció) egy olaj folt kerül, valamint a kisrobotot áthelyezzük egy másik cellára, amelyet véletlenszerűen választunk.
Láthatóság: +.
- **void onTurnStart():** Ha a keményen dolgozó kisrobot végzett egy folt feltakarításával, vagy éppen nincsen cél cella kijelölve neki, akkor a pályától elkeríti a foltokat tároló listát, ezek közül manhattam távolság alapján kiválasztja a legközelebbi cellát, majd beállítja a *destinationCell* attribútumára. A kisrobot addig lépked vertikális irányban amíg el nem ért abba a sorba, ahol a *destinationCell* található. Ekkor pedig horizontálisan kezd el mozogni, míg el nem éri a *destinationCell*-t. Ha elérte a cél cellát, egy kör alatt feltakarítja.
Láthatóság: +.
- **void onTurnEnd():** Kör végén nincs dolga a keményen dolgozó kisrobotnak.
Láthatóság: +.

8.2 A tesztek részletes tervezeti leírásuk a teszt nyelvén

8.2.1 Az idő lejár

- **Leírás**

A játék végét vizsgáljuk. A kellő kör letelte után a játéknak vége kell, hogy legyen. Ekkor a program kiírja, hogy az egyes játékosok mekkora utat tettek meg, majd győztest hirdet.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A játéknak vége kell, hogy legyen adott kör után. A nagyobb távolságra jutó játékost kell győztesnek kihirdetnie. Egyenlő távolságok megtétele esetén az eredmény döntetlen. Várható hiba lehet, hogy a játék végét jelző szöveg nem jelenik meg adott kör letelte után, illetve nem a nagyobb távolságot megtevő játékost hirdeti meg győztesnek.

- **Bemenet**

start map.txt

set-clock 9

move N

end-turn

```

end-turn
start map.txt
set-clock 9
end-turn
    move N
end-turn
start map.txt
set-clock 9
end-turn
end-turn

```

- **Elvárt kimenet**

```

Player 0: Cell(22,8) Speed: 2 CanChangeDirection: True
Player 0: New Position: Cell(20, 8)
Player 1: Cell(22,10) Speed: 2 CanChangeDirection: True
Player 0: Distance - 2; Player 1: Distance - 0; Winner: Player 0
Player 0: Cell(22,8) Speed: 2 CanChangeDirection: True
Player 1: Cell(22,10) Speed: 2 CanChangeDirection: True
Player 1: New Position: Cell(20, 10)
Player 0: Distance - 0; Player 1: Distance - 2; Winner: Player 1
Player 0: Cell(22,8) Speed: 2 CanChangeDirection: True
Player 1: Cell(22,10) Speed: 2 CanChangeDirection: True
Player 0: Distance - 0; Player 1: Distance - 0; Winner: Draw

```

8.2.2 Mozgás

- **Leírás**

A játékos mozgatja a hozzá tartozó robotot. Meg kell adnia, hogy melyik irányba szeretne lépni (észak, északkelet, északnyugat, dél, délkelet, délnyugat, kelet, nyugat).

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Ezzel a teszesettel ellenőrizhető, hogy a játékos által irányított robot a megfelelő cellára lépett-e. Várható hiba lehet, hogy a robot nem a megfelelő irányba ugrik. A program ki fogja írni a kezdőpozició koordinátáit, majd a célpozició koordinátáit, így a helyes lépés könnyen ellenőrizhető a megfelelő koordináták egymásból kivonásával.

- **Bemenet**

```
start map.txt
```

```
move N
```

- **Elvárt kimenet**

```

Player 0: Cell(22,8) Speed: 2 CanChangeDirection: True
Player 0: New Position: Cell(20, 8)

```

8.2.3 Nagy robot ugrik kisrobotra

- **Leírás**

A játékos által irányított robot ráugrik a takarító robotra, vagyis a játékos egy olyan cellára mozog, amelyen egy kisrobot található,

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A teszesetben két dolgot kell ellenőrizni: Először is a kisrobotnak meg kell semmisülnie, majd helyén egy olajfolt kell, hogy keletkezzen. Várható hiba lehet, hogy a kisrobot nem semmisül meg, illetve, hogy nem keletkezik a helyén egy olajfolt. Ezt a cellát leíró metódus meghívásával ellenőrizhetjük, amely kiírja a cellán található játék-objektumot (kisrobot vagy folt).

- **Bemenet**

```
start map.txt
```

hardworking-little-robot-random off
 robot-move 0 22 6
 move W

- **Elvárt kimenet**

Player 0: Cell(22,8) Speed: 2 CanChangeDirection: True
 Hardworking-little-robot 0: New Position: Cell(22, 6)
 Player 0: New Position: Cell(22, 6)
 Cell(22, 6): GameObject: OilStain

8.2.4 Nagy robotok ütköznek

- **Leírás**

A két játékos által irányított robot közül az egyik arra a cellára ugrik, amelyen a másik játékos által irányított robot tartózkodik.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A teszesetben két dolgot kell ellenőrizni: A két robot közül a lassabbnak össze kell törnie, és a gyorsabb fél sebességének meg kell változnia. Sok esetben a robotok egyenlő sebességgel haladnak majd, ez esetben az számít, hogy ki volt az elszenevedő fél, és ki a kezdeményező. Várható hiba lehet, hogy nem a lassabb robot török össze, azaz nem ő veszíti el a játékot. Mivel két játékosunk van, ezért egy ilyen ugrás biztosan a játék végét is jelenti. Másik várható hiba, hogy a gyorsabb robot sebességének iránya, és nagysága nem változik meg. Ezt az ugrás után a játékos tulajdonságait leíró metódus meghívásával ellenőrizhetjük.

- **Bemenet**

start map.txt

move E

- **Elvárt kimenet**

Player 0: Cell(22,8) Speed: 2 CanChangeDirection: True
 Player 0: New Position: Cell(22, 10)
 Player 0: Distance - 2; Player 1: Distance - 0; Winner: Player 0

8.2.5 Érvénytelen cellára ugrás

- **Leírás**

Az egyik robot érvénytelen cellára ugrik. Mivel két játékosunk van, ezért ez rögtön a játék végét is jelenti. A kiugrott robot a vesztes, a másik pedig a győztes.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Ebben a teszesetben kell ellenőrizni, hogyha egy robot érvénytelen cellára ugrik, az kiesik-e a játékból, azaz véget ér-e a küzdelem. Lehetséges hiba, hogy hiába ugrik egy játékos egy invalid cellára, mégsem esik ki, hanem folytatódik a játék. Másik lehetséges hiba, hogy a program a rossz játékost hirdeti ki győztesnek. Ezeket a játék végét jelző szöveg elolvasásával ellenőrizhetjük.

- **Bemenet**

start map.txt

move S

- **Elvárt kimenet**

Player 0: Cell(22,8) Speed: 2 CanChangeDirection: True
 Player 0: New Position: Cell(24, 8)
 Player 0: Distance - 2; Player 1: Distance - 0; Winner: Player 1

8.2.6 Olajfoltos cellára ugrás

- **Leírás**

Az egyik játékos olyan cellára ugrik, amelyen olajfolt található. Ekkor a játékos sebességének meg kell feleződni.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Itt ellenőrizzük, hogy miután egy játékos egy olajfoltos cellára ugrott, a speed attribútuma valóban megfeleződött-e (2-ről 1-re állítódott-e). Ezeket a játékos tulajdonságait leíró metódus meghívásával ellenőrizhetjük.

