Szoftver laboratórium 4.

29 – Fearlesscode

Konzulens:
Bozóki Szilárd

Csapattagok

Dányi Bence	K7YLW5	madbence@gmail.com
Dudás Zsolt	I5QC0Y	ifj.dudas.zsolt@gmail.com
Kern Ádám	HTZ5FL	kernadam@freemail.hu
Kolozsi Tibor Zsolt	QHB625	tibi.kolozsi@gmail.com

2012. május 14.

Tartalomjegyzék

2. Követelmény, project, funkcionalitás

2.1. Követelmény definíció

2.1.1. A program célja, alapvető feladata

A program célja a Continuity nevű flash alapú játék saját implementációjának elkészítése. A játék egy logikai játék, ami egy régi képkirakós játéknak a digitális megvalósítása. A játékmenete során először fel kell vennünk egy kulcsot, aminek a birtokában ki tudunk menni a pálya egy másik részén elhelyezett ajtón. Minden pályán egy és csak egy ajtó, illetve tetszőleges számú kulcs helyezkedik el. A játékmenetről a feladatleírásnál részletesebb leírás olvasható.

2.1.2. Fejlesztőkörnyezet

A fejlesztés Windows 7 és OS X Lion operációs rendszereken, Eclipse illetve NetBeans integrált fejlesztői környezetekben folyik. Mivel a csapatban mindenki kódol is, ezért a kódot valahogyan szinkronban kell tartanunk. Ehhez Git elosztott verziókezelőt alkalmazunk. Mivel bármikor bármi történhet a csapat bármely tagjának a számítógépével, ezért biztonsági, illetve projekt menedzsment célokból is használunk egy központi tárhelyet a verziókezelésre. Erre a célra a Bitbucket (bitbucket.org) szolgáltatását használjuk. Választásunk azért esett erre, mert nem csak tárhelyet tudnak biztosítani a kód számára, hanem wiki rendszerük is van, illetve a hibák jelzésére is nyújtanak integrált szolgáltatásokat, ennek következtében ezeket nagyon kényelmesen, egy helyen tudjuk kezelni. Élve a szolgáltatásokkal a dokumentációt ebben a wiki rendszerben készítjük el, amiből egy alkalmazás segítségével Latex formátumú dokumentációt készítünk.

2.1.3. A futtatáshoz szükséges környezet

Az Irányítástechnika és Informatika Tanszék (továbbiakban a 'megrendelő') azt a követelményt tűzte ki, hogy az alkalmazásnak a Hallgatói Számítógép Központ (továbbiakban a 'HSZK') számítógépein kell futnia. Ennek megfelelően az alkalmazás Java Runtime Environment (továbbiakban 'JRE') 1.6 verziószámú futtatókörnyezet a feltétele az alkalmazás elindításának. Mivel az alkalmazás egy vizuális játék, ezért a használathoz a számítógéphez kapcsolt megjelenítő eszköz (monitor) illetve adatok bevitelére szolgáló billentyűzet szükséges. Az alkalmazás billentyűzetről vezérelhető, ezért a játék maximális élményéhez egér használata nem szükséges.

2.1.4. Minőségi tényezők

Újrafelhasználhatóság A szoftver teljes egészét objektum orientáltan készítjük el, azért, hogy a maximális újrafelhasználhatóságot elérjük. A hibaminimalizálás és későbbi változtatások miatt szem előtt tartjuk a DRY (don't repeat yourself) eszmét, vagyis úgy tervezünk, hogy minden funkció mindenhol csak egyszer szerepeljen így nem keletkezik probléma abból, ha egy funkciót módosítanunk kell, hiszen ez mindenhol ugyanazt a hatást fogja elérni.

Rugalmasság és teljesítmény Azért, hogy a játékmenetet a későbbiekben könnyen módosítani tudjuk, mindent paraméteresen egy központi helyen fogunk meghatározni, hogy a szoftver használata során a felhasználó élményt finomhangolni tudjuk, ezzel maximalizálni a megrendelő elégedettségét. Természetesen optimális megoldásokra törekszünk, de teljesítmény problémák nem lesznek, hiszen az alkalmazás semilyen magas számításigényű funkciót sem valósít meg.

Felhasználhatóság A játékmenet újszerűsége miatt minimális tájékoztatást igényel a felhasználó, hogy átlássa az alkalmazást, de ez legrosszabb esetben is fél percet vesz igénybe, hiszen a játékmenet az alkalmazás teljes futásideje alatt megegyezik. Ezek következtében csak alapvető felhasználói tudás szükségeltetik.

A Szoftver minősítése A szoftver funkcionalitását, illetve a kódot unit tesztekkel, míg a végső futást, a játékmenetet felhasználó tesztekkel, valamint integrációs teszttel kívánjunk minőségbiztosítani. Hogy ne csak az alkalmazás, hanem a kód is minőségi legyen GRASP irányelveket követünk, hogy SOLID kódot kapjunk.

Kibocsátás A szoftver kibocsátásra, illetve a megrendelő általi átvételre az ütemtervben meghatározott időben kerül sor. Mivel a projekt, illetve a megrendelés oktatási célokat szolgál ezért az átvételt követően a világhálón is elérhető lesz, valamint módosítás nélkül szabadon terjeszthető is.

2.2. Project terv

2.2.1. Mérföldkövek

A fejlesztés során három fontos mérföldkövet különböztetünk meg, ezek elérésekor nem csak a határidőnek megfelelő dokumentációt készítjük el, hanem a fejlesztés megfelelő szakaszához tartozó programot is.

Szkeleton Az első lépcsőfok a szkeleton előállítása, ebben a változatban pusztán az általunk megalkotott modell helyességét vizsgáljuk, hogy az a kitűzött feladat modelljét alkotja-e. Minden - a végső változatban jelen lévő - business objektum szerepel a programban, azonban ezeknek csak az interfésze definiált, algoritmusokat nem implementálnak a metódusaik, azok csak tesztelési céllal a saját nevüket kiírják a standard kimenetre, majd meghívják a szolgáltatás által megkívánt metódusokat. Esetenként szükség lehet elágazásokra, ilyenkor a standard bemenetről befolyásolható a működés. A szkeleton célja, hogy az analízis modell részben megalkotott forgatókönyvekhez tartozó szekvenciadiagramokat tesztelje. Különösen fontos fázis a project életében, gondos tervezést igényel, cserébe könnyen felfedi a modell hiányosságait, esetleges hibáit.

Proto A prototípus mérföldkő elérésekor minden algoritmus teljes, így a program teljes egészében tesztelhető és működőképes, mivel azonban grafikus felületet nem tartalmaz, így természetesen elkülönül a megjelenítéstől, ezáltal a fölötte elhelyezkedő GUI könnyen cserélhető, fejleszthető.

Grafikus A grafikus változat a program minden tekintetében teljes és végleges verziója. Fejlesztése során - mivel a működési logika elkészült - leginkább a könnyű kezelhetőségre, ergonómikus kialakításra kell fókuszálni, a felhasználói élmény fokozására ezek segítségével, ezzel biztosítva kellemes kikapcsolódást a felhasználónak.

2.2.2. Határidők

A projekt elkészítése során szinte minden héten fontos határidő áll előttünk, ezek betartása a feladat szempontjából kritikus prioritású, mivel az esetleg felhalmozódott feladatokat egyre nehezebb egységnyi idő alatt elvégezni.

#	**Határidő**	**Feladat**
1	február 10.	Regisztráció
2	február 20.	Követelmény, projekt, funkcionalitás
3	február 27.	Analízis modell kidolgozása 1.
4	március 5.	Analízis modell kidolgozása 2.
5	március 12.	Szkeleton tervezése
6	március 19.	Szkeleton **program**
7	március 26.	Prototípus koncepciója
8	április 2.	Részletes tervek
9	április 9.	
10	április 16.	Prototípus **program**
11	április 23.	Grafikus felület specifikációja
12	április 30.	
13	május 7.	Grafikus változat **program**
14	május 11.	Összefoglalás

2.2.3. Csapat

A csapat négy főből áll, a feladatokat lehetőség szerint igyekszünk egyenlően elosztani, ügyelve arra, hogy senki se legyen egy-egy feladat egyedüli felelőse, így egy esetleges betegség, vagy akár más problémák miatt munkaképtelen csapattag nem tartja föl a folyamatban lévő feladatokat.

Név	**Személyes preferencia**
Dudás Zsolt	Kód, teszt
Kern Ádám	Kód, UML
Kolozsi Tibor Zsolt	Dokumentáció, kód, teszt
Dányi Bence (csapatvezető)	Dokumentáció, kód

2.2.4. Kommunikációs modell

A hatékony munkavégzés egyik fontos alappillére a hatékony és gördülékeny kommunikáció és információcsere, e nélkül a feladatok elvégzése káoszba fullad, ezért nekünk is érdekünk az optimális csatornák megtalálása.

Verziókezelés Több ember közös projectben való részvétele esetén fontos, hogy mindenki friss változattal rendelkezzen a kódbázisról, konkurrens kódszerkesztés is a lehető legkisebb kellemetlenségekkel járjon, valamint hogy a már elkészült kód élettörténe tetszőlegesen visszakövethető legyen bármelyik előző állapotára, a tárolása pedig a maximális biztonság érdekében redundáns legyen.

Ennek fényében döntöttünk a Git verziókezelő rendszer mellett, ami egy gyors, elosztott verziókezelő rendszer, népszerűségét és hatékonyságát mi sem mutatja jobban, mint hogy a Linux kernel is ezen keresztül van menedzselve.

A számtalan interneten fellelhető ingyenes megoldásból mi a BitBucket szolgáltatását választottuk, a privát tárolók létrehozása, és a mellé elérhető wiki rendszer miatt, ami a következő pontban van részletezve.

Wiki A project nagyobb hányada dokumentációk készítéséből áll, így ehhez is egy hatékony rendszert kellett találni, végül a BitBucket wiki rendszere mellett tettük le a voksunk. Egyszerű szintaxisa miatt bármikor könnyen szerkeszthető, a többi csapattag számára bármikor elérhető, és érdekességképpen maga a wiki rendszere is egy git tároló.

Levelezési lista Aszinkron kommunikációs forma, mivel nem igényli minden csapattag internetes jelenlétét, leginkább a későbbi megbeszélések megszervezése, valamint olyan információk cseréje zajlik, ami a wiki rendszer keretei közé nem fér, de nem szabad, hogy mégis feledésbe merüljön.

Szinkron kommunikáció A kód illetve a dokumentáció konzisztenciája mellett a másik legfontosabb dolog a csapattagok közti közvetlen párbeszéd hatékony lebonyolítása.

Skype Konferenciabeszélgetések lebonyolítására rendkívül alkalmas rendszer, így amennyiben fizikailag nem tudunk egy helyiségben lenni, akkor a megbeszélések ezen a platformon történnek.

Egyéb A legfontosabb döntések meghozásához mindenképpen szükséges a személyes kontaktus, ilyenkor lehet csak a csapat tagjaitól a maximális koncentrációt elvárni.

2.2.5. Átadás

A Határidők szekcióban szereplő alkalmakkor papír alapú dokumentációkat nyújtunk be a formai előírásoknak megfelelően, ezen felül a Mérföldkövek fejezetben leírt fázisoknál működő program is bemutatásra kerül, amit a labor számítógépein teszünk meg.

2.3. Feladat leírás

2.3.1. Eredeti feladatkiírás

Folytonosság (continuity)
A feladat az alábbi játék elkészítése: http://continuitygame.com/playcontinuity.html

2.3.2. Részletes feladatleírás

A megvalósítandó feladat egy egyszemélyes logikai játék, ami a Folytonosság (Continuity) nevet viseli.

Ebben a játékban a játékos feladata, hogy felszedje a kulcsot vagy kulcsokat és ezután a pálcikaembert eljuttasa az ajtóhoz. Csak akkor számít teljesítettnek az adott pálya, hogyha a játékos mielött az ajtóhoz ért volna felszedi az összes kulcsot, ha nem így tesz akkor az ajtóhoz érve nem történik semmi, az ajtó zárva marad, ha pedig sikerült az összes kulcsot összegyűjteni, és ezután az pálcikaembert elnavigálni az ajtóhoz, akkor sikerült megcsinálni a pályát.

A játékos három parancsot tud adni a pálcikaembernek. Jobbra vagy balra haladhat, valamint ugorhat. A vízszintes mozgás és a függőleges teljesen függetlenek egymástól, tetszőlegesen kombinálhatóak, tehát - a fizika törvényeit meghazudtolva - akár a levegőben is irányt válthatunk.

A pálcikaember csak egy adott területen, a játéktéren mozoghat. A játéktér vízszintes illetve függőleges vonalakból kialakított zárt alakzatokból áll, amik belseje tömör, ezeken a játékos nem tud áthatolni. Egy pálya tipikus kialakítása során a játékos olyan elemekkel találkozik, mint egy lépcső, vagy egy platform, de természetesen bármilyen más konstrukció is elképzelhető. Ha a pálcikaember lába alatt nincsen talaj, akkor leesik.

A játékos nem csak a pálcikaembert tudja irányítani, hanem a pályablokkokat is. Minden pálya pályablokkokra van bontva. A pályablokkok tartalmazzák a játékteret felépítő elemeket, entitásokat (fal, ajtó, kulcs). Egy pálya széltében és magasságában 1-nél nagyobb egész szamú blokkból állhat, a pálya nehézségi szintjétől, sorszámától függően, minnél nehezebb egy pálya annál több pályablokk építi fel, illetve a játéktér bonyolultsága is nő, értve ezalatt azt, hogy nehezebb megszerezni a kulcsot/kulcsokat, és nehezebb eljutni az ajtóhoz. A pályán található blokkok közül az egyik mindig üresblokk, ami nem tartalmaz játékteret felépítő elemeket, illetve mivel teljesen üres, meg se jelenik, nincs kerete. A játékos a játék során valójában az üresblokkot mozgatja. A blokkok mozgatása egy egyszerű csere, mindig egy üresblokk és egy pályablokk cserél helyet. Miközben a játékos a kulcsot, vagy éppen már az ajtót próbálja elérni kénytelen mozgatni a blokkokat, hogy kialakítsa a pálcikaember útját a játéktéren. A blokkok mozgatásával olyan utat kell kialakítani a

játékosnak, amin a pálcikaember végig tud haladni. A játék bármelyik pillanatban megállítható, ilyenkor van lehetőség a pályablokkok mozgatására. Ha a pálcikaember olyan pozícióban van, hogy több pályablokk határán áll, és így átlóg egyik pályablokkból a másikba, akkor ezeknek a blokkoknak a mozgatása nem megengedett. Két blokk között az átjárás megengedett pontosan akkor, ha szélük tökéletesen illeszkedik, azaz ha az egyik blokknak a másik blokkal illeszkedő egy pontján fal található, akkor a másik blokk azonos pontján is falnak kell lennie (ha pedig nem található az érintkező ponton fal, akkor kötelezően a másik oldal sem tartalmazhat azon a ponton falat), ez a feltétel igaz mind az egymás alatt-fölött illetve egymás mellett található blokkokra.

Ha a játékos valami olyan hibát vét, amiből nem létezik értelmes folytatása a játéknak, és eddig még nem szedte fel a kulcsot, akkor a pálcikaember a kezdőpozícióba kerül vissza, ha pedig már szedett fel kulcsot, akkor az utolsó felvett kulcs poziciójába kerül vissza. Ezen felül, hogyha a játékos úgy érzi, hogy valami olyat tett, ami miatt az adott pályát már nem tudja végigcsinálni, akkor dönthet úgy, hogy újrakezdi az adott pályát, ilyenkor is visszakerül a pálcikaember a kezdőpozícióba.

A játék célja az egymás után következő pályák végigjátszása. A játék akkor számít befejezettnek, hogyha a játékos az összes pályát végigjátszotta. A pályák nehézsége a játék előrehaladásával egyre nő, a játék így igyekszik fenntartani a játékos érdeklődését a játék kezdetétől a legvégéig.

A játékos teljesítménye nem kerül rangsorolásra, mivel a játékban nincsenek pontok. Egy adott pálya végigvitele nincs időhöz kötve, bármennyi idő rendelkezésre áll.

2.4. Szótár

ajtó Minden pályán van egy piros téglalap, amit el kell érni a kulcsok felszedése után a pálya teljesítéséhez.

átjárás Két szomszédos pályablokk esetén átjutás egyikről a másikra.

blokk A játéktér egyenlő négyzet alakú részekre, blokkokra van osztva. Egy blokk helyén lehet pályablokk vagy semmi. Ilyenkor üres blokknak hívjuk.

blokk mód Ilyen módban lehet mozgatni a pályablokkokat az üres blokk helyére, viszont a pálcikaembert nem lehet ilyenkor irányítani. Az idő ekkor a pálcikaember számára megáll.

blokk mozgatás Bármely üres blokk melletti blokkot az üres blokk helyére lehet vinni.

illeszkedő blokkok Két blokk akkor illeszkedik, ha egymás mellett vannak, és érintkező oldalaikon pontosan akkor van fal, ha a másik blokk oldalának ugyanazon részén is fal van.

játék megnyerése A játékot akkor nyeri meg a játékos, ha eljut az ajtóig a legutolsó pályán.

játék mód Ebben a módban lehet a pálcikaembert mozgatni, viszont a pályablokkokat nem.

kezdőpozíció A pályának azon része, ahol a pálcikaember a pálya elkezdésekor áll.

kiesik a pályáról Akkor történik a pálcikaemberrel, ha olyan helyen mozog ki egy pályablokk alján, ahol nincs a pályablokkhoz csatlakoztatva semmi (nincs alatta pályablokk, vagy nem illeszkedik)

kulcs Minden pályán elhelyezett egy vagy több piros tárgy, melyet fel kell venni, és csak azután lehet átmenni az ajtón.

kulcs pozíció A pályának azon része, ahol az egyes kulcsok vannak/voltak.

leesik Ha a pálcikaember olyan helyre mozog, ahol nincs alatta talaj akkor lefele mozog(esik) mindaddig, amíg platformig nem ér vagy ki nem esik a pályáról.

lépcső Rövid platformok lépcsőt alakítva helyezkednek el egymás mellett melyeken könnyű felfelé vagy lefelé haladni.

menü A játék indításakor megjelenő képernyőfelület. Itt különböző menüpontokból választhatunk mint pl. Új Játék.

nyitott ajtó Minden kulcs felszedése után az ajtó zártból nyitottá válik, át lehet rajta menni, azaz be lehet fejezni a pályát.

pálcikaember A játék főhőse, őt irányítja a játékos.

pálya A játék egy szintje(egysége). Az ajtó elérésekor számít befejezettnek és kezdődik egy újabb.

pályablokk A pálya részegysége, minden üres blokk melletti pályablokk mozgatható az üres blokk helyére. Az összes pályablokk egyforma méretű.

pálya újrakezdés Visszaugrás az adott pálya elejére, a kezdőállapotba.

pálya nehézség A pályák végigjátszásához szükséges logikai készség relatív mértéke.

pálya végigjátszás A pálya kezdetétől az ajtó eléréséig tartó játék.

platform Egymás fölött/alatt elhelyezkedő vagy egymással egyszinten levő vízszintes felületek amelyeken a pálcikaember mozoghat.

újjáéledés Ha a pálcikaember kiesik egy pályablokkról úgy, hogy annak nincs folytatása akkor megjelenik a kezdőpozícióban vagy kulcs felvétele esetén a legutolsó felvett kulcs pozíciójában és a játék onnan folytatható.

újrakezdés Ha a játékos elakadt, akkor lehetősége van az adott pályát vagy akár az egész játékot újrakezdeni az elejétől. Ehhez vissza kell lépni a menübe.

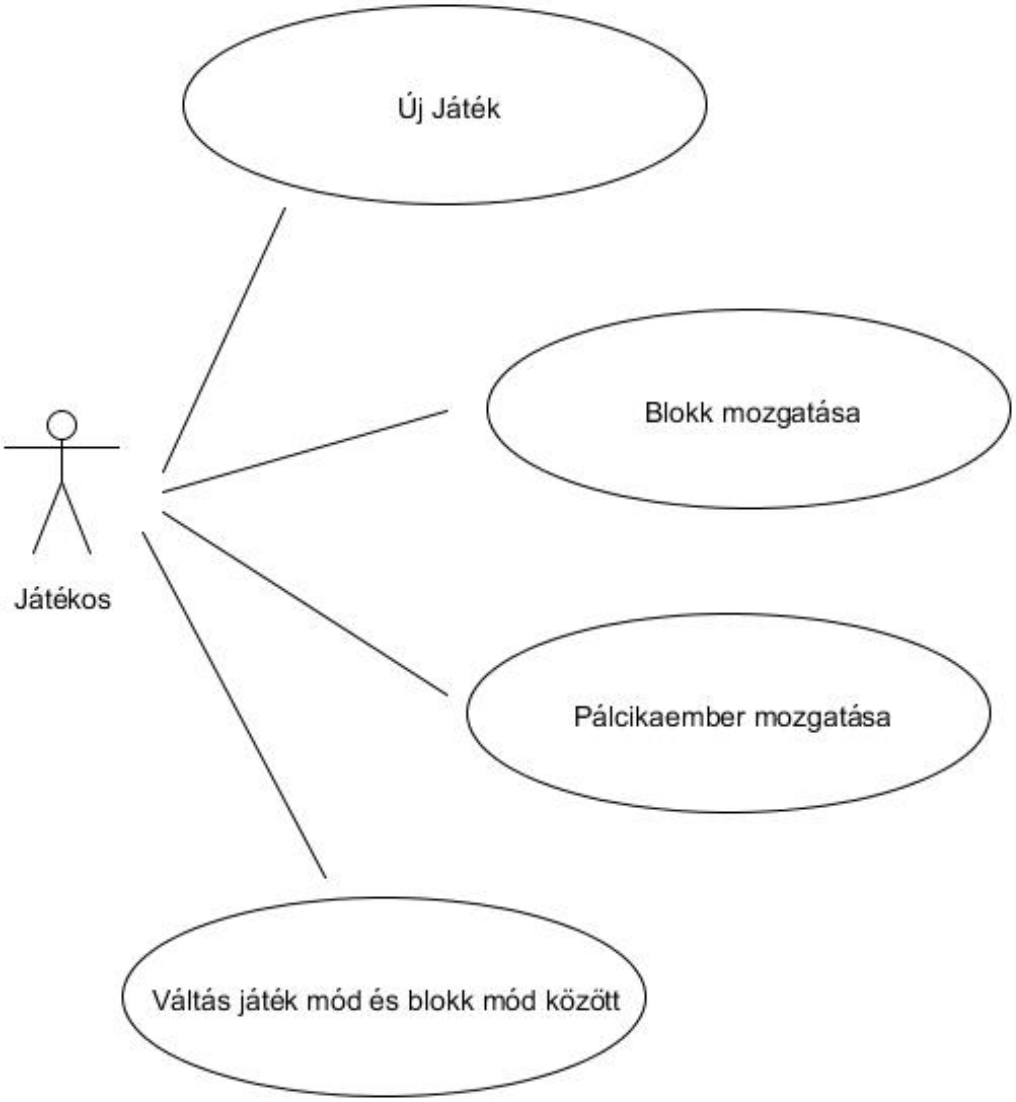
üres blokk Egy pályablokk méretű hely ahol nincs semmi, ide mozgatható egy pályablokk.

zárolt blokk Ha a pálcikaember két (vagy több) blokk között áll, akkor azok a blokkok addig nem mozgathatók.

zárt ajtó Az ajtón addig nem lehet átmenni(tehát zárt), amíg van nem felszedett kulcs a pályán.

2.5. Essential use-case-ek

2.5.1. Use-case diagram



2.5.2. Use-case leírások

Use-case neve	**Új Játék**
Rövid leírás	Új játék kezdése
Aktorok	Játékos
Forgatókönyv	1. A játékos kiválasztja a menüben az "Új játék indítása" címkét és rákattint. 2. Az első pálya töltődik be, a pálcikaember a kezdőpozícióban áll.

Use Case neve	**Pálcikaember mozgatása**
Rövid leírás	Emberke mozgatása
Aktorok	Játékos
Forgatókönyv	Játék módba kapcsolunk.A pálcikaembert jobbra/balra irányítjuk vagy/és ugrunk vele.

Use Case neve	**Blokk mozgatása**
Rövid leírás	Játéktér-blokkok mozgatása
Aktorok	Játékos
Forgatókönyv	Blokk módba kapcsolunk. Üresblokk helyére valamelyik szomszédos pályablokkot bemozgatjuk.

Use Case neve	**Váltás játék mód és blokk mód között**
Rövid leírás	Játéktér-blokkok mozgatása
Aktorok	Játékos
Forgatókönyv	Játék módból átváltunk blokk módba vagy fordítva.

2.6. Napló

Kezdet	Időtartam	Résztvevők	Leírás
2012.02.15. 9:00	1 óra	Dányi Dudás Kern Kolozsi	Első heti feladatok megbeszélése, feladatok kiosztása: Dányi: Projekt terv Dudás: Követelmény definíció Kern: Use case-ek Kolozsi: Feladatleírás
2012.02.15. 19:00	2 óra	Kolozsi	Feladatleírás elkészítése
2012.02.15. 21:00	2 óra	Dányi	Project terv elkészítése
2012.02.15. 16:00	2 óra	Kern	Use-Case-ek
2012.02.17. 21:00	1 óra	Dányi Kern	Eddig elkészült anyagok értékelése, újabb feladatok kiosztása: Dányi: Wiki2LaTeX átalakító megírása Kern: Szótár megírása
2012.02.17. 23:00	2.5 óra	Dányi	Wiki2LaTeX átalakító script megírása
2012.02.18. 15:00	1 óra	Dányi Kolozsi	Feladatleírás véglegesítés
2012.02.18. 15:15	1 óra	Dudás	Követelmény leírása
2012.02.18. 21:00	2 óra	Dányi Dudás Kern Kolozsi	Feladatok véglegesítése, következő meeting hétfő 9:00 E épület

3. Analízis modell kidolgozása 1.

3.1. Objektum katalógus

3.1.1. Game

A játék fő mozzanatait (pálya betöltés, játék megnyerése) kezeli, és kezdetben inicializál minden objektumot. Felelőssége a pálya betöltése (és inicializálása).

3.1.2. Player

A játékos irányította pálcikaembert reprezentálja. Számontartja a megszerzett kulcsok számát, valamint, hogy mely blokkokban van jelen.

3.1.3. PlayField

Ez az objektum tárolja el a pályán található blokkokat, amik közül pontosan egy darab üres. Az ő felelőssége mozgatni őket, és ellenőrizni a szabályosságát a mozgatásnak, és a játékos által felvett utolsó kulcs pozíciójának (vagy ha ilyen nincs, a kezdőpozíciónak) a tárolása, és a játékos visszahelyzése ebbe a pontba, amennyiben szükséges.

3.1.4. EmptyBlock

Egy üres blokkot reprezentál, a játékos semmilyen körülmények között nem tud rálépni, illetve nem tartalmaz kulcsot, ajtót, kezdőpontot, vagy falat. A PlayField ezt a blokkot "mozgatja" a játék során (valójában egy sima blokkot tol be a helyére, de technikailag mindig ez a blokk mozog)

3.1.5. FilledBlock

Egy sima blokkot reprezentál, tárolja a benne fellelhető objektumokat (Key, Door, Wall, SpawnPoint), valamint a rajta található játékost (amennyiben van rajta). Felelőssége tájékoztatni a rajta található objektumokat a velük történt eseményekről (ütközések, kilépés a blokkból).

3.1.6. Key

Egy kulcsot reprezentál, amit a játékosnak föl kell szednie. Amennyiben játékossal találkozik, és még nem szerezte meg a kulcsot (ez a saját belső állapota), úgy értesíti a játékost a fölvételről.

3.1.7. Door

A pályán található egyetlen ajtót reprezentálja, amit a játékosnak ki kell nyitnia, majd eljutni hozzá. Ha játékossal találkozik, a Player objektumtól elkéri a megszerzett kulcsok számát, és értesíti a PlayField objektumot a pálya megnyeréséről, ha ez a szám egyezik a saját maga által tárolt számmal (összes kulcs száma).

3.1.8. SpawnPoint

Minden pálya indulásakor a játékos ezen a pozíción kezd, és mindig ide helyeződik vissza, ha leesik a pályáról. Szerepét egy Key objektum veszi át, amint a játékos megszerez egyet.

3.1.9. Wall

A pálya azon szilárd részét (ez lehet ténylegesen egy fal, de egy platform, vagy akár lépcső is) reprezentálja, amin a játékos nem tud áthaladni. Amennyiben a játékos összeütközik vele, értesíti a játékost.

3.2. Osztályok leírása

3.2.1. Block (absztrakt)

- **Felelőssége**
  A szomszédos blokkokat nyilvántartja, és biztosítja a közöttük lévő kölcsönös kapcsolatot, valamint értesíti a rajta található objektumokat a játékos interakciójáról (rájuk lép, kilép a blokkból).
- **Ősosztályok**
  Nincs
- **Interfészek**
  Nincs
- **Attribútumok**
  - **neighbours** A blokk szomszédjainak listája.
  - **entities** A blokkon található objektumok (és pozíciójuk).
  - **player** A játékos referenciája (és pozíciója).
- **Metódusok**

- **setPlayer(player, pos)** Beállítja a játékos referenciáját és pozícióját az adott blokkon.
- **addEntity(position, entity)** Hozzáad egy objektumot a blokkhoz a megadott helyen.
- **getNeighbour(dir)** Visszaadja a megadott irányban található szomszédot. A dir egy szám, 0 jelenti az északot, és az óramutató járásával megegyező irányban történik a számozás.
- **setNeighbours(neighbours)** Felülírja az eddigi szomszédait, és a kapottakat állítja be.
- **setNeighbour(neighbour, dir, bool)** A megadott szomszédot beállítja a megfelelő irányban, a harmadik paraméter függvényében (igaz/hamis) viszont beállítja a szomszédságot. Ez a harmadik paraméter a végtelen ciklusok elkerülésére van bevezetve.
- **abstract processCollisions()** Lefuttatja az ütközésdetektálást.
- **abstract checkBorders()** A blokkok közötti mozgást kezeli le.

3.2.2. Door

- **Felelőssége**
  A Door értesíti a PlayFieldet, ha a pályának vége, ezt akkor teszi, ha a játékosnál lévő kulcsok száma megegyezik az ajtó kinyitásához szükséges kulcsok számával.
- **Ősosztályok**
  Entity
- **Interfészek**
  Nincs
- **Attribútumok**
  - **requiredKeys** A kinyitáshoz szükséges kulcsok száma.
- **Metódusok**
  - **meetPlayer(player)** Ha a játékosnál lévő kulcsok száma megegyezik a szükséges kulcsok számával, értesíti a PlayFieldet, hogy a pályának vége.

3.2.3. EmptyBlock

- **Felelőssége**
  Az üres blokkot reprezentálja, gyakorlatilag a Block osztály default (üres) implemetációja.
- **Ősosztályok**
  Block
- **Interfészek**
  Nincs
- **Attribútumok**
  Nincs
- **Metódusok**
  - **processCollisions()** Mivel üres blokk, semmi sincs benne, így ez a metódus nem csinál semmit.
  - **checkBorders()** Mivel üres blokk, semmi sincs benne, így ez a metódus nem csinál semmit.

3.2.4. Entity (absztrakt)

- **Felelőssége**
  Az Entity egy blokkon található statikus (nem mozgó) objektumot reprezentál, minden leszármazottnak kötelessége viselkedést definiálni a játékossal való találkozásra.
- **Ősosztályok**
  Nincs
- **Interfészek**
  Nincs
- **Attribútumok**
  - **playField** PlayField referencia, a leszármazottak a viselkedés leírásának megkönnyítése érdekében felhasználhatják.
- **Metódusok**
  - **abstract meetPlayer(player)** Kötelezően implementálandó metódus, a játékossal való találkozás forgatókönyvét írja le.

3.2.5. FilledBlock

- **Felelőssége**
  A sima blokkot reprezentálja, kezeli a benne található objektumokat (objektumok interakciója a játékossal)
- **Ősosztályok**
  Block
- **Interfészek**
  Nincs
- **Attribútumok**
  Nincs
- **Metódusok**
  - **processCollisions()** Végignézi az összes objektumot, hogy ütközik-e a játékossal, ha pedig igen, meghívja a meetPlayer metódusukat.

3.2.6. Game

- **Felelőssége**
  A Game osztály felelőssége betölteni, és inicializáltatni a pályát.
- **Ősosztályok**
  Nincs
- **Interfészek**
  Nincs
- **Attribútumok**
  - **playField** Az aktuális pálya referenciája
- **Metódusok**
  - **start()** Elindítja a játékot (betölti a megfelelő pályát)
  - **loadNextLevel()** Betölti a következő pályát

3.2.7. Key

- **Felelőssége**
  A Key számon tartja magáról, hogy fölvették-e, és a játékossal való találkozása esetén ennek függvényében változik a viselkedése.
- **Ősosztályok**
  Entity
- **Interfészek**
  Nincs
- **Attribútumok**
  - **isObtained** Fel lett-e véve már a kulcs.
- **Metódusok**
  - **meetPlayer(player)** Ha a kulcs fel lett véve, semmi sem történik, egyébként megnöveli a játékosnál lévő kulcsok számát, a saját állapotát pedig megváltoztatja (felvettre).

3.2.8. Player

- **Felelőssége**
  A Player számon tartja magáról, hogy mely blokkokban van jelen, valamint a nála lévő kulcsok számát.
- **Ősosztályok**
  Entity
- **Interfészek**
  Nincs
- **Attribútumok**
  - **obtainedKeys** A megszerzett kulcsok száma.
  - **activeBlocks** A játékos jelenleg ezekben a blokkokban van jelen. Ha ez több egynél, akkor a mozgatás nem feltétlenül lehetséges.
  - **speed** A játékos jelenlegi sebessége, ebből számítható a következő pozíciója.
- **Metódusok**
  - **addKey()** Megnöveli a nála lévő kulcsok számát.

- **enterBlock(block)** A játékos belép a megadott blokkba.
- **leaveBlock(block)** A játékos kilép a megadott blokkból.
- **move(dir)** A játékos sebességét a megadott irányban megnöveli. Ténylegesen nem mozgatja a játékost, csak jelzi a mozgás irányát.
- **getObtainedKeys()** A játékos által birtokolt kulcsok számát adja vissza.
- **getActiveBlocks()** A játékos által elfoglalt blokkokat adja vissza.

3.2.9. PlayField

- **Felelőssége**
  A PlayField felelős a rajta található blokkok mozgatásáért, ismeri az aktuális játékmódot (blokk/játék), és tudja hova kell áthelyeznie a játékost (ezért is ő a felelős), ha kiesik a pályáról.
- **Ősosztályok**
  Nincs
- **Interfészek**
  Nincs
- **Attribútumok**
  - **blocks** A rajta található blokkok (és pozíciójuk).
  - **game** Egy Game-re mutató referencia a kommunikációhoz.
  - **player** A játékosra mutató referencia.
  - **spawnPosition** A játékos aktuális újraéledési helye, ha leesik a pályáról.
  - **blockMode** Az aktuális játékmód (blokk/játék)
- **Metódusok**
  - **setPlayer(player)** Beállítja a játékos pozícióját.
  - **addBlock(position,block)** Hozzáadja a megadott blokkot a megadott pozícióban az aktuális pályához.
  - **setSpawnPoint(entity)** Beállítja, hogy az újraéledés melyik objektumon történjen.
  - **toggleMode()** Játékmódot vált (blokk/játék).
  - **win()** Meghívja a Game loadNextLevel() metódusát.
  - **move(block,dir)** A megadott blokkot elmozdítja a megadott irányba.

- **resetPlayer()** A játékost visszahelyezi az elmentett kezdőpozícióba.
- **tick()** A játékos ténylegesen csak ebben a metódusban mozdul meg. Ütközésdetektálás és blokkok közötti mozgás vizsgálata után beállítja az új pozícióját. Részletesen lásd a szekvenciadiagramon.

3.2.10. PlayFieldBuilder

- **Felelőssége**
  A PlayField összerakásáért felelős, elkészíti a blokkokat, elhelyezi rajtuk az objektumokat, a játékost, a blokkokat összerendezi a PlayField objektumban.
- **Ősosztályok**
  Nincs
- **Interfészek**
  Nincs
- **Attribútumok**
  Nincs
- **Metódusok**
  - **createPlayField(spec)** A paraméterben megadott specifikációnak megfelelő PlayFieldet összerakja, majd a hívónak visszaadja.