- **Bemenet**

```
start map.txt
end-turn
move N
end-turn
```

- **Elvárt kimenet**

```
Player 0: Cell(22,8) Speed: 2 CanChangeDirection: True
Player 1: Cell(22,10) Speed: 2 CanChangeDirection: True
Player 1: New Position: Cell(20, 10)
Player 0: Cell(22,8) Speed: 2 CanChangeDirection: True
Player 1: Cell(20,10) Speed: 2 CanChangeDirection: False
```

8.2.7 Ragacsfoltos cellára ugrás

- **Leírás**

Az egyik játékos olyan cellára ugrik, amelyen ragacsfolt található. A játékos a következő körben nem választhat irányt. A ragacsnak pedig kopottabbnak kell lennie.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Itt ellenőrizzük, hogy miután egy játékos egy ragacsfoltos cellára ugrott, a canChangeDirection attribútuma valóban hamisra állítódott-e. Ezeket a játékos tulajdonságait leíró metódus meghívásával ellenőrizhetjük.

- **Bemenet**

```
start map.txt
move N
end-turn
end-turn
```

- **Elvárt kimenet**

```
Player 0: Cell(22,8) Speed: 2 CanChangeDirection: True
Player 0: New Position: Cell(20, 8)
Player 1: Cell(22,10) Speed: 2 CanChangeDirection: True
Player 0: Cell(20,8) Speed: 1 CanChangeDirection: True
```

8.2.8 Olajfolt lerakása

- **Leírás**

Az adott játékos a mozgása után olajfoltot rak le arra cellára, amelyre megérkezett.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Ebben a teszesetben ellenőrizzük, hogy az olajfolt lerakása helyesen működik-e. A robotnak a jó foltot kell leraknia (azaz hiba lehet, hogy másik típusú foltot tesz le, amikor olajfolt lerakására kapott utasítást), a foltnak pedig meg kell jelennie a cellán. Ezeket a cella tulajdonságait leíró metódus meghívásával ellenőrizhetjük.

- **Bemenet**

```
start map.txt
put-stain O
```

- **Elvárt kimenet**

```
Player 0: Cell(22,8) Speed: 2 CanChangeDirection: True
Cell(22, 8): GameObject: OilStain
```

8.2.9 Ragacsfolt lerakása

- **Leírás**

Az adott játékos a mozgása után ragacsfoltot rak le a cellára, amelyre megérkezett.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Itt ellenőrizzük, hogy a ragacsfolt lerakása helyesen működik-e. A robotnak a jó foltot kell leraknia (azaz hiba lehet, hogy másik típusú foltot tesz le, amikor ragacsfolt lerakására kapott utasítást), a foltnak pedig meg kell jelennie a cellán. Ezeket a cella tulajdonságait leíró metódus meghívásával ellenőrizhetjük.

- **Bemenet**

```
start map.txt
```

```
put-stain G
```

- **Elvárt kimenet**

```
Player 0: Cell(22,8) Speed: 2 CanChangeDirection: True
Cell(22, 8): GameObject: GlueStain
```

8.2.10 Olajfolt felszáradása

- **Leírás**

Adott kör elteltével az olajfoltnak el kell tűnnie a celláról.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Ebben a tesztesetben ellenőrizzük, hogy adott kör elteltével az olajfolt valóban eltűnik-e a cellájáról. Hiba lehet, hogy egy kör elteltével a folt eventCount attribútuma nem csökken egygyel, illetve, hogy az eventCount 0-ra csökkenésével nem tűnik el a celláról. Ezek ellenőrzéséhez adott kör elteltével (jelen tesztesetben ez négy kör) meg kell hívunk a cellát leíró metódust, hogy lássuk valóban eltűnt-e folt.

- **Bemenet**

```
start map.txt
```

```
end-turn
```

- **Elvárt kimenet**

```
Player 0: Cell(22,8) Speed: 2 CanChangeDirection: True
Player 1: Cell(22,10) Speed: 2 CanChangeDirection: True
Player 0: Cell(22,8) Speed: 2 CanChangeDirection: True
Player 1: Cell(22,10) Speed: 2 CanChangeDirection: True
Player 0: Cell(22,8) Speed: 2 CanChangeDirection: True
Player 1: Cell(22,10) Speed: 2 CanChangeDirection: True
Player 0: Cell(22,8) Speed: 2 CanChangeDirection: True
Player 1: Cell(22,10) Speed: 2 CanChangeDirection: True
Cell(20, 10): GameObject: null
```

8.2.11 Ragacsfolt elhasználódása

- **Leírás**

A ragacsfoltnak el kell tűnnie a celláról, miután négyeszer beleugrottak.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Ebben a tesztesetben ellenőrizzük, hogy a folt valóban eltűnik-e a cellájáról, miután robotok négyeszer beleugrottak. Hiba lehet, hogy egy beleugrás után a folt eventCount attribútuma nem csökken egygyel, illetve, hogy az eventCount 0-ra csökkenésével nem tűnik el a celláról. Ezek ellenőrzéséhez négy ugrás után meg kell hívunk a cellát leíró metódust, hogy lássuk valóban eltűnt-e folt.

- **Bemenet**

```
start map.txt  
move 20 8  
end-turn  
end-turn  
move 20 9  
end-turn  
end-turn  
move 20 8  
end-turn  
end-turn  
move 20 9  
end-turn  
end-turn  
move 20 8  
end-turn  
end-turn  
move 20 9  
end-turn  
end-turn  
move 20 8  
end-turn  
end-turn
```

• Elvárt kimenet

Player 0: Cell(22,8) Speed: 2 CanChangeDirection: True
Player 0: New Position: Cell(20, 8)
Player 1: Cell(22,10) Speed: 2 CanChangeDirection: True
Player 0: Cell(20,8) Speed: 2 CanChangeDirection: True
Player 0: New Position: Cell(20, 9)
Player 1: Cell(22,10) Speed: 2 CanChangeDirection: True
Player 0: Cell(20,9) Speed: 2 CanChangeDirection: True
Player 0: New Position: Cell(20, 8)
Player 1: Cell(22,10) Speed: 2 CanChangeDirection: True
Player 0: Cell(20,8) Speed: 2 CanChangeDirection: True
Player 0: New Position: Cell(20, 9)
Player 1: Cell(22,10) Speed: 2 CanChangeDirection: True
Player 0: Cell(20,9) Speed: 2 CanChangeDirection: True
Player 0: New Position: Cell(20, 8)
Player 1: Cell(22,10) Speed: 2 CanChangeDirection: True
Player 0: Cell(20,8) Speed: 2 CanChangeDirection: True
Player 0: New Position: Cell(20, 9)
Player 1: Cell(22,10) Speed: 2 CanChangeDirection: True
Player 0: Cell(20,9) Speed: 2 CanChangeDirection: True
Player 0: New Position: Cell(20, 8)
Player 1: Cell(22,10) Speed: 2 CanChangeDirection: True
Cell(20, 10): GameObject: null

8.2.12 Kisrobot takarítása

• Leírás

Egy kisrobot, ha olyan cellára kerül, amelyen folt található, akkor azt egy kör alatt feltakarítja, azaz egy kör műlva már nem lesz a cellán semmilyen folt.

• **Ellenőrzött funkcionalitás, várható hibahelyek**

Itt ellenőrizzük, hogy miután a kisrobot egy foltos cellára került, a folt valóban eltűnik-e egy kör elteltével. Hiba lehet, hogy a folt nem tűnik el a celláról, azaz egy kör elteltével is jelzi a rajta levő foltot a cellát leíró metódus.