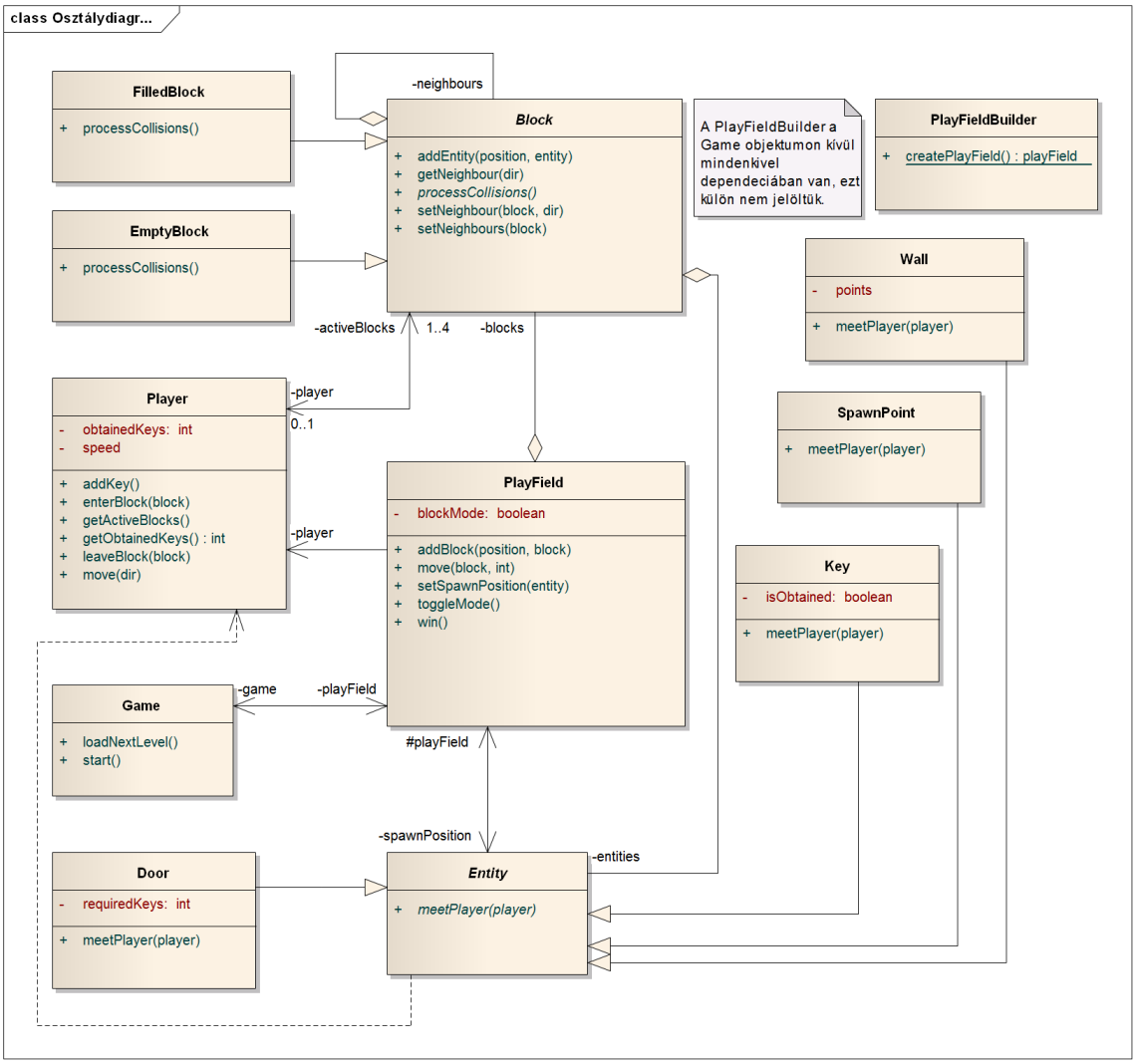
3.2.11. SpawnPoint

- **Felelőssége**
  Az Entity absztrakt osztály default (üres) implementációja, nem reagál a játékossal, viszont a játékost ide lehet visszahelyezni, ha leesik a pályáról.
- **Ősosztályok**
  Entity
- **Interfészek**
  Nincs
- **Attribútumok**
  Nincs
- **Metódusok**
  - **meetPlayer(player)** Üres metódus, nem reagál a játékossal.

3.2.12. Wall

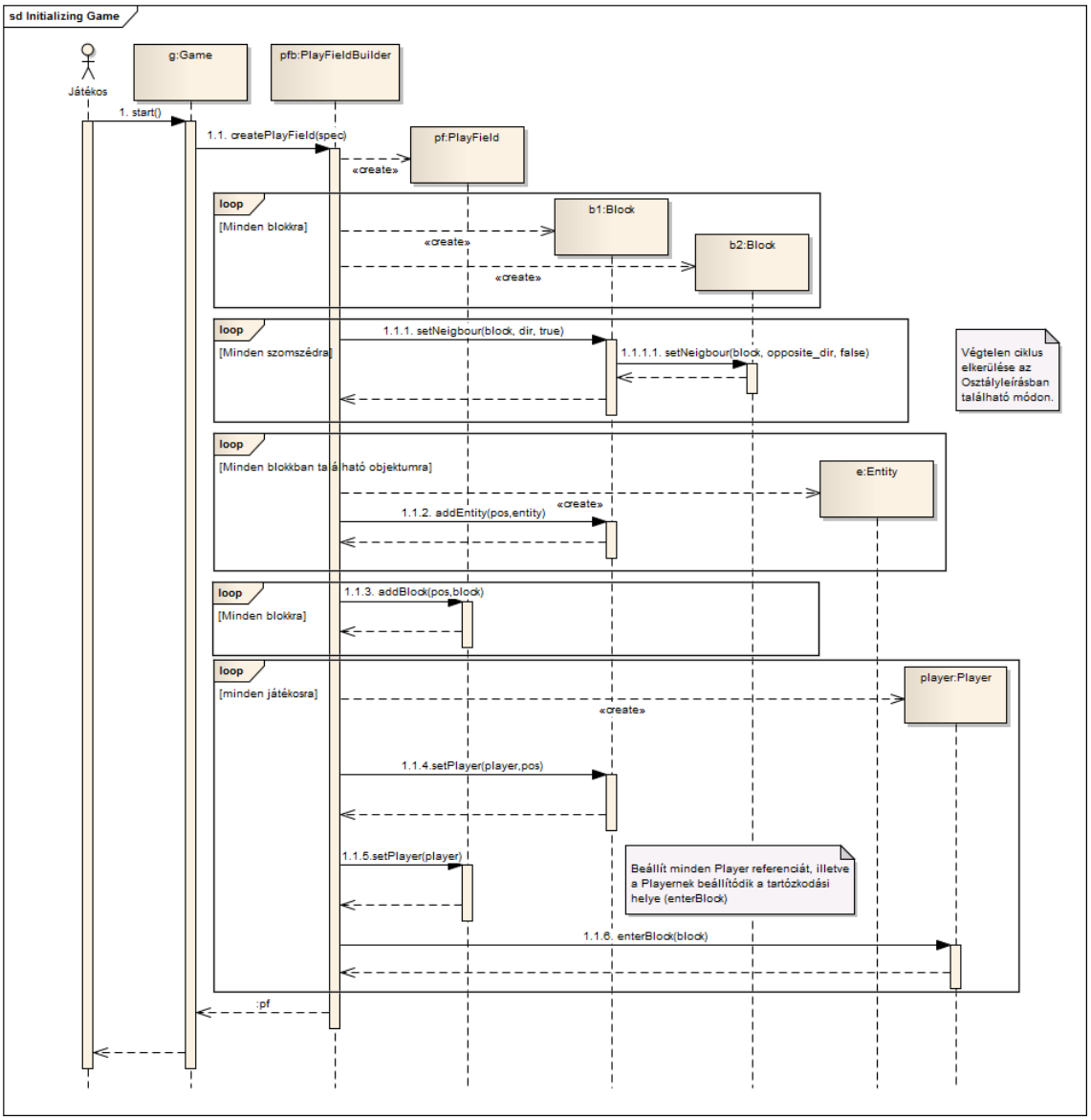
- **Felelőssége**
  A Wall egy olyan poligon, ami megállítja a játékost, ha összeütközik vele, a feladata ennek a viselkedésnek a helyes elvégzése.
- **Ősosztályok**
  Entity
- **Interfészek**
  Nincs
- **Attribútumok**
  - **points** A falat reprezentáló poligon pontjai.
- **Metódusok**
  - **meetPlayer(player)** Ha összeütközik a játékossal, akkor megállítja a mozgásban (vízszintes, vagy függőleges irányban)

3.3. Statikus struktúra diagramok

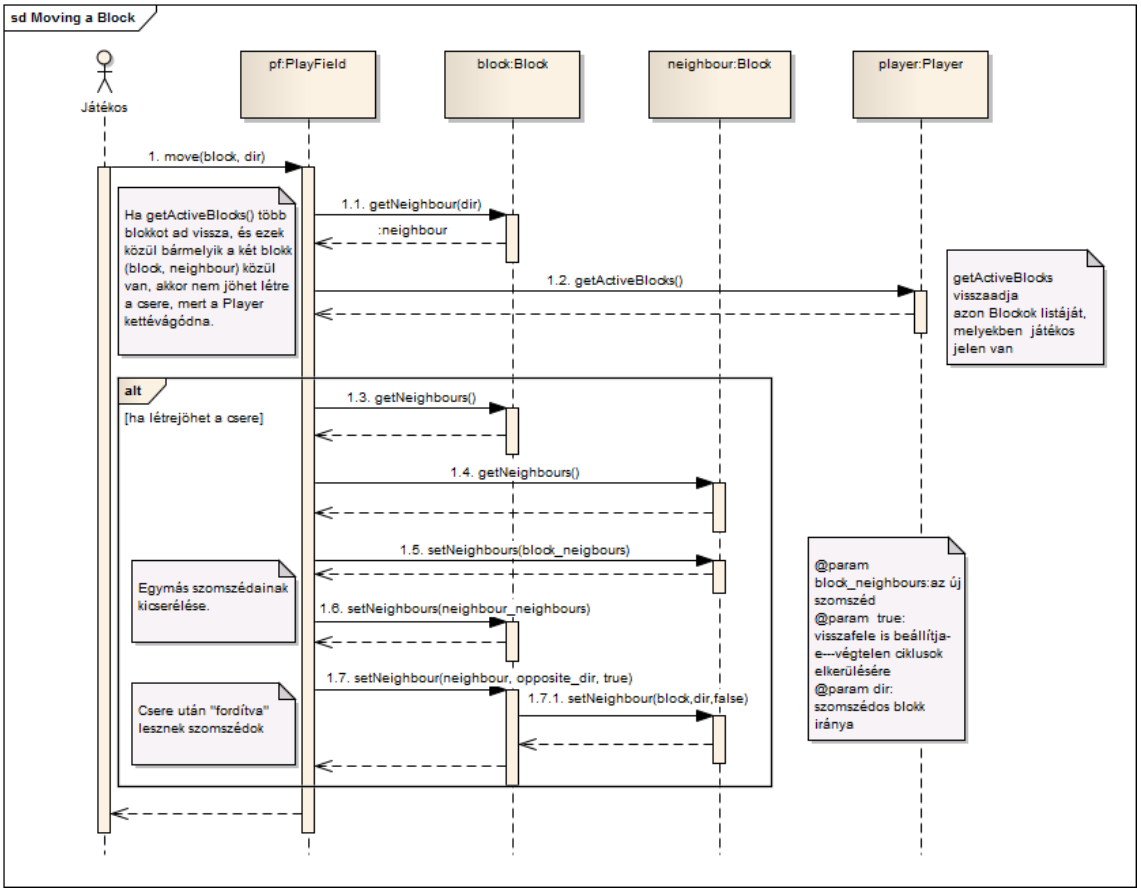


3.4. Szekvencia diagramok

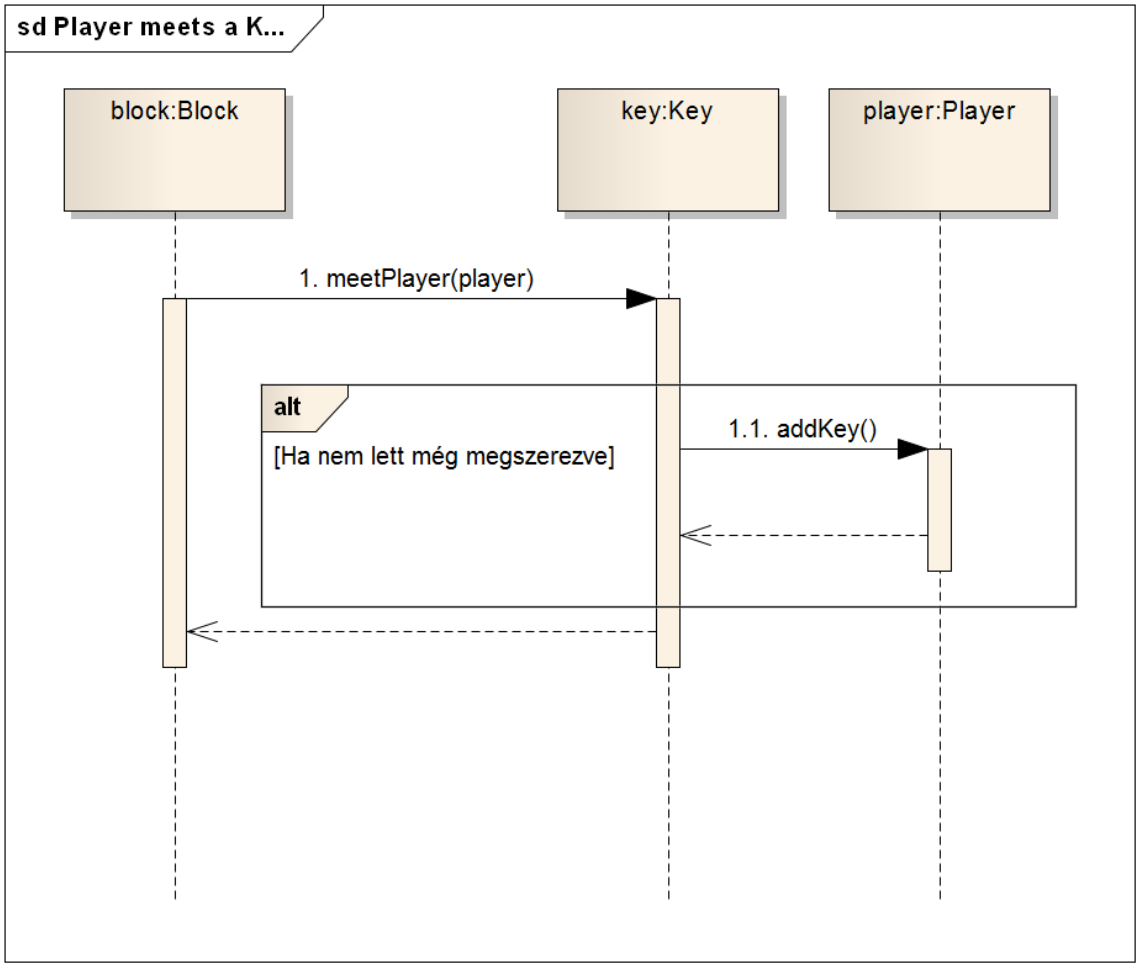
3.4.1. Inicializálás



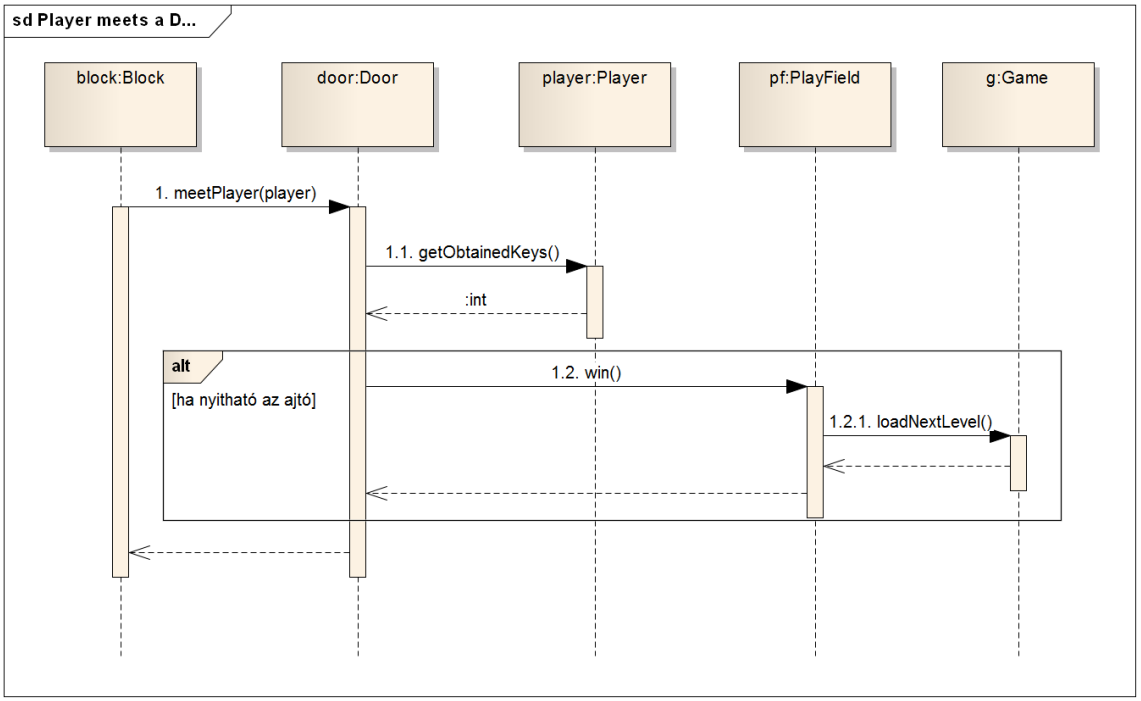
3.4.2. Blokk mozgatás



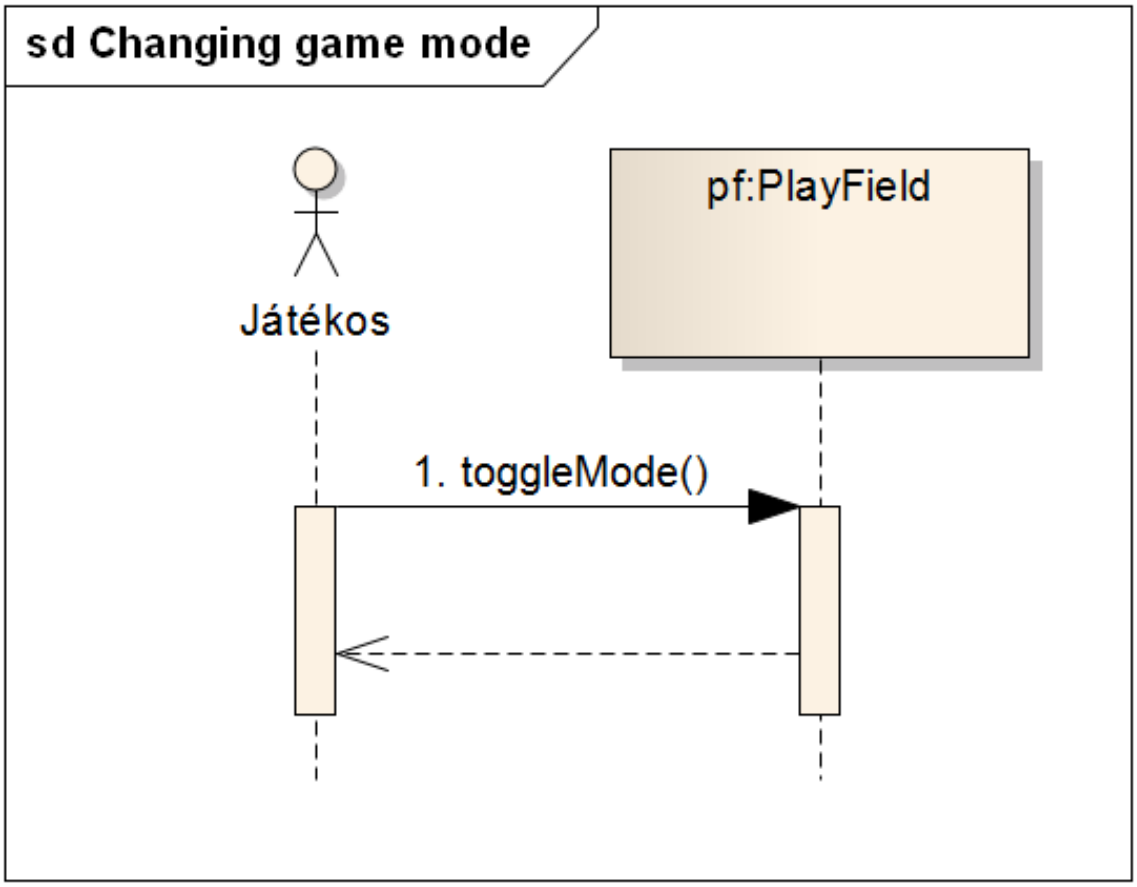
3.4.3. Kulcs felvétele



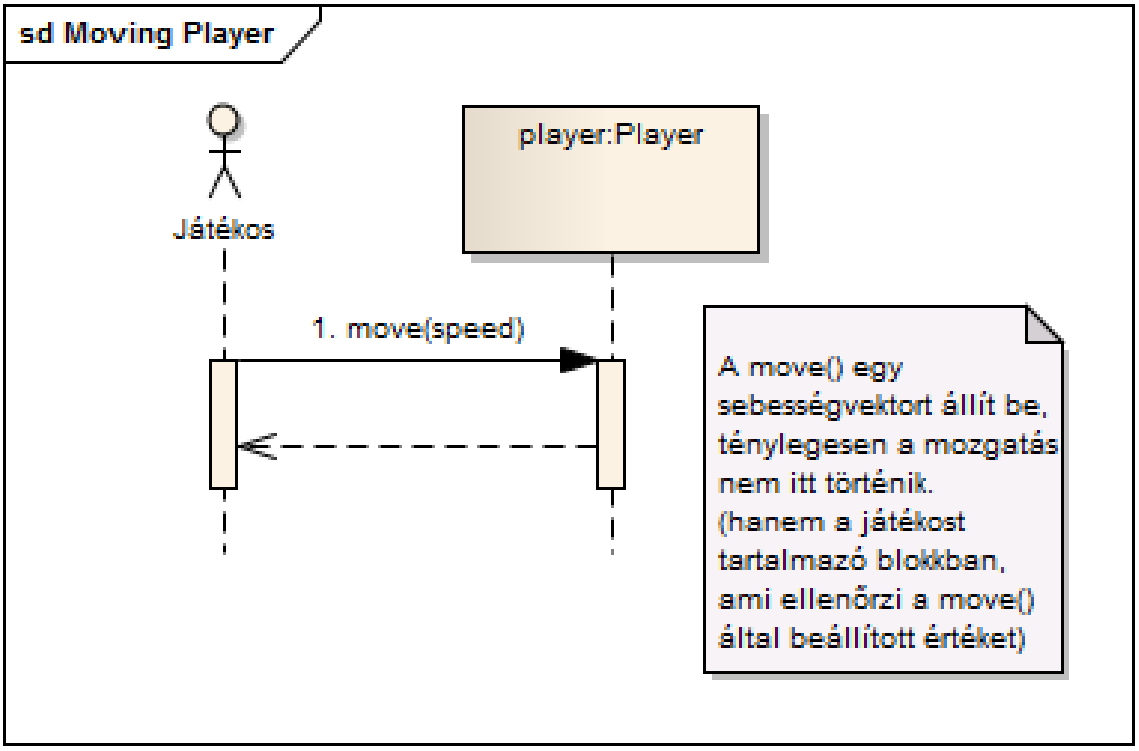
3.4.4. Ajtó interakció



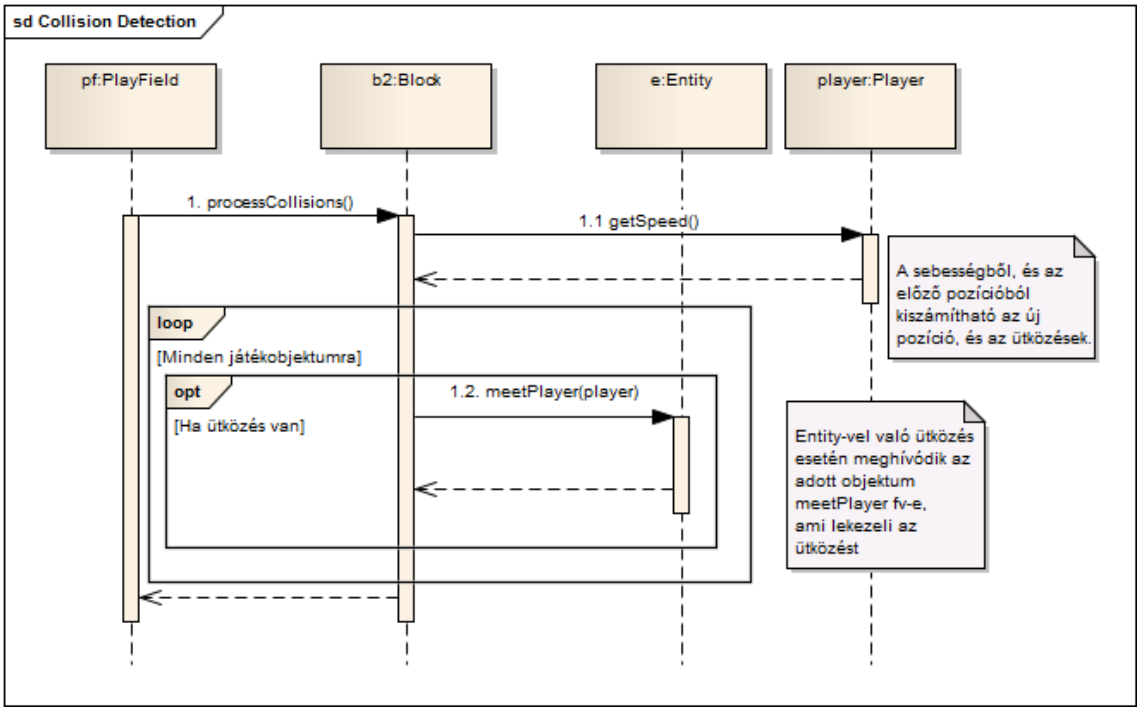
3.4.5. Játékmód váltás



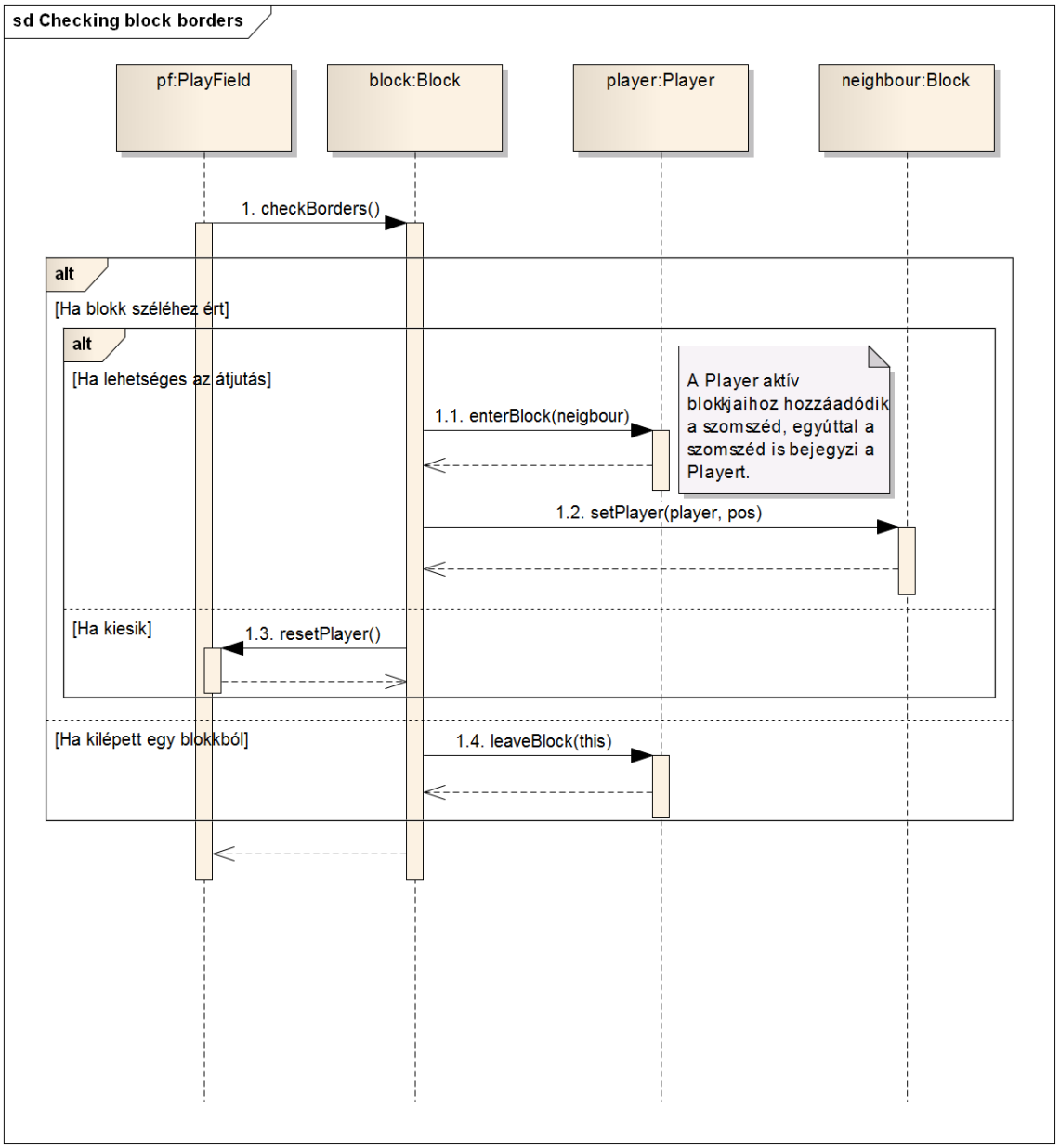
3.4.6. Játékos mozgatása



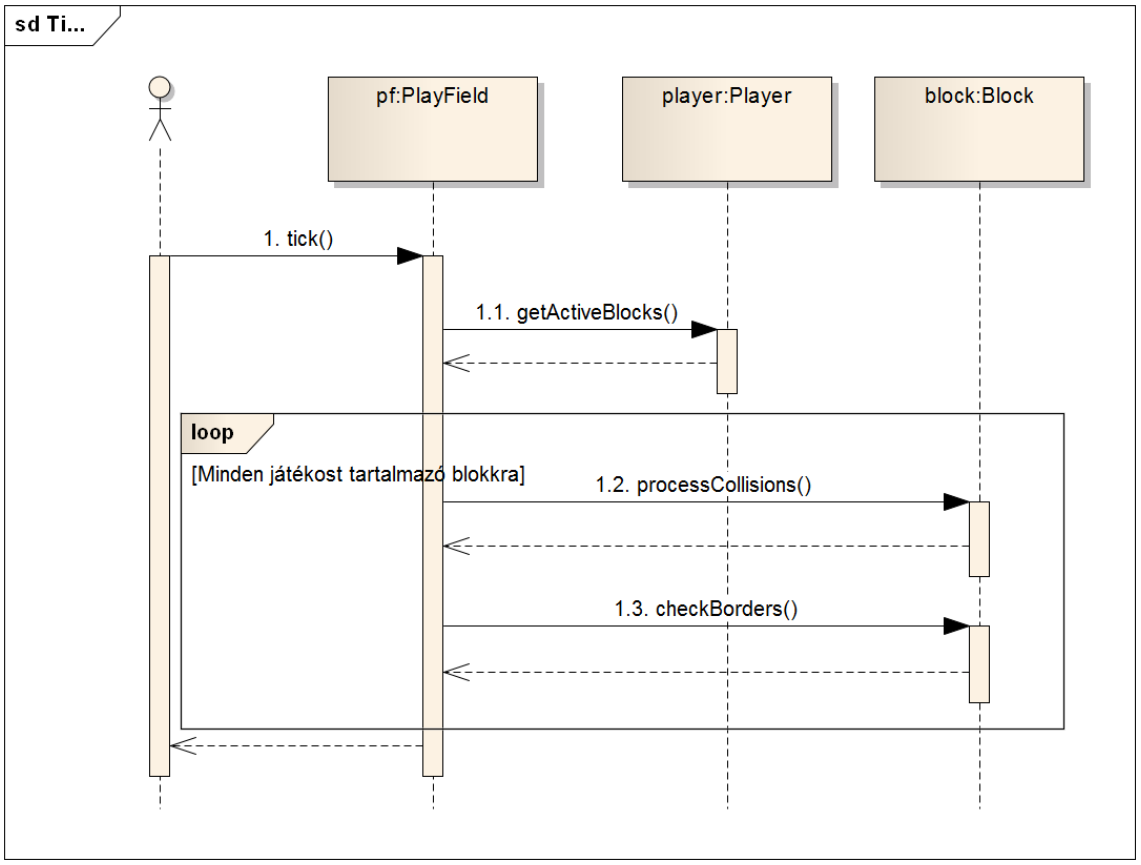
3.4.7. Ütközések kezelése



3.4.8. Blokkok közötti mozgás

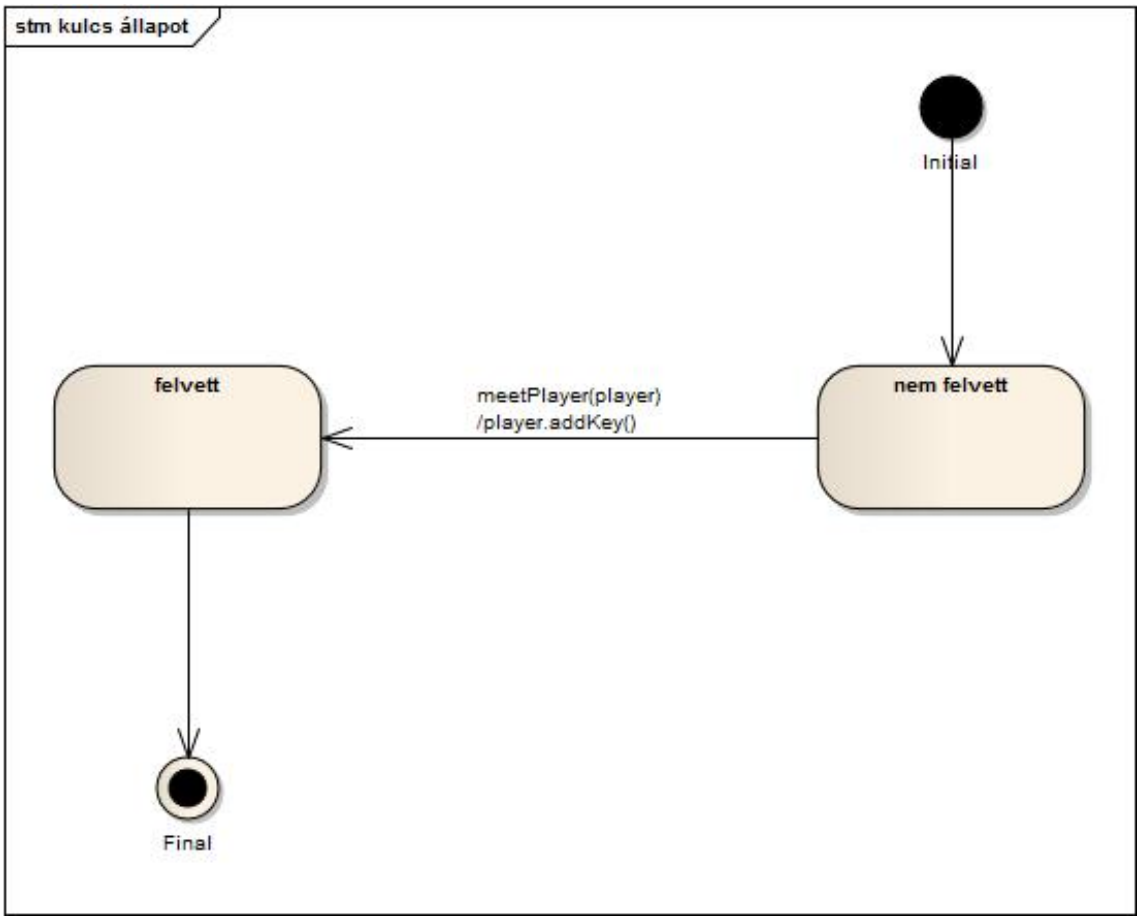


3.4.9. Tick

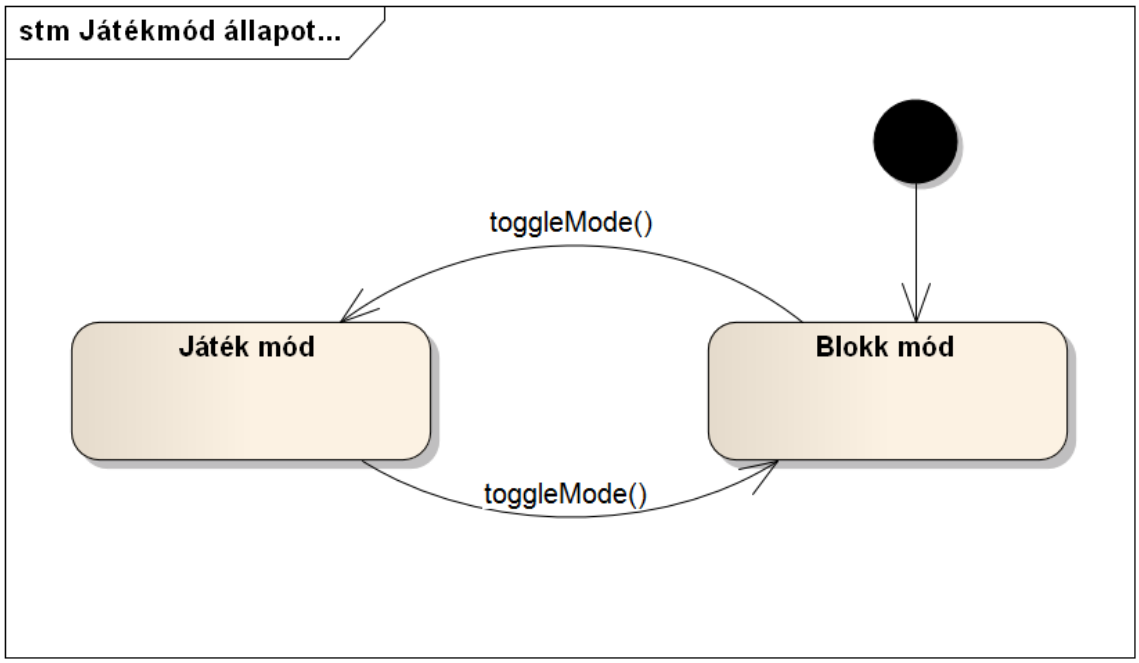

sd Ti...
pf:PlayField
player:Player
block:Block
1. tick()
1.1. getActiveBlocks()
loop
[Minden játékost tartalmazó blokkra]
1.2. processCollisions()
1.3. checkBorders()

3.5. State-chartok

3.5.1. Key állapota



3.5.2. PlayField állapota



3.6. Napló

Kezdet	Időtartam	Résztvevők	Leírás
2012.02.20. 9:00	1 óra	Dányi Dudás Kern Kolozsi	Analízis modell kidolgozásának elkezdése közösen
2012.02.22. 9:45	0,5 óra	Dányi Dudás Kern Kolozsi	Rövid megbeszélés a következő meetingek időpontjairól
2012.02.22. 18:30	3,5 óra	Dányi Kern Kolozsi	Objektum katalógus elkészítése
2012.02.24. 14:00	4 óra	Dányi Dudás Kern Kolozsi	Osztályhierarchia, szekvencia diagrammok, és state chartok kidolgozása közösen
2012.02.24. 18:00	1 óra	Kolozsi	Diagramok digitalizálása
2012.02.26. 16:00	4 óra	Dányi	Diagramok véglegesítése, köztük lévő konzisztencia biztosítása

4. Analízis modell kidolgozása 2.

4.1. Objektum katalógus

4.1.1. Game

A játék fő mozzanatait (pálya betöltés, játék megnyerése) kezeli, és kezdetben inicializál minden objektumot. Felelőssége a pálya betöltése (és inicializálása).

4.1.2. Player

A játékos irányította pálcikaembert reprezentálja. Számontartja a megszerzett kulcsok számát, valamint, hogy mely blokkokban van jelen.

4.1.3. PlayField

Ez az objektum tárolja el a pályán található blokkokat, amik közül pontosan egy darab üres. Az ő felelőssége mozgatni őket, és ellenőrizni a szabályosságát a mozgatásnak, és a játékos által felvett utolsó kulcs pozíciójának (vagy ha ilyen nincs, a kezdőpozíciónak) a tárolása, és a játékos visszahelyzése ebbe a pontba, amennyiben szükséges.

4.1.4. EmptyBlock

Egy üres blokkot reprezentál, a játékos semmilyen körülmények között nem tud rálépni, illetve nem tartalmaz kulcsot, ajtót, kezdőpontot, vagy falat. A PlayField ezt a blokkot "mozgatja" a játék során (valójában egy sima blokkot tol be a helyére, de technikailag mindig ez a blokk mozog)

4.1.5. FilledBlock

Egy sima blokkot reprezentál, tárolja a benne fellelhető objektumokat (Key, Door, Wall, SpawnPoint), valamint a rajta található játékost (amennyiben van rajta). Felelőssége tájékoztatni a rajta található objektumokat a velük történt eseményekről (ütközések, kilépés a blokkból).

4.1.6. Key

Egy kulcsot reprezentál, amit a játékosnak föl kell szednie. Amennyiben játékossal találkozik, és még nem szerezte meg a kulcsot (ez a saját belső állapota), úgy értesíti a játékost a fölvételről.

4.1.7. Door

A pályán található egyetlen ajtót reprezentálja, amit a játékosnak ki kell nyitnia, majd eljutni hozzá. Ha játékossal találkozik, a Player objektumtól elkéri a megszerzett kulcsok számát, és értesíti a PlayField objektumot a pálya megnyeréséről, ha ez a szám egyezik a saját maga által tárolt számmal (összes kulcs száma).

4.1.8. SpawnPoint

Minden pálya indulásakor a játékos ezen a pozíción kezd, és mindig ide helyeződik vissza, ha leesik a pályáról. Szerepét egy Key objektum veszi át, amint a játékos megszerez egyet.

4.1.9. Wall

A pálya azon szilárd részét (ez lehet ténylegesen egy fal, de egy platform, vagy akár lépcső is) reprezentálja, amin a játékos nem tud áthaladni. Amennyiben a játékos összeütközik vele, értesíti a játékost.

4.2. Osztályok leírása

4.2.1. Block (absztrakt)

- **Felelőssége**
  A szomszédos blokkokat nyilvántartja, és biztosítja a közöttük lévő kölcsönös kapcsolatot, valamint értesíti a rajta található objektumokat a játékos interakciójáról (rájuk lép, kilép a blokkból).
- **Ősosztályok**
  Nincs
- **Interfészek**
  Nincs
- **Attribútumok**
  - **neighbours** A blokk szomszédjainak listája.
  - **entities** A blokkon található objektumok (és pozíciójuk).
  - **player** A játékos referenciája (és pozíciója).
- **Metódusok**

- **setPlayer(player, pos)** Beállítja a játékos referenciáját és pozícióját az adott blokkon.
- **addEntity(position, entity)** Hozzáad egy objektumot a blokkhoz a megadott helyen.
- **getNeighbour(dir)** Visszaadja a megadott irányban található szomszédot. A dir egy szám, 0 jelenti az északot, és az óramutató járásával megegyező irányban történik a számozás.
- **setNeighbours(neighbours)** Felülírja az eddigi szomszédait, és a kapottakat állítja be.
- **setNeighbour(neighbour, dir, bool)** A megadott szomszédot beállítja a megfelelő irányban, a harmadik paraméter függvényében (igaz/hamis) viszont beállítja a szomszédságot. Ez a harmadik paraméter a végtelen ciklusok elkerülésére van bevezetve.
- **abstract processCollisions()** Lefuttatja az ütközésdetektálást.
- **abstract checkBorders()** A blokkok közötti mozgást kezeli le.

4.2.2. Door

- **Felelőssége**
  A Door értesíti a PlayFieldet, ha a pályának vége, ezt akkor teszi, ha a játékosnál lévő kulcsok száma megegyezik az ajtó kinyitásához szükséges kulcsok számával.
- **Ősosztályok**
  Entity
- **Interfészek**
  Nincs
- **Attribútumok**
  - **requiredKeys** A kinyitáshoz szükséges kulcsok száma.
- **Metódusok**
  - **meetPlayer(player)** Ha a játékosnál lévő kulcsok száma megegyezik a szükséges kulcsok számával, értesíti a PlayFieldet, hogy a pályának vége.

4.2.3. EmptyBlock

- **Felelőssége**
  Az üres blokkot reprezentálja, gyakorlatilag a Block osztály default (üres) implemetációja.
- **Ősosztályok**
  Block
- **Interfészek**
  Nincs
- **Attribútumok**
  Nincs
- **Metódusok**
  - **processCollisions()** Mivel üres blokk, semmi sincs benne, így ez a metódus nem csinál semmit.
  - **checkBorders()** Mivel üres blokk, semmi sincs benne, így ez a metódus nem csinál semmit.

4.2.4. Entity (absztrakt)

- **Felelőssége**
  Az Entity egy blokkon található statikus (nem mozgó) objektumot reprezentál, minden leszármazottnak kötelessége viselkedést definiálni a játékossal való találkozásra.
- **Ősosztályok**
  Nincs
- **Interfészek**
  Nincs
- **Attribútumok**
  - **playField** PlayField referencia, a leszármazottak a viselkedés leírásának megkönnyítése érdekében felhasználhatják.
- **Metódusok**
  - **abstract meetPlayer(player)** Kötelezően implementálandó metódus, a játékossal való találkozás forgatókönyvét írja le.

4.2.5. FilledBlock

- **Felelőssége**
  A sima blokkot reprezentálja, kezeli a benne található objektumokat (objektumok interakciója a játékossal)
- **Ősosztályok**
  Block
- **Interfészek**
  Nincs
- **Attribútumok**
  Nincs
- **Metódusok**
  - **processCollisions()** Végignézi az összes objektumot, hogy ütközik-e a játékossal, ha pedig igen, meghívja a meetPlayer metódusukat.

4.2.6. Game

- **Felelőssége**
  A Game osztály felelőssége betölteni, és inicializáltatni a pályát.
- **Ősosztályok**
  Nincs
- **Interfészek**
  Nincs
- **Attribútumok**
  - **playField** Az aktuális pálya referenciája
- **Metódusok**
  - **start()** Elindítja a játékot (betölti a megfelelő pályát)
  - **loadNextLevel()** Betölti a következő pályát

4.2.7. Key

- **Felelőssége**
  A Key számon tartja magáról, hogy fölvették-e, és a játékossal való találkozása esetén ennek függvényében változik a viselkedése.
- **Ősosztályok**
  Entity
- **Interfészek**
  Nincs
- **Attribútumok**
  - **isObtained** Fel lett-e véve már a kulcs.
- **Metódusok**
  - **meetPlayer(player)** Ha a kulcs fel lett véve, semmi sem történik, egyébként megnöveli a játékosnál lévő kulcsok számát, a saját állapotát pedig megváltoztatja (felvettre).

4.2.8. Player

- **Felelőssége**
  A Player számon tartja magáról, hogy mely blokkokban van jelen, valamint a nála lévő kulcsok számát.
- **Ősosztályok**
  Entity
- **Interfészek**
  Nincs
- **Attribútumok**
  - **obtainedKeys** A megszerzett kulcsok száma.
  - **activeBlocks** A játékos jelenleg ezekben a blokkokban van jelen. Ha ez több egynél, akkor a mozgatás nem feltétlenül lehetséges.
  - **speed** A játékos jelenlegi sebessége, ebből számítható a következő pozíciója.
- **Metódusok**
  - **addKey()** Megnöveli a nála lévő kulcsok számát.

- **enterBlock(block)** A játékos belép a megadott blokkba.
- **leaveBlock(block)** A játékos kilép a megadott blokkból.
- **move(dir)** A játékos sebességét a megadott irányban megnöveli. Ténylegesen nem mozgatja a játékost, csak jelzi a mozgás irányát.
- **getObtainedKeys()** A játékos által birtokolt kulcsok számát adja vissza.
- **getActiveBlocks()** A játékos által elfoglalt blokkokat adja vissza.

4.2.9. PlayField

- **Felelőssége**
  A PlayField felelős a rajta található blokkok mozgatásáért, ismeri az aktuális játékmódot (blokk/játék), és tudja hova kell áthelyeznie a játékost (ezért is ő a felelős), ha kiesik a pályáról.
- **Ősosztályok**
  Nincs
- **Interfészek**
  Nincs
- **Attribútumok**
  - **blocks** A rajta található blokkok (és pozíciójuk).
  - **game** Egy Game-re mutató referencia a kommunikációhoz.
  - **player** A játékosra mutató referencia.
  - **spawnPosition** A játékos aktuális újraéledési helye, ha leesik a pályáról.
  - **blockMode** Az aktuális játékmód (blokk/játék)
- **Metódusok**
  - **setPlayer(player)** Beállítja a játékos pozícióját.
  - **addBlock(position,block)** Hozzáadja a megadott blokkot a megadott pozícióban az aktuális pályához.
  - **setSpawnPoint(entity)** Beállítja, hogy az újraéledés melyik objektumon történjen.
  - **toggleMode()** Játékmódot vált (blokk/játék).
  - **win()** Meghívja a Game loadNextLevel() metódusát.
  - **move(block,dir)** A megadott blokkot elmozdítja a megadott irányba.

- **resetPlayer()** A játékost visszahelyezi az elmentett kezdőpozícióba.
- **tick()** A játékos ténylegesen csak ebben a metódusban mozdul meg. Ütközésdetektálás és blokkok közötti mozgás vizsgálata után beállítja az új pozícióját. Részletesen lásd a szekvenciadiagramon.

4.2.10. PlayFieldBuilder

- **Felelőssége**
  A PlayField összerakásáért felelős, elkészíti a blokkokat, elhelyezi rajtuk az objektumokat, a játékost, a blokkokat összerendezi a PlayField objektumban.
- **Ősosztályok**
  Nincs
- **Interfészek**
  Nincs
- **Attribútumok**
  Nincs
- **Metódusok**
  - **createPlayField(spec)** A paraméterben megadott specifikációnak megfelelő PlayFieldet összerakja, majd a hívónak visszaadja.

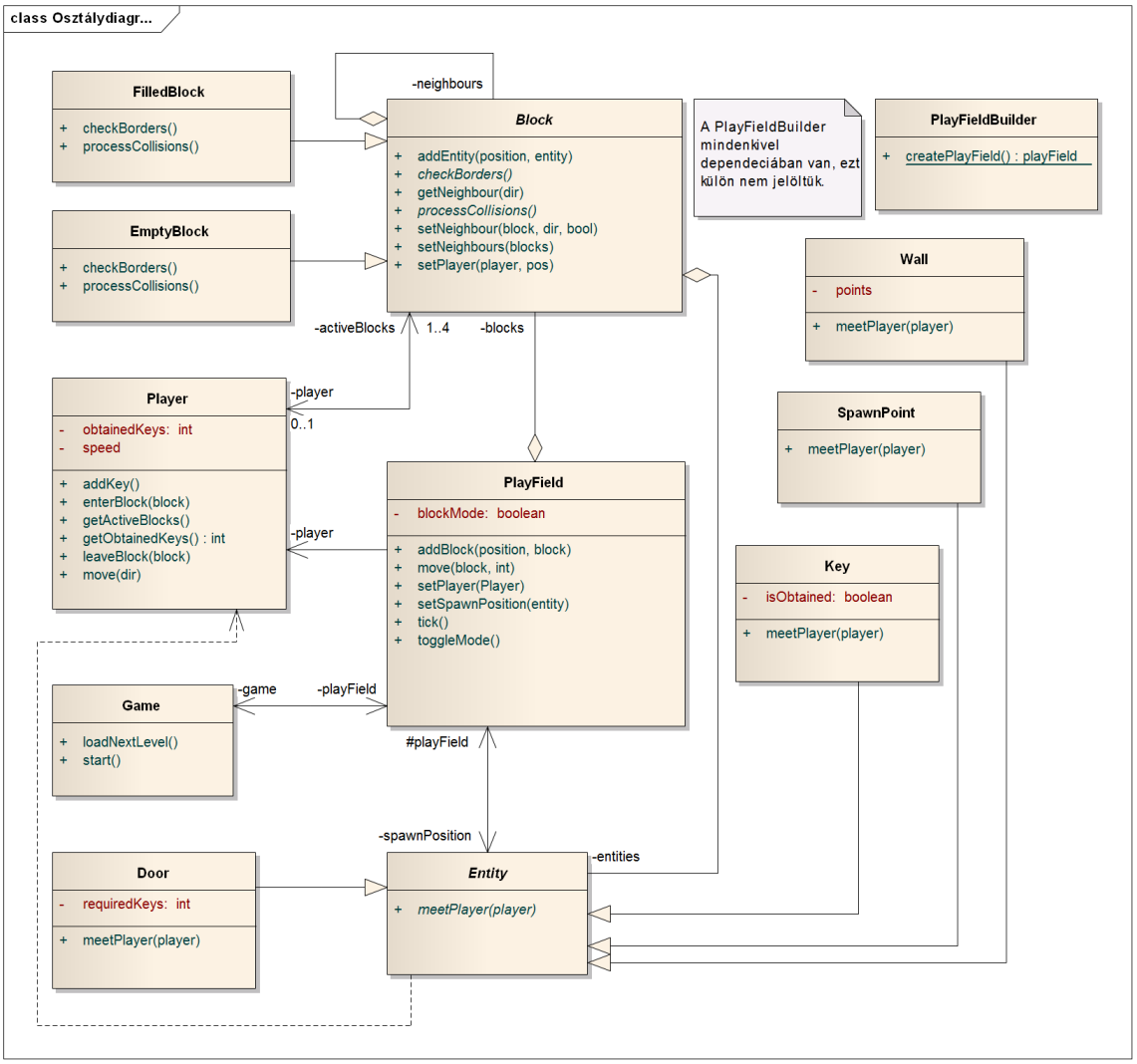
4.2.11. SpawnPoint

- **Felelőssége**
  Az Entity absztrakt osztály default (üres) implementációja, nem reagál a játékossal, viszont a játékost ide lehet visszahelyezni, ha leesik a pályáról.
- **Ősosztályok**
  Entity
- **Interfészek**
  Nincs
- **Attribútumok**
  Nincs
- **Metódusok**
  - **meetPlayer(player)** Üres metódus, nem reagál a játékossal.

4.2.12. Wall

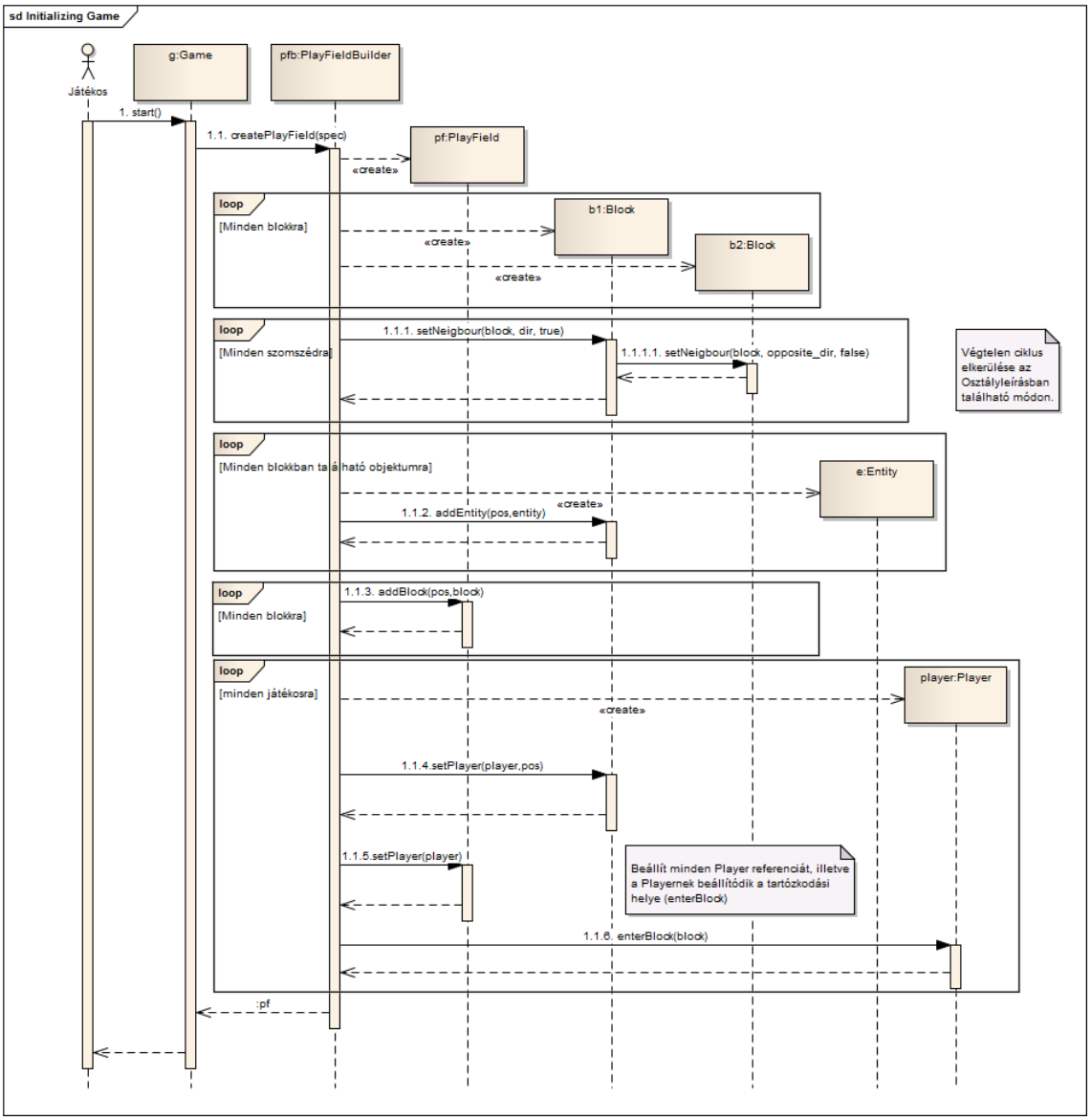
- **Felelőssége**
  A Wall egy olyan poligon, ami megállítja a játékost, ha összeütközik vele, a feladata ennek a viselkedésnek a helyes elvégzése.
- **Ősosztályok**
  Entity
- **Interfészek**
  Nincs
- **Attribútumok**
  - **points** A falat reprezentáló poligon pontjai.
- **Metódusok**
  - **meetPlayer(player)** Ha összeütközik a játékossal, akkor megállítja a mozgásban (vízszintes, vagy függőleges irányban)