- **Bemenet**

```
start map.txt
hardworking-little-robot-random off
robot-move 0 20 10
```

- **Elvárt kimenet**

Hardworking-little-robot 0: New Position: Cell(20, 10)
 Cell(20, 10): GameObject: Hardworking-little-robot

8.2.13 Kisrobot nagyrobotnak ütközik

- **Leírás**

A kisrobot, miután nekiütközött egy nagyrobotnak (azaz olyan cellára lép, amelyen egy nagyrobot tartózkodik), irányt vált, azaz választ egy másik cellát a környezetéből.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Ebben a teszesetben ellenőrizzük, hogy a robot által választott új cella különbözik-e a régitől. Azaz a teszt akkor sikeres, ha az új cella nem egyezik a régivel. Ezt a koordináták összehasonlításával ellenőrizhetjük.

- **Bemenet**

```
start map.txt
hardworking-little-robot-random off
robot-move 0 22 8
```

- **Elvárt kimenet**

Hardworking-little-robot 0: Collision; New Position: Cell(20, 8)

8.2.14 Kisrobot kisrobotnak ütközik

- **Leírás**

Egy kisrobot nekiütközik egy másik kisrobotnak. A vétkes fél irányt vált, azaz választ egy másik cellát a környezetéből.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Ebben a teszesetben ellenőrizzük, hogy a vétkes kisrobot által választott új cella különbözik-e a régitől. Ez nagyban hasonlít az előző teszesetre, annyi különbséggel, hogy a nagyrobot helyett egy másik kisrobot szerepel benne.

- **Bemenet**

```
start map.txt
hardworking-little-robot-random off
move-robot 0 10 10
move-robot 1 10 10
```

- **Elvárt kimenet**

Hardworking-little-robot 0: New Position: Cell(10, 10)
 Hardworking-little-robot 1: New Position: Cell(10, 10)
 Hardworking-little-robot 0: Collision; New Position: Cell(8, 10)
 Hardworking-little-robot 1: Collision; New Position: Cell(12, 10)

8.3 Kiegészítés a tesztekhez

A tesztesetekhez használt pálya file: map.txt

Tartalom (szemléletesen):

Formátum jelentése:

'0' - invalid cella

‘1’ - valid cella, folt nélkül

'2' - valid cella, olajfolttal

'2' - vallat cella, olajfronttal

‘4’ - első játékos kezdeti pozíciója

8.4 A tesztelést támogató programok tervei

8.4.1 paladiff

Az ellenőrzés fő eszköze egy egyszerű, a UNIX rendszerekből jól ismert diff parancsra emlékeztető segédprogram lesz. A parancs két fájlt vár bemenetként, az egyik az elvárt kimenet, a másik pedig a főprogram fájlba irányított kimenete. Amennyiben a kapott és az elvárt kimenet ugyanaz, a teszt sikeresnek tekintjük, különben a teszt sikertelen.

8.5 Napló

Kezdet	Időtartam	Résznevők	Leírás
2015.04.03. 16:30	1,5 óra	Verbőczy	Tevékenység: Az osztályok és metódusok terveinek elkészítése.
2015.04.04. 10:15	1,5 óra	Herold	Tevékenység: Osztály leírások pontosítása, kijavítása.
2015.04.04. 20:45	1,5 óra	Graics	Tevékenység: Tesztek részletes terveinek leírása, várható hibahelyek felsorolása.
2015.04.05. 18:30	0,5 óra	Sallai	Tevékenység: A tesztelést támogató programok terveinek leírása
2015.04.06. 21:30	2,5 óra	Graics Herold Sallai Verbőczy	Tevékenység: Tesztek kimeneteinek és bemeneteinek leírása.

0. Változtatások a tesztesetekben

Minden tesztesetben a “reset” utasítást kicseréltük “start map.txt” utasításra.

A tesztesetekben bizonyos utasítások megváltoztak (utasítások sorrendje, új utasítások is kerültek be), ezért tests mappában található utasítások használatát javasoljuk.

10. Prototípus beadása

10.1 Fordítási és futtatási útmutató

10.1.1 Fájllista

Fájl neve	Méret	Keletkezés ideje	Tartalom
Paladiff.java	6375 byte	2015.04.19. 22:58	paladiff
Cell.java	4600 byte	2015.04.19. 23:07	Cella osztály
CellType.java	120 byte	2015.04.19. 22:58	CellType enum
Game.java	18447 byte	2015.04.19. 23:01	Game osztály.
GameException.java	190 byte	2015.04.19. 22:58	GameException osztály
GameObject.java	799 byte	2015.04.19. 23:01	GameObject interface.
GlueStain.java	1047 byte	2015.04.19. 22:58	GlueStain osztály.
Logger.java	2662 byte	2015.04.19. 22:57	Logger osztály.
Main.java	1433 byte	2015.04.19. 22:58	Main osztály.
Map.java	3476 byte	2015.04.19. 23:01	Map osztály.
OilStain.java	1015 byte	2015.04.19. 22:58	OilStain osztály.
Player.java	6666 byte	2015.04.19. 23:01	Player osztály.
Robot.java	5786 byte	2015.04.19. 23:10	Kisrobot osztály.
Stain.java	976 byte	2015.04.19. 23:01	Stain osztály.
map.txt	949 byte	2015.04.19. 17:55	A térkép fájl.
glue-stain.in	80 byte	2015.04.19. 21:52	Glue-stain teszteset bemenet.
glue-stain.out	202 byte	2015.04.19. 18:03	Glue-stain teszteset kimenete.
glue-used-up.in	230 byte	2015.04.19. 19:09	A glue-used-up teszeset bemenete.
glue-used-up.out	1112 byte	2015.04.19. 21:52	A glue-used-up teszteset kimenete.
hardworking-little-robot-cleans.in	83 byte	2015.04.19. 21:52	A hardworking-little-robot-cleans teszteset bemenete.
hardworking-little-robot-cleans.out	192 byte	2015.04.19. 21:52	A hardworking-little-robot-cleans teszteset kimenete.
invalid-cell.in	71 byte	2015.04.19. 21:52	A invalid-cell teszteset bemenete.
invalid-cell.out	156 byte	2015.04.19. 18:03	A invalid-cell teszteset kimenete.
little-robot-player-collision.in	82 byte	2015.04.19. 21:52	A little-robot-player-collision teszteset bemenete.
little-robot-player-collision.out	228 byte	2015.04.19. 21:52	A little-robot-player-collision teszteset kimenete.
little-robots-collide.in	93 byte	2015.04.19. 18:22	A little-robots-collide teszteset bemenete.

little-robots-collide.out	229 byte	2015.04.19. 21:52	A little-robots-collide teszteset kimenete.
move.in	62 byte	2015.04.19. 21:52	A move teszteset bemenete.
move.out	91 byte	2015.04.19. 18:03	A move teszteset kimenete.
oil-dries-up.in	82 byte	2015.04.19. 21:52	A oil-dries-up teszteset bemenete.
oil-dries-up.out	25 byte	2015.04.19. 21:52	A oil-dries-up teszteset kimenete.
oil-stain.in	89 byte	2015.04.19. 21:52	A oil-stain teszteset bemenete.
oil-stain.out	289 byte	2015.04.19. 21:52	A oil-stain teszteset kimenete.
player-collision.in	71 byte	2015.04.19. 21:52	A player-collision teszteset bemenete.
player-collision.out	157 byte	2015.04.19. 21:52	A player-collision teszteset kimenete.
player-little-robot-collision.in	89 byte	2015.04.19. 21:52	A player-little-robot-collision teszteset bemenete.
player-little-robot-collision.out	233 byte	2015.04.19. 21:52	A player-little-robot-collision teszteset kimenete.
put-glue.in	67 byte	2015.04.19. 21:52	A put-glue teszteset bemenete.
put-glue.out	88 byte	2015.04.19. 21:52	A put-glue teszteset kimenete.
put-oil.in	67 byte	2015.04.19. 21:52	A put-oil teszteset bemenete.
put-oil.out	87 byte	2015.04.19. 21:52	A put-oil teszteset kimenete.
time-up-draw.in	85 byte	2015.04.19. 21:52	A time-up-draw teszteset bemenete.
time-up-draw.out	230 byte	2015.04.19. 21:52	A time-up-draw teszteset kimenete.
time-up-player0.in	92 byte	2015.04.19. 21:52	A time-up-player0 teszteset bemenete.
time-up-player0.out	266 byte	2015.04.19. 21:52	A time-up-player0 teszteset kimenete.
time-up-player1.in	92 byte	2015.04.19. 21:52	A time-up-player1 teszteset bemenete.
time-up-player1.out	267 byte	2015.04.19. 21:52	A time-up-player1 teszteset kimenete.