4.3. Statikus struktúra diagramok

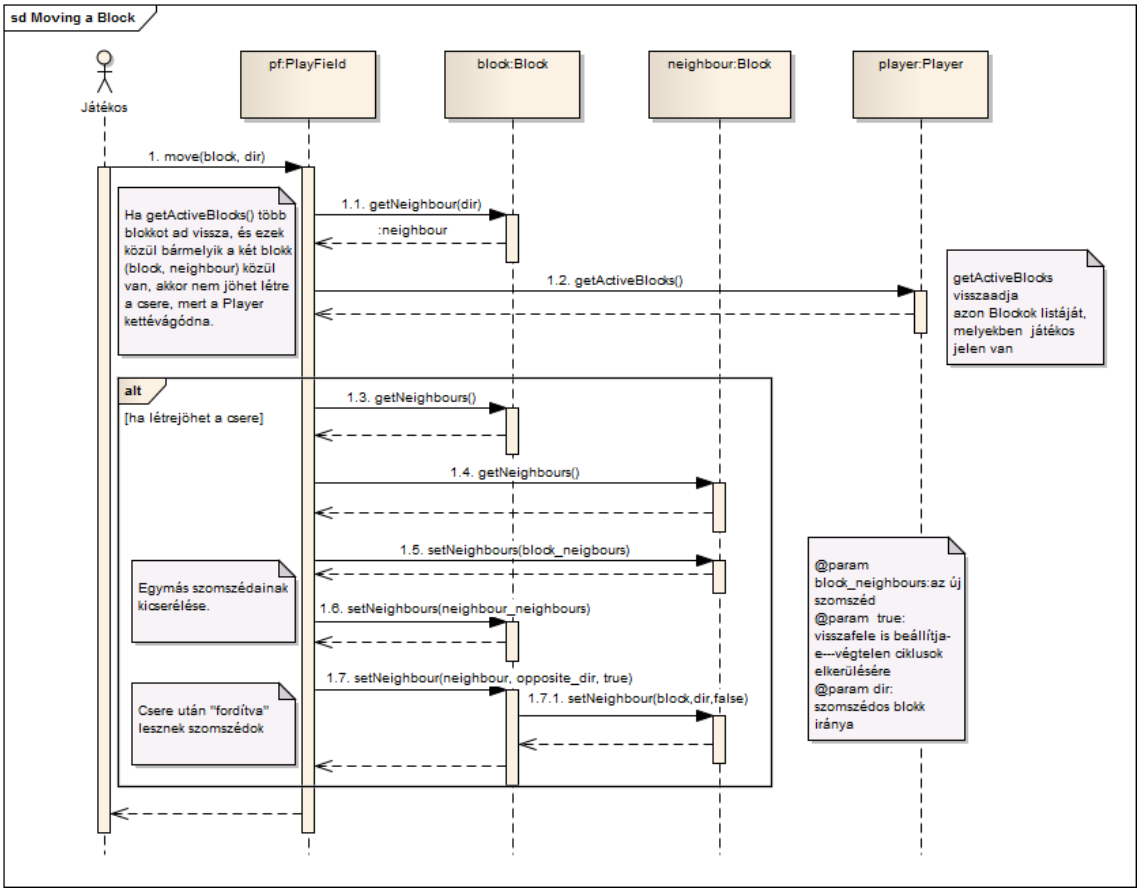


4.4. Szekvencia diagramok

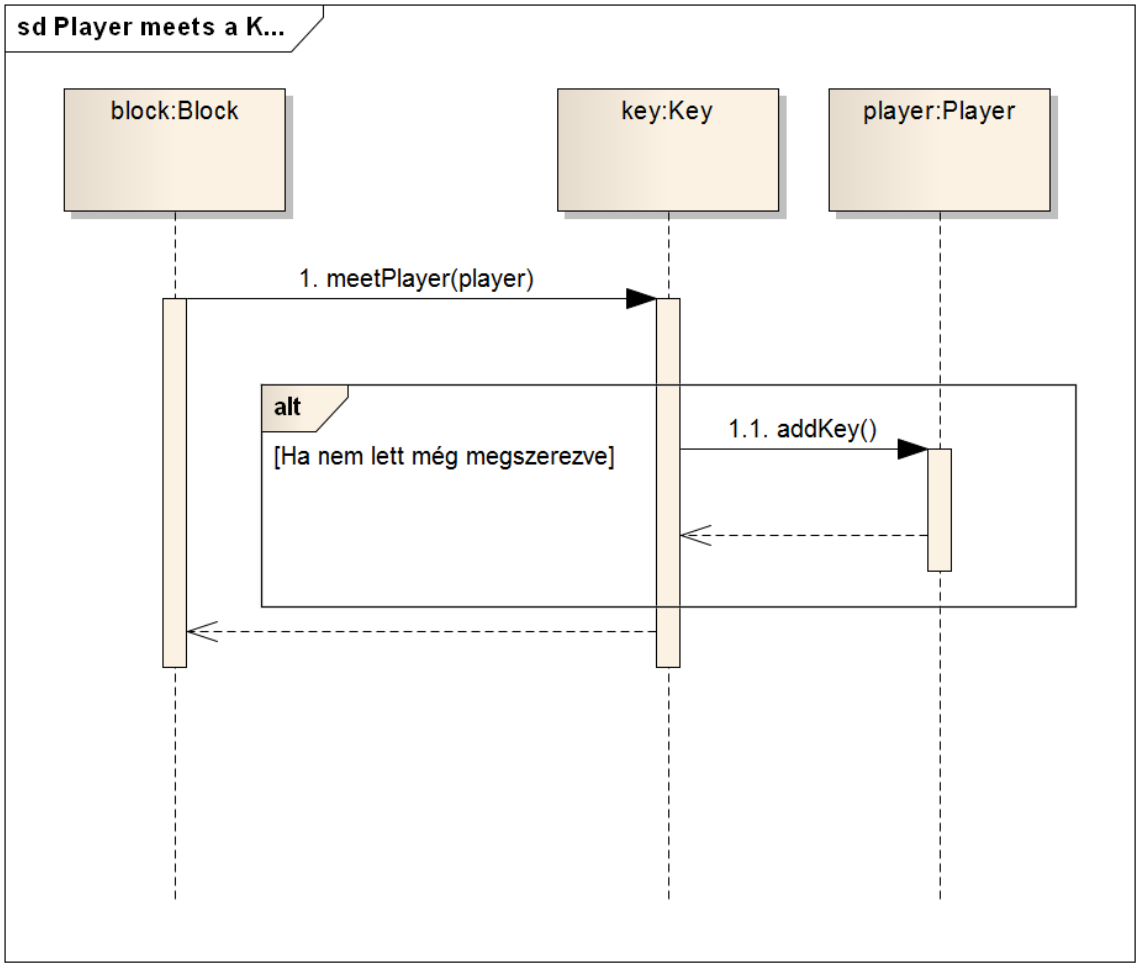
4.4.1. Inicializálás



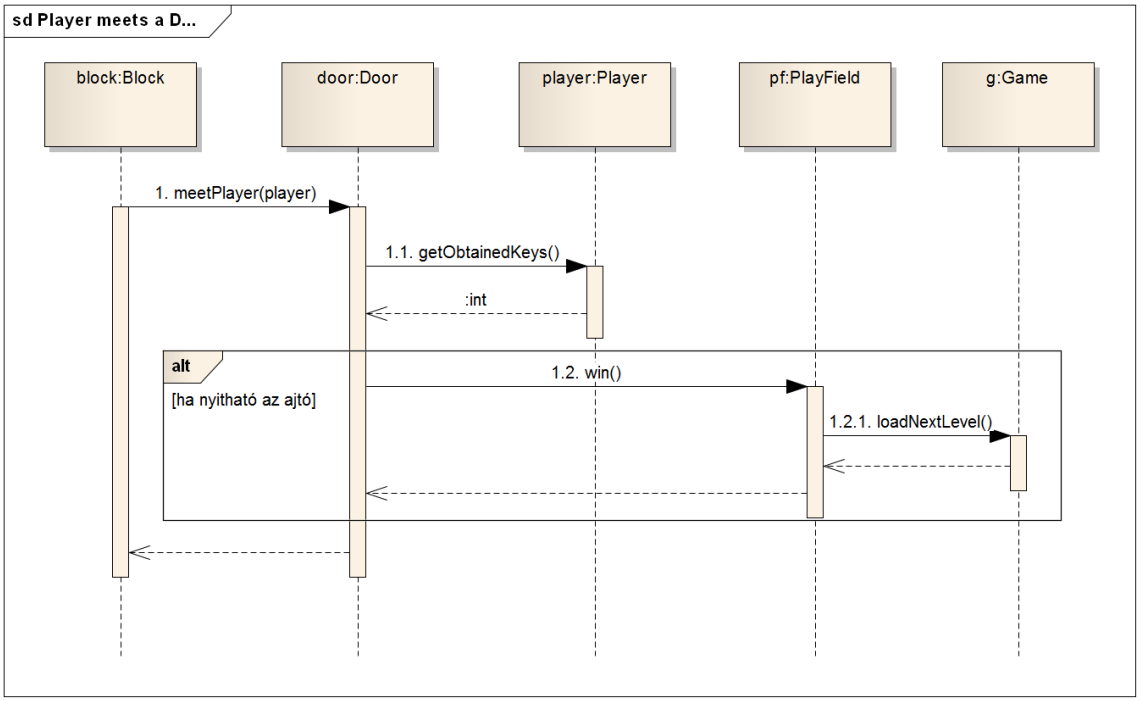
4.4.2. Blokk mozgatás



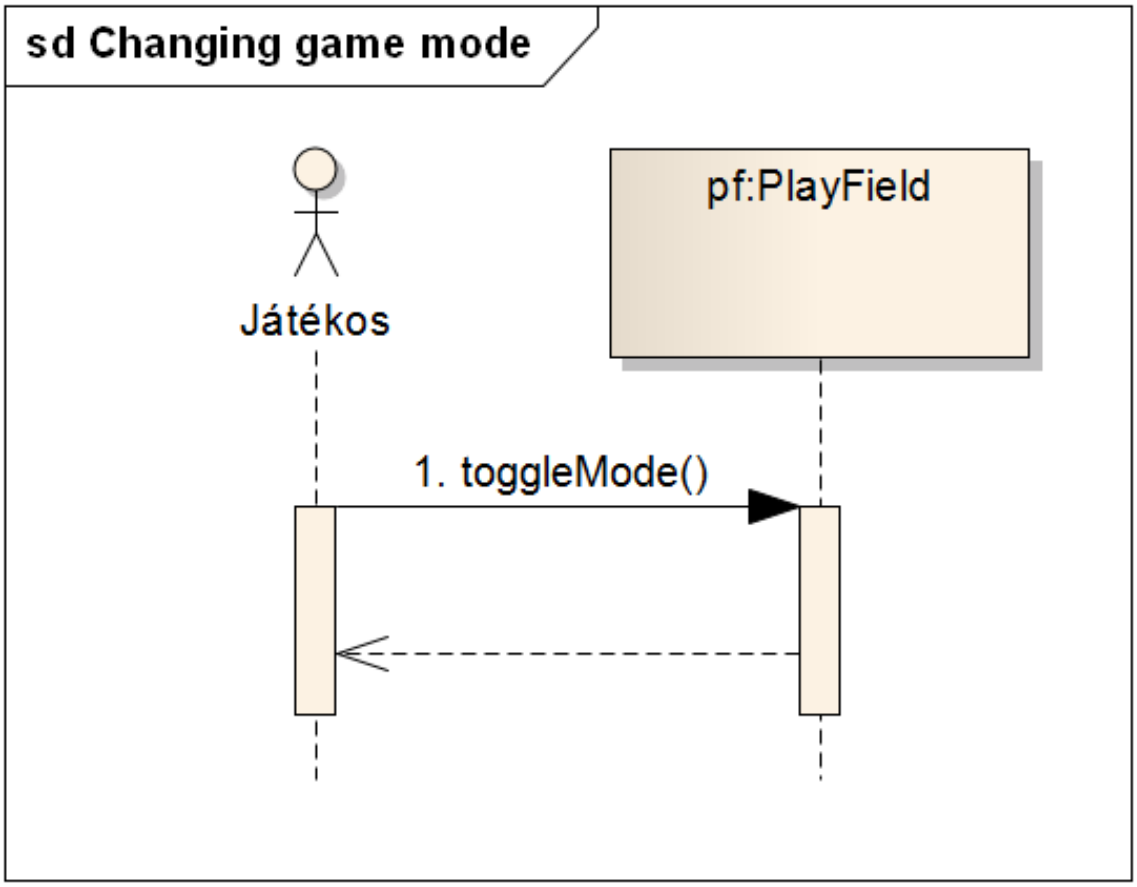
4.4.3. Kulcs felvétele



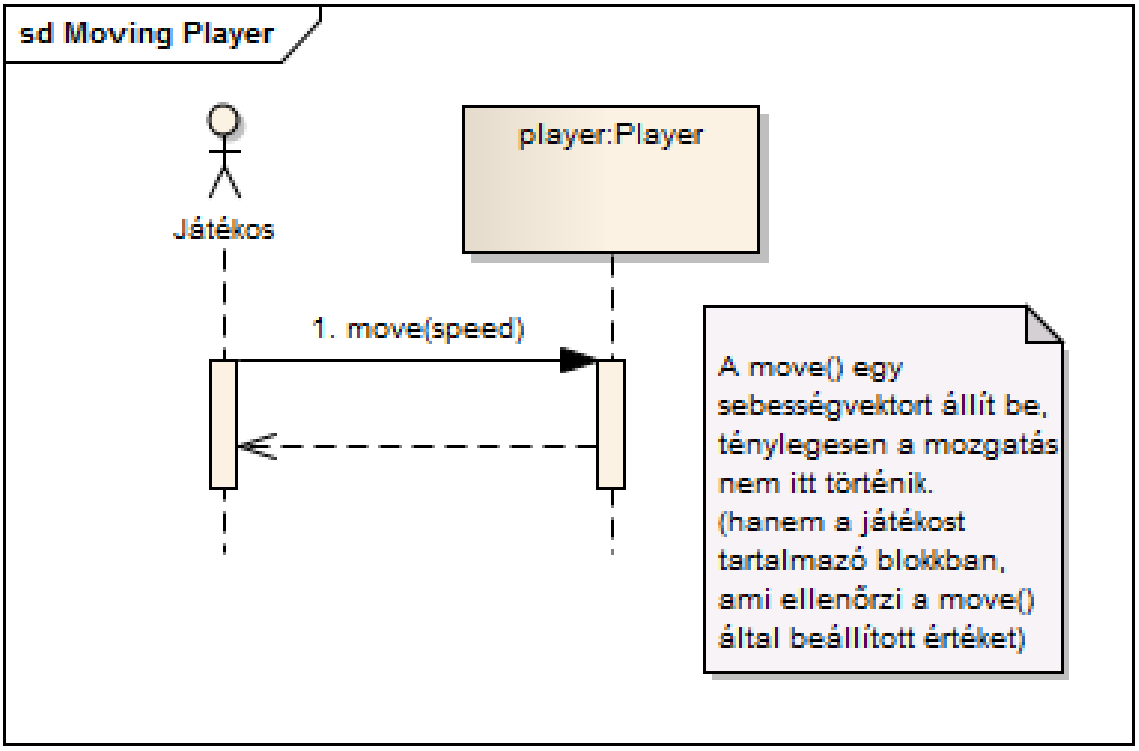
4.4.4. Ajtó interakció



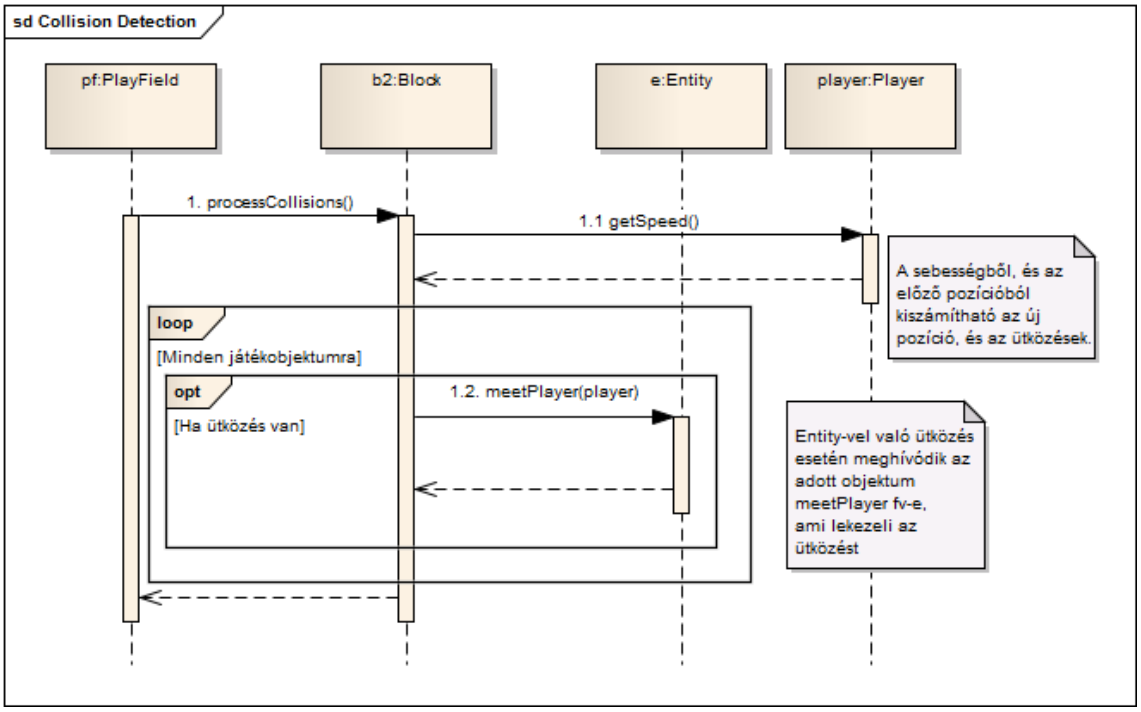
4.4.5. Játékmód váltás



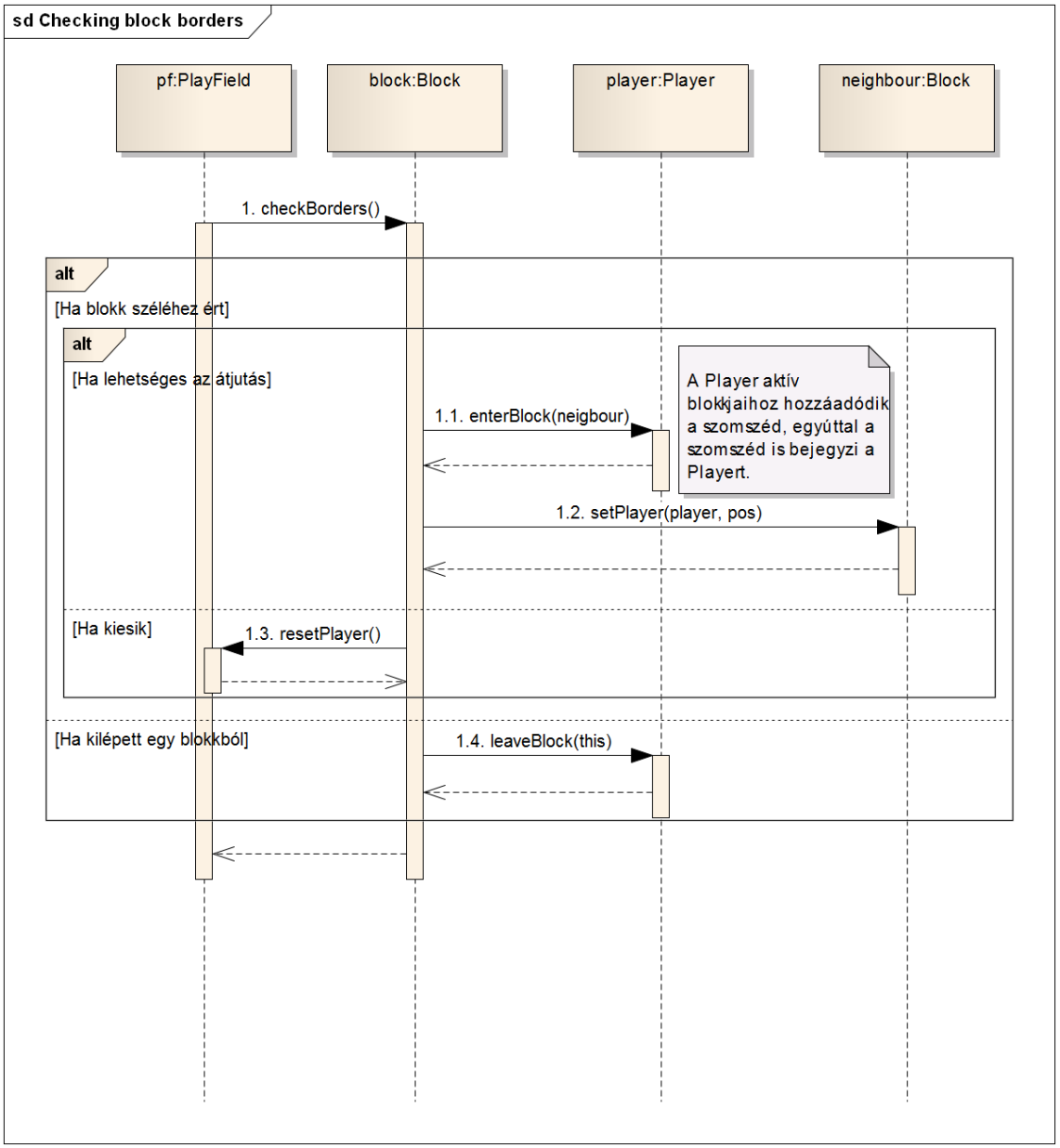
4.4.6. Játékos mozgatása



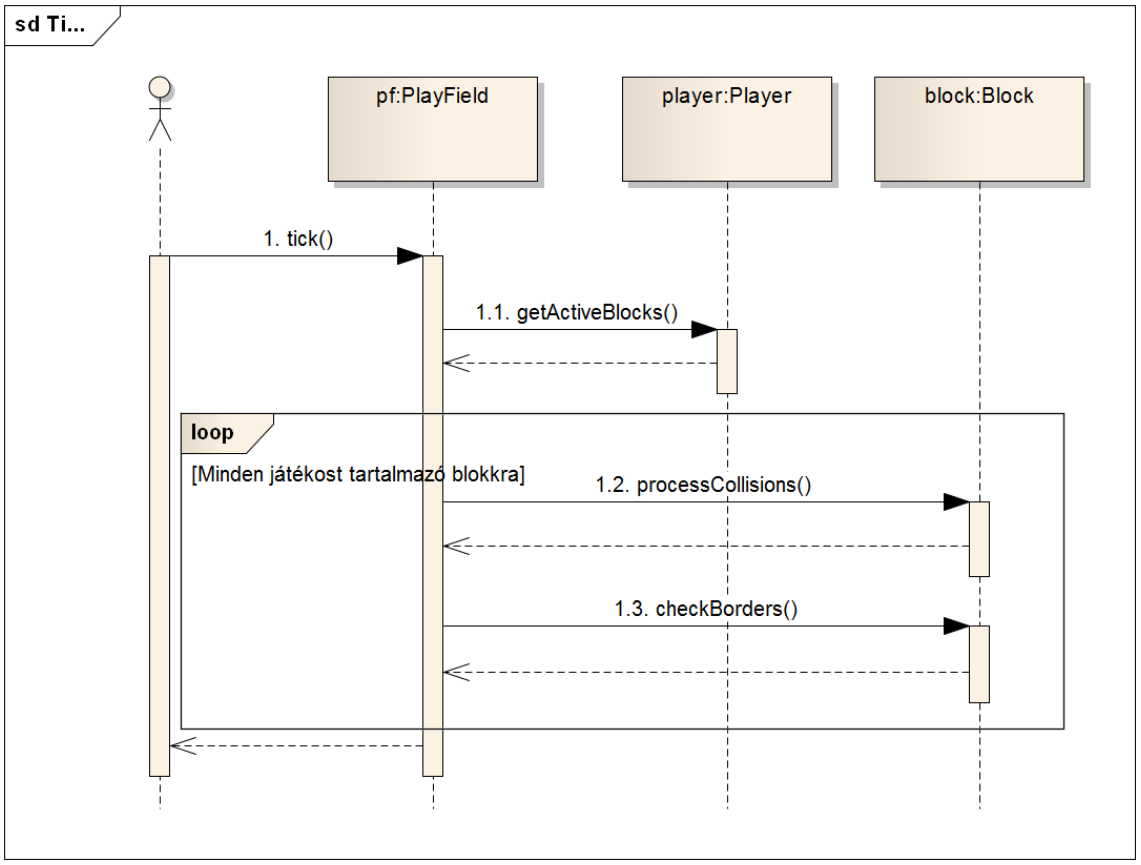
4.4.7. Ütközések kezelése



4.4.8. Blokkok közötti mozgás

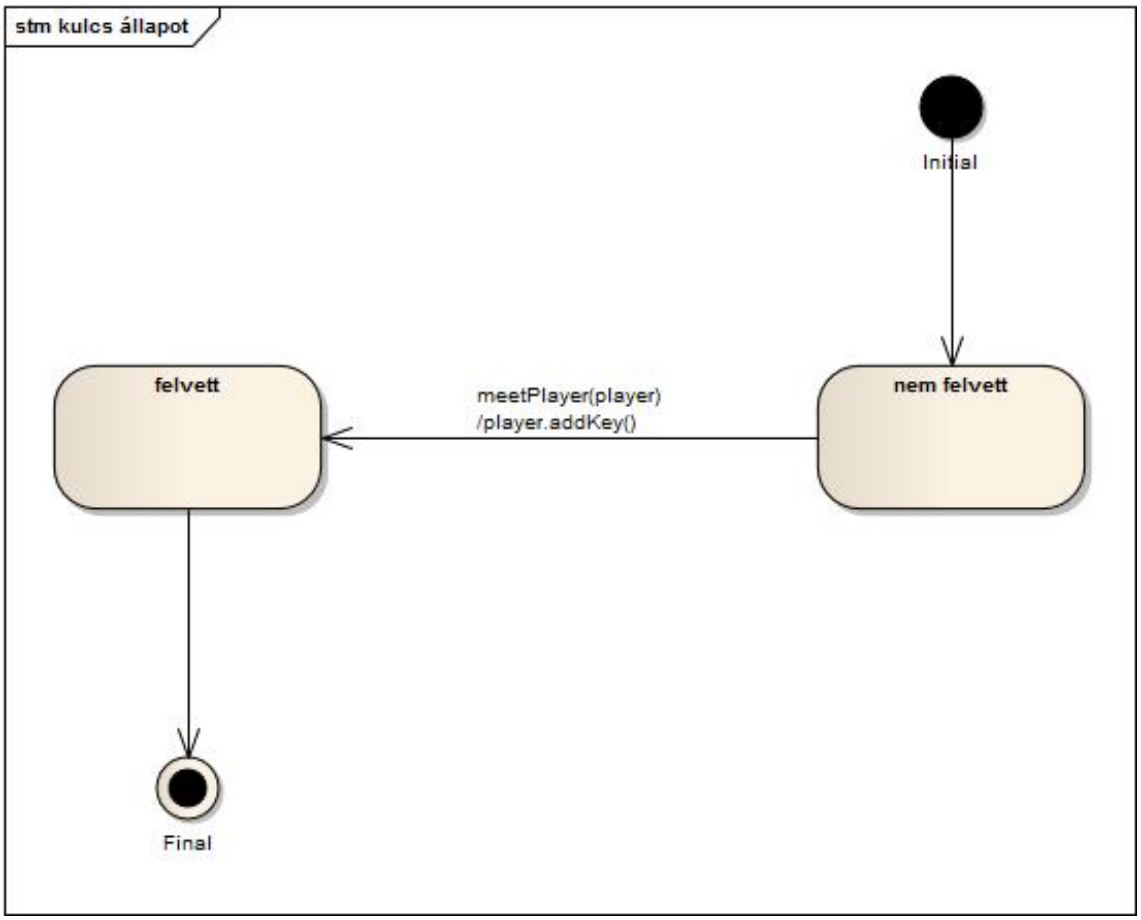


4.4.9. Tick

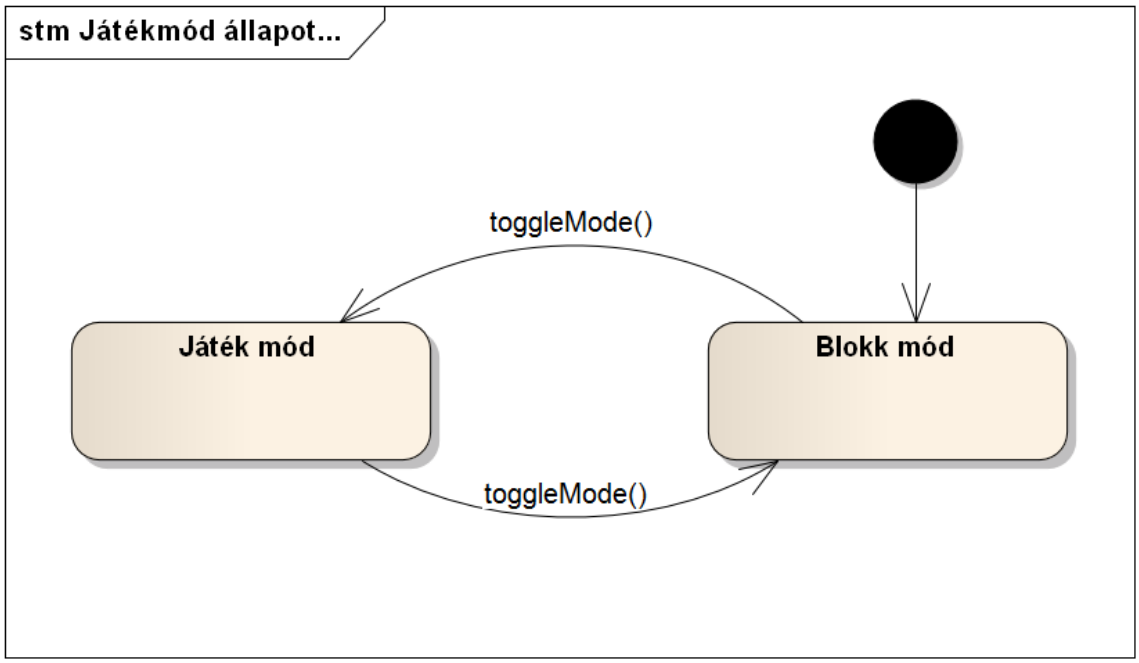

sd Ti...
pf:PlayField
player:Player
block:Block
1. tick()
1.1. getActiveBlocks()
loop
[Minden játékost tartalmazó blokkra]
1.2. processCollisions()
1.3. checkBorders()

4.5. State-chartok

4.5.1. Key állapota



4.5.2. PlayField állapota



4.6. Napló

Kezdet	Időtartam	Résztvevők	Leírás
2012.03.01. 21:00	2,5 óra	Dányi Dudás Kern	A modell hibáinak megbeszélése, és javítása.
2012.03.04. 14:00	2 óra	Kern	PlayFieldBuilderrel kapcsolatos hibák javítása.
2012.03.04. 21:00	2 óra	Dányi	Diagramok összehangolása, egyéb felmerült hibák javítása.

5. Szkeleton tervezése

5.1. A szkeleton modell valóságos use-case-ei

5.1.1. Use-case diagram

5.1.2. Use-case leírások

Use-case neve	**Idő léptetése**
Rövid leírás	Játék belső órájának egy egységgel való növelése
Aktorok	Timer
Forgatókönyv	Az idő mindig automatikusan egységnyivel nő
Use-case neve	**Tartalmazó blokkok ellenőrzése**
Rövid leírás	Lekérdezés a játékostól, hogy mely blokkokban van jelen
Aktorok	Timer
Forgatókönyv	A PlayField megkérdezi a játékostól,hogy mely blokkok tartalmazzák
Use-case neve	**Ütközés detektálás**
Rövid leírás	Ellenőrzés,hogy a játékos ütközött-e valaminek
Aktorok	Timer
Forgatókönyv	1.Minden Játékobjektumra a blokkban történik ellenőrzés 2.Ütközés esetén a blokk továbbadja a kezelést az adott entitásnak
Use-case neve	**Ütközés Ajtóval**
Rövid leírás	Pálcikaember eljut az ajtóhoz
Aktorok	Timer
Forgatókönyv	1.Az ajtó megkérdezi a játékostól,hogy megszerezte-e az összes kulcsot 2.Ha nem,nem nyílik ki,egyébként a pályát megnyertnek tekintjük és betöltődik az új pálya
Use-case neve	**Ütközés Kulccsal**
Rövid leírás	Pálcikaember eljut egy kulcshoz
Aktorok	Timer
Forgatókönyv	Ha a kulcsot korábban még nem vettük fel,akkor most a játékos megszerzett kulcsainak számát egyel növeljük
Use-case neve	**Ütközés fallal**
Rövid leírás	Pálcikaembert a játékos egy falnak irányítja
Aktorok	Timer
Forgatókönyv	Ha a pálcikaember következő lépésben falnak menne, a játék nem engedi arra haladni.

Use-case neve	**Falnak ütközés felülről**
Rövid leírás	Pálcikaember felülről ütközik falnak
Aktorok	Timer
Forgatókönyv	Ha a pálcikaember lefele haladás közben következő lépésben falnak ütközne, akkor megáll
Use-case neve	**Falnak ütközés oldalról**
Rövid leírás	Pálcikaember oldalról ütközik falnak
Aktorok	Timer
Forgatókönyv	Ha a pálcikaember oldalra haladás közben következő lépésben falnak ütközne, akkor a játék nem engedi arra haladni
Use-case neve	**Blokk szélének ellenőrzése**
Rövid leírás	Annak ellenőrzése,hogy a pálcikaember átjuthat-e másik blokkba
Aktorok	Timer
Forgatókönyv	Ha a játékos egy blokk széléről átmozogna egy másikba,ellenőrizzük, hogy a két blokk szélei egyeznek-e
Use-case neve	**Játékos leesik**
Rövid leírás	Játékos kiesik egy blokkból
Aktorok	Timer
Forgatókönyv	Ha a játékos úgy mozog ki egy blokk alján, hogy alatta nincs egyező szélű blokk, akkor a játékost vissza rakjuk a legutoljára felvett kulcs pozíciójába, vagy ha ilyen nincs, akkor a kezdőpozícióba
Use-case neve	**Átmozgatás másik blokkba**
Rövid leírás	Pálcikaember átjut egyik blokkból a másikba
Aktorok	Timer
Forgatókönyv	Ha egyezik két blokk széle akkor a pálcikaember mozoghat a két blokk között
Use-case neve	**Blokkok Mozgatása**
Rövid leírás	Egy üres blokkal szomszédos blokknak az üres blokk helyére való mozgatása
Aktorok	Player
Forgatókönyv	1.Ellenőrizzük,hogy a pálcikaember nem áll-e két blokk között, melyek ilyenkor nem mozgathatók 2.Ha ez nem áll fenn, akkor az üres blokk helyére mozdíthatunk egy vele szomszédos blokkot

Use-case neve	**Új Játék Inditása**
Rövid leírás	Új játék kezdése
Aktorok	Player
1. Forgatókönyv	A játékos kiválasztja a menüben az "Új játék indítása" címkét és rákattint. 2. Az első pálya töltődik be, a pálcikaember a kezdőpozícióban áll

Use-case neve	**Pálya Betöltése**
Rövid leírás	Következő pálya megjelenítése
Aktorok	Player
Forgatókönyv	Ha egy pályán elértünk egy nyitott ajtót, akkor a következő pálya megjelenik, melynek kezdőpzíciójába kerül a pálcikaember

Use-case neve	**Pálcikaember mozgatása**
Rövid leírás	Pálcikaember irányítása
Aktorok	Player
Forgatókönyv	Játék módba kapcsolunk. A pálcikaembert jobbra/balra irányítjuk vagy/és ugrunk vele

Use-case neve	**Ugrás**
Rövid leírás	A pálcikaember ugrik
Aktorok	Player
Forgatókönyv	A pálcikaember felfelé mozog, ha nincs felette fal.

Use-case neve	**Jobbra/Balra mozgás**
Rövid leírás	A pálcikaember vízszintesen mozog.
Aktorok	Player
Forgatókönyv	A pálcikaember vízszintesen mozog ha nincs előtte fal

Use-case neve	**Váltás Blokk Mód és Játék Mód között**
Rövid leírás	Váltás blokk mód és játék mód között
Aktorok	Player
Forgatókönyv	Játék módból átváltunk blokk módba vagy fordítva

Use-case neve	**Blokk módba váltás**
Rövid leírás	Blokkok mozgatásának aktiválása
Aktorok	Player
Forgatókönyv	Játék módból blokk módba váltunk

Use-case neve	**Játék módba váltás**
Rövid leírás	Pálcikaember mozgatásának aktiválása
Aktorok	Player
Forgatókönyv	Blokk módból játék módba váltunk

5.2. Architektúra

5.2.1. Tesztpálya

16 Blokkból áll. 15 FilledBlock és 1 ÜresBlokk. Az egyik blokk tartalmaz egy falat, egy kulcsot, egy ajtót és egy játékost, a többi blokk üres.

5.2.2. Első teszteset

Azt vizsgáljuk, hogy Timer, mint aktor megfelelően működik-e. A tick() meghívásával ezzel a use-case-ek többségét le is fedjük.

5.2.3. Második teszteset

Blokk mozgatás helyességének ellenőrzését vizsgáljuk, valamint azt, hogy a játékos, ha kiesik, helyesen visszaugrik-e a kezdőpozícióba.

5.3. A szkeleton kezelői felületének terve, dialógusok

A szkeleton program azért készül, hogy a megtervezett modell megfelel-e a valóságnak, azaz a működése valóban megfelel-e a modellnek. A programban jelen van minden olyan business objektum, ami a teljes értékű grafikus programban is szerepelni fog, azonban algoritmust nem implementálnak, innen következik, hogy stateless objektumok, nincs állapotuk. Futáskor minden metódus kiírja a saját nevét a standard kimenetre, majd meghívja a működéséhez szükséges egyéb metódusokat. Elágazás esetén a standard bemeneten érkező válasz alapján halad tovább a program.
Példa a kimenetre:

```
> ->[1:Door].meetPlayer(player)
>   ->[1:Player].getObtainedKeys()
>   <-[1:Player].getObtainedKeys()
<   "Nyithato az ajto?": y
>   ->Playfield.win()
>     ->Game.loadNextLevel()
//...
>     <-Game.loadNexrLevel()
>   <-PlayField.win()
> <-[1:Door].meetPlayer(player)
```

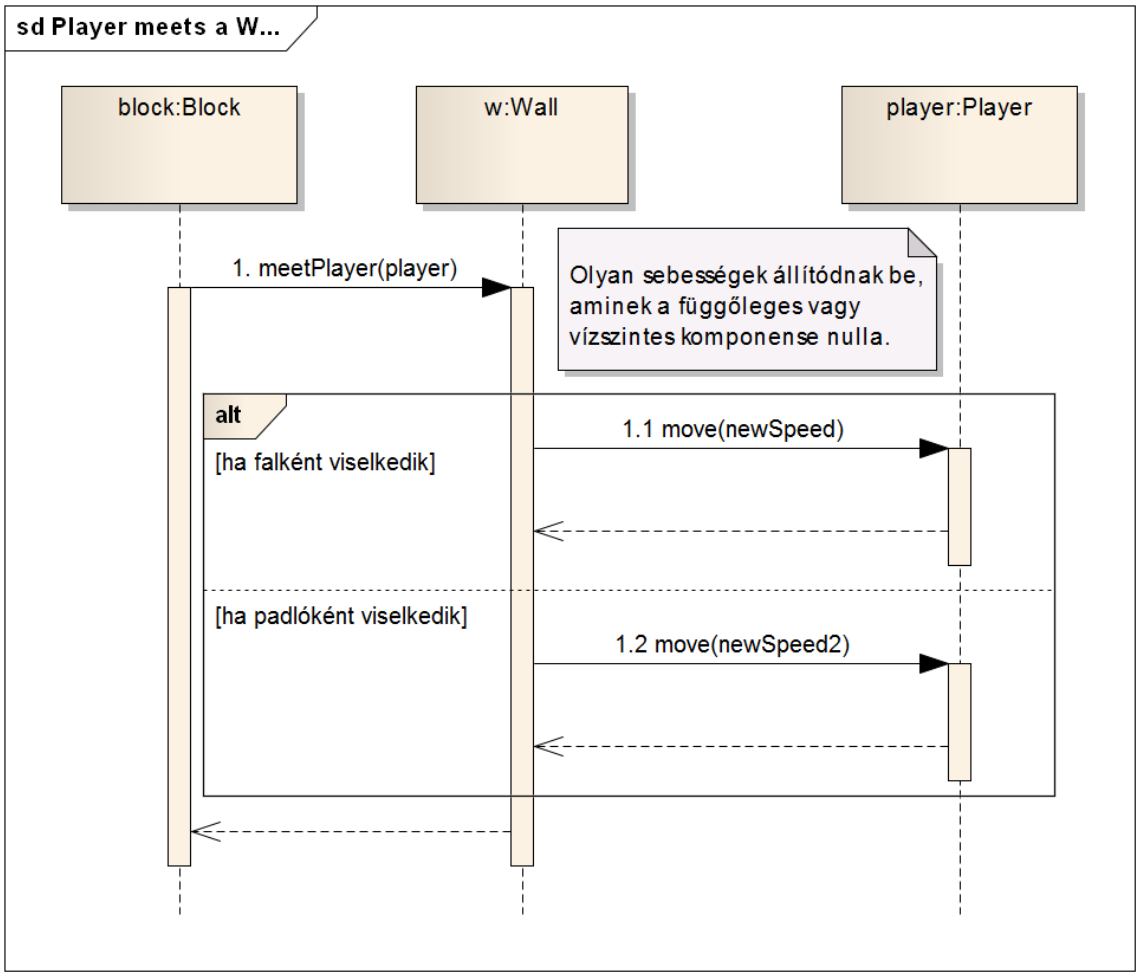
A függvényhívásokat jobbra mutató nyíl jelöli, a visszatéréseket egy balra mutató. Az egymásba ágyazott hívások a behúzásokkal jól követhetőek. Az elágazásokat jelképező

kérdések idézőjelben vannak, válaszolni rájuk tipikusan 1-2 karakterrel lehet. Az egyes metódusok neve elé odakerül az osztály neve (így egyértelmű marad a kimenet), valamint az egyszerűsített paraméterlistája.

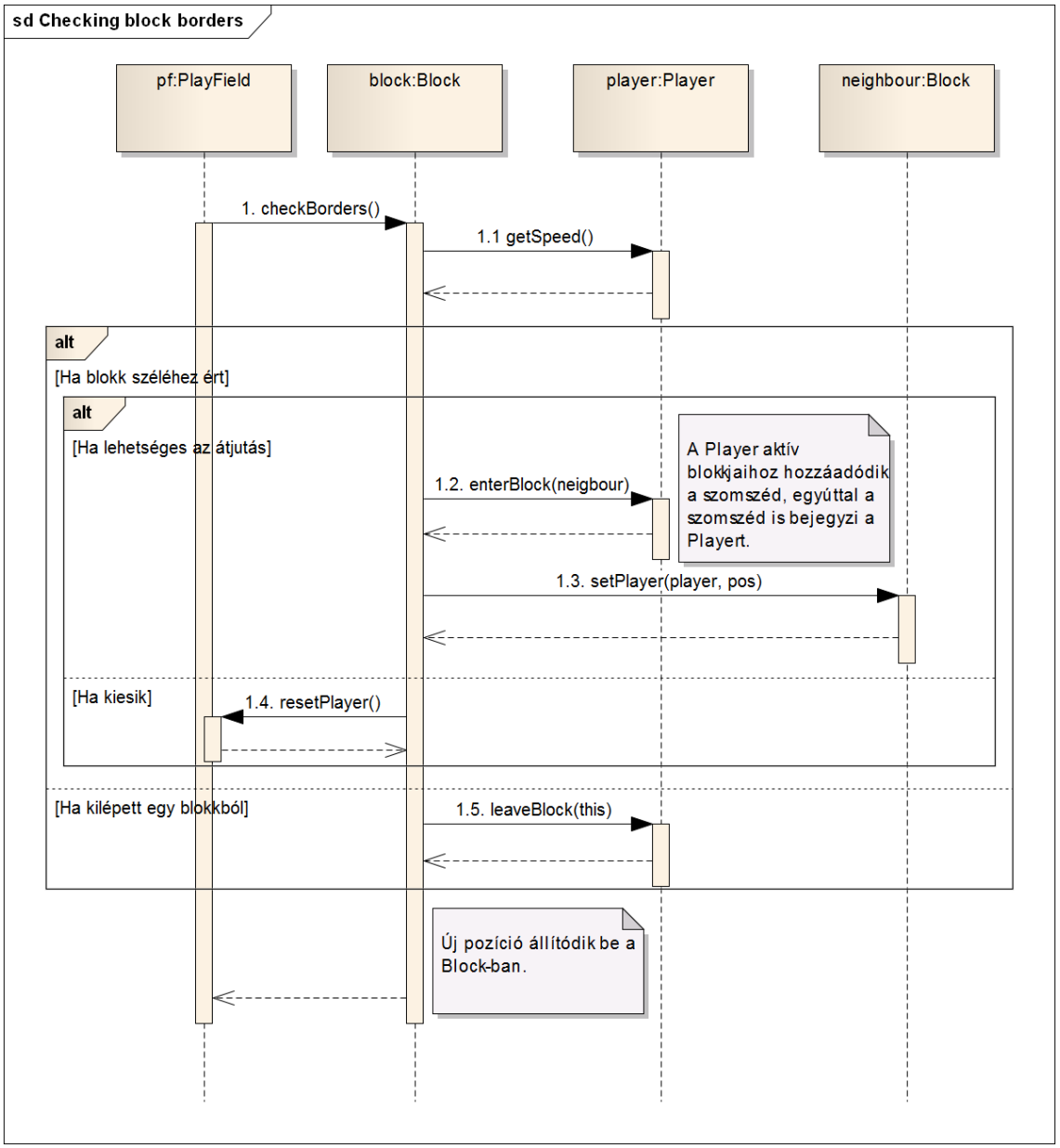
5.4. Szekvencia diagramok a belső működésre

A negyedik fejezet szekvenciadiagramjai ennél a fejezetnél is érvényesek, az alábbi kiegészítésekkel:

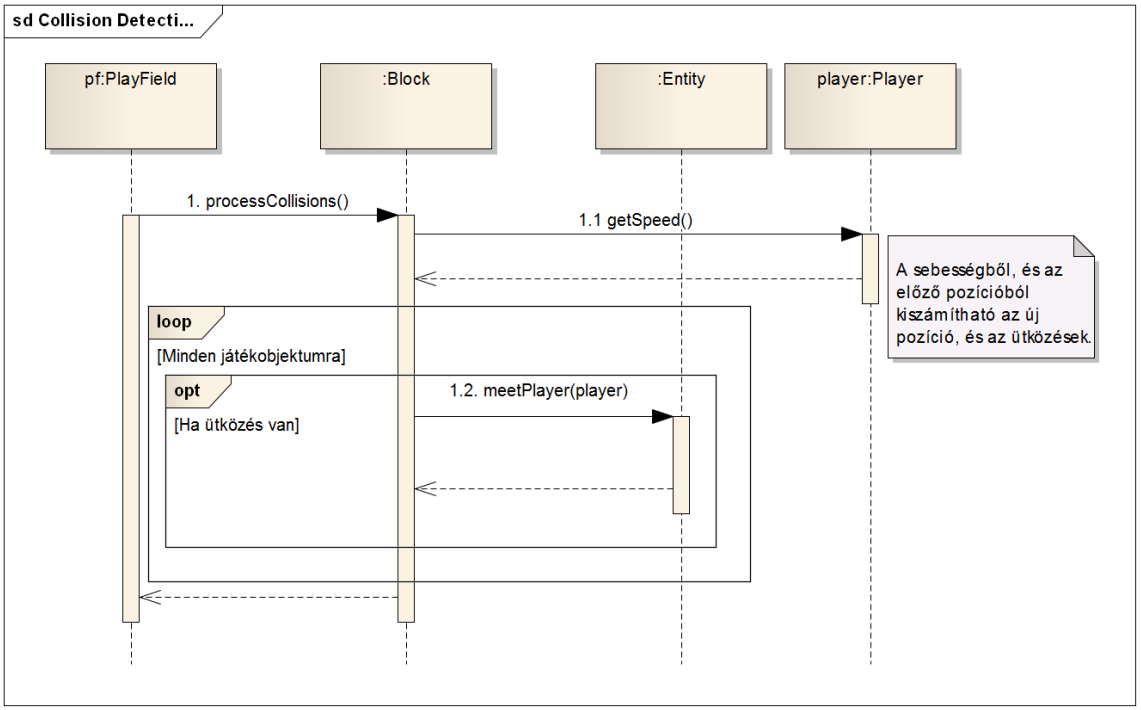
5.4.1. A játékos fallal ütközik



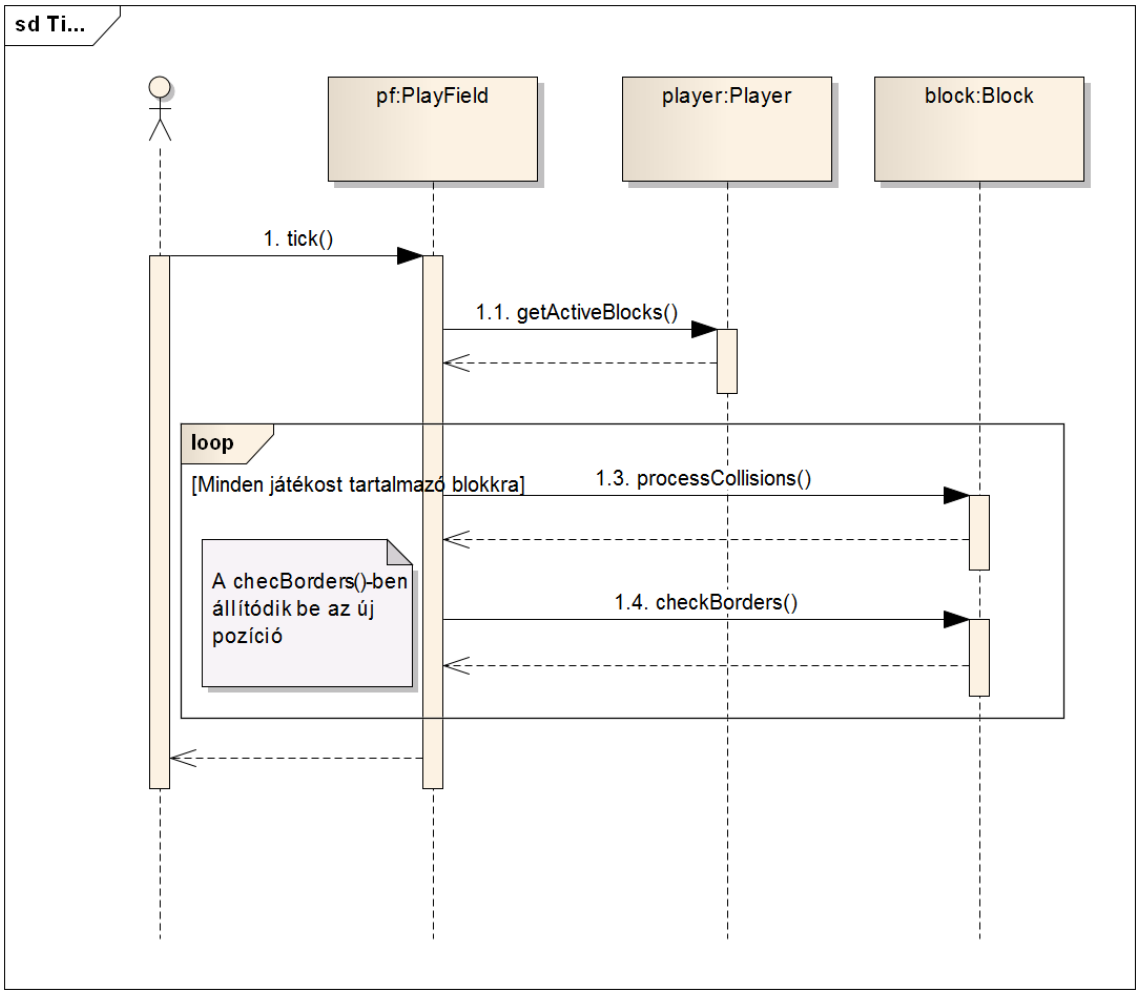
5.4.2. Blokkok közötti mozgás



5.4.3. Ütközésdetektálás



5.4.4. Idő léptetése



5.5. Napló

Kezdet	Időtartam	Résztvevők	Leírás
2012.03.07. 18:45	0,5 óra	Dudás Kern Kolozsi	Gyors megbeszélés
2012.03.08. 20:00	3 óra	Dányi Kern Kolozsi	Use-casek összeszedése, Diagram elkészítése
2012.03.10. 16:00	2 óra	Kern	Use-Case leírások és kimenet szerkesztése
2012.03.11. 21:00	3 óra	Dányi Kern	Szekvenciadiagramok és a szkeleton ki/bemenetének specifikálása.Tesztesetek megadása.

6. Szkeleton

6.1. Fordítási és futtatási útmutató

6.1.1. Fájllista

Fájl neve	Méret	Keletkezés Ideje	Tartalom
src/fearlesscode/util	1298 bájt	2012.03.18. 21:04:55	Logger.java-t tartalmazó mappa
src/fearlesscode	15109 bájt	2012.03.15. 13:07:32	Forrásfile-okat tartalmazó mappa
Block.java	4135 bájt	2012.03.15. 19:04:55	Blokkot reprezetáló absztrakt ősosztály
compile.bat	219 bájt	2012.03.18. 15:42:25	A kódot lefordító szkript
compile&run.bat	11 bájt	2012.03.18. 18:22:54	A kódot lefordító és futtató szkript
Door.Java	1069 bájt	1012.03.17. 16:42:53	Entitás leszármazott, az ajtót jelképezi
EmptyBlock.java	857 bájt	2012.03.15. 16:52:41	Block leszármazott, üresblokként funkcionál
Entity.java	848 bájt	2012.03.17. 11:24:55	Entitás ősosztály, a különböző játékelemek őse
EntityContainer.java	350 bájt	2012.03.16. 17:32:31	Entitásokat tartalmazó osztály
EntityPosition.java	117 bájt	2012.03.16. 17:52:26	Entitások helyzetét adja meg
FilledBlock.java	1649 bájt	2012.03.15. 14:25:76	A játék blokkjait reprezentálja
Game.java	1020 bájt	2012.03.14. 15:25:09	Játék inicializálását kezdeményező osztály
Key.java	840 bájt	2012.03.17. 12:54:52	Entitás leszármazott, a kulcsot jelképezi
Logger.java	1661 bájt	2012.03.18. 21:10:43	Szkeleton helyes működéséért felelős. Kiírja a képernyőre ha függvényhívás vagy visszatérés történik. Elágazásnál a kérdéseket is ez az osztály teszi fel.

Fájl neve	Méret	Keletkezés Ideje	Tartalom
PlayField.java	3083 bájt	2012.03.16. 17:54:00	Játéktér ami a blokkokat tartalmazza
PlayFieldBuilder.java	2233 bájt	2012.03.16. 17:39:21	Az inicializálást végzi
Player.java	2251 bájt	2012.03.16. 17:21:36	Játékost jelképezi
PlayerContainer.java	423 bájt	2012.03.17. 15:02:33	A játékos pozícióját tároló osztály
Position.java	72 bájt	2012.03.18. 18:54:23	Blokkok pozícióját tárolja

6.1.2. Fordítás

A fordítás legegyszerűbben parancssorból történhet. A fordításhoz a parancssorból elérhetőnek kell lennie a javac parancsnak.

A compile.bat hatására elkészülnek a futtatáshoz szükséges .class fájlok is a forrásokhoz így futtathatóvá válik a program.

6.1.3. Futtatás

A run.bat futtatásával elindítható a már lefordított program. A parancssorból elérhetőnek kell lennie a java parancsnak.

6.2. Értékelés

Tag neve	**Munka százalékban**
Dányi Bence	36%
Dudás Zsolt	11%
Kern Ádám	30%
Kolozsi Tibor	23%

6.3. Napló

Kezdet	**Időtartam**	**Résztvevők**	**Leírás**
2012.03.15. 16:45	0.5 óra	Kolozsi	Kód generálás
2012.03.16. 10:00	2 óra	Kolozsi	Kiosztott osztályok kódolása
2012.03.16. 14:00	2 óra	Dányi	Game és Logger osztályok implementálása
2012.03.18. 09:00	2 óra	Kern	Osztályok kódolása
2012.03.18. 22:00	3.5 óra	Dányi	Fordítás és Futtatás előkészítése, osztályok hibajavítása
2012.03.18. 18:00	3 óra	Dudás	Osztályok kódolása
2012.03.19. 9:00	4.5 óra	Dányi Kern Kolozsi	Szkeleton megvalósítás ellenőrzése

7. Prototípus koncepciója

7.1. Prototípus interface-definíciója

A prototípus feladata a játék tesztelése, grafikus felület nélkül. A program ekkor karakteres felületről kezelhető. A standard bemeneten a program parancsokat vár, így az interfész leginkább egy shellre emlékeztet. A prototípus végrehajtja a megadott parancsot, aminek hatására megjelenik (amennyiben van) a kimeneten az eredménye. A parancsok tetszőleges kombinációjával minden teszteset elkészíthető, így a prototípus képes teljesíteni a feladatát.

A program tesztelése emberi munkával hosszadalmas, így a prototípus fel van készítve az automatikus tesztelésre. Ezt a standard input és output használatával éri el, hiszen ekkor a program bemenete a billentyűzet helyett lehet egy fájl is, és a kimenet a képernyő helyett lehet egy fájl is, így a teszt eredménye könnyen összehasonlítható egy előre elkészített, elvárt eredményt tartalmazó kimenettel, így a modell tesztelése sokkal egyszerűbbé, és gyorsabbá válik.

7.1.1. Az interfész általános leírása

A prototípus a szabványos be és kimeneten kommunikál a felhasználóval, így a bemenetet vagy konzolból (a billentyűzetről) vagy parancsfájlból kapja („kacsacsőr" `<` operátor), a kimenet pedig vagy a képernyőre, vagy tetszőleges fájlba történik (`>` operátor). A tesztpálya betöltése fájlból történik (részletesen 7.1.2)

Alapértelmezetten a program nem ír ki semmilyen általános információt a parancsok végrehajtása után, amennyiben a tesztelő a pálya/blokkok/játékobjektumok/játékosok állapotára kíváncsi, ezt az erre szolgáló parancsokkal megteheti. Így elkerülhető, hogy a túl sok információ miatt átláthatatlan lesz a program kimenete.

7.1.2. Bemeneti nyelv

A program nem tartalmaz véletlen elemeket, így determinisztikusan képes működni. Az ütközések is mindig azonos sorrendben zajlanak le (minden a pálya felépítésekor történő referencia beregisztrálás sorrendjében történik), attól függetlenül, hogy egy „pillanatban" történtek-e. A program nem használ több szálat a játék eseménykezelője futtatásához, a párhuzamosítást nem indokolja semmi, különösen számításigényes algoritmusok nincsenek a programban.

tick

Leírás: Az időt előreléptеti `n` darab lépéssel, vagy 1-el, ha nincs paraméter (`PlayField.tick()` metódus). Példa: `tick 2` (2 lépés történik), `tick` (1 lépés történik)

Opciók: `tick n`

moveBlock

Leírás: A megadott `dir` irányba (0 a fölfele, óramutató járása szerint haladunk 90 fokonként) mozgatja a megadott `id`-vel rendelkező blokkot. Példa: `moveBlock 1 2` (az 1-es id-vel rendelkező blokkot lefelé mozdítja el).

Opciók: `moveBlock id dir`

toggleMode

Leírás: Váltás játék mód és blokk mód között. Példa: `toggleMode` (ha játékmódban vagyunk, akkor blokkmódba lépünk, és fordítva)

Opciók: Nincs

movePlayer

Leírás: A megadott `id`-vel rendelzekő játékosnak ad egy `x`, `y` komponensekkel rendelkező sebességimpulzust. Példa: `movePlayer 0 0 -30` (a 0. játékos ugrik egyet, mivel a koordináták jobbra és lefele nőnek)

Opciók: `movePlayer id x y`

getBlockInfo

Leírás: Kilistáz minden blokkot az aktuális pályán, a hozzájuk kapcsolódó adatokkal (formátum a 7.1.3 fejezetben)

Opciók: Nincs

getEntityInfo

Leírás: Kilistáz minden Entity-t a pályán, a hozzájuk kapcsolódó adatokkal (formátum a 7.1.3 fejezetben)

Opciók: Nincs

getPlayerInfo

Leírás: Kilistáz minden játékost (a specifikáció szerint kettőt), a hozzájuk kapcsolódó adatokkal (formátum a 7.1.3 fejezetben)

Opciók: Nincs

exit

Leírás: Befejezi a tesztelést, és kilép a programból.

Opciók: Nincs

loadMap

Leírás: Betölt a hivatkozott `file` fájlból egy pályát. Az aktuális pálya felülírásra kerül. Példa: `loadMap example.json` (az example.json pályát betölti)

Opciók: `loadMap file`

include

Leírás: Betölti és végrehajtja a hivatkozott fájlt, ami szintén a proto számára érthető parancsokat tartalmaz. A tesztelőnek ügyelnie kell, hogy parancsfájlok ne hivatkozzanak egymásra, ennek ellenőrzését a tesztelőre bízzuk, a prototípusnak nem feladata ennek ellenőrzése.

Opciók: `include file`

Pályát megadó fájlok nyelvtana A pályák egy JSON formátumú fájlból kerülnek betöltésre (`loadMap` parancs), mely formátuma az alábbi:

```
{
    "size":
    {
        "x": 2,
        "y": 2
    }
    "blocks":
    [
        //blokkok leirasa kesobb
    ],
}
```

A gyökér elemben található a pálya mérete (vízszintesen és függőlegesen a blokkok száma), valamint a `blocks` tömbben találhatóak a blokkok felsorolva.

```
{
    "type": "FilledBlock",
    "position":
    {
        "x": 1,
        "y": 1
    },
    "entities":
    [
        //Entitasok leirasa kesobb
    ]
}
```

Egy blokk tartalmazza, hogy a `PlayField`-en belül melyik koordinátán helyezkedik el (`position`, a bal fölső sarok az `(1,1)`, a koordináták jobbra illetve lefele nőnek), a típusát (ami lehet `FilledBlock` vagy `EmptyBlock`), valamint a benne található entitások tömbjét (`entities`).

```
{
    "type": "Key",
    "position":
    {
        "x": 13.2,
        "y": 34.8
    }
}
```

A kulcs objektumnál meg kell adni a pozíciót (`position`, két lebegőpontos szám), mást nem szükséges megadni.

```
{
    "type": "Door",
    "position":
    {
        "x": 88.1,
        "y": 30
    },
    "requiredKeys": 3
}
```

Az ajtónál meg kell adni szintén a pozíciót, valamint a kinyitásához szükséges kulcsok számát (`requiredKeys`).

```
{
    "type": "Wall",
```

```
    "position":
    {
        "x": 0,
        "y": 120.79
    },
    "width": 80.9,
    "height": 30.1
}
```

A fal objektumnál meg kell adni a pozícióját, majd a szélességét és magasságát (`width` és `height` lebegőpontos számok, a fal egy téglalap alakú objektum, komplexebb alakzatokat több fal egymásra helyezésével lehet létrehozni)

```
{
    "type": "SpawnPoint",
    "position":
    {
        "x": 0,
        "y": 120.79
    },
    "playerID": 0
}
```

A SpawnPoint tartalmazza a pozícióját, valamint a játék kezdetén ott feléledő játékos azonosítóját (`playerID`).

7.1.3. Kimeneti nyelv

A parancsokra adandó választ a program a standard kimeneten jeleníti meg, formátuma pedig a következő:

```
['ID':'Objektum típusa'] has 'Esemény leírása és a paraméterei'.
```

Példa

```
[3:Player] has moved to position (10,15).
```

Hiba esetén (értelmezhetetlen parancs, rossz paraméter):

```
Error:'Hiba leírása'
```

Példa:

```
Error: Invalid command
```

Grafikus interface hiányában a pályán található objektum paramétereit szövegesen jelenítjük meg, az alábbi nyelvtan szerint:

```
['ID':'Objektum típusa']
    'Paraméter neve':'Paraméter értéke'
    ...
```

Példa:

```
[3:Player]
    Coordinates:(10,15)
    Obtained keys:5
```

7.2. Összes részletes use-case

Use-case neve	**Teszt inicializálása**
Rövid leírás	A program kezdeti állapotának beállítása, ezután kezdődhet a tesztelés
Aktorok	Tesztelő
Forgatókönyv	Pálya betöltése egy adott File-ból.Ez a `loadMap` paranccsal tehető meg.
Use-case neve	**Léptetés**
Rövid leírás	A program léptetése
Aktorok	Tesztelő
Forgatókönyv	A program időzítőjének egy vagy több egységgel való léptetése a `tick` paranccsal.
Use-case neve	**Teszt befejezése**
Rövid leírás	A tesztelés befejezése
Aktorok	Tesztelő
Forgatókönyv	A program befejezi a működést. Az `exit` parancs után azonnal kilép.
Use-case neve	**Pálcikaember mozgatása**
Rövid leírás	A tesztelő megadja a pálcikaembernek, hogy melyik irányba mozogjon
Aktorok	Tesztelő
Forgatókönyv	A játékos és az irányok kiválasztása, mely a `movePlayer`-el lehetséges.
Use-case neve	**Mód váltás**
Rövid leírás	Váltás blokk mód és játék mód között
Aktorok	Tesztelő
Forgatókönyv	Blokk módból játék módba vagy fordítva vált a tesztelő, ha kiadja a `toggleMode` parancsot.

Use-case neve	**Blokk mozgatása**
Rövid leírás	Üresblokk melletti blokk üresblokk helyére mozgatása
Aktorok	Tesztelő
Forgatókönyv	Egy blokkot a megfelelő irányba mozgat a tesztelő, ha lehetséges, amennyiben beírja a `moveBlock`-ot.
Use-case neve	**Entitás lekérdezés**
Rövid leírás	Entitások információjának lekérdezése
Aktorok	Tesztelő
Forgatókönyv	A pályán levő összes entitás adatainak kiíratása a `getEntityInfo` paranccsal.
Use-case neve	**Játékos lekérdezés**
Rövid leírás	Játékosok információjának lekérdezése
Aktorok	Tesztelő
Forgatókönyv	A két játékos adatainak kiíratása a`getPlayerInfo` paranccsal.
Use-case neve	**Blokk lekérdezés**
Rövid leírás	Blokkok információjának lekérdezése
Aktorok	Tesztelő
Forgatókönyv	A blokkok adatainak kiíratása a`getBlockInfo` beírásával.

7.3. Tesztelési terv

A tesztelés során keletkezett kimenet egy az egyben összevethető egy már előre elkészített elvárt kimenettel, ennek a módja a 7.4 fejezetben található.

Teszt-eset neve	**A pálya beolvasása és kiírása**
Rövid leírás	A program beolvas majd kiír egy pályát (játékossal, ajtóval, kulcsokkal és a falakkal.
Teszt célja	Leellenőrizni, hogy a pálya rendesen (a specifikáltaknak megfelelően) felépül. A teszt a `PlayFieldBuilder.createPlayField()` metódus helyességét teszteli (és a szekvenciadiagramon található többi metódust)
Teszt-eset neve	**Játékos esik**
Rövid leírás	A teszt során a játékos alatt nincs talaj, ekkor a gravitáció miatt lefele kell elmozdulnia, a teszt akkor eredményes, ha a játékos a `PlayField.tick()` metódus lefutása után megfelelően elmozdult (lefelé) a megfelelő pozícióba.
Teszt célja	A gravitáció vizsgálata. A teszt a `PlayField.tick()` és a `Player.move()` metódusokat teszteli

Teszt-eset neve	**Játékos padlóra érkezik**
Rövid leírás	A játékos a teszt elején a padló felett helyezkedik el, a teszt azt vizsgálja, hogy a játékos zuhanását valóban megállítja a fal vízszintes része (padló).
Teszt célja	A fal ütközési logikájának vizsgálata. A teszt a `Wall.meetPlayer()` metódust teszteli

Teszt-eset neve	**Játékos balra lép**
Rövid leírás	Egy felépített pályán a játékos balra mozog. A teszt során a játékos kap egy bal oldali irányba ható impulzust, amit a `PlayField.tick()` lekezel, ezek után pedig megvizsgáljuk, hogy a játékos valóban a megfelelő irányban (a megfelelő helyre) mozdult-e el.
Teszt célja	A pálya funkciójának leellenőrzése, és a játékos balra mozgásának vizsgálata.

Teszt-eset neve	**Játékos jobbra lép**
Rövid leírás	Egy felépített pályán a játékos jobbra mozog. A teszt során a játékos kap egy jobb oldali irányba ható impulzust, amit a `PlayField.tick()` lekezel, ezek után pedig megvizsgáljuk, hogy a játékos valóban a megfelelő irányban (a megfelelő helyre) mozdult-e el.
Teszt célja	A pálya funkciójának leellenőrzése, és a játékos jobbra mozgásának vizsgálata.

Teszt-eset neve	**Játékos ugrik**
Rövid leírás	Egy felépített pályán a játékos ugrik. A teszt során a játékos kap egy felfelé impulzust, amit a `PlayField.tick()` lekezel, ezek után pedig megvizsgáljuk, hogy a játékos valóban a megfelelő irányban (a megfelelő helyre) mozdult-e el.
Teszt célja	A pálya funkciójának leellenőrzése, és a játékos ugró mozgásának vizsgálata.

Teszt-eset neve	**Játékos ajtóval ütközik (kinyílik)**
Rövid leírás	A játékos ütközik az ajtóval, és az ajtó nyitható. A teszt az ajtónyitási logikát teszteli (az ajtónak helyes működés esetén ki kell nyílnia)
Teszt célja	Annak ellenőrzése, hogy az ajtó logikája megfelelő-e. A `Door.meetPlayer()` metódust teszteli.

Teszt-eset neve	**Játékos ajtóval ütközik (nem nyílik ki)**
Rövid leírás	A játékos ütközik az ajtóval, és az ajtó nem nyitható. A teszt az ajtónyitási logikát teszteli (az ajtónak helyes működés esetén nem szabad kinyílnia)
Teszt célja	Annak ellenőrzése, hogy az ajtó logikája megfelelő-e. A `Door.meetPlayer()` metódust teszteli.
Teszt-eset neve	**Játékos kulccsal ütközik (még nem megszerzett)**
Rövid leírás	A játékos összeütközik egy kulccsal, amit eddig még nem szerzett meg egy játékos sem. A teszt végén a kulcs állapotának meg kell változnia, a játékosnál lévő kulcsok számának pedig növekednie kell.
Teszt célja	Annak ellenőrzése, hogy a játékos és a kulcs interrakciója megfelelően lezajlik-e. A `Key.meetPlayer()` metódust teszteli.
Teszt-eset neve	**Játékos fallal ütközik balról**
Rövid leírás	A játékos nekimegy a falnak, és megáll hiszen nem tud tovább abba az irányba (jobbra) haladni. A teszt során a játékos jobbra halad, majd az ütközés után a pozíciójából megvizsgáljuk, hogy valóban megállt-e a falnál (nem haladt át rajta)
Teszt célja	Annak ellenőzése, hogy a játékos és a fal balról történő interrakciója megfelelően lezajlik-e. A `Wall.meetPlayer()` metódust teszteli
Teszt-eset neve	**Játékos fallal ütközik jobbról**
Rövid leírás	A játékos nekimegy a falnak, és megáll hiszen nem tud tovább abba az irányba (balra) haladni. A teszt során a játékos balra halad, majd az ütközés után a pozíciójából megvizsgáljuk, hogy valóban megállt-e a falnál (nem haladt át rajta)
Teszt célja	Annak ellenőzése, hogy a játékos és a fal balról történő interrakciója megfelelően lezajlik-e. A `Wall.meetPlayer()` metódust teszteli
Teszt-eset neve	**Játékos irányítása és blokkmozgatás közötti váltás.**
Rövid leírás	A játékos vált a két mód között. A teszt során a módváltás után ellenőrizzük, hogy valóban megváltozott-e a `PlayField` állapota.
Teszt célja	Annak ellenőrzése, hogy megtörténik e a blokk mozgatás és a játékos irányítás közötti váltás. A `PlayField.toggleMode()` metódust teszteli.
Teszt-eset neve	**Blokk mozgatás (lehetséges)**
Rövid leírás	A játékos mozgat egy blokkot. Valójában az üres blokkot mozgatja ellentétes irányba.
Teszt célja	Annak ellenőrzése, hogy a pálya szerkezete megfelelően változik-e amikor blokk mozgatás történik. A teszt a `PlayField.move()` metódust ellenőrzi.

Teszt-eset neve	**Blokk mozgatás (nem lehetséges)**
Rövid leírás	A játékos mozgat egy blokkot, de nem lehetséges, mivel a játékos éppen két blokk között áll.
Teszt célja	Annak ellenőrzése, hogy a játékos megfelelően letiltja-e a blokkmozgatást, ha blokkhatáron van. A teszt a `PlayField.move()` metódust ellenőrzi.
Teszt-eset neve	**Blokkok közötti mozgás (lehetséges)**
Rövid leírás	A játékos az egyik blokkból áthalad egy szomszédos blokkba.
Teszt célja	Annak ellenőrzése, hogy a játékos megfelelően tud-e blokkok között mozogni. A teszt a `Block.checkBorders()` metódust ellenőrzi.
Teszt-eset neve	**Blokkok közötti mozgás (nem lehetséges)**
Rövid leírás	A játékos az egyik blokkból megpróbál áthaladni egy szomszédos blokkba, de nem tud, mert a két blokk között nem megengedett az átjárás.
Teszt célja	Annak tesztelése, hogy a blokkok közötti mozgás engedélyezésének feltétele megfelelően teljesül-e. A teszt a `Block.checkBorders()` metódust ellenőrzi.
Teszt-eset neve	**Játékos kiesik a blokkból (meghal)**
Rövid leírás	A játékos addig esik míg a blokk széléhez nem ér, de alatta vagy nincs blokk, vagy olyan blokk van amibe nem tud áthaladni.
Teszt célja	Annak ellenőrzése, hogy bekövetkezik e a játékos halála. A teszt a `Block.checkBorders()` és a `PlayField.resetPlayer()` metódusokat teszteli.
Teszt-eset neve	**Játékos átesik egy alattalévő blokkba**
Rövid leírás	A játékos addig esik míg a blokk széléhez nem ér, és alatta olyan blokk van amibe át tud haladni.
Teszt célja	Annak tesztelése, hogy a játékos képes e átmozogni egy alatta lévő blokkba. A teszt a `Block.checkBorders()` metódust ellenőrzi.

7.4. Tesztelést támogató segéd- és fordítóprogramok specifikálása

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

Hiba esetén a program kijelzi, hogy hol, és milyen eltérést talált.

7.5. Naplo

Kezdet	Időtartam	Résztvevők	Leírás
2012.03.22. 15:30	1.5 óra	Kolozsi	E heti feladatok előkészitése, teszt-esetek
2012.03.22. 19:00	3.5 óra	Dányi Dudás Kern Kolozsi	Megbeszélés, Dányi tesztpályákat készít, Dudás a kimeneti nyelvet specifikálja, Kern összegyűjti a use-case-eket, Kolozsi pedig elkészíti a maradék teszt-eseteket
2012.03.25. 13:30	2.5 óra	Dányi	Az interfész általános leírásának elkészítése, a pályát megadó nyelvtan specifikálása, tesztesetek finomítása.
2012.03.25. 14:00	1.5 óra	Kern	Use-case-ek javítása
2012.03.25. 18:00	1 óra	Dudás	Kimeneti nyelv specifikálása
2012.03.25. 22:00	2 óra	Dányi	Tesztelési terv javítása javítása, tesztelést segítő segédprogramok specifikálása. A dokumentáció formázása.

8. Részletes tervek

8.1. Osztályok és metódusok tervei

8.1.1. Block (absztrakt)

Felelőssége
Absztarkt osztály a speciális blokkok leszármaztatásához. A játéktér azonos méretű elemkre osztott darabokból áll. Ezek típusa lehet üres vagy játékblokk. Mindkét verzió egy közös ősből száramazik, mégpedig ebből, az úgynevezett Block osztályból.

Ősosztályok
Nincs

Interfészek
BlockInfo

Attribútumok

#entities: ArrayList<EntityContainer> Egy adott Blockban található Entityk tárolására szolgál.

#neighbours: Block[] A szomszédossági kapcsolatokat írja le a többi a játéktéren található szomszédos Blockokhoz.

#players: ArrayList<PlayerContainer> Egy adott Blockban található Playerek tárolására szolgál.

#playField: PlayField PlayFieldre mutató referencia.

-count: int Privát statikus számláló.

#ID: int Protected azonosító.

Metódusok

+getID(): int A Block ID-jának gettere. Visszatérési érték: A Block ID-ja.

+addEntity(EntityPosition position, Entity entity): void Hozzáad egy objektumot a blokkhoz a megadott helyen. `position` paraméter: Entitás pozicíója. `entity` paraméter: Entitás referenciája.

+getEntities(): ArrayList<EntityContainer> Visszaadja a tartalmazott entitásokat. Visszatérési érték: A blokkban található játékobjektumok

+checkBorders(): void A blokkok közötti mozgást kezeli le.

+getNeighbour(int dir): Block Visszaadja a megadott irányban található szomszédot. A dir egy szám, 0 jelenti az északot, és az óramutató járásával megegyező irányban történik a számozás. `dir` paraméter: A szomszédos Block iránya. Visszatérési érték: Visszaadja a paraméterként meghatározott irányú szomszédot.

+getNeighbours(): Block[] Visszaadja az adott Block szomszédait. Visszatérési érték: A Blockokat tartalmazó tömb.

+processCollisions(): void Lefuttatja az ütközésdetektálást.

+setNeighbour(Block neighbour, int dir, boolean bool): void A megadott szomszédot beállítja a megfelelő irányban a harmadik paraméter függvényében (igaz/hamis) fordított irányban is beállítja a szomszédságot. Ez a harmadik paraméter a végtelen ciklusok elkerülésére van bevezetve. `neighbour` paraméter: A szomszédos block. `dir` paraméter: A szomszédos block iránya. `bool` paraméter: Visszairányba is beállítja e vagy sem.

+setNeighbours(Block[] neighbours): void Felülírja az eddigi szomszédait, és a kapottakat állítja be. `neighbours` paraméter: Az új szomszédok.

+setNeighbours(Block[] neighbours,boolean bool): void Felülírja az eddigi szomszédait, és a kapottakat állítja be. Bool paraméter függvényében visszafele is állítja a szomszédságot. `bool` paraméter: Visszafele is állít e. `neighbours` paraméter: Az új szomszédok.

+addPlayer(Player player, EntityPosition position): void Beállítja a játékosok referenciáját és pozícióját az adott blokkon. `player` paraméter: A player referenciája. `position` paraméter: A player poziciója.

+removePlayer(Player player): void Eltávolítja a játékost a blokkról. `player` paraméter: Az eltávolítandó játékos.

+getPlayers(): ArrayList<PlayerContainer> A Block-ban található Playereket lekérdező metódus. Visszatérési érték: A Block-ban található Playerek.

+matches(Block other, int dir, boolean callback): boolean Absztrakt metódus, ami 2 blokk szélének egyezőségét adja meg. `other` paraméter: A másik blokk. `dir` paraméter: A vizsgálat mely oldalon történjen. `callback` paraméter: Történjen-e visszahívás (másik oldalról is elvégzi a vizsgálatot).

8.1.2. BlockMatcher

Felelőssége
Két blokk egyezőségét (a megfelelő széleken) vizsgálja.

Ősosztályok
Nincs

Interfészek
Nincs

Attribútumok

-block1: Block Az egyik blokk.

-block2: Block A másik blokk.

-dir: int A vizsgálat iránya.

-width: double Az aktuális entitás szélessége.

-height: double Az aktuális entitás magassága.

Metódusok

+matches(): boolean Elvégzi a vizsgálatot. Visszatérési érték: Igaz, ha illeszkedik a két blokk.

-matchesNorth(): boolean Az északi oldal egyezőségét vizsgálja. Visszatérési érték: Igaz, ha illeszkedik a két blokk.

-matchesEast(): boolean A keleti oldal egyezőségét vizsgálja. Visszatérési érték: Igaz, ha illeszkedik a két blokk.

-matchesSouth(): boolean A déli oldal egyezőségét vizsgálja. Visszatérési érték: Igaz, ha illeszkedik a két blokk.

-matchesWest(): boolean A nyugati oldal egyezőségét vizsgálja. Visszatérési érték: Igaz, ha illeszkedik a két blokk.

+setWidth(double w): void Beállítja az aktuális entitás a matchernek a saját szélességét. `w` paraméter: Az entitás szélessége.

+setHeight(double h): void Beállítja az aktuális entitás a matchernek a saját magasságát. `h` paraméter: Az entitás magassága.

8.1.3. EmptyBlock

Felelőssége
Az üres blokkot reprezentálja, gyakorlatilag a Block osztály default (üres) implemetációja.

Ősosztályok
Block

Interfészek
Nincs

Attribútumok

Nincs

Metódusok

+checkBorders(): void Mivel üres blokk, semmi sincs benne, így ez a metódus nem csinál semmit.

+processCollisions(): void Mivel üres blokk, semmi sincs benne, így ez a metódus nem csinál semmit.

+getInfo(Position pos): String Az EmptyBlock információk lekérésére. Visszatérési érték: A saját koordinátái a PlayField-en belül, és a szomszédai.

+getName(): String A név és az ID lekérésére szolgáló metódus. Visszatérési érték: Szögletes zárójelek között visszaadja az ID-t és a nevet. ([ID:név])

+matches(Block other, int dir, boolean callback): boolean Metódus, ami 2 blokk szélének egyezőségét adja meg. `other` paraméter: A másik blokk. `dir` paraméter: A vizsgálat mely oldalon történjen. `callback` paraméter: Történjen-e visszahívás (másik oldalról is elvégzi a vizsgálatot).

8.1.4. FilledBlock

Felelőssége
A sima blokkot reprezentálja, kezeli a benne található objektumokat (objektumok interakciója a játékossal).

Ősosztályok
Block

Interfészek
Nincs

Attribútumok

Nincs

Metódusok

+checkBorders(): void A blokkok kközötti mozgást kezeli le.

+processCollisions(): void Végignézi az összes objektumot, hogy ütközik-e a játékosokkal, ha pedig igen, meghívja a meetPlayer metódusukat.

+getInfo(Position pos): String A FilledBlock információk lekérésére. `pos` paraméter: A FilledBlock poziciója. Visszatérési érték: A Block koordinátáit a PlayField-en belül, melyik Player-ek, és milyen Entity-k tartózkodnak benne, és hogy milyen szomszédai vannak.

+getName(): String A név és az ID lekérésére szolgáló metódus. Visszatérési érték: Szögletes zárójelek között visszaadja az ID-t és a nevet. ([ID:név])

+matches(Block other, int dir, boolean callback): boolean Metódus, ami 2 blokk szélének egyezőségét adja meg. `other` paraméter: A másik blokk. `dir` paraméter: A vizsgálat mely oldalon történjen. `callback` paraméter: Történjen-e visszahívás (másik oldalról is elvégzi a vizsgálatot).

8.1.5. BlockContainer

Felelőssége
Blokkot és a hozzátartozó pozíciót tömörítő wrapper osztály.

Ősosztályok
Nincs

Interfészek
Nincs

Attribútumok

-position: Position A blokkohoz tartozó pozíció.

-block: Block Az adott blokk.

-playField: PlayField A tartalmazó PlayField referenciája.

Metódusok

+getPosition(): Position Position getter.

+getBlock(): Block Block getter.

+setPosition(Position position): void Position setter.

+setBlock(Block block): void Block setter.

+setPlayField(PlayField pf): void Beállítja a tartalmazó PlayFieldet. `pf` paraméter: A tartalmazó PlayField.

+getPlayField(): PlayField Lekéri a tartalmazó PlayFieldet. Visszatérési érték: A tartalmazó PlayField.

8.1.6. PlayerContainer

Felelőssége
A Playert és a Player pozicióját tároló segédosztály. A Block (és leszármazottai) tartalmazzák, így tárolódik el egy-egy játékos pozíciója a blokkon belül.

Ősosztályok
Nincs

Interfészek
Nincs

Attribútumok

-player: Player A tárolt játékos referenciája

-position: EntityPosition A tárolt pozíció referenciája (a játékos pozíciója).

Metódusok

+getPlayer(): Player A tárolt játékos gettere. Visszatérési érték: A tárolt játékos.

+getPosition(): EntityPosition A tárolt pozíció gettere. Visszatérési érték: A tárolt pozíció.

+setPosition(EntityPosition pos): void A tárolt pozíció settere. `pos` paraméter: Az új pozíció.

8.1.7. PlayField

Felelőssége
A pályát (Blokkok összességét) reprezentáló objektum. Ez az objektum tárolja el a pályán található blokkokat, amik közül pontosan egy darab üres. Az ő felelőssége

mozgatni őket, és ellenőrizni a szabályosságát a mozgatásnak, és a játékos által felvett utolsó kulcs pozíciójának (vagy ha ilyen nincs, a kezdőpozíciónak) a tárolása, és a játékos visszahelyzése ebbe a pontba, amennyiben szükséges.

Ősosztályok
Nincs

Interfészek
Nincs

Attribútumok

-blockMode: boolean A PlayField állapota. Blokk módban az értéke igaz, egyébként hamis.

-game: Game A tárolt referencia a Game objektumra, ezt értesíti a pálya megnyerésekor.

-players: ArrayList<PlayerSpawnPoint> A Player-eket tároló lista.

-blocks: ArrayList<BlockContainer> A Block-okat tároló lista.

Metódusok

+addBlock(Position position, Block block): void Egy új blokkot ad hozzá a PlayField objektumhoz a megadott koordinátán. `position` paraméter: A blokk helye. `block` paraméter: Az elhelyezendő blokk.

+move(Block block, int direction): void A megadott blokkot a megadott irányba elmozdítja (kicseréli a megadott irány beli szomszédjával). `block` paraméter: Az elmozdítandó blokk. `direction` paraméter: A mozgatás iránya.

+addPlayer(Player player, Entity spawnPoint): void Egy új játékos referenciát ad hozzá a játékosok listájához. `player` paraméter: Az új játékos referenciája. `spawnPoint` paraméter: Az új játékos SpawnPoint-jának referenciája.

+getPlayers(): ArrayList<PlayerSpawnPoint> Visszaadja a játékosok listáját. Visszatérési érték: A PlayField objektumban eltárolt játékosok és spawnpoint-juk listája.

+getBlock(Position pos): Block Adott Position-ű Block-ot visszaadó metódus. `pos` paraméter: Az adott Position.

+setSpawnPosition(Player player, Entity entity): void Beállítja a megadott játékos újraéledési helyét a pályán belül (egy Entity objektumra). `player`

paraméter: A módosítani kívánt játékos referenciája. `entity` paraméter: Az új újraéledési pont.

+tick(): void A játék-mód eseménykezelője. Minden egyes frissítésre lefuttatja az ütközésdetektálásokat, és a mozgást. (blokkon belüli, és kívüli).

+toggleMode(): void Váltás játék-mód és blokk-mód között. Játék módban a blokkok nem mozognak, blokk módban a tick nem fut.

+win(): void A metódus bejelenti a Game objektumnak, hogy a pálya végére ért a játékos. Ha minden feltétel teljesül, a pálya véget ér, és betöltődik a következő pálya.

+resetPlayer(PlayerContainer player): void A megadott játékost visszahelyezi az újraéledési helyére. `player` paraméter: Az áthelyezni kívánt játékos.

+getBlocks(): ArrayList<BlockContainer> Visszaadja a PlayField blokkjait. Visszatérési érték: A PlayFieldben lévő minden blokk.

+getBlock(int id): Block Visszaadja a megadott azonosítójú blokkot. Ha nincs ilyen, akkor kivétel keletkezik (CommandException‘). `id` paraméter: A blokk azonosítója. Visszatérési érték: A kért blokk.

+getBlockPosition(int id): Position Visszaadja a megadott azonosítójú blokk pozicióját. Ha nincs ilyen, akkor kivétel keletkezik (CommandException‘). `id` paraméter: A blokk azonosítója. Visszatérési érték: A kért blokk pozíciója.