10.1.2 Fordítás

javac src/*/*.java

10.1.3 Futtatás

cd src

Ha a paladiffet akarjuk indítani:

java paladiff.Paladiff tests

Ha a játékot akarjuk indítani:

java phoebe.Main

Megjegyzés: a játékot a “start map.txt” parancssal indítsuk(a java phoebe.Main után)!

10.2 Tesztek jegyzőkönyvei

10.2.1 Glue_stain

Tesztelő neve	Verbőczy Kristóf
Teszt időpontja	2015.04.19.

10.2.2 Glue_used_up

Tesztelő neve	Verbőczy Kristóf
Teszt időpontja	2015.04.19.

10.2.3 Hardworking_little_robot_clean

Tesztelő neve	Graics Bence
Teszt időpontja	2015.04.19.

Tesztelő neve	Graics Bence
Teszt időpontja	2015.04.18.
Teszt eredménye	A kisrobot nem takarította fel a foltot a celláról, amikor odaérkezett.
Lehetséges hibaok	A cella gameObject referencia nem lett átállítva.
Változtatások	A cella gameObject referenciaját ráállítjuk a kisrobotra.

10.2.4 Invalid_cell

Tesztelő neve	Graics Bence
Teszt időpontja	2015.04.19.

10.2.5 Little_robot_player_collision

Tesztelő neve	Herold Kristóf
Teszt időpontja	2015.04.19.

Tesztelő neve	Herold Kristóf
Teszt időpontja	2015.04.19.
Teszt eredménye	A kisrobot cellája nem változik meg.
Lehetséges hibaok	Rossz implementáció a Robot move metódusában. Nem állítja át a currentCell referenciát egy másik cellára.
Változtatások	A metódus kiegészítése. CurrentCell átállítása egy másik cellára.

Player 0: Cell(22,8) Speed: 2 CanChangeDirection: true

Hardworking-little-robot 0: New Position: Cell(22, 8)

Hardworking-little-robot 0: Collision; New Position: Cell(2, 2)

Player 1: Cell(22,10) Speed: 2 CanChangeDirection: true

10.2.6 Little_Robots_Collide

Tesztelő neve	Herold Kristóf
Teszt időpontja	2015.04.19.

10.2.7 Move

Tesztelő neve	Herold Kristóf
Teszt időpontja	2015.04.19.

10.2.8 Oil_dries

Tesztelő neve	Herold Kristóf
Teszt időpontja	2015.04.18.

Tesztelő neve	Herold Kristóf
Teszt időpontja	2015.04.19
Teszt eredménye	Az olajfolt nem szárad fel 4 kör elteltével.
Lehetséges hibaok	Az onTurnStart metódus nem csökkenti a folt eventCount attribútumát.
Változtatások	A metódus kiegészítése a fenti funkcionalitással.

Player 0: Cell(22,8) Speed: 2 CanChangeDirection: true

Player 1: Cell(22,10) Speed: 2 CanChangeDirection: true

Player 0: Cell(22,8) Speed: 2 CanChangeDirection: true

Cell(20,10) GameObject: null

Player 1: Cell(22,10) Speed: 2 CanChangeDirection: true

10.2.9 Oil_stain

Tesztelő neve	Herold Kristóf
Teszt időpontja	2015.04.19.

10.2.10 Player_collision

Tesztelő neve	Herold Kristóf
Teszt időpontja	2015.04.19.

Tesztelő neve	Herold Kristóf
----------------------	----------------

Teszt időpontja	2015.04.19.
Teszt eredménye	Nem a gyorsabb nyert játékos.
Lehetséges hibaok	Alapértelmezetten az első játékos nyer.
Változtatások	Az ezért felelős metódusba bele kellett írni, hogy a gyorsabb nyer.

10.2.11 Player_little_robot_collision

Tesztelő neve	Sallai Gyula
Teszt időpontja	2015.04.19.

Tesztelő neve	Sallai Gyula
Teszt időpontja	2015.04.19
Teszt eredménye	A kisrobot nem hagy olajfoltot, miután megsemmisül.
Lehetséges hibaok	A kisrobot megsemmisüléskor nem a jó cellára teszi le az olajfoltot,
Változtatások	A cella megvizsgálása, currentCell-nek kell lennie a folt cellájának.

10.2.12 Put_glue

Tesztelő neve	Sallai Gyula
Teszt időpontja	2015.04.19.

10.2.13 Put_oil

Tesztelő neve	Sallai Gyula
Teszt időpontja	2015.04.19.

10.2.14 Time_up_draw

Tesztelő neve	Sallai Gyula
Teszt időpontja	2015.04.19.

Tesztelő neve	Sallai Gyula
Teszt időpontja	2015.04.19.
Teszt eredménye	Az első játékos nyert.
Lehetséges hibaok	Az alapértelmezett győztes az első játékos.
Változtatások	Egy elágazás, ami figyeli, hogy egyforma távot tettek-e meg a robotok.

10.2.15 Time_up_player0

Tesztelő neve	Sallai Gyula
Teszt időpontja	2015.04.19.

10.2.16 Time_up_player1

Tesztelő neve	Sallai Gyula
Teszt időpontja	2015.04.19.

10.3 Értékelés

Tag neve	Munka százalékban
Graics Bence	25.75%
Herold Kristóf	27%
Sallai Gyula	27%
Verbőczy Kristóf	20.25%