+getPlayer(int id): Player Visszaadja a megadott azonosítójú játékost. Ha nincs ilyen, akkor kivétel keletkezik (CommandException‘). `id` paraméter: A játékos azonosítója. Visszatérési érték: A kért játékos.

+isBlockMode(): boolean Visszaadja, hogy a játéktér blokk-módban van-e. Visszatérési érték: Blokk módban vagyunk-e.

8.1.8. EntityContainer

Felelőssége
Az Entity-t és az Entity pozicióját tároló segédosztály. A Block (és leszármazottai) tartalmazzák, így tárolódik el egy-egy Entity pozíciója a blokkon belül.

Ősosztályok
Nincs

Interfészek
Nincs

Attribútumok

-entity: Entity A tárolt Entity referenciája

-position: EntityPosition A tárolt pozíció referenciája (az Entity pozíciója).

-block: Block A konténer ezen az objektumon belül van.

Metódusok

+getEntity(): Entity A tárolt Entity gettere. Visszatérési érték: A tárolt Entity.

+getPosition(): EntityPosition A tárolt pozíció gettere. Visszatérési érték: A tárolt pozíció.

+getBlock(): Block A tartalmazó objektum gettere. Visszatérési érték: A konténert tartalmazó objektum.

+setBlock(Block block): void A tartalmazó objektum settere. `block` paraméter: A beállítandó objektum.

8.1.9. Game

Felelőssége
A játék fő mozzanatait kezelő osztály. A játék fő mozzanatait (pálya betöltés, játék megnyerése) kezeli. Kezdetben inicializál minden objektumot. Felelőssége a pálya betöltése (és inicializálása).

Ősosztályok
Nincs

Interfészek
Nincs

Attribútumok

-playField: PlayField A tárolt PlayField referenciája.

Metódusok

+loadNextLevel(): void Betölti a következő pályát. Az aktuális pálya ismeretében megkeresi a következő pálya specifikációját, és legyártatja a PlayFieldBuilder osztállyal.

+start(PlayField pf): void Elindítja a játékot. Betölti a megfelelő specifikációval rendelkező pályát, majd elindítja a játékot vele. `pf` paraméter: Az elindítandó pálya.

+getPlayField(): PlayField A tárolt PlayField gettere. Visszatérési érték: A tárolt PlayFiled.

8.1.10. Door

Felelőssége
A pályán megjelenő ajtót reprezentáló objektum. A Door objektum az Entity osztály kiterjesztése így ez is megvalósítja a meetPlayer metódust. Abban az esteben, ha a játékos rendelkezik az összes megszerzett kulccsal, akkor a játék végetér.

Ősosztályok
Entity

Interfészek
Nincs

Attribútumok

-requiredKeys: int Az ajtó kinyitásához szükséges kulcsok száma.

Metódusok

+meetPlayer(PlayerContainer player): void Ha a játékosnál lévő kulcsok száma megegyezik a szükséges kulcsok számával, értesíti a PlayFieldet, hogy a pályának vége. `player` paraméter: Az adott Player objektum, amelyikkel a Door objektum találkozott.

+getInfo(EntityPosition pos): String A Door információinak lekérése. Visszatérési érték: A saját koordinátái a Block-on belül, elvárt kulcsok száma.

+getName(): String A név és az ID lekérésére szolgáló metódus. Visszatérési érték: Szögletes zárójelek között visszaadja az ID-t és a nevet. ([ID:név])

+getBoundingBox(): Rectangle Visszaadja a befoglaló dobozt. Visszatérési érték: Az objektum befoglaló doboza.

8.1.11. Entity (absztrakt)

Felelőssége
Az Entity egy blokkon található statikus (nem mozgó) objektumot reprezentál, minden leszármazottnak kötelessége viselkedést definiálni a játékossal való találkozásra.

Ősosztályok
Nincs

Interfészek
Info, Collideable

Attribútumok

-count: int Privát statikus számláló.

#ID: int Protected azonosító.

#block: Block Referencia a Block-ra.

#playField: PlayField PlayField referencia, a leszármazottak a viselkedés leírásának megkönnyítése érdekében felhasználhatják.

#container: EntityContainer A befoglaló konténer.

Metódusok

+getContainer(): EntityContainer Visszaadja a tartalmazó objektumot. Visszatérési érték: Az objektum, amiben az Entity van.

+setContainer(EntityContainer con): void Beállítja a konténer objektumot. `con` paraméter: Az objektum, ami tartalmazza az Entity-t.

+setBlock(Block block): void A tartalmazó Block referenciáját beállító metódus.

+meetPlayer(PlayerContainer p): void Kötelezően implementálandó metódus, a játékossal való találkozás forgatókönyvét írja le. `p` paraméter: A Player tárolója.

+getID(): int Az azonosító gettere.

+accept(BlockMatcher matcher): void A visitor pattern alapján egy BlockMatcherrel kommunikál. Alapértelmezetten kivételt dob, Ezzel jelezve, hogy az illeszkedés vizsgálatában nincs szerepe. `matcher` paraméter: A kommunikációt kezdeményező objektum.

8.1.12. Key

Felelőssége
Egy kulcsot reprezentál, amit a játékosnak föl kell szednie. Amennyiben játékossal találkozik, és még nem szerezte meg a kulcsot (ez a saját belsőállapota), akkor értesíti a játékost a fölvételről.

Ősosztályok
Entity

Interfészek
Nincs

Attribútumok

-isObtained: boolean A kulcs állapotát leíró boolean. Ha true akkor a kulcs már fel lett véve. Ha false akkor még nem lett felvéve.

Metódusok

+meetPlayer(PlayerContainer player): void Ha a játékos ütközik egy kulcs objektummal, akkor a megszerzettsége állapotától függően megnöveli a játékos kulcsainak a számát. `player` paraméter: A játékos, akivel a kulcs interakcióban van.

+getInfo(EntityPosition pos): String A Key információinak lekérése. `post` paraméter: A Key pozíciója. Visszatérési érték: A saját koordinátái a Block-on belül, és hogy felvették-e.

+getName(): String A név és az ID lekérésére szolgáló metódus. Visszatérési érték: Szögletes zárójelek között visszaadja az ID-t és a nevet. ([ID:név])

+getBoundingBox(): Rectangle Visszaadja a befoglaló dobozt.

8.1.13. SpawnPoint

Felelőssége
Az Entity osztály üres implementációja, a játékosok alapértelmezetten ennek az osztály egy példányának a helyén jönnek létre.

Ősosztályok
Entity

Interfészek
Nincs

Attribútumok

Nincs

Metódusok

+meetPlayer(PlayerContainer player): void Ha a játékos egy SpawPointtal ütközik, nem történik semmi, így a metódus implementációja teljesen üres. `player` paraméter: A játékos, akivel ütközik az objektum.

+getInfo(EntityPosition pos): String A SpawnPoint információinak lekérése. `pos` paraméter: A SpawnPoint poziciója. Visszatérési érték: A saját koordinátája a Block-on belül.

+getName(): String A név és az ID lekérésére szolgáló metódus. Visszatérési érték: Szögletes zárójelek között visszaadja az ID-t és a nevet. ([ID:név])

+getBoundingBox(): Rectangle Visszaadja a befoglaló dobozt.

8.1.14. CommandException

Felelőssége
Hibás parancs esetén ilyen kivétel keletkezik

Ősosztályok
Exception

Interfészek
Nincs

Attribútumok

Nincs

Metódusok

Nincs

8.1.15. Wall

Felelőssége
Falat reprezentáló osztály.

Ősosztályok
Entity

Interfészek
Nincs

Attribútumok

-x: double A fal szélessége.

-y: double A fal magassága.

Metódusok

+meetPlayer(PlayerContainer player): void A fal játékossal való találkozás eseménykezelője. Ha a játékos és a fal találkozási felülete függőleges, úgy az X irányú mozgás szűnik meg a játékos részéről, ha vízszintes, akkor pedig a függőleges (Y). @ param player A játékos, akivel a fal interakcióba került

+getInfo(EntityPosition pos): String A Wall információinak lekérése. `post` paraméter: A Wall pozíciója. Visszatérési érték: A saját koordinátája a Block-on belül, szélessége, magassága.

+getName(): String A név és az ID lekérésére szolgáló metódus. Visszatérési érték: Szögletes zárójelek között visszaadja az ID-t és a nevet. ([ID:név])

+getWidth(): double Visszaadja a fal szélességeét. Visszatérési érték: A fal szélessége.

+getHeight(): double Visszaadja a fal magasságát. Visszatérési érték: A fal magassága.

+getBoundingBox(): Rectangle Visszaadja a befoglaló dobozt.

+accept(BlockMatcher matcher): void A visitor pattern alapján egy BlockMatcherrel kommunikál. Beállítja a saját szélességét, magasságát. `matcher` paraméter: A kommunikációt kezdeményező objektum.

8.1.16. Shape

Felelőssége
Wrapper osztály általános alakzatok leszármaztatásához.

Ősosztályok
Nincs

Interfészek
Nincs

Attribútumok

Nincs

Metódusok

Nincs

8.1.17. EntityPosition

Felelőssége
Az Entity pozicióját tároló segédosztály.

Ősosztályok
Nincs

Interfészek
Nincs

Attribútumok

-x: double A tárolt x koordináta.

-y: double A tárolt y koordináta.

Metódusok

+getX(): double A tárolt x koordinata gettere. Visszatérési érték: A tárolt x koordinata.

+getY(): double A tárolt y koordinata gettere. Visszatérési érték: A tárolt y koordinata.

+setX(double x): void A tárolt x koordinata settere.

+setY(double y): void A tárolt y koordinata settere.

8.1.18. CollisionProcesser

Felelőssége
Ütközések detektálására szolgáló objektum. Az objektum segítségével különböző Shape objektumok ütközését tudjuk meghatározni. A játék során lényegében csak Rectangle objektumokat fogunk ütköztetni, de a modularitásnak köszönhetően ez könnyen bővíthető.

Ősosztályok
Nincs

Interfészek
Nincs

Attribútumok

Nincs

Metódusok

+isCollied(EntityPosition mainPos, Rectangle main, EntityPosition otherPos, R
Két Rectangle objektum ütközéséről dönt. `main` paraméter: A egyik ütköztetendő Rectangle. `other` paraméter: A másik ütköztetendő Rectangle. Visszatérési érték: A visszatérési érték true, ha a két téglalap ütközött, false ha nem.

-isPointInInterval(double point, double intervalStart, double intervalEnd): bool
Egy pont egy adott intervallumba eséséről dönt. `point` paraméter: A vizsgálandó pont. `intervalStart` paraméter: Az intervallum kezdete. `intervalStart` paraméter: Az intervallum vége. Visszatérési érték: True értékkel tér vissza, ha az adott pont a meghatározott intervallumba esik.

-isPointInRectangle(double px, double py, EntityPosition pos, Rectangle rectan
Egy pont egy adott téglalapba eséséről dönt. `px` paraméter: A vizsgálandó pont X koordinátája. `py` paraméter: Az vizsgálandó pont Y koordinátája. `rectangle` paraméter: A vizsgálandó téglalap Visszatérési érték: True értékkel tér vissza, ha az adott pont a meghatározott intervallumba esik.

8.1.19. Rectangle

Felelőssége
Wrapper osztály téglalap tárolásához. Egy-egy téglalapot négy paraméterrel reprezentálunk. Egy pont és a hozzá tartozó szélességi, valamint magassági adatokkal. A téglalap kezdőpontjának a bal felső sarkot tekintjük.

Ősosztályok
Shape

Interfészek
Nincs

Attribútumok

-width: double A téglalap szélességét jelenti.

-height: double A téglalap magasságát jelenti.

Metódusok

+getWidth(): double A szélessége gettere. Visszatérési érték: A téglalap szélessége.

+getHeight(): double A magasság gettere. Visszatérési érték: A téglalap magassága.

8.1.20. Player

Felelőssége
A játékos(okat) reprezentáló osztály. A Player számon tartja magáról, hogy mely blokkokban van jelen, valamint a nála lévő kulcsok számát.

Ősosztályok
Nincs

Interfészek
Info, Collideable

Attribútumok

-speed: Speed A játékos jelenlegi sebessége, ebből számítható a következő poziciója.

-obtainedKeys: int A megszerzett kulcsok száma.

-activeBlocks: ArrayList<Block> A játékos jelenleg ezekben a blokkokban van jelen. Ha ez több egynél, akkor a mozgatása nem feltétlenül lehetséges.

-ID: int A játékos azonosítója.

Metódusok

+getSpeed(): Speed A sebesség gettere. Visszatérési érték: A sebesség.

+addKey(): void A felvett kulcsok számának növelésére szolgáló metódus.

+resetKeys(): void A felvett kulcsok számát nullázza.

+reset(): void Eltávolítja a Playerre vonatkozó referenciákat.

+enterBlock(Block block, EntityPosition pos): void A Block-ba történő beléptetést végző metódus. `block` paraméter: Referencia, hogy melyik Blockba történik a belépés.

+getActiveBlocks(): ArrayList<Block> Az aktív blokkok gettere. Visszatérési érték: Az aktiív blokkok listája.

+getObtainedKeys(): int A felvett kulcsok számának gettere. Visszatérési érték: A felvett kulcsok száma.

+leaveBlock(Block block): void A játékos kilép a megadott blokkból. `block` paraméter: Referecina, hogy melyik Blockból történik a kilépés.

+move(Speed newSpeed): void A játékos sebességét megváltoztatja a megadott mértékben. Ténylegesen nem mozgatja a játékost, csak egy sebességvektort állít. `newSpeed` paraméter: Az új sebesség.

+getInfo(EntityPosition pos): String A Player információinak lekérésére. Visszatérési érték: A saját koordinátája a Block-on belül, és hogy hány kulcs van nála.

+getName(): String A név és az ID lekérésére szolgáló metódus. Visszatérési érték: Szögletes zárójelek között visszaadja az ID-t és a nevet. ([ID:név])

+getID(): int Visszaadja a játékos azonosítóját.

+getBoundingBox(): Rectangle Visszaadja a befoglaló téglalap méretét.

+getNextPosition(EntityPosition current): EntityPosition Az aktuális sebesség alapján visszaadja a következő pozíciót. `current` paraméter: Az aktuális pozíció. Visszatérési érték: A sebesség és a paraméter alapján számolt pozíció.

8.1.21. PlayFieldBuilder

Felelőssége
A PlayField-et felépítő osztály. A PlayField összerakásáért felelős, elkészíti a blokkokat, elhelyezi rajtuk az objektumokat, a játékost, a blokkokat összerendezi a PlayField objektumban.

Ősosztályok
Nincs

Interfészek
Nincs

Attribútumok

Nincs

Metódusok

+createPlayField(Game game, String file): PlayField A PlayField-et felépítő metódus. Felépíti a PlayField-et a megadott file alapján, felhasználva a MapFromJson segédosztályt. A szekvencia diagrammok között megadott 4.4.1-es diagram szerint. A MapFromJson osztályból a pályára vonatkozó összes információ lekérhető get metódusok segítségével, igy ezekből az információkból az PlayField könnyen felépíthető. `game` paraméter: A Game objektum referenciája. `file` paraméter: A pályát tartalmazó file elérési útja.

8.1.22. Speed

Felelőssége
A játékos sebességét reprezentáló osztály.

Ősosztályok
Nincs

Interfészek
Nincs

Attribútumok

-x: double Vízszintes irányú sebesség.

-y: double Függőleges irányú sebesség.

Metódusok

+getX(): double A vízszintes irányú sebességet adja vissza. Visszatérési érték: x Vízszintes irányú sebesség.

+getY(): double A függőleges irányú sebességet adja vissza. Visszatérési érték: y Függőleges irányú sebesség.

8.1.23. PlayerSpawnPoint

Felelőssége
A játékost, és újraéledési helyét tároló segédosztály. Számon tartja, hogy egy-egy játékos mely játékobjektumon köteles újraéledni.

Ősosztályok
Nincs

Interfészek
Nincs

Attribútumok

-player: Player Az objektumban tárolt játékos.

-spawPoint: Entity A játékos újraéledési helye.

Metódusok

+getPlayer(): Player Visszaadja a tárolt játékost. Visszatérési érték: Az eltárolt játékos

+getSpawnPoint(): Entity Visszaadja az újjáéledési helyet. Visszatérési érték: Az eltárolt újraéledési hely

+setSpawnPoint(Entity entity): void Beállít egy új újjáéledési pontot. `e` paraméter: Az új újjáéledési pont.

8.1.24. Position

Felelőssége
Egy blokk pozícióját tárolja a PlayField objektumon belül.

Ősosztályok
Nincs

Interfészek
Nincs

Attribútumok

-x: int Vízszintes pozíció.

-y: int Függőleges pozíció.

Metódusok

+getX(): int A vízszintes pozíció gettere. Visszatérési érték: A tárolt vízszintes pozíció.

+getY(): int A függőleges pozíció gettere. Visszatérési érték: A tárolt függőleges pozíció.

8.1.25. MapFromJson

Felelőssége
A pálya információkat JSON file-ból betöltő osztály. A pályára vonatkozó összes információt lekérdező osztály.

Ősosztályok
Nincs

Interfészek
Nincs

Attribútumok

-rawMap: JSONObject Az egész JSON állományt tartalmazó változó.

Metódusok

-getSize(): JSONObject A Map méretét tároló JSONObject-et lekérdező privát metódus. Visszatérési érték: A Map méretének JSONObject-je.

-getBlocks(): JSONArray A Map Block-jait tároló JSONArray-t lekérdező privát metódus. Visszatérési érték: A Map Block-jainak JSONArray-je.

-getBlock(int numberOfBlock): JSONObject A Mapból egy Block-ot tároló JSONObjectet lekérdező privát metódus. `numberOfBlock` paraméter: A Block sorszáma. Visszatérési érték: A Map egy Block-ját tároló JSONObject.

-getEntity(int numberOfBlock, int numberOfEntity): JSONObject A Map egy Blockjában egy Entity-t tároló JSONObjectet lekérdező privát metódus. `numberOfBlock` paraméter: A Block sorszáma. `numberOfEntity` paraméter: Az Entity sorszáma. Visszatérési érték: A Map Block-jának Entity-jét tároló JSONObject.

+getMapSizeX(): int A Map x irányú méretének lekérdezésére szolgáló metódus. Visszatérési érték: A Map x irányú mérete.

+getMapSizeY(): int A Map y irányú méretének lekérdezésére szolgáló metódus. Visszatérési érték: A Map y irányú mérete.

+getNumberOfBlocks(): int A Map-ben található Block-ok számának lekérdezésére szolgáló metódus. Visszatérési érték: A Map-ben található Block-ok száma.

+isFilledBlock(int numberOfBlock): boolean Egy adott Block- típusának lekérdezésére szolgáló metódus. `numberOfBlock` paraméter: Annak a Block-nak a sorszáma, amiről információt szeretnénk. Visszatérési érték: Az adott Block típusa boolean-ként. Ha FilledBlock akkor true, ha EmptyBlock akkor false.

+getXPositionOfBlock(int numberOfBlock): int Egy adott Block x poziciójának lekérdezésére szolgáló metódus. `numberOfBlock` paraméter: Annak a Block-nak a sorszáma, amiről információt szeretnénk. Visszatérési érték: Az adott Block x poziciója.

+getYPositionOfBlock(int numberOfBlock): int Egy adott Block y poziciójának lekérdezésére szolgáló metódus. `numberOfBlock` paraméter: Annak a Block-nak a sorszáma, amiről információt szeretnénk. Visszatérési érték: Az adott Block y poziciója.

+getNumberOfEntitiesInBlock(int numberOfBlock): int Egy adott Block-ban lévő Entity-k számának lekérdezésére szolgáló metódus. `numberOfBlock` paraméter: Annak a Block-nak a sorszáma, amiről információt szeretnénk. Visszatérési érték: Az adott Block-ban található Entity-k száma.

+getTypeOfEntityInBlock(int numberOfBlock,int numberOfEntity): String Egy adott Block-ban lévő adott Entity típusát lekérdező metódus. `numberOfBlock` paraméter: Annak a Block-nak a sorszáma, amiről információt szeretnénk. `numberOfEntity` paraméter: Annak az Entity-nek a sorszáma, amiről információt szeretnénk. Visszatérési érték: Az adott Entity típusa Stringként. (Wall, Key, Door, SpawnPoint)

+getXPositionOfEntityInBlock(int numberOfBlock, int numberOfEntity): double Egy adott Block-ban lévő adott Entity x pozicióját lekérdező metódus. `numberOfBlock` paraméter: Annak a Block-nak a sorszáma, amiről információt szeretnénk. `numberOfEntity` paraméter: Annak az Entity-nek a sorszáma, amiről információt szeretnénk. Visszatérési érték: Az adott Entity x pozíciója double-ként.

+getYPositionOfEntityInBlock(int numberOfBlock, int numberOfEntity): double Egy adott Block-ban lévő adott Entity y pozicióját lekérdező metódus. `numberOfBlock` paraméter: Annak a Block-nak a sorszáma, amiről információt szeretnénk. `numberOfEntity` paraméter: Annak az Entity-nek a sorszáma, amiről információt szeretnénk. Visszatérési érték: Az adott Entity y pozíciója double-ként.

+getNumberOfRequiredKeys(int numberOfBlock, int numberOfEntity): int Egy adott Door elvárt kulcsainak számát lekérdező metódus. `numberOfBlock` paraméter: Annak a Block-nak a sorszáma, amiről információt szeretnénk. `numberOfEntity` paraméter: Annak az Entity-nek a sorszáma, amiről információt szeretnénk. Visszatérési érték: Az adott Door által elvárt kulcsok száma. Ha nem Door-ról van szó akkor a visszatérési érték -1.

+getPlayerIDofSpawnPoint(int numberOfBlock, int numberOfEntity): int Egy adott SpawnPoint PlayerID-ját lekérdező metódus. `numberOfBlock` paraméter: Annak a Block-nak a sorszáma, amiről információt szeretnénk. `numberOfEntity` paraméter: Annak az Entity-nek a sorszáma, amiről információt szeretnénk. Visszatérési érték: Az adott Door által elvárt kulcsok száma. Ha nem Door-ról van szó akkor a visszatérési érték -1.

+getXPositionOfWallInBlock(int numberOfBlock,int numberOfEntity): double Egy adott Block-ban lévő adott Wall poziciójának x koordinátáját lekérő metódus. `numberOfBlock` paraméter: Annak a Block-nak a sorszáma, amiről információt szeretnénk. `numberOfEntity` paraméter: Annak az Entity-nek a sorszáma, amiről információt szeretnénk. Visszatérési érték: Az adott Wall x koordinátája.

+getYPositionOfWallInBlock(int numberOfBlock,int numberOfEntity): double Egy adott Block-ban lévő adott Wall poziciójának y koordinátáját lekérő metódus. `numberOfBlock` paraméter: Annak a Block-nak a

sorszáma, amiről információt szeretnénk. `numberOfEntity` paraméter: Annak az Entity-nek a sorszáma, amiről információt szeretnénk. Visszatérési érték: Az adott Wall y koordinátája.

+getWidthOfWall(int numberOfBlock,int numberOfEntity): double
Egy adott Block-ban lévő adott Wall szélességét visszaadó metódus. `numberOfBlock` paraméter: Annak a Block-nak a sorszáma, amiről információt szeretnénk. `numberOfEntity` paraméter: Annak az Entity-nek a sorszáma, amiről információt szeretnénk. Visszatérési érték: Az adott Wall szélessége.

+getHeightOfWall(int numberOfBlock,int numberOfEntity): double
Egy adott Block-ban lévő adott Wall magasságát visszaadó metódus. `numberOfBlock` paraméter: Annak a Block-nak a sorszáma, amiről információt szeretnénk. `numberOfEntity` paraméter: Annak az Entity-nek a sorszáma, amiről információt szeretnénk. Visszatérési érték: Az adott Wall magassága.

8.2. A tesztek részletes tervei, leírásuk a teszt nyelvén

Teszteset1 A tesztbemenetekben hivatkozott fájlok tartalma az egyes bemenetek után találhatóak.

Leírás: Az első teszteset egy pályát hoz létre és azt ellenőrzi, hogy minden objektum helyesen jött-e létre.

Ellenőrzött funkcionalitás, várható hibahelyek: Előfordulhat, hogy nem hozza létre a program a megfelelő objektumokat, vagy hibásan hozza őket létre. Az is előfordulhat hogy a blokkok egymáshoz való viszonya hibás.

Bemenet:

```
loadMap maps/level1.json
getBlockInfo
getEntityInfo
getPlayerInfo
exit
```

A tesztpálya tartalma:

```
{
    "size":
    {
        "x": 2,
        "y": 2
    },
    "blocks":
```

```
[
{
    "type": "FilledBlock",
    "position":
    {
        "x": 1,
        "y": 1
    },
    "entities":
    [
    ]
},
{
    "type": "FilledBlock",
    "position":
    {
        "x": 2,
        "y": 1
    },
    "entities":
    [
        {
        "type": "Key",
        "position":
        {
            "x": 150,
            "y": 75
        }
    },
    {
        "type": "Door",
        "position":
        {
            "x": 170,
            "y": 75
        },
        "requiredKeys": 1
    },
    {
        "type": "Wall",
        "position":
        {
            "x": 0,
            "y": 95
        },
        "width": 200,
        "height": 55
    },
    {
```

```
            "type": "SpawnPoint",
            "position":
            {
                "x": 50,
                "y": 75
            },
            "playerID": 1
        },
        {
            "type": "SpawnPoint",
            "position":
            {
                "x": 50,
                "y": 75
            },
            "playerID": 2
        }
        ]
    },
    {
        "type": "EmptyBlock",
        "position":
        {
            "x": 1,
            "y": 2
        }
    },
    {
        "type": "FilledBlock",
        "position":
        {
            "x": 2,
            "y": 2
        },
        "entities":
        [
        ]
    }
  ]
}
```

Elvárt kimenet:

```
[1:FilledBlock]
  Coordinates:(1,1)
  Players:none
  Entities:none
  Neighbours:[none][2:FilledBlock][3:EmptyBlock][none]
```

```
[2:FilledBlock]
  Coordinates:(2,1)
  Players:[1:Player][2:Player]
  Entities:[1:Key][2:Door][3:Wall][4:SpawnPoint][5:SpawnPoint]
  Neighbours:[none][none][4:FilledBlock][1:FilledBlock]
[3:EmptyBlock]
  Coordinates:(1,2)
  Players:none
  Entities:none
  Neighbours:[1:FilledBlock][4:FilledBlock][none][none]
[4:FilledBlock]
  Coordinates:(2,2)
  Players:none
  Entities:none
  Neighbours:[2:FilledBlock][none][none][3:EmptyBlock]
[1:Key]
  Coordinates:(150.0,75.0)
  isObtained:false
[2:Door]
  Coordinates:(170.0,75.0)
  requiredKeys:1
[3:Wall]
  Coordinates: (0.0,95.0)
  Width: 200.0
  Height: 55.0
[4:SpawnPoint]
  Coordinates:(50.0,75.0)
[5:SpawnPoint]
  Coordinates:(50.0,75.0)
[1:Player]
  Coordinates:(50.0,75.0)
  Obtained Keys:0
  Speed:(0.0,0.0)
[2:Player]
  Coordinates:(50.0,75.0)
  Obtained Keys:0
  Speed:(0.0,0.0)
```

8.2.1. Teszteset2

Leírás: Pálcikaember esését vizsgáljuk. Megnézzük, hogy a gravitáció hatására a pálcikaember milyen pozícióba kerül.

Ellenőrzött funkcionalitás, várható hibahelyek: A pálcikaember rossz pozícióba kerülhet egy léptetést követően. Az is lehet hogy egyáltalán nem történik léptetés.

Bemenet:

```
loadMap maps/level2.json
tick 1
getPlayerInfo
exit
```

A tesztpálya tartalma:

```
{
    "size":
    {
        "x": 2,
        "y": 2
    },
    "blocks":
    [
    {
        "type": "FilledBlock",
        "position":
        {
            "x": 1,
            "y": 1
        },
        "entities":
        [
        ]
    },
    {
        "type": "FilledBlock",
        "position":
        {
            "x": 2,
            "y": 1
        },
        "entities":
        [
             {
            "type": "SpawnPoint",
            "position":
            {
                "x": 50,
                "y": 75
            },
            "playerID": 1
        }

        ]
    },
```

```
    {
        "type": "EmptyBlock",
        "position":
        {
            "x": 1,
            "y": 2
        }
    },
    {
        "type": "FilledBlock",
        "position":
        {
            "x": 2,
            "y": 2
        },
        "entities":
        [
        ]
    }
  ]
}
```

Elvárt kimenet:

```
[1:Player] is now at (50.0,76.0) in [2:FilledBlock]
[1:Player]
  Coordinates:(50.0,76.0)
  Obtained Keys:0
  Speed:(0.0,1.0)
```

8.2.2. Teszteset3

Leírás: Azt vizsgáljuk, hogy a játékos esését megállítja-e a padló. A pálya kezdetekor a pálcikaember a padló fölött helyezkedik el és el kezd zuhanni a következő tickre.

Ellenőrzött funkcionalitás, várható hibahelyek: A pálcikaember "belemehet" a padlóba, ha nem működik helyesen az ütközésdetektálás.

Bemenet:

```
loadMap maps/level3.json
tick 2
getPlayerInfo 1
exit
```

A tesztpálya tartalma:

```
{
    "size":
    {
        "x": 2,
        "y": 2
    },
    "blocks":
    [
    {
        "type": "FilledBlock",
        "position":
        {
            "x": 1,
            "y": 1
        },
        "entities":
        [
        ]
    },
    {
        "type": "FilledBlock",
        "position":
        {
            "x": 2,
            "y": 1
        },
        "entities":
        [
        {
            "type": "SpawnPoint",
            "position":
            {
                "x": 50,
                "y": 65
            },
            "playerID": 1
        },
        {
            "type": "Wall",
            "position":
            {
                "x": 0,
                "y": 95
            },
            "width": 200,
            "height": 55
        }
```

```
            ]
        },
        {
            "type": "EmptyBlock",
            "position":
            {
                "x": 1,
                "y": 2
            }
        },
        {
            "type": "FilledBlock",
            "position":
            {
                "x": 2,
                "y": 2
            },
            "entities":
            [
            ]
        },
    ]
}
```

Elvárt kimenet:

```
[1:Player] is now at (50.0,66.0) in [2:FilledBlock]
[1:Player] is now at (50.0,68.0) in [2:FilledBlock]
[1:Player]
  Coordinates:(50.0,68.0)
  Obtained Keys:0
  Speed:(0.0,2.0)
```

8.2.3. Teszteset4

Leírás: Egy felépített pályán a játékos balra mozog. A teszt során a játékos kap egy bal oldali irányba ható impulzust, ezek után pedig megvizsgáljuk, hogy a játékos valóban a megfelelő irányba (a megfelelő helyre) mozdult-e el.

Ellenőrzött funkcionalitás, várható hibahelyek: Lehetséges, hogy a pálcika-ember rossz irányba mozdult el vagy esetleg egyáltalán nem mozdult el.

Bemenet:

```
loadMap maps/level4.json
movePlayer 1 -10 0
tick
getPlayerInfo
exit
```

A tesztpálya tartalma:

```
{
    "size":
    {
        "x": 2,
        "y": 2
    },
    "blocks":
    [
    {
        "type": "FilledBlock",
        "position":
        {
            "x": 1,
            "y": 1
        },
        "entities":
        [
        ]
    },
    {
        "type": "FilledBlock",
        "position":
        {
            "x": 2,
            "y": 1
        },
        "entities":
        [
            {
            "type": "Wall",
            "position":
            {
                "x": 0,
                "y": 95
            },
            "width": 200,
            "height": 55
        },
        {
            "type": "SpawnPoint",
            "position":
```

```
            {
                "x": 50,
                "y": 75
            },
            "playerID": 1
        }
        ]
    },
    {
        "type": "EmptyBlock",
        "position":
        {
            "x": 1,
            "y": 2
        }
    },
    {
        "type": "FilledBlock",
        "position":
        {
            "x": 2,
            "y": 2
        },
        "entities":
        [
        ]
    }
  ]
}
```

Elvárt kimenet:

```
[1:Player] has collided with [1:Wall]
[1:Player] is now at (40.0,75.0) in [2:FilledBlock]
[1:Player]
  Coordinates:(40.0,75.0)
  Obtained Keys:0
  Speed:(-10.0,0.0)
```

8.2.4. Teszteset5

Leírás: Egy felépített pályán a játékos jobbra mozog. A teszt során a játékos kap egy jobb oldali irányba ható impulzust, ezek után pedig megvizsgáljuk, hogy a játékos valóban a megfelelő irányba (a megfelelő helyre) mozdult-e el.

Ellenőrzött funkcionalitás, várható hibahelyek: Lehetséges, hogy a pálcikaember rossz irányba mozdult el vagy esetleg egyáltalán nem mozdult el.

Bemenet:

```
loadMap maps/level5.json
movePlayer 1 10 0
tick
getPlayerInfo
exit
```

A tesztpálya tartalma:

```
{
    "size":
    {
        "x": 2,
        "y": 2
    },
    "blocks":
    [
    {
        "type": "FilledBlock",
        "position":
        {
            "x": 1,
            "y": 1
        },
        "entities":
        [
        ]
    },
    {
        "type": "FilledBlock",
        "position":
        {
            "x": 2,
            "y": 1
        },
        "entities":
        [
            {
            "type": "Wall",
            "position":
            {
                "x": 0,
                "y": 95
            },
            "width": 200,
```

```
            "height": 55
        },
        {
            "type": "SpawnPoint",
            "position":
            {
                "x": 50,
                "y": 75
            },
            "playerID": 1
        }
        ]
    },
    {
        "type": "EmptyBlock",
        "position":
        {
            "x": 1,
            "y": 2
        }
    },
    {
        "type": "FilledBlock",
        "position":
        {
            "x": 2,
            "y": 2
        },
        "entities":
        [
        ]
    }
  ]
}
```

Elvárt kimenet:

```
[1:Player] has collided with [1:Wall]
[1:Player] is now at (60.0,75.0) in [2:FilledBlock]
[1:Player]
  Coordinates:(60.0,75.0)
  Obtained Keys:0
  Speed:(10.0,0.0)
```

8.2.5. Teszteset6

Leírás: Egy felépített pályán a játékos ugrik. A teszt során a játékos kap egy felfelé impulzust, ezek után pedig megvizsgáljuk, hogy a játékos valóban a megfelelő irányban (a megfelelő helyre) mozdult-e el.

Ellenőrzött funkcionalitás, várható hibahelyek: A pálcikaember az ugrást követően hibás pozícióba kerülhet.

Bemenet:

```
loadMap maps/level6.json
movePlayer 1 0 -10
tick
getPlayerInfo
exit
```

A tesztpálya tartalma:

```
{
    "size":
    {
        "x": 2,
        "y": 2
    },
    "blocks":
    [
    {
        "type": "FilledBlock",
        "position":
        {
            "x": 1,
            "y": 1
        },
        "entities":
        [
        ]
    },
    {
        "type": "FilledBlock",
        "position":
        {
            "x": 2,
            "y": 1
        },
        "entities":
        [
            {
```

```
                "type": "Wall",
                "position":
                {
                    "x": 0,
                    "y": 95
                },
                "width": 200,
                "height": 55
            },
            {
                "type": "SpawnPoint",
                "position":
                {
                    "x": 50,
                    "y": 75
                },
                "playerID": 1
            }
            ]
        },
        {
            "type": "EmptyBlock",
            "position":
            {
                "x": 1,
                "y": 2
            }
        },
        {
            "type": "FilledBlock",
            "position":
            {
                "x": 2,
                "y": 2
            },
            "entities":
            [
            ]
        }
      ]
    }
```

Elvárt kimenet:

```
[1:Player] is now at (50.0,66.0) in [2:FilledBlock]
[1:Player]
  Coordinates:(50.0,66.0)
  Obtained Keys:0
```

```
Speed:(0.0,-9.0)
```

8.2.6. Teszteset7

Leírás: A játékos ütközik az ajtóval, és az ajtó nyitható. A teszt az ajtónyitási logikát teszteli.

Ellenőrzött funkcionalitás, várható hibahelyek: Lehetséges hogy az ajtó nem nyílik ki annak ellenére, hogy nincs felszedetlen kulcs a pályán.