10.4 Napló

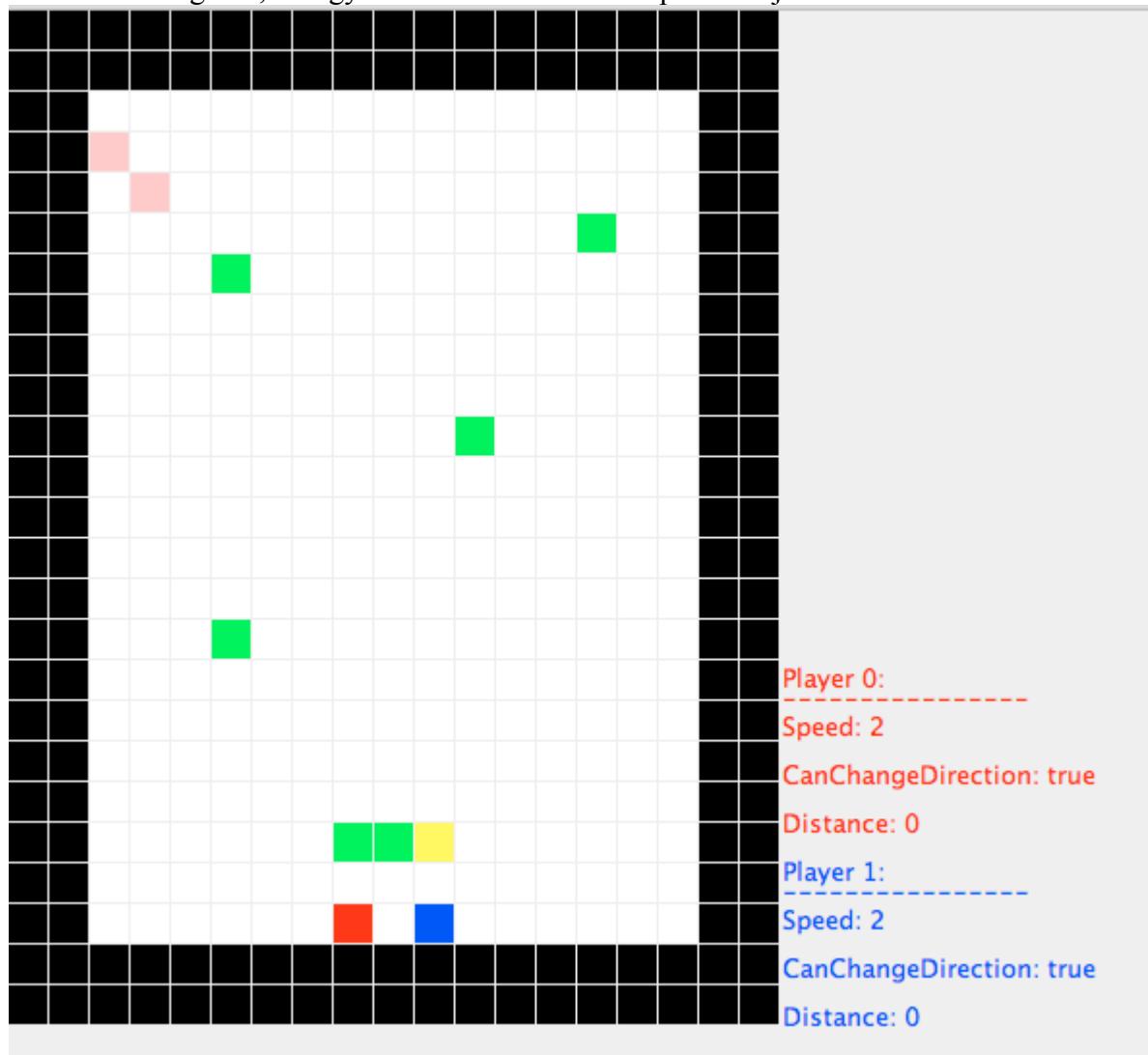
Kezdet	Időtartam	Résztvevők	Leírás
2015.04.14. 11:00	1 óra	Herold	Új osztály felvétele: Robot, implementálás.
2015.04.18. 17:23	2 óra	Sallai	Paladiff implementálása.
2015.04.19. 13:00	8 óra	Herold, Sallai	Kód implementálás.
2015.04.19 19:00	6 óra	Graics, Verbőczy	Tesztesetek ellenőrzése. Dokumentáció elkészítése.

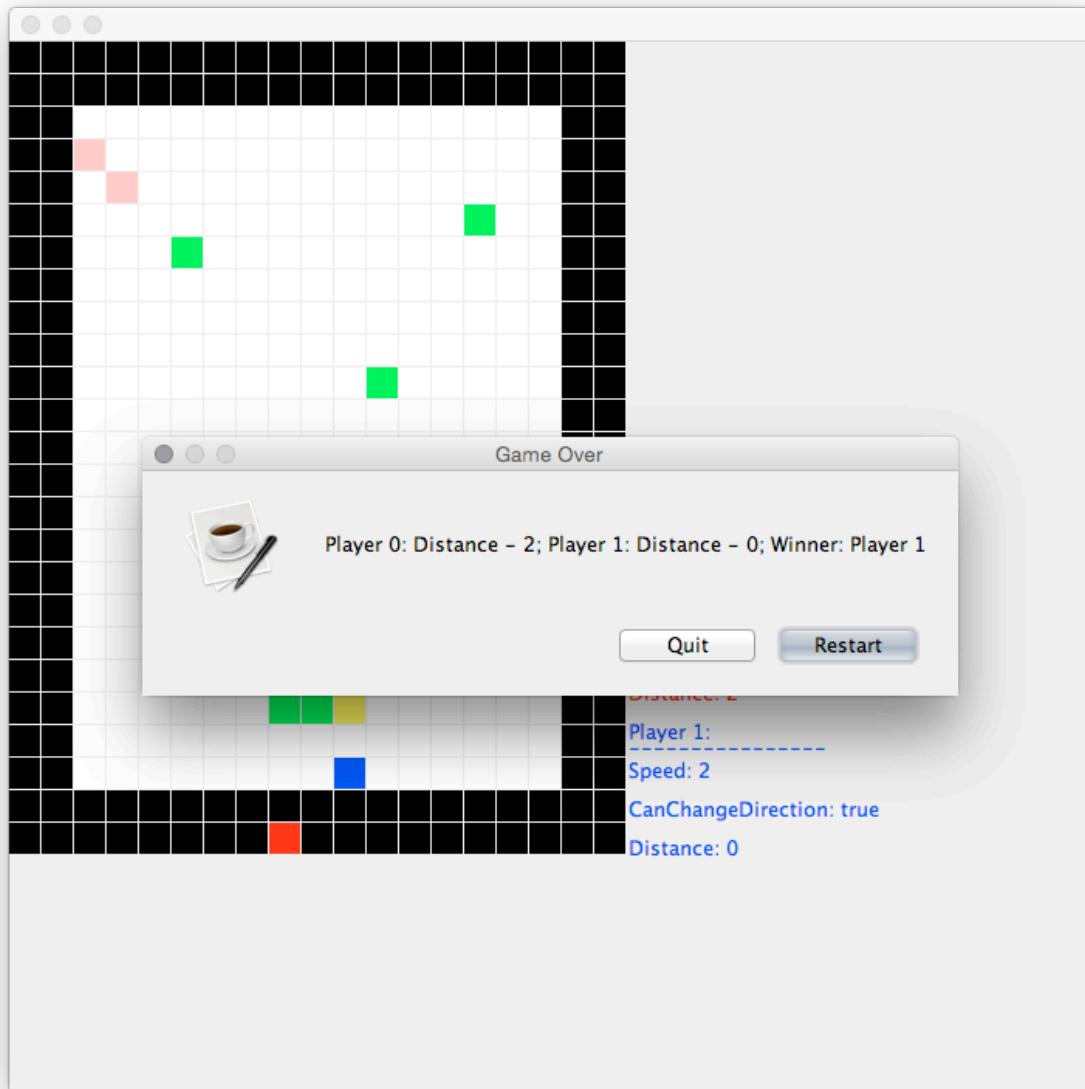
11. Grafikus felület specifikációja

11.1 A grafikus interfész

A játék kezelőfelülete oly mértékben egyszerű, hogy nem indokolja külön menü használatát. Ezen okból kifolyólag nem készítünk menüt; a program induláskor rögtön belép a játékba. Ezt az első kép szemlélteti. Amint látható, a különböző objektumokat különböző színű négyzetek reprezentálják. A játékosok robotja piros, illetve kék színű. A keményen dolgozó kisrobotok rózsaszín színűek. A ragacsfoltok zöld színűek. Az olajfoltok sárga színűek. Az érvényes cellákat fehér, az érvényteleneket fekete szín reprezentálja. A robotok állapotáról a felhasználók a térképtől jobbra elhelyezkedő szövegekből tájékozódhatnak (sebesség, tud-e irányt váltani, megtett távolság). Az alkalmazás a felhasználókkal felugró ablakokkal fog kommunikálni:

- Ha egy játékos beleugrik egy foltba, akkor ezt az alkalmazás egy felugró ablakban jelzi.
- Ha egy felhasználó újra akarja indítani a játékot, egy felugró ablak megkérdezi, hogy biztosan ezt akarja-e.
- Ha vége a játéknak, akkor felugrik egy új ablak, amelyről leolvashatjuk a megtett távolságokat, és a győztest. Ezt a második kép ábrázolja.