Bemenet:

```
loadMap maps/level7.json
movePlayer 1 10 0
tick
exit
```

A tesztpálya tartalma:

```
{
    "size":
    {
        "x": 2,
        "y": 2
    },
    "blocks":
    [
    {
        "type": "FilledBlock",
        "position":
        {
            "x": 1,
            "y": 1
        },
        "entities":
        [
        ]
    },
    {
        "type": "FilledBlock",
        "position":
        {
            "x": 2,
            "y": 1
        },
        "entities":
```

```
        [
        {
            "type": "Door",
            "position":
            {
                "x": 170,
                "y": 75
            },
            "requiredKeys": 0
        },
        {
            "type": "Wall",
            "position":
            {
                "x": 0,
                "y": 95
            },
            "width": 200,
            "height": 55
        },
        {
            "type": "SpawnPoint",
            "position":
            {
                "x": 165,
                "y": 75
            },
            "playerID": 1
        }

        ]
    },
    {
        "type": "EmptyBlock",
        "position":
        {
            "x": 1,
            "y": 2
        }
    },
    {
        "type": "FilledBlock",
        "position":
        {
            "x": 2,
            "y": 2
        },
        "entities":
        [
```

```
        ]
    }
  ]
}
```

Elvárt kimenet:

```
[1:Player] has collided with [1:Door]
[1:Door] has opened.
Level finished!
[1:Player] has collided with [2:Wall]
[1:Player] is now at (175.0,75.0) in [2:FilledBlock]
```

8.2.7. Teszteset8

Leírás: A játékos ütközik az ajtóval, és az ajtó nem nyitható. A teszt az ajtónyitási logikát teszteli.

Ellenőrzött funkcionalitás, várható hibahelyek: Hibás lehet a működés, ha az ajtó nyitható miközben van még felszedetlen kulcs a pályán.

Bemenet:

```
loadMap maps/level8.json
movePlayer 1 10 0
tick
exit
```

A tesztpálya tartalma:

```
{
    "size":
    {
        "x": 2,
        "y": 2
    },
    "blocks":
    [
    {
        "type": "FilledBlock",
        "position":
        {
            "x": 1,
            "y": 1
        },
```

```
        "entities":
        [
        ]
    },
    {
        "type": "FilledBlock",
        "position":
        {
            "x": 2,
            "y": 1
        },
        "entities":
        [
        {
            "type": "Door",
            "position":
            {
                "x": 170,
                "y": 75
            },
            "requiredKeys": 1
        },
        {
            "type": "Wall",
            "position":
            {
                "x": 0,
                "y": 95
            },
            "width": 200,
            "height": 55
        },
        {
            "type": "SpawnPoint",
            "position":
            {
                "x": 165,
                "y": 75
            },
            "playerID": 1
        }

        ]
    },
    {
        "type": "EmptyBlock",
        "position":
        {
            "x": 1,
```

```
            "y": 2
        }
    },
    {
        "type": "FilledBlock",
        "position":
        {
            "x": 2,
            "y": 2
        },
        "entities":
        [
        ]
    }
  ]
}
```

Elvárt kimenet:

```
[1:Player] has collided with [1:Door]
[1:Door] has not opened.
[1:Player] has collided with [2:Wall]
[1:Player] is now at (175.0,75.0) in [2:FilledBlock]
```

8.2.8. Teszteset9

Leírás: A játékos összeütközik egy kulccsal, amit eddig még nem szerzett meg egy játékos sem, és utána mégegyszer megpróbálja felszedni ugyan ezt a kulcsot.

Ellenőrzött funkcionalitás, várható hibahelyek: Lehet , hogy a kulcs állapota nem változik meg vagy a játékos nem veszi fel a kulcsot.

Bemenet:

```
loadMap maps/level9.json
movePlayer 1 10 0
tick
getEntityInfo 1
getPlayerInfo 1
movePlayer 1 -20 0
tick
movePlayer 1 20 0
tick
getEntityInfo 1
getPlayerInfo 1
exit
```

A tesztpálya tartalma:

```
{
    "size":
    {
        "x": 2,
        "y": 2
    },
    "blocks":
    [
    {
        "type": "FilledBlock",
        "position":
        {
            "x": 1,
            "y": 1
        },
        "entities":
        [
        ]
    },
    {
        "type": "FilledBlock",
        "position":
        {
            "x": 2,
            "y": 1
        },
        "entities":
        [
        {
            "type": "Key",
            "position":
            {
                "x": 170,
                "y": 75
            }
        },
        {
            "type": "Wall",
            "position":
            {
                "x": 0,
                "y": 95
            },
            "width": 200,
            "height": 55
        },
        {
```

```
            "type": "SpawnPoint",
            "position":
            {
                "x": 155,
                "y": 75
            },
            "playerID": 1
        }

        ]
    },
    {
        "type": "EmptyBlock",
        "position":
        {
            "x": 1,
            "y": 2
        }
    },
    {
        "type": "FilledBlock",
        "position":
        {
            "x": 2,
            "y": 2
        },
        "entities":
        [
        ]
    }
  ]
}
```

Elvárt kimenet:

```
[1:Player] has collided with [1:Key]
[1:Key] has been picked up.
[1:Player] has collided with [2:Wall]
[1:Player] is now at (165.0,75.0) in [2:FilledBlock]
[1:Key]
  Coordinates:(170.0,75.0)
  isObtained:true
[1:Player]
  Coordinates:(165.0,75.0)
  Obtained Keys:1
  Speed:(10.0,0.0)
[1:Player] has collided with [2:Wall]
[1:Player] is now at (155.0,75.0) in [2:FilledBlock]
```

```
[1:Player] has collided with [1:Key]
[1:Key] has already been picked up.
[1:Player] has collided with [2:Wall]
[1:Player] is now at (165.0,75.0) in [2:FilledBlock]
[1:Key]
  Coordinates:(170.0,75.0)
  isObtained:true
[1:Player]
  Coordinates:(165.0,75.0)
  Obtained Keys:1
  Speed:(10.0,0.0)
```

8.2.9. Teszteset10

Leírás: A játékos nekimegy a falnak, és megáll hiszen nem tud tovább abba az irányba haladni. A teszt során a játékos jobbra halad.

Ellenőrzött funkcionalitás, várható hibahelyek: A játékos belemehet a falba.

Bemenet:

```
loadMap maps/level10.json
movePlayer 1 10 0
tick
getPlayerInfo
exit
```

A tesztpálya tartalma:

```
{
    "size":
    {
        "x": 2,
        "y": 2
    },
    "blocks":
    [
    {
        "type": "FilledBlock",
        "position":
        {
            "x": 1,
            "y": 1
        },
        "entities":
        [
```

```
        ]
    },
    {
        "type": "FilledBlock",
        "position":
        {
            "x": 2,
            "y": 1
        },
        "entities":
        [
        {
            "type": "Wall",
            "position":
            {
                "x": 170,
                "y": 40
            },
            "width": 10,
            "height": 55
        },
        {
            "type": "Wall",
            "position":
            {
                "x": 0,
                "y": 95
            },
            "width": 200,
            "height": 55
        },
        {
            "type": "SpawnPoint",
            "position":
            {
                "x": 155,
                "y": 75
            },
            "playerID": 1
        }

        ]
    },
    {
        "type": "EmptyBlock",
        "position":
        {
            "x": 1,
            "y": 2
```

```
        }
    },
    {
        "type": "FilledBlock",
        "position":
        {
            "x": 2,
            "y": 2
        },
        "entities":
        [
        ]
    }
  ]
}
```

Elvárt kimenet:

```
[1:Player] has collided with [1:Wall]
[1:Player] has collided with [2:Wall]
[1:Player] is now at (155.0,75.0) in [2:FilledBlock]
[1:Player]
  Coordinates:(155.0,75.0)
  Obtained Keys:0
  Speed:(0.0,0.0)
```

8.2.10. Teszteset11

Leírás: A játékos nekimegy a falnak, és megáll hiszen nem tud tovább abba az irányba haladni. A teszt során a játékos balra halad.

Ellenőrzött funkcionalitás, várható hibahelyek: A játékos belemehet a falba.

Bemenet:

```
loadMap maps/level11.json
movePlayer 1 -10 0
tick
getPlayerInfo
exit
```

A tesztpálya tartalma:

```
{
    "size":
    {
        "x": 2,
        "y": 2
    },
    "blocks":
    [
    {
        "type": "FilledBlock",
        "position":
        {
            "x": 1,
            "y": 1
        },
        "entities":
        [
        ]
    },
    {
        "type": "FilledBlock",
        "position":
        {
            "x": 2,
            "y": 1
        },
        "entities":
        [
        {
            "type": "Wall",
            "position":
            {
                "x": 170,
                "y": 40
            },
            "width": 10,
            "height": 55
        },
        {
            "type": "Wall",
            "position":
            {
                "x": 0,
                "y": 95
            },
            "width": 200,
            "height": 55
        },
        {
```

```
            "type": "SpawnPoint",
            "position":
            {
                "x": 185,
                "y": 75
            },
            "playerID": 1
        }

        ]
    },
    {
        "type": "EmptyBlock",
        "position":
        {
            "x": 1,
            "y": 2
        }
    },
    {
        "type": "FilledBlock",
        "position":
        {
            "x": 2,
            "y": 2
        },
        "entities":
        [
        ]
    }
  ]
}
```

Elvárt kimenet:

```
[1:Player] has collided with [1:Wall]
[1:Player] has collided with [2:Wall]
[1:Player] is now at (185.0,75.0) in [2:FilledBlock]
[1:Player]
  Coordinates:(185.0,75.0)
  Obtained Keys:0
  Speed:(0.0,0.0)
```

8.2.11. Teszteset12

Leírás: A játékos vált a két mód között.

Ellenőrzött funkcionalitás, várható hibahelyek: Lehetséges, hogy a program nem reagál a módváltásra.

Bemenet:

```
undefined
```

A tesztpálya tartalma:

```
{
    "size":
    {
        "x": 2,
        "y": 2
    },
    "blocks":
    [
    {
        "type": "FilledBlock",
        "position":
        {
            "x": 1,
            "y": 1
        },
        "entities":
        [
        ]
    },
    {
        "type": "FilledBlock",
        "position":
        {
            "x": 2,
            "y": 1
        },
        "entities":
        [
            {
            "type": "Key",
            "position":
            {
                "x": 150,
                "y": 75
            }
        },
        {
            "type": "Door",
            "position":
            {
```

```
                "x": 170,
                "y": 75
            },
            "requiredKeys": 1
        },
        {
            "type": "Wall",
            "position":
            {
                "x": 10,
                "y": 95
            },
            "width": 200,
            "height": 55
        },
        {
            "type": "SpawnPoint",
            "position":
            {
                "x": 50,
                "y": 75
            },
            "playerID": 1
        },
        {
            "type": "SpawnPoint",
            "position":
            {
                "x": 50,
                "y": 75
            },
            "playerID": 2
        }
        ]
    },
    {
        "type": "EmptyBlock",
        "position":
        {
            "x": 1,
            "y": 2
        }
    },
    {
        "type": "FilledBlock",
        "position":
        {
            "x": 2,
            "y": 2
```

```
        },
        "entities":
        [
        ]
    }
  ]
}
```

Elvárt kimenet:

```
Game mode has been toggled
[1:Block] has moved to Position(1,2)
Game mode has been toggled
[1:Player] has moved to Position(40,75)
[1:FilledBlock]
  Coordinates:(1,1)
  Players:none
  Entities:none
  Neighbours:none[4:Block][3:Block]none
[2:FilledBlock]
  Coordinates:(2,1)
  Players:[1:Player][2:Player]
  Entities:[1:Key][2:Wall][3:SpawnPoint][4:Door]
  Neighbours:[3:Block]nonenone[4:Block]
[3:EmptyBlock]
  Coordinates:(1,2)
  Players:none
  Entities:none
  Neighbours:none[2:Block]none[1:Block]
[4:FilledBlock]
  Coordinates:(2,2)
  Players:none
  Entities::none
  Neighbours:[1:Block]none[2:Block]none
[1:Player]
  Coordinates: (50,75)
  Obtained Keys: 0
[1:FilledBlock]
  Coordinates:(1,1)
  Players:none
  Entities:none
  Neighbours:none[4:Block][3:Block]none
[2:FilledBlock]
  Coordinates:(2,1)
  Players:[1:Player][2:Player]
  Entities:[1:Key][2:Wall][3:SpawnPoint][4:Door]
  Neighbours:[3:Block]nonenone[4:Block]
[3:EmptyBlock]
```

```
  Coordinates:(1,2)
  Players:none
  Entities:none
  Neighbours:none[2:Block]none[1:Block]
[4:FilledBlock]
  Coordinates:(2,2)
  Players:none
  Entities:none
  Neighbours:[1:Block]none[2:Block]none
[1:Player]
  Coordinates: (40,75)
  Obtained Keys: 0
```

8.2.12. Teszteset13

Leírás: A játékos mozgat egy blokkot. Valójában az üres blokkot mozgatja ellentétes irányba.

Ellenőrzött funkcionalitás, várható hibahelyek: Blokk nem mozdul. Lehet az is,hogy a blokk rossz helyre kerül.

Bemenet:

```
loadMap maps/level13.json
moveBlock 1 2
getBlockInfo
exit
```

A tesztpálya tartalma:

```
{
  "size":
  {
    "x": 2,
    "y": 2
  },
  "blocks":
  [
    {
      "type": "FilledBlock",
      "position":
      {
        "x": 1,
        "y": 1
      },
      "entities":
      [
```

```
      ]
    },
    {
      "type": "FilledBlock",
      "position":
      {
        "x": 2,
        "y": 1
      },
      "entities":
      [
        {
          "type": "SpawnPoint",
          "position":
          {
            "x": 50,
            "y": 75
          },
          "playerID": 1
        },
      ]
    },
    {
      "type": "EmptyBlock",
      "position":
      {
        "x": 1,
        "y": 2
      }
    },
    {
      "type": "FilledBlock",
      "position":
      {
        "x": 2,
        "y": 2
      },
      "entities":
      [
      ]
    }
  ]
}
```

Elvárt kimenet:

```
[1:FilledBlock]
  Coordinates:(1,2)
```

```
  Players:none
  Entities:none
  Neighbours:[3:EmptyBlock][4:FilledBlock][none][none]
[2:FilledBlock]
  Coordinates:(2,1)
  Players:[1:Player]
  Entities:[1:SpawnPoint]
  Neighbours:[none][none][4:FilledBlock][3:EmptyBlock]
[3:EmptyBlock]
  Coordinates:(1,1)
  Players:none
  Entities:none
  Neighbours:[none][2:FilledBlock][1:FilledBlock][none]
[4:FilledBlock]
  Coordinates:(2,2)
  Players:none
  Entities:none
  Neighbours:[2:FilledBlock][none][none][1:FilledBlock]
```

8.2.13. Teszteset14

Leírás: A játékos mozgat egy blokkot, de nem lehetséges, mivel a játékos éppen két blokk között áll.

Ellenőrzött funkcionalitás, várható hibahelyek: A blokk mégi elmozdul.

Bemenet:

```
loadMap maps/level14.json
movePlayer 1 10 0
tick
moveBlock 1 2
getBlockInfo
exit
```

A tesztpálya tartalma:

```
{
    "size":
    {
        "x": 2,
        "y": 2
    },
    "blocks":
    [
    {
```

```
        "type": "FilledBlock",
        "position":
        {
            "x": 1,
            "y": 1
        },
        "entities":
        [
          {
            "type": "Wall",
            "position":
            {
                "x": -10,
                "y": 95
            },
            "width": 220,
            "height": 55
        },
        {
            "type": "SpawnPoint",
            "position":
            {
                "x": 185,
                "y": 75
            },
            "playerID": 1
        }
        ]
    },
    {
        "type": "FilledBlock",
        "position":
        {
            "x": 2,
            "y": 1
        },
        "entities":
        [
        {
            "type": "Wall",
            "position":
            {
                "x": -10,
                "y": 95
            },
            "width": 220,
            "height": 55
        }
        ]
```

```
    },
    {
        "type": "EmptyBlock",
        "position":
        {
            "x": 1,
            "y": 2
        }
    },
    {
        "type": "FilledBlock",
        "position":
        {
            "x": 2,
            "y": 2
        },
        "entities":
        [
        ]
    }
  ]
}
```

Elvárt kimenet:

```
[1:Player] has collided with [1:Wall]
[1:Player] has entered [2:FilledBlock]
[1:Player] is now at (195.0,75.0) in [1:FilledBlock]
[1:FilledBlock]
  Coordinates:(1,1)
  Players:[1:Player]
  Entities:[1:Wall][2:SpawnPoint]
  Neighbours:[none][2:FilledBlock][3:EmptyBlock][none]
[2:FilledBlock]
  Coordinates:(2,1)
  Players:[1:Player]
  Entities:[3:Wall]
  Neighbours:[none][none][4:FilledBlock][1:FilledBlock]
[3:EmptyBlock]
  Coordinates:(1,2)
  Players:none
  Entities:none
  Neighbours:[1:FilledBlock][4:FilledBlock][none][none]
[4:FilledBlock]
  Coordinates:(2,2)
  Players:none
  Entities:none
  Neighbours:[2:FilledBlock][none][none][3:EmptyBlock]
```

8.2.14. Teszteset15

Leírás: A játékos az egyik blokkból áthalad egy szomszédos blokkba.

Ellenőrzött funkcionalitás, várható hibahelyek: Pálcikaember nem tud átmenni egyik blokkból a másikba. Esetleg rossz blokkba kerül.

Bemenet:

```
loadMap maps/level15.json
movePlayer 1 10 0
tick
getPlayerInfo 1
tick
getPlayerInfo 1
exit
```

A tesztpálya tartalma:

```
{
    "size":
    {
        "x": 2,
        "y": 2
    },
    "blocks":
    [
    {
        "type": "FilledBlock",
        "position":
        {
            "x": 1,
            "y": 1
        },
        "entities":
        [
          {
            "type": "Wall",
            "position":
            {
                "x": 0,
                "y": 95
            },
            "width": 200,
            "height": 55
        },
        {
            "type": "SpawnPoint",
```

```
            "position":
            {
                "x": 189,
                "y": 75
            },
            "playerID": 1
        }
        ]
    },
    {
        "type": "FilledBlock",
        "position":
        {
            "x": 2,
            "y": 1
        },
        "entities":
        [
        {
            "type": "Wall",
            "position":
            {
                "x": 0,
                "y": 95
            },
            "width": 200,
            "height": 55
        }
        ]
    },
    {
        "type": "EmptyBlock",
        "position":
        {
            "x": 1,
            "y": 2
        }
    },
    {
        "type": "FilledBlock",
        "position":
        {
            "x": 2,
            "y": 2
        },
        "entities":
        [
        ]
    }
```

```
  ]
}
```

Elvárt kimenet:

```
[1:Player] has collided with [1:Wall]
[1:Player] has entered [2:FilledBlock]
[1:Player] is now at (199.0,75.0) in [1:FilledBlock]
[1:Player]
  Coordinates:(199.0,75.0)
  Obtained Keys:0
  Speed:(10.0,0.0)
[1:Player]
  Coordinates:(-10.0,75.0)
  Obtained Keys:0
  Speed:(10.0,0.0)
[1:Player] has left [1:FilledBlock]
[1:Player] has collided with [3:Wall]
[1:Player] is now at (0.0,75.0) in [2:FilledBlock]
[1:Player]
  Coordinates:(0.0,75.0)
  Obtained Keys:0
  Speed:(10.0,0.0)
```

8.2.15. Teszteset16

Leírás: A játékos az egyik blokkból megpróbál áthaladni egy szomszédos blokkba, de nem tud, mert a két blokk között nem megengedett az átjárás.

Ellenőrzött funkcionalitás, várható hibahelyek: A pálcikaember mégis átjut egy másik blokkba. Az is előfordulhat, hogy 2 blokk közöttt ragad.

Bemenet:

```
loadMap maps/level16.json
movePlayer 1 10 0
tick
getPlayerInfo 1
exit
```

A tesztpálya tartalma:

```
{
    "size":
    {
```

```
        "x": 2,
        "y": 2
    },
    "blocks":
    [
    {
        "type": "FilledBlock",
        "position":
        {
            "x": 1,
            "y": 1
        },
        "entities":
        [
          {
            "type": "Wall",
            "position":
            {
                "x": -10,
                "y": 95
            },
            "width": 220,
            "height": 55
        },
        {
            "type": "SpawnPoint",
            "position":
            {
                "x": 185,
                "y": 75
            },
            "playerID": 1
        }
        ]
    },
    {
        "type": "FilledBlock",
        "position":
        {
            "x": 2,
            "y": 1
        },
        "entities":
        [
        {
            "type": "Wall",
            "position":
            {
                "x": -10,
```

```
                "y": 50
            },
            "width": 220,
            "height": 55
        }
        ]
    },
    {
        "type": "EmptyBlock",
        "position":
        {
            "x": 1,
            "y": 2
        }
    },
    {
        "type": "FilledBlock",
        "position":
        {
            "x": 2,
            "y": 2
        },
        "entities":
        [
        ]
    }
  ]
}
```

Elvárt kimenet:

```
[1:Player] has collided with [1:Wall]
[1:Player] has collided with the border of [1:FilledBlock]
[1:Player] is now at (185.0,75.0) in [1:FilledBlock]
[1:Player]
  Coordinates:(185.0,75.0)
  Obtained Keys:0
  Speed:(0.0,0.0)
```

8.2.16. Teszteset17

Leírás: A játékos addig esik míg a blokk széléhez nem ér, de alatta vagy nincs blokk, vagy olyan blokk van amibe nem tud áthaladni.

Ellenőrzött funkcionalitás, várható hibahelyek: A pálcikaember áteshet egy másik blokkba, két blokk között ragadhat vagy esetleg megállhat a blokk határán.

Bemenet:

```
loadMap maps/level17.json
tick 3
exit
```

A tesztpálya tartalma:

```
{
    "size":
    {
        "x": 2,
        "y": 2
    },
    "blocks":
    [
    {
        "type": "FilledBlock",
        "position":
        {
            "x": 1,
            "y": 1
        },
        "entities":
        [
        {
            "type": "SpawnPoint",
            "position":
            {
                "x": 185,
                "y": 125
            },
            "playerID": 1
        }
        ]
    },
    {
        "type": "FilledBlock",
        "position":
        {
            "x": 2,
            "y": 1
        },
        "entities":
        [
        ]
    },
    {
        "type": "EmptyBlock",
```

```
            "position":
            {
                "x": 1,
                "y": 2
            }
        },
        {
            "type": "FilledBlock",
            "position":
            {
                "x": 2,
                "y": 2
            },
            "entities":
            [
            ]
        }
    ]
}
```

Elvárt kimenet:

```
[1:Player] is now at (185.0,126.0) in [1:FilledBlock]
[1:Player] is now at (185.0,128.0) in [1:FilledBlock]
[1:Player] has been reset to [1:SpawnPoint]
```

8.2.17. Teszteset18

Leírás: A játékos addig esik míg a blokk széléhez nem ér, és alatta olyan blokk van amibe át tud haladni.

Ellenőrzött funkcionalitás, várható hibahelyek: A pálcikaember kieshet a pályáról, rossz blokkba kerülhet.

Bemenet:

```
loadMap maps/level18.json
tick 2
getPlayerInfo
exit
```

A tesztpálya tartalma:

```
{
    "size":
    {
        "x": 2,
        "y": 2
    },
    "blocks":
    [
    {
        "type": "FilledBlock",
        "position":
        {
            "x": 1,
            "y": 1
        },
        "entities":
        [
        {
            "type": "SpawnPoint",
            "position":
            {
                "x": 190,
                "y": 128
            },
            "playerID": 1
        }
        ]
    },
    {
        "type": "EmptyBlock",
        "position":
        {
            "x": 2,
            "y": 1
        }
    },
    {
        "type": "FilledBlock",
        "position":
        {
            "x": 2,
            "y": 1
        },
        "entities":
        [
          {
            "type": "Wall",
            "position":
            {
```

```
                "x": 0,
                "y": 50
            },
            "width": 200,
            "height": 100
        }
        ]
    },
    {
        "type": "FilledBlock",
        "position":
        {
            "x": 2,
            "y": 2
        },
        "entities":
        [
        ]
    }
  ]
}
```

Elvárt kimenet:

```
[1:Player] is now at (190.0,129.0) in [1:FilledBlock]
[1:Player] has entered [3:FilledBlock]
[1:Player] is now at (190.0,131.0) in [1:FilledBlock]
[1:Player]
  Coordinates:(190.0,131.0)
  Obtained Keys:0
  Speed:(0.0,2.0)
[1:Player]
  Coordinates:(190.0,-20.0)
  Obtained Keys:0
  Speed:(0.0,2.0)
```

8.3. Tesztelést támogató segéd- és fordítóprogramok specifikálása

8.3.1. Tesztesetek ellenőrzése

Mivel egy-egy teszteset lefutása során teljesen ismert a kimeneti eredmény, s ennek a nyelvezete teljesen specifikált, ezért a könnyebb ellenőrzés végett a tényleges kimenet összehasonlítását a várttal automatizálni lehet. A segédprogramunk egy a diff jellegű

funkciót fog megvalósítani. Nem csak azt határozza meg, hogy a két komparálandó szöveg (mert a kimeneteket szövegként dolgozzuk fel) megegyezik-e, hanem a különbséget is meghatározza.

8.3.2. A program működése

Az segédprogram egy key-value párosításon alapul, amelyben meghatározzuk, hogy milyen bemenetre, milyen kimenetet várunk. Miután a értelmeztük a bemeneti parancsot, tudni fogjuk, hogy milyen kimeneti eredményre kell számítanunk, amellyel majd el is végezzük a komparálást. Az adatok párosítása egyszerű szöveges fájlban található:

```
"bemenet1.txt","kimenet2.txt"
"bemenet2.txt","kimenet2.txt"
...
```

Példa a 'Teszteset1' alapján

Key

```
loadMap level1.json
...
<további parancsok lsd.: Teszteset1 pontos specifikációja>
...
exit
```

Value

```
[1:FilledBlock]
    Players:none
    Keys:none
    Walls:none
    SpawnPoints:none
    Doors:none
    Neighbours:nonenonenonenone
...
<további kimenet lsd.: Teszteset1 pontos specifikációja>
...
```

8.3.3. Felhasználói interfész

Az információ visszacsatolás során egy összevágott kemenetet fog látni a felhasználó, amely tartalmazza tartalmazza mind a várt eredményt, mind pedig a ténylegesen generált kimenetet. Optimális esetben a két szöveg megegyezik, azonban hiba esetén jelezni

kell a különbséget. Ezt a különbséget a sor elején található '++' illetve '–' jelekkel fogjuk szemléltetni. A '++' jel jelenti azt, hogy valami olyan pluszt kaptunk a generált kimenetnél, ami nem volt specifikálva a várt kimenetben. A másik jel, a '–' ennek az ellenkezőjét fogja jelenteni. Olyan sor fog ez jelölni, ahol várnánk egy bizonyos kimenetet, de ez mégse kaptuk meg.

8.3.4. Példa

Várt kimenet

```
[1:FilledBlock]
    Players:none
    Keys:none
    Walls:none
    SpawnPoints:none
    Doors:none
    Neighbours:nonenonenone
```

Teszteset által generált kimenet

```
[1:FilledBlock]
    Players:none
    Keys:none
    Walls:none
    Obtained keys:5
    SpawnPoints:none
    Doors:none
```

Segédprogram által generált kimenet

```
  [1:FilledBlock]
      Players:none
      Keys:none
      Walls:none
++    Obtained keys:5
      SpawnPoints:none
      Doors:none
--    Neighbours:nonenonenone
```

8.4. Napló

Kezdet	Időtartam	Résztvevők	Leírás
2012.03.27. 19:00	2 óra	Dányi Dudás Kern Kolozsi	Feladatok megbeszélése és kiosztása. Dányi elkészíti a tesztelési segédprogramot, és segíti az osztályreferenciák írását, Dudás az ütközésdetektálás megtervezéséért felelős. Kern leírja a teszteket és az elvárt kimeneteket az egyes tesztesetekhez, Kolozsi a pályabetöltés megtervezéséért felelős.
2012.03.28. 14:00	3 óra	Dányi	Osztályreferenciát rendszerező program megírása.
2012.03.29. 16:00	1 óra	Dudás	Ütközésdetektálás megtervezése
2012.03.29. 19:00	1 óra	Dányi Kolozsi	JSON pályabetöltés egyeztetése.
2012.03.29. 20:30	1 óra	Kern Kolozsi	A felmerült kérdések megbeszélése.
2012.03.30. 9:30	3,5 óra	Kolozsi	JSON pályabetöltés megvalósítás, kiosztott osztályok dokumentálása.
2012.03.30. 17:00	2,5 óra	Kern	Tesztesetek megírása
2012.04.01. 19:00	6 óra	Dányi, Dudás	Véglegesítés, javítások
2012.04.01. 22:00	3 óra	Kern	Véglegesítés, javítások
2012.04.02. 10:00	2 óra	Dudás	Tesztprogram specifikálása
2012.04.02. 10:00	4 óra	Dányi Kern Kolozsi	Utolsó ellenörzés, véglegesítés

10. Prototípus

10.1. Fordítási és futtatási útmutató

10.1.1. Fájllista

Fájl neve	Méret	Keletkezés Ideje	Tartalom
src/fearlesscode/util	1298 bájt	2012.03.18. 21:04:55	Logger.java-t és Diff.java-t tartalmazó mappa
src/fearlesscode/org/json	103207 bájt	2012.04.06. 16:00:01	JSON segédosztályokat tartalmazza
src/fearlesscode	15109 bájt	2012.03.15. 13:07:32	Forrásfile-okat tartalmazó mappa
src/fearlesscode/maps	30206 bájt	2012.03.15. 13:07:32	Pályákat tartalmazza
Block.java	5720 bájt	2012.03.15. 19:04:55	Blokkot reprezetáló absztrakt ősosztály
BlockContainer.java	923 bájt	2012.03.22. 19:01:55	Blokkot és pozícióját tárolja
BlockInfo.java	396 bájt	2012.03.30. 12:32:15	Blokk lekérdezéséhez interfész
BlockMatcher.java	6072 bájt	2012.04.11. 14:12:12	Blokkok oldalának egyezését figyeli
Collideable.java	216 bájt	2012.04.05. 20:00:00	Befoglaló dobozhoz interfész
CollisionProcesser.java	2636 bájt	2012.03.23.10:44:23	Ütközések detektálására szolgál
CommandException.java	278 bájt	2012.04.08. 21:56:45	Kivételkezelő osztály
compile.bat	214 bájt	2012.03.18. 15:42:25	A kódot lefordító szkript
compile&run.bat	12 bájt	2012.03.18. 18:22:54	A kódot lefordító és futtató szkript
Diff.Java	3870 bájt	1012.04.10. 22:24:51	Tesztkimenetet hasonlítja össze az elvárt kimenettel
Door.Java	2083 bájt	1012.03.17. 16:42:53	Entitás leszármazott, az ajtót jelképezi
EmptyBlock.java	857 bájt	2012.03.15. 16:52:41	Block leszármazott, üresblokként funkcionál
Entity.java	2124 bájt	2012.03.17. 11:24:55	Entitás ősosztály, a különböző játékelemek őse
EntityContainer.java	1458 bájt	2012.03.16. 17:32:31	Entitásokat tartalmazó osztály
EntityPosition.java	1061 bájt	2012.03.16. 17:52:26	Entitások helyzetét adja meg
FilledBlock.java	5496 bájt	2012.03.15. 14:25:76	A játék blokkjait reprezentálja
Game.java	1260 bájt	2012.03.14. 15:25:09	Játék inicializálását kezdeményező osztály

10.1.2. Fordítás

A fordítás szükséges feltétele a Java SDK 6-os verziója, azaz a parancssorból elérhető „`javac -version`" parancs.

A fordítás Windows platformon a `compile` parancs kiadásával történhet. Ekkor elkészülnek a binárisok a `build/` mappában. Az esetlegesen létező régebbi binárisok törlődnek, így nem fordulhat elő, hogy fordítási hiba esetén a program régebbi verziója fut.

Egyéb platformokon a fordítás az alábbi módon történik (`javac` helyére az aktuális Java fordító kerül):

```
javac -classpath .\src\ -encoding UTF-8 -d .\build\ .\src\fearlesscode\*.java .\src\fearlesscode\uti
```

10.1.3. Futtatás

A futtatás szükséges feltétele a Java 6-os verziója, azaz a parancssorba a „`java -version`" parancsot kiírva a Java kiírja a verziószámát (1.6.x).

A futtatás (a már lefordított binárisok birtokában) a `run` parancs kiadásával történik. A fordítás és a futtatás kényelmesen és gyorsan elvégezhető a `compile\&run` parancs kiadásával.

A futtatás manuálisan is lehetséges, ha kiadjuk az alábbi parancsot (`java` helyére az aktuális Java virtuális gép binárisa kerül):

```
java -classpath .\build\ fearlesscode.Proto
```

10.2. Tesztek jegyzőkönyvei

Mivel részletes (egység)teszteket végeztünk a program írása közben, és a program akkori állapotában nem tudtuk lefuttatni a funkcionális teszteket (hiányzó implementációk), így a program végső (kész) állapotában tudtunk csak erre sort keríteni, így hibás funkcionalitást nem érzékeltünk a tesztek során. Az egyedüli eltérések a kimeneti nyelv nyelvtani, illetve formázási hibái voltak, azokat külön nem tüntettük föl, mivel a modell szempontjából nem relevánsak.