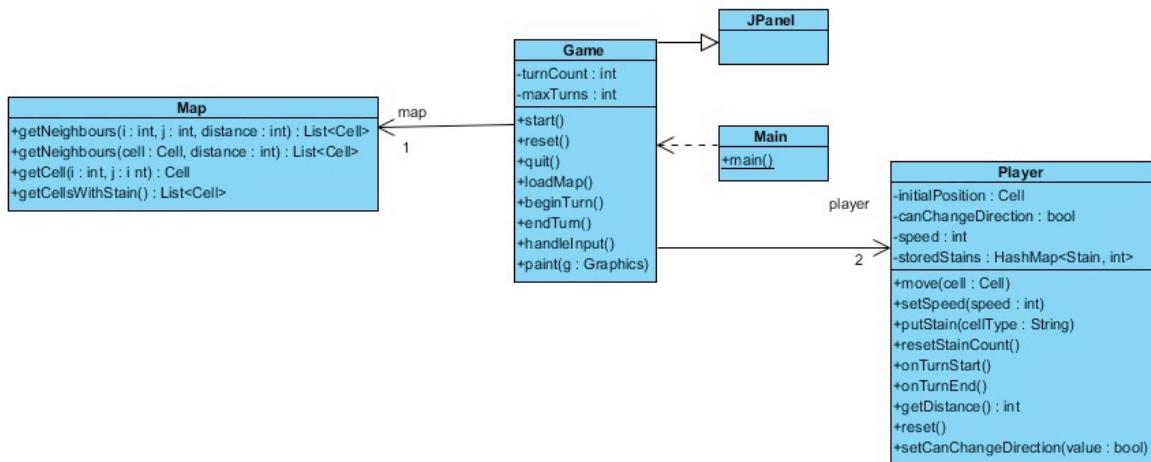


11.2 A grafikus rendszer architektúrája

11.2.1 A felület működési elve

A projekt kezdetétől való igényes tervezésnek köszönhetően a játék modelljében nem történik sok változás. Lényegében csak egy grafikus felület lesz rátámasztva a prototípusra. A Game osztály mint View fogja a megjelenítést végezni a Map mint Model adatai alapján. Ennek megvalósításához a Game osztály leszármazik a Java beépített JPanel osztályából, hogy rajzolhasson a képernyőre. A képernyő frissítése minden kör elején megtörténik: a Game rajzolásáért felelős metódusa lekéri a Map-tól az egyes cellák tartalmát (ez a prototípusban is meg volt valósítva), majd a tartalomnak megfelelően rajzol a képernyőre. A körökön belül nincs szükség újrarendelezésre, hiszen a játék körökre osztott, így az állás egy körben csak egyszer változik. A felhasználó a játékban történő eseményekről pop-up ablakokban fog értesülni.

11.2.2 A felület osztály-struktúrája



11.3 A grafikus objektumok felsorolása

Nem került be új osztály a programba. Egyedül a Game osztály bővült egy kirajzolási funkcióval. A Game-hez kapcsolódó osztályok (Map, Mapon keresztül a Cell, Player) semmiben sem változtak az előző verzióhoz képest.

11.3.1 Game

- **Felelősség**

A Game osztály felelőssége a pálya megjelenítése a Map osztály által szolgáltatott Cell adatok alapján. Kirajzolja a képernyőre a térképet, és a megfelelő cellákra a megfelelő objektumokat. A térképtől balra megjeleníti az egyes robotok aktuális állapotát. Értesíti a felhasználókat a körben történt eseményekről (játék vége, foltba lépés).

- **Ősosztályok**

JPanel

- **Attribútumok**

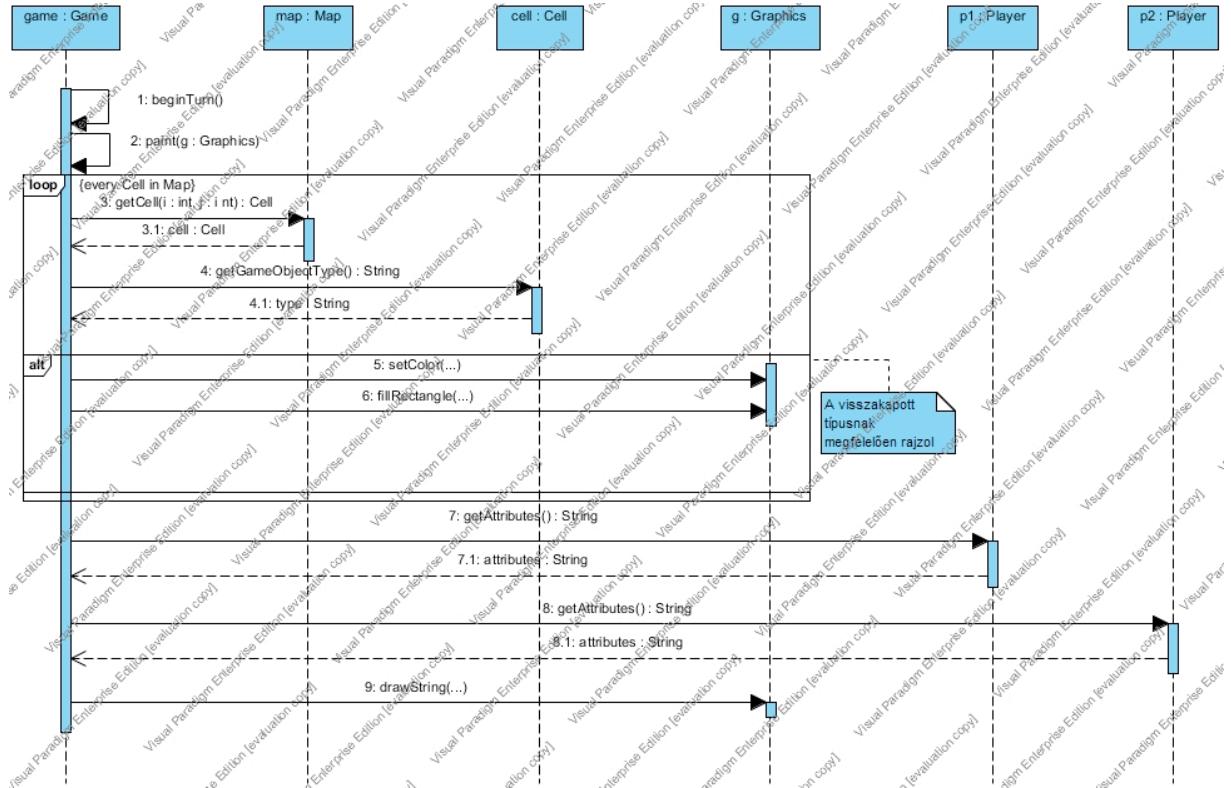
Nincs változás az előző verziókhöz képest.

- **Metódusok**

- **void beginTurn()**: Annyiban különbözik az előző verziótól, hogy elsőként meghívja a paint(Graphics g) metódust.
- **void paint(Graphics g)**: JPanel ősosztály paint(Graphics g) felüldefiniálása. A metódus sorban lekéri a Map minden cellájának a tartalmát. A tartalomtól függően, és a cellának a koordinátája szerint rajzol egy színes négyzetet a képernyőre. Ezután lekéri a játékosok tulajdonságait, majd azokat kiírja a képernyőre.
- **void handleInput()**: Nyomógombok lekezelése: ‘R’ billentyű esetén modális ablak megjelenítése, a következő szöveggel: “Biztosan újraindítja a játékot?” az erre adott interakció függvényében pedig vagy újraindul a játék, vagy folytatódik az aktuális. Irány kiválasztására a ‘D’ billentyű lenyomásával lesz lehetőség. Kör befejezése: ENTER billentyű. ‘A’ billentyű lenyomása esetén olajfoltot helyezhetünk el a pályán, ‘S’ lenyomása esetén pedig ragacsfoltot helyezhetünk el.

11.4 Kapcsolat az alkalmazói rendszerrel

A játék grafikus megjelenítése, tehát a térkép(pálya) kirajzolása az ablakos felületre.



11.5 Napló

Kezdet	Időtartam	Résztvevők	Leírás
2015.04.24. 18:00	1 óra	Graics Herold Sallai Verbőczy	Megbeszélés. Koncepció kialakítása. Közben szurkolás a magyar válogattnak.
2015.04.25. 22:00	2 óra	Herold Graics Verbőczy	Tevékenység: Herold létrehozza a screenshotokat. Graics elkészíti az új szekvenciadiagramot és osztálydiagramot. Verbőczy dokumentálja a változásokat,
2015.04.26. 18:00	3 óra	Graics Verbőczy	Tevékenység: Verbőczy elkészít egy szekvenciadiagramot, dokumentálja a grafikus interfést. Graics elkészíti a Game osztály leírását.

0. Változtatások

0.1 Paladrawin

Paladrawin osztály a rajzolásért felelős osztály, űosztálya a JPanel. A játék logikát a Game osztály tartalmazza, ezáltal a Game osztály rendelkezik egy Paladrawin attribútummal, melynek biztosítja az adatokat a kirajzoláshoz, ezek a map és a playerek.

0.2 Irányítás

A játékos az alábbi módon tudja irányítani a robotot:

W - balra-fel, E - fel, R - jobbra-fel

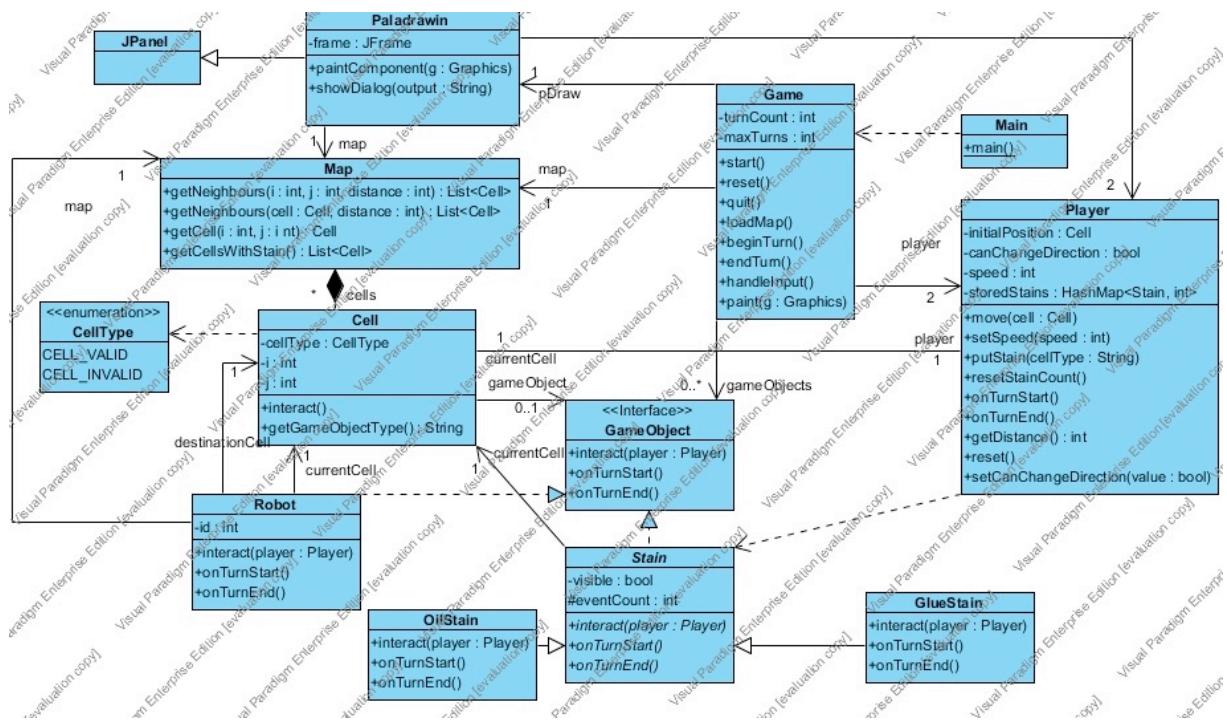
S - balra, F - jobbra

X - balra-le, C - le, V - jobbra-le

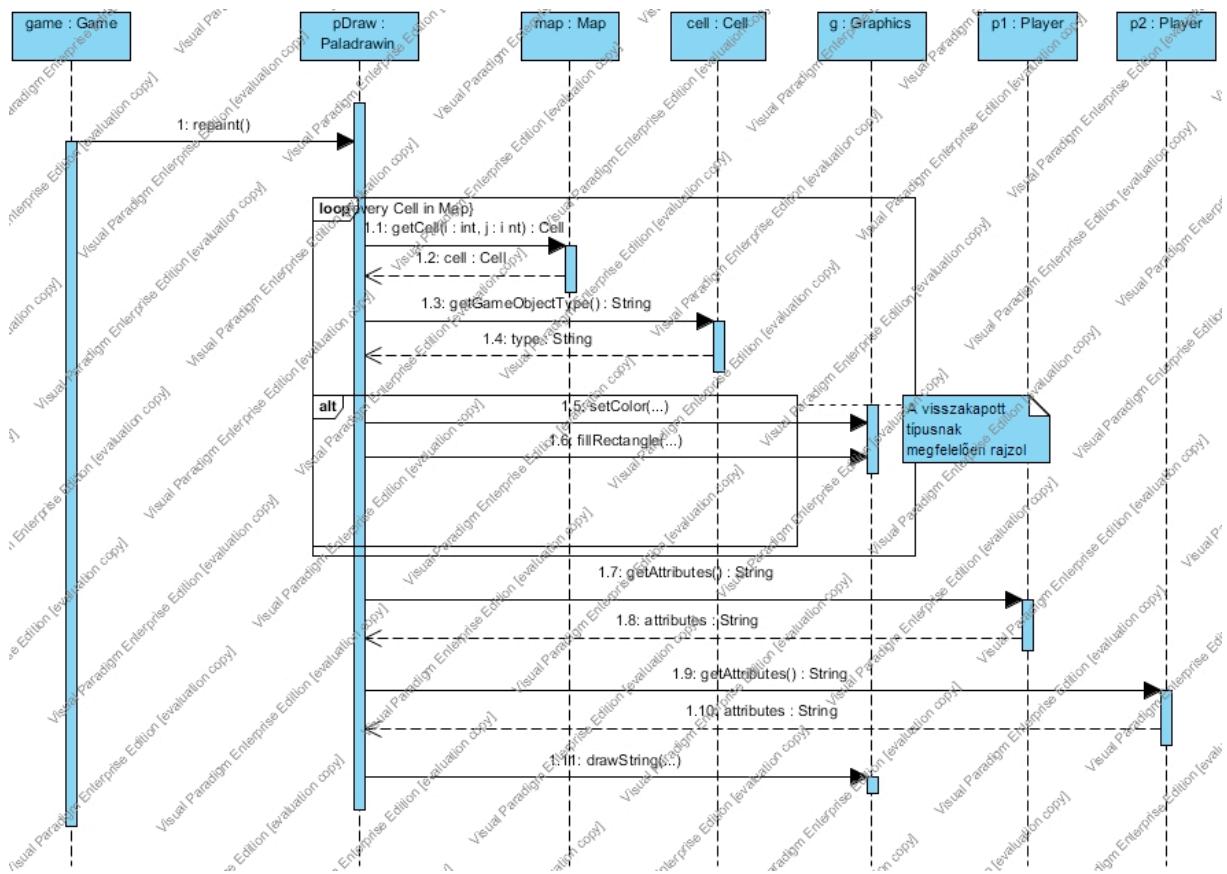
G - ragacsfolt lerakása, O - olajfolt lerakása

Q - játék újraindítása

0.3 Módositott osztálydiagram



0.4 Módosult szekvenciadiagram



13. Grafikus változat beadása

13.1 Fordítási és futtatási útmutató

13.1.1 Fájllista

Fájl neve	Méret	Keletkezés ideje	Tartalom
Cell.java	4600 byte	2015.04.19. 23:07	Cella osztály.
CellType.java	120 byte	2015.04.19. 22:58	CellType enum.
Game.java	16706 byte	2015.05.10. 22:45	Game osztály.
GameException.java	190 byte	2015.04.19. 22:58	GameException osztály.
GameObject.java	799 byte	2015.04.19. 23:01	GameObject interface.
GlueStain.java	1047 byte	2015.04.19. 22:58	GlueStain osztály.
Logger.java	2662 byte	2015.03.20. 22:57	Logger osztály.
Main.java	1289 byte	2015.05.10. 22:45	Main osztály.
Map.java	3476 byte	2015.04.19. 23:01	Map osztály.
OilStain.java	1015 byte	2015.05.10. 22:45	OilStain osztály.
Paladravin.java	3774 byte	2015.05.10. 22:45	Paladravin osztály.
Player.java	6653 byte	2015.05.10. 19:01	Player osztály.
Robot.java	5141 byte	2015.05.10. 22:45	Robot osztály.
Stain.java	976 byte	2015.04.19. 23:01	Stain osztály.
map.txt	949 byte	2015.04.19. 17:55	A térkép fájl.
balage.jpg	2780 byte	2015.05.07. 16:18	Dr. Goldschmidt Balázs arcképe.

13.1.2 Fordítás és telepítés

javac src/phoebe/*.java

13.1.3 Futtatás

cd src
java phoebe.Main

13.2 Értékelés

Tag neve	Munka százalékban
Graics Bence	25.93%
Herold Kristóf	25.51%
Sallai Gyula	24.01%
Verbőczy Kristóf	24.55%

13.3 Napló

Kezdet	Időtartam	Résznevők	Leírás
2015.05.01. 22:00	1,5 óra	Sallai Verbőczy	Grafikus kirajzolásért felelős osztály (Paladrawin) felvétele. Kezdeti implementálása.
2015.05.08. 21:00	3 óra	Verbőczy	Tevékenység: Részletes terv kidolgozása.
2015.05.09. 21:00	3 óra	Graics Sallai Verbőczy	Reset implementálása, billentyűzet kezelése, Paladrawin teljes elkészítése.
2015.05.10. 21:00	2 óra	Graics Herold Verbőczy	Dokumentáció elkészítése.

1. Projektre fordított munkaidő:

Graics Bence	Herold Kristóf	Sallai Gyula	Verbőczy Kristóf	Összesen
47.2 óra	46.45 óra	43.7 óra	44.7 óra	182.05 óra

2. Forrássorok száma:

Szkeleton	922 sor
Prototípus	1754 sor
Grafikus	1884 sor

3. Projekt összegzés:

a. Mit tanultak a projektból konkrétan és általában?

Megismerkedtünk új technológiákkal. A verziókezeléshez Gitet használtunk, így azt mindenképpen mélyebben is meg kellett ismernünk. A diagramokat a projekt felétől kezdve Visual Paradigmban rajzoltuk, ez a másik szoftver, amelyben elmélyedtünk.

Fejlesztőkörnyezetnek Eclipse-t használtunk, megtapasztaltuk az erősségeit és hibáit is. Általánosságban megtanultuk, hogy nem jó az utolsó pillanatban nekiállni bizonyos dolgoknak, és időben el kell kezdeni a munkával foglalkozni, hogy egy-egy új ismeret, gondolat kialakuljon. Saját bőrünkön tapasztaltuk meg a csoportmunka előnyeit és hátrányait. Jobban megismertük egymást, rájöttünk, hogyan érdemes kommunikálni, hogy mindenkből a lehető legjobb teljesítményt hozzuk elő.

b. Mi volt a legnehezebb és a legkönnyebb?

A projekt eleje volt talán a legnehezebb, és a vége felé egyre könnyebb lett. Még sosem ültünk le megtervezni egy egész projektet, ez új volt mindannyiunk számára, de hamar belejöttünk. Nehéz volt elsőre egy jól átgondolt modellt készíteni, nem is sikerült tökéletesen, de miután megvolt, a későbbi munka is sokkal könnyebb lett. Nehéz volt időnként úgy elosztani a munkát, hogy mindenki ugyanannyit dolgozzon. A legnehezebb, vagy inkább legrosszabb, az OpenAmeos modellező program használata volt. Nagyon nehezen kezelhető szoftver, és sokszor meggyűlt vele a bajunk. (Pl.: Az entitások elnevezése osztálydiagramon, szekvenciadiagramon inkonzisztenciákat okozott. Grafikus hibák megjelenése szekvenciadiagramokon stb.)

Legkönnyebb a csapaton belüli kommunikáció volt, hiszen mind a négyen egy szobában lakunk, így nem ment el felesleges idő a megbeszélések szervezésére, levezénylésére. A projekt körülbelül felénél átálltunk az OpenAmeosról a Visual Paradigmra, és utána már a diagram rajzolás is könnyebbé vált.

c. Összhangban állt-e az idő és a pontszám az elvégzendő feladatokkal?

Teljes mértékben. Már említettük, hogy nekünk ez nem volt gond, sok időt megspóroltunkazzal, hogy egy helyen laktunk.

d. Milyen változtatási javaslatuk van?

Szerintünk jó ötlet lenne, ha a csapatok követhetnék hány pontot kaptak az egyes dokumentációkra. (Fel lehetne tölteni a tárgyhonlapra az eredményeket.)

A projekt kezdetekor lehetne ajánlásokat tenni az egyes segédtechnológiák megválasztására. Pl.: Milyen verziókezelő szoftvert érdemes használni, milyen modellező eszközt érdemes használni.

A leadásnál kitöltendő teszteket lehetne módosítani, mert nem minden állt összhangban a teszteléssel az ott szereplő kérdések egy része (pl. a szkeleton tesztelésénél volt olyan kérdés, ami még nem tartozott oda, 2009-es dátum szerepel a tesztes táblázatok fejlécében.)

e, Milyen feladatot ajánlanának a projektre?

Egy autós játékot. Hasonló lehetne, mint az idei, de robotok helyett autók versenyeznének, valós időben. Lehetnének akadályok a pályán, amelyeket ki kell kerülni. Továbbá az autók felvehetnének power-upokat, így több játékossal szórakoztatóbb lenne a játék. A módosításkor bővülhetne a program további power-upokkal.