10.2.1. Teszteset1

Tesztelő neve	**Kern**
Teszt időpontja	2012.04.16 10:00:00

10.2.2. Teszteset2

Tesztelő neve	**Kern**
Teszt időpontja	2012.04.16 10:00:00

10.2.3. Teszteset3

Tesztelő neve	**Kern**
Teszt időpontja	2012.04.16 10:00:00

10.2.4. Teszteset4

Tesztelő neve	**Kern**
Teszt időpontja	2012.04.16 10:00:00

10.2.5. Teszteset5

Tesztelő neve	**Kern**
Teszt időpontja	2012.04.16 10:00:00

10.2.6. Teszteset6

Tesztelő neve	**Kern**
Teszt időpontja	2012.04.16 10:00:00

10.2.7. Teszteset7

Tesztelő neve	**Kolozsi**
Teszt időpontja	2012.04.16. 13:00

10.2.8. Teszteset8

Tesztelő neve	**Kolozsi**
Teszt időpontja	2012.04.16. 13:00

10.2.9. Teszteset9

Tesztelő neve	**Kolozsi**
Teszt időpontja	2012.04.16. 13:00

10.2.10. Teszteset10

Tesztelő neve	**Kolozsi**
Teszt időpontja	2012.04.16. 13:00

10.2.11. Teszteset11

Tesztelő neve	**Kolozsi**
Teszt időpontja	2012.04.16. 13:00

10.2.12. Teszteset12

Tesztelő neve	**Kolozsi**
Teszt időpontja	2012.04.16. 13:00

10.2.13. Teszteset13

Tesztelő neve	**Dányi**
Teszt időpontja	2012.04.16. 13:00

10.2.14. Teszteset14

Tesztelő neve	**Dányi**
Teszt időpontja	2012.04.16. 13:00

10.2.15. Teszteset15

Tesztelő neve	**Dányi**
Teszt időpontja	2012.04.16. 13:00

10.2.16. Teszteset16

Tesztelő neve	**Dányi**
Teszt időpontja	2012.04.16. 13:00

10.2.17. Teszteset17

Tesztelő neve	**Dányi**
Teszt időpontja	2012.04.16. 13:00

10.2.18. Teszteset18

Tesztelő neve	**Dányi**
Teszt időpontja	2012.04.16. 13:00

10.3. Értékelés

Tag neve	**Munka százalékban**
Dányi Bence	36%
Dudás Zsolt	11%
Kern Ádám	30%
Kolozsi Tibor	23%

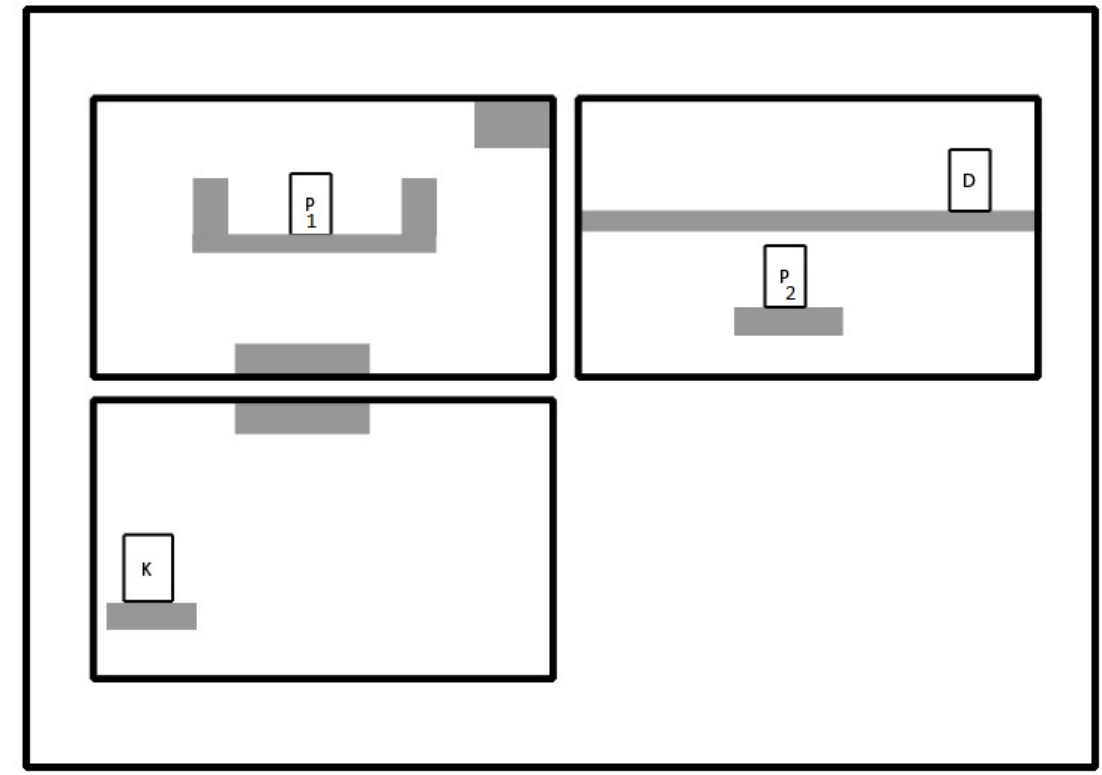
10.4. Napló

Kezdet	Időtartam	Résztvevők	Leírás
2012.04.03. 19:00	0.5 óra	Dányi Dudás Kern Kolozsi	Feladatok megbeszélése és kiosztása.
2012.04.04. 10:00	2 óra	Dányi	Door implementálása, a modellben található fordítási hibák feloldása.
2012.04.04. 16:45	1 óra	Dányi	Game implementálása, Futtatási és Fordítás útmutató megírása.
2012.04.05. 18:00	3 óra	Kolozsi	MapFromJson, PlayFieldBuilder implementálása.
2012.04.05. 14:00	3 óra	Kern	Proto+egyéb osztályok
2012.04.06. 20:00	3 óra	Kolozsi	Block, BlockContainer, EmptyBlock, Entity implementálása.
2012.04.09. 21:00	2 óra	Kern	Proto fix+tesztesetek javítása
2012.04.10. 13:00	3 óra	Dányi	FilledBlock és kapcsolódó osztályok kódolása.
2012.04.11. 16:00	2 óra	Dudás	Osztályok kódolása.
2012.04.12. 19:00	3 óra	Kolozsi	Szintekhez tartozó mapok, log metódusok.
2012.04.13. 12:00	3 óra	Dányi	Blokk illeszkedés vizsgálatának implementálása.
2012.04.13. 17:00	2 óra	Kolozsi	getEntityInfo, getBlockInfo, getPlayerInfo véglegesítés
2012.04.14 16:00	2 óra	Kern	Teszteset kimenet+ debug.
2012.04.14. 20:00	4 óra	Kolozsi	moveBlock hiba keresés, javítás, tesztelés.
2012.04.13. 14:00	4 óra	Dányi	Blokk átmenetek implementálása.
2012.04.13. 21:00	6 óra	Dányi	Debuggolás, tesztelés.
2012.04.15. 19:00	3 óra	Dudás	Diff kódolása.
2012.04.16. 10:00	2 óra	Dudás	Diff debuggolása.
2012.04.16. 10:00	4 óra	Dányi Kolozsi Kern	Debug

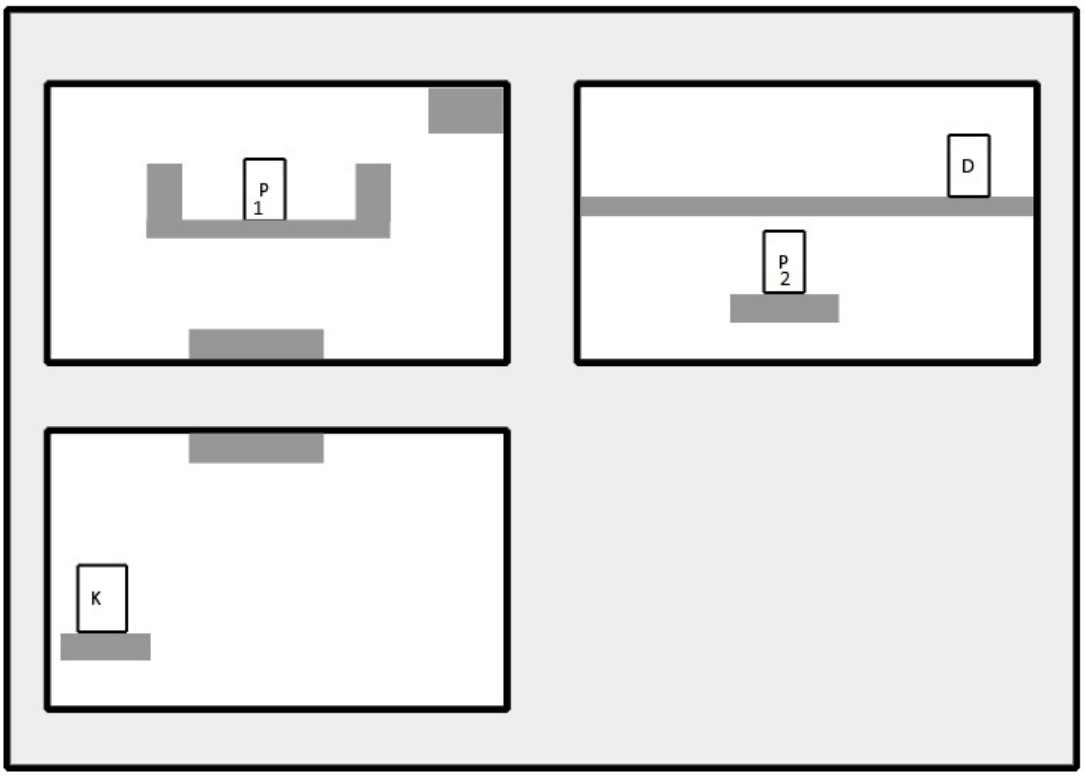
11. Grafikus felület specifikációja

11.1. A grafikus interfész

11.1.1. PLAYERMODE



11.1.2. BLOCKMODE



Blokk módban az egyes blokkok kisebbek lesznek, eltávolodnak egymástól, valamint megvátozik a háttér színe.

11.1.3. MENU

A menüben a fel/le nyilakkal lehet navigálni, az aktuálisan kiválasztott menüpont színe eltérő.

11.1.4. NEXTLEVEL

A pálya végén megjelenik egy statikus képernyő, majd adott idő elteltével a következő pálya.

11.1.5. ENDGAME

A játék végén eltérő képernyő jelenik meg, utána visszalépünk a főmenübe (adott idő elteltével).

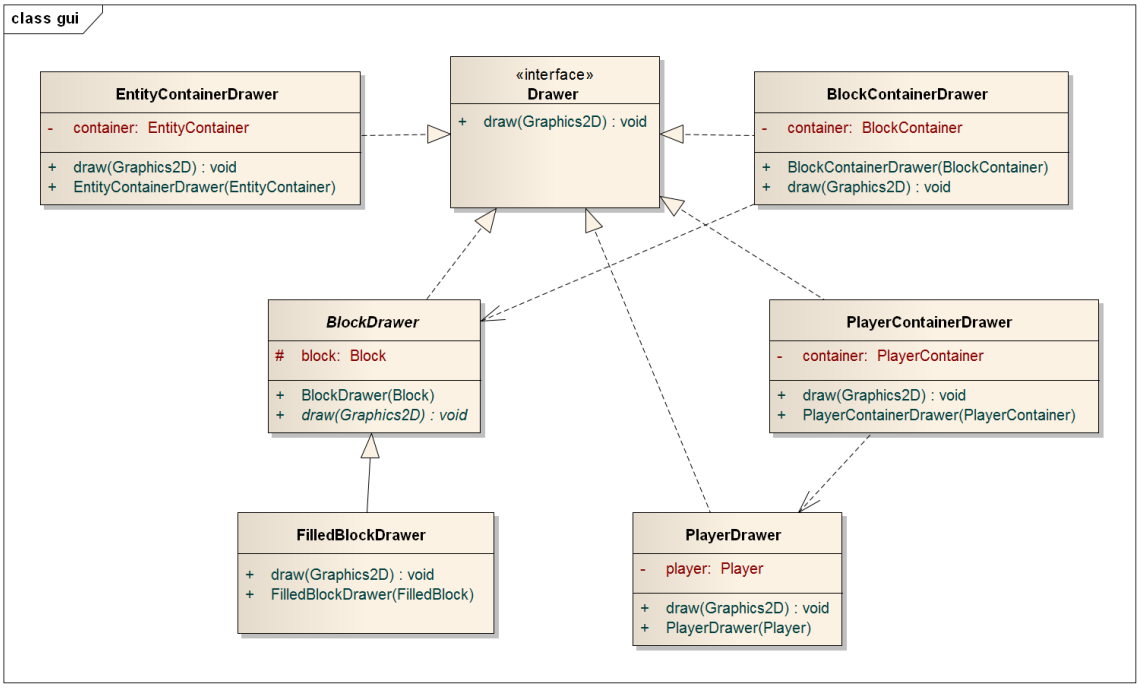
11.2. A grafikus rendszer architektúrája

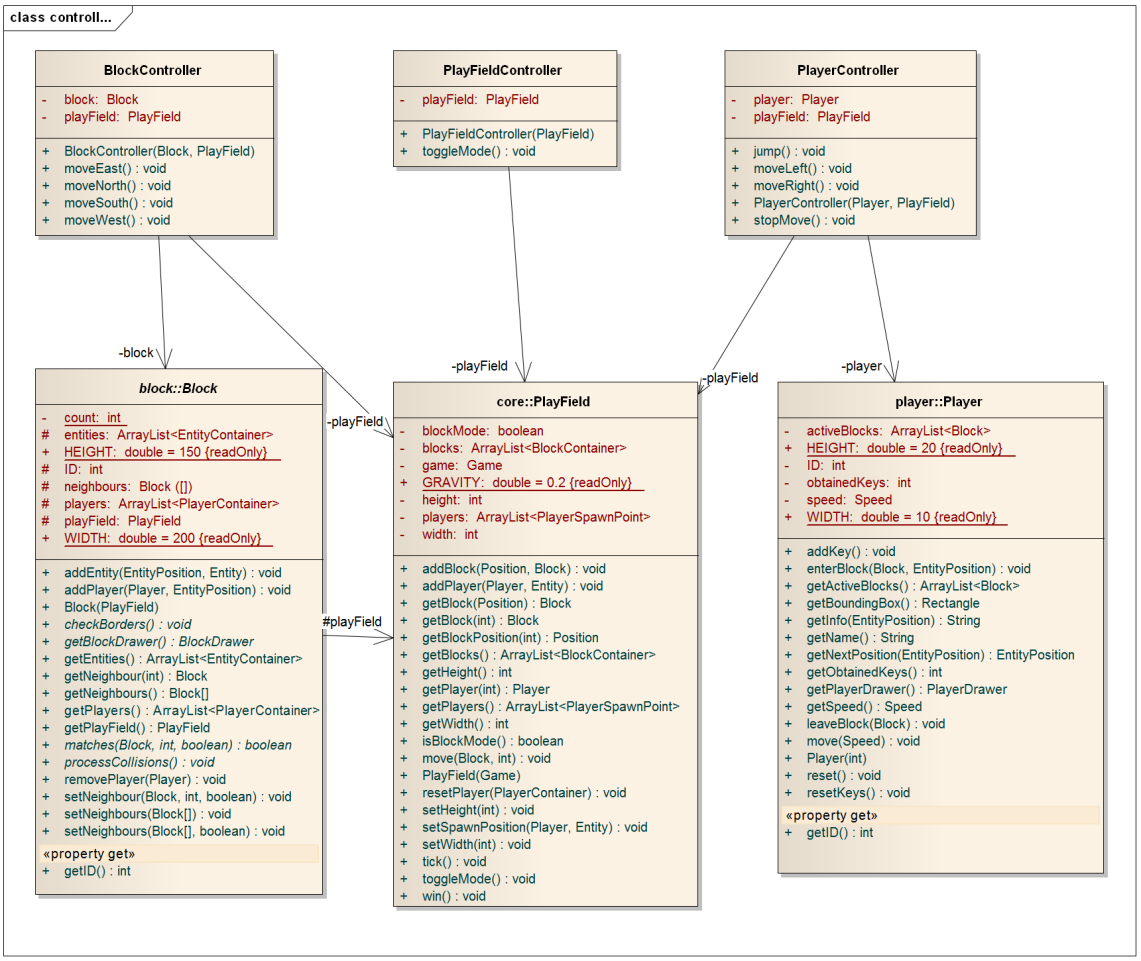
11.2.1. A felület működési elve

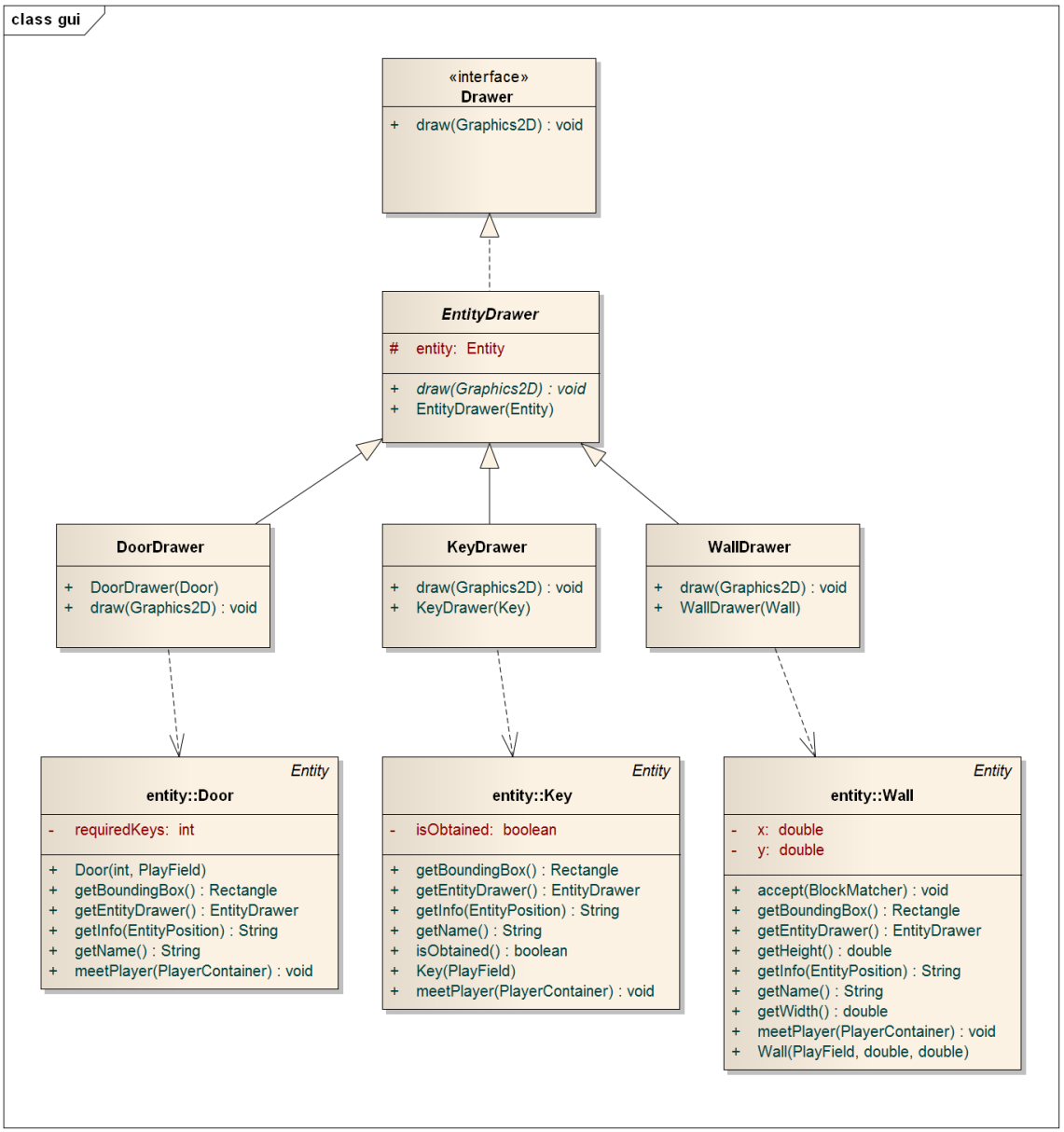
A grafikus rendszer felépítése a pull architektúrát követi, azaz a rendszer a modell állapotát minden újrarajzolásnál lekéri, és ez alapján rajzolja ki a pályát. Hasonlóan a Swing rendszerhez, a kirajzolás rekurzívan történik, erre maga a modell is remek lehetőséget nyújt a hierarchikus fa szerkezetével. Minden objektum a benne található elemek kirajzolásáért felelős, egészen a legalsó szintig, ahol ténylegesen megtörténik a rajzolás.

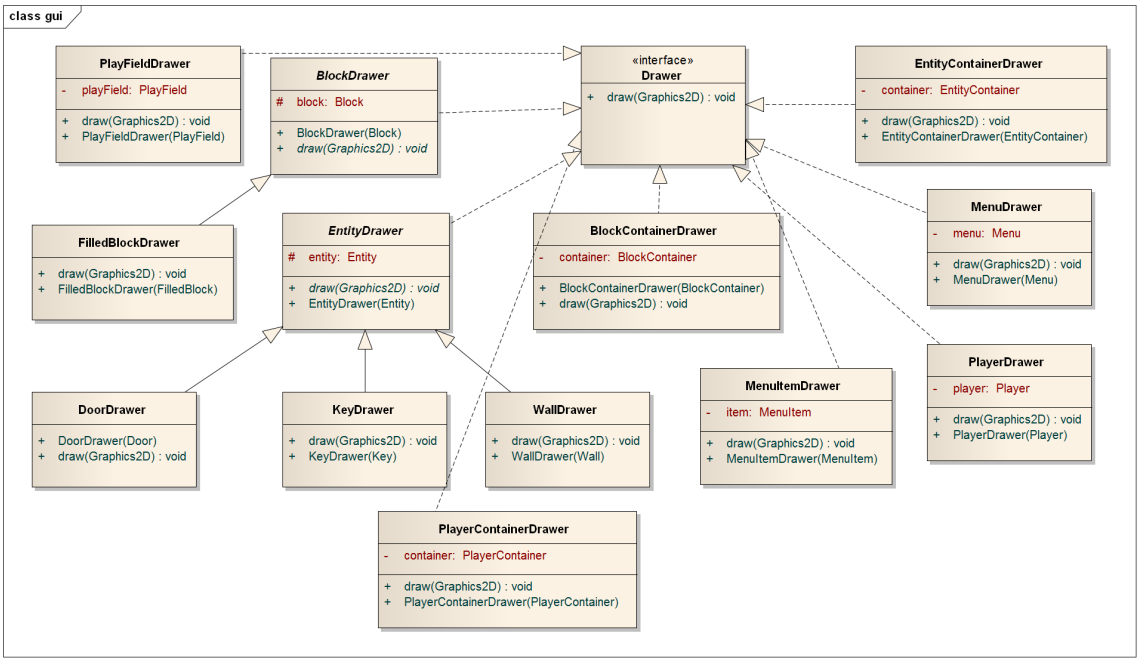
A grafikus alrendszer a natív Java AWT csomagját használja, mivel az minden, a program számára szükséges funkciót implementál: képek rajzolása, transzformációk, stb.

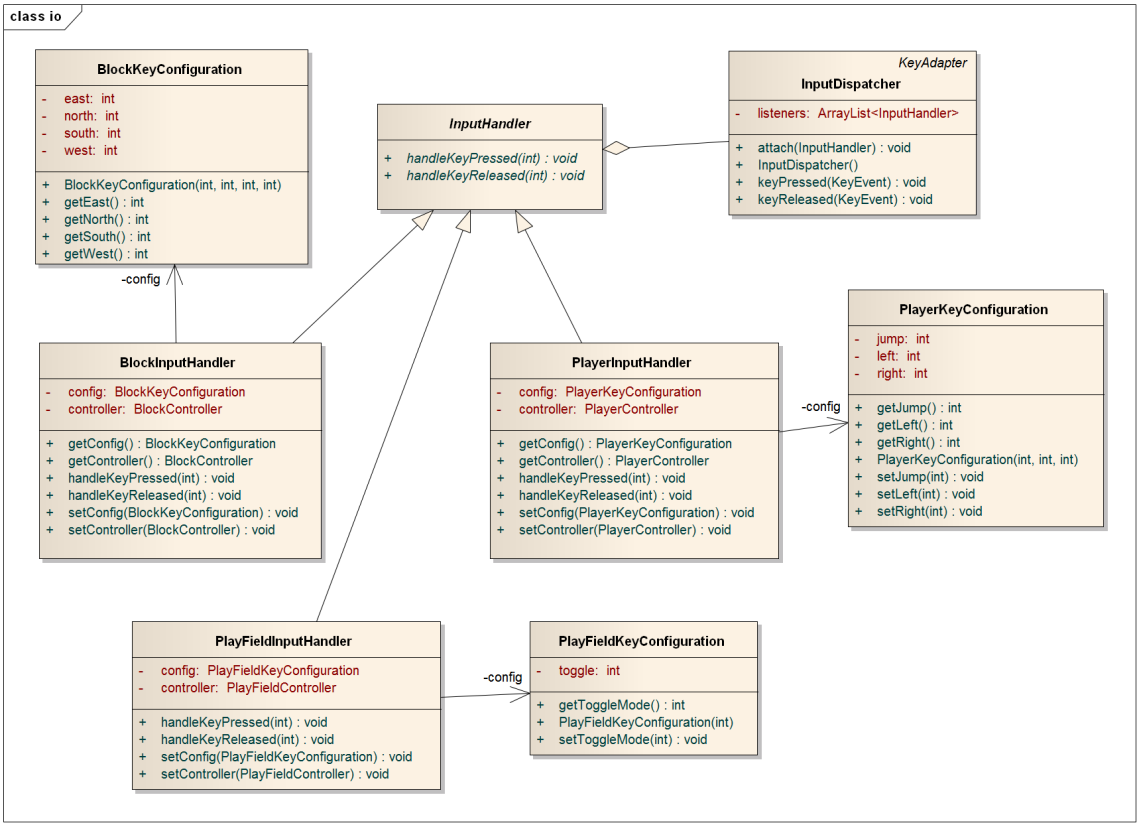
11.2.2. A felület osztály-struktúrája

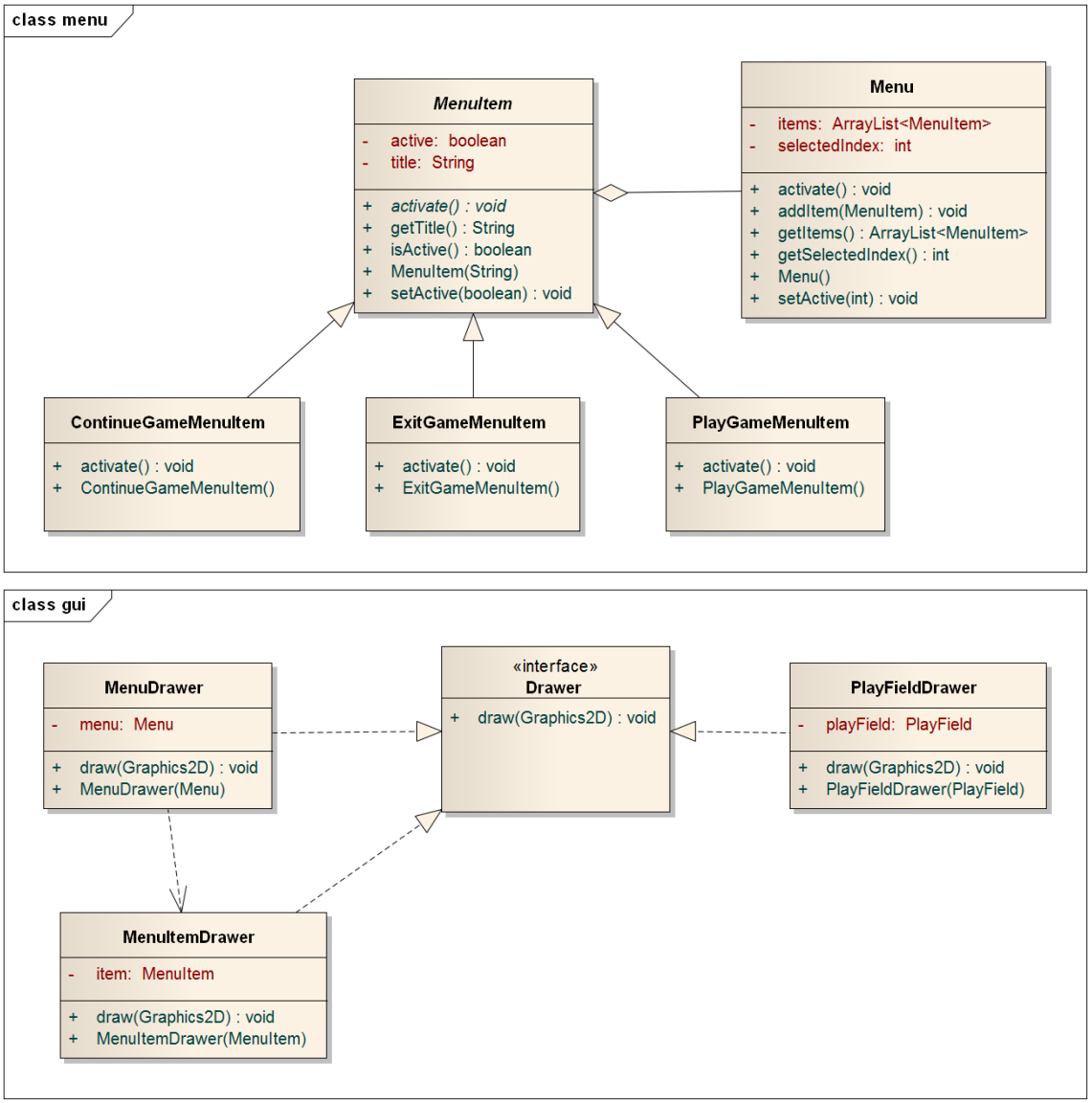



class controll...
BlockController
- block: Block
- playField: PlayField
+ BlockController(Block, PlayField)
+ moveEast() : void
+ moveNorth() : void
+ moveSouth() : void
+ moveWest() : void
PlayFieldController
- playField: PlayField
+ PlayFieldController(PlayField)
+ toggleMode() : void
PlayerController
- player: Player
- playField: PlayField
+ jump() : void
+ moveLeft() : void
+ moveRight() : void
+ PlayerController(Player, PlayField)
+ stopMove() : void
-block
-playField
-playField
-player
block::Block
- count: int
entities: ArrayList<EntityContainer>
+ HEIGHT: double = 150 {readOnly}
# ID: int
# neighbours: Block ([])
# players: ArrayList<PlayerContainer>
# playField: PlayField
+ WIDTH: double = 200 {readOnly}
+ addEntity(EntityPosition, Entity) : void
+ addPlayer(Player, EntityPosition) : void
+ Block(PlayField)
+ checkBorders() : void
+ getBlockDrawer() : BlockDrawer
+ getEntities() : ArrayList<EntityContainer>
+ getNeighbour(int) : Block
+ getNeighbours() : Block[]
+ getPlayers() : ArrayList<PlayerContainer>
+ getPlayField() : PlayField
+ matches(Block, int, boolean) : boolean
+ processCollisions() : void
+ removePlayer(Player) : void
+ setNeighbour(Block, int, boolean) : void
+ setNeighbours(Block[]) : void
+ setNeighbours(Block[], boolean) : void
«property get»
+ getID() : int
-playField
#playField
core::PlayField
- blockMode: boolean
- blocks: ArrayList<BlockContainer>
- game: Game
+ GRAVITY: double = 0.2 {readOnly}
- height: int
- players: ArrayList<PlayerSpawnPoint>
- width: int
+ addBlock(Position, Block) : void
+ addPlayer(Player, Entity) : void
+ getBlock(Position) : Block
+ getBlock(int) : Block
+ getBlockPosition(int) : Position
+ getBlocks() : ArrayList<BlockContainer>
+ getHeight() : int
+ getPlayer(int) : Player
+ getPlayers() : ArrayList<PlayerSpawnPoint>
+ getWidth() : int
+ isBlockMode() : boolean
+ move(Block, int) : void
+ PlayField(Game)
+ resetPlayer(PlayerContainer) : void
+ setHeight(int) : void
+ setSpawnPosition(Player, Entity) : void
+ setWidth(int) : void
+ tick() : void
+ toggleMode() : void
+ win() : void
player::Player
- activeBlocks: ArrayList<Block>
+ HEIGHT: double = 20 {readOnly}
- ID: int
- obtainedKeys: int
- speed: Speed
+ WIDTH: double = 10 {readOnly}
+ addKey() : void
+ enterBlock(Block, EntityPosition) : void
+ getActiveBlocks() : ArrayList<Block>
+ getBoundingBox() : Rectangle
+ getInfo(EntityPosition) : String
+ getName() : String
+ getNextPosition(EntityPosition) : EntityPosition
+ getObtainedKeys() : int
+ getPlayerDrawer() : PlayerDrawer
+ getSpeed() : Speed
+ leaveBlock(Block) : void
+ move(Speed) : void
+ Player(int)
+ reset() : void
+ resetKeys() : void
«property get»
+ getID() : int


class gui
«interface»
Drawer
+ draw(Graphics2D) : void
EntityDrawer
entity: Entity
+ draw(Graphics2D) : void
+ EntityDrawer(Entity)
DoorDrawer
+ DoorDrawer(Door)
+ draw(Graphics2D) : void
KeyDrawer
+ draw(Graphics2D) : void
+ KeyDrawer(Key)
WallDrawer
+ draw(Graphics2D) : void
+ WallDrawer(Wall)
Entity
entity::Door
- requiredKeys: int
+ Door(int, PlayField)
+ getBoundingBox() : Rectangle
+ getEntityDrawer() : EntityDrawer
+ getInfo(EntityPosition) : String
+ getName() : String
+ meetPlayer(PlayerContainer) : void
Entity
entity::Key
- isObtained: boolean
+ getBoundingBox() : Rectangle
+ getEntityDrawer() : EntityDrawer
+ getInfo(EntityPosition) : String
+ getName() : String
+ isObtained() : boolean
+ Key(PlayField)
+ meetPlayer(PlayerContainer) : void
Entity
entity::Wall
- x: double
- y: double
+ accept(BlockMatcher) : void
+ getBoundingBox() : Rectangle
+ getEntityDrawer() : EntityDrawer
+ getHeight() : double
+ getInfo(EntityPosition) : String
+ getName() : String
+ getWidth() : double
+ meetPlayer(PlayerContainer) : void
+ Wall(PlayField, double, double)


class gui
PlayFieldDrawer
- playField: PlayField
+ draw(Graphics2D) : void
+ PlayFieldDrawer(PlayField)
BlockDrawer
block: Block
+ BlockDrawer(Block)
+ draw(Graphics2D) : void
«interface»
Drawer
+ draw(Graphics2D) : void
EntityContainerDrawer
- container: EntityContainer
+ draw(Graphics2D) : void
+ EntityContainerDrawer(EntityContainer)
MenuDrawer
- menu: Menu
+ draw(Graphics2D) : void
+ MenuDrawer(Menu)
FilledBlockDrawer
+ draw(Graphics2D) : void
+ FilledBlockDrawer(FilledBlock)
EntityDrawer
entity: Entity
+ draw(Graphics2D) : void
+ EntityDrawer(Entity)
BlockContainerDrawer
- container: BlockContainer
+ BlockContainerDrawer(BlockContainer)
+ draw(Graphics2D) : void
PlayerDrawer
- player: Player
+ draw(Graphics2D) : void
+ PlayerDrawer(Player)
DoorDrawer
+ DoorDrawer(Door)
+ draw(Graphics2D) : void
KeyDrawer
+ draw(Graphics2D) : void
+ KeyDrawer(Key)
WallDrawer
+ draw(Graphics2D) : void
+ WallDrawer(Wall)
MenuItemDrawer
- item: MenuItem
+ draw(Graphics2D) : void
+ MenuItemDrawer(MenuItem)
PlayerContainerDrawer
- container: PlayerContainer
+ draw(Graphics2D) : void
+ PlayerContainerDrawer(PlayerContainer)


class io
BlockKeyConfiguration
- east: int
- north: int
- south: int
- west: int
+ BlockKeyConfiguration(int, int, int, int)
+ getEast() : int
+ getNorth() : int
+ getSouth() : int
+ getWest() : int
InputHandler
+ handleKeyPressed(int) : void
+ handleKeyReleased(int) : void
KeyAdapter
InputDispatcher
- listeners: ArrayList<InputHandler>
+ attach(InputHandler) : void
+ InputDispatcher()
+ keyPressed(KeyEvent) : void
+ keyReleased(KeyEvent) : void
-config
BlockInputHandler
- config: BlockKeyConfiguration
- controller: BlockController
+ getConfig() : BlockKeyConfiguration
+ getController() : BlockController
+ handleKeyPressed(int) : void
+ handleKeyReleased(int) : void
+ setConfig(BlockKeyConfiguration) : void
+ setController(BlockController) : void
PlayerInputHandler
- config: PlayerKeyConfiguration
- controller: PlayerController
+ getConfig() : PlayerKeyConfiguration
+ getController() : PlayerController
+ handleKeyPressed(int) : void
+ handleKeyReleased(int) : void
+ setConfig(PlayerKeyConfiguration) : void
+ setController(PlayerController) : void
-config
PlayerKeyConfiguration
- jump: int
- left: int
- right: int
+ getJump() : int
+ getLeft() : int
+ getRight() : int
+ PlayerKeyConfiguration(int, int, int)
+ setJump(int) : void
+ setLeft(int) : void
+ setRight(int) : void
PlayFieldInputHandler
- config: PlayFieldKeyConfiguration
- controller: PlayFieldController
+ handleKeyPressed(int) : void
+ handleKeyReleased(int) : void
+ setConfig(PlayFieldKeyConfiguration) : void
+ setController(PlayFieldController) : void
-config
PlayFieldKeyConfiguration
- toggle: int
+ getToggleMode() : int
+ PlayFieldKeyConfiguration(int)
+ setToggleMode(int) : void



11.3. A grafikus objektumok felsorolása

A modell változásai rendkívül apróak voltak, így külön külön felsorolni nincs értelme, az alábbi változtatások történtek:

Az `Entity` osztály kapott egy `+getEntityDrawer():EntityDrawer` absztrakt metódust, az egyes entitások ezeket definiálják fölül az alábbi módon: A `Door` osztály egy `DoorDrawer` objektumot ad vissza, a `Key` egy `KeyDrawer`-t, a `Wall` egy `WallDrawer`-t, a `SpawnPoint` egy `null`-t, mivel azt nem kell kirajzolni.

A `Player` is kapott egy `+getPlayerDrawer():PlayerDrawer` metódust, ez egy játékos rajzolót ad vissza.

11.3.1. Grafikus

Felelőssége
A program grafikus változatának fő osztálya.

Ősosztályok
Nincs

Interfészek
Nincs

Attribútumok

-clock: Timer A játék órája.

+level: int Az aktuális pálya száma.

-instance: Grafikus A Grafikus példányának tárolására (sigleton pattern).

Metódusok

+main(String[] args): void Létrehoz egy Grafikus objektumot, majd betölti rajta a főmenüt. `args` paraméter: Parancssori paraméterek (nem használjuk).

+getInstance(): Grafikus Visszaadja a Grafikus egyetlen példányát (sigleton pattern).

+createInputDispatcher(): InputDispatcher Létrehozza a játékhoz szükséges InputHandlereket (részletesen a szekvenciadiargamokon).

+play(int n): void A megadott pályától indít egy játékot. `n` paraméter: A pálya száma.

+playNext(): void Elindítja a következő pályát, és növeli az aktuális pálya számát ennek megfelelően.

+loadMainMenu(): void Betölti a főmenüt.

+getLevel(): int Visszaadja az aktuális pálya számát.

+loadNextLevel(): void Megjelenít egy statikus képernyőt ("Loading next level..." felirattal). Majd betölti a következő pályát.

+endGame(): void Megjeleníti a játék vége képernyőt, majd visszatér a főmenübe.

11.3.2. BlockContainerDrawer

Felelőssége
Egy blokk konténerének kirajzolásáért felelős.

Ősosztályok
Nincs

Interfészek
Drawer

Attribútumok

-container: BlockContainer A kirajzolandó blokk konténer objektum.

Metódusok

+draw(Graphics2D g): void A kapott grafikus felületet eltranszformálja a megadott pozícióba, a játékmód állapotától függően nagyít rajta, majd kirajzolja a konténerben találhatóü blokkot (meghívja a blokk kirajzolójának draw metódusát). g paraméter: A grafikus felület, amire rajzolunk.

11.3.3. DoorDrawer

Felelőssége
Egy ajtót kirajzoló osztály.

Ősosztályok
EntityDrawer

Interfészek
Nincs

Attribútumok

Nincs

Metódusok

+draw(Graphics2D g): void Kirajzol egy ajtót. Natív AWT metódusokat használ. g paraméter: A grafikus felület, amire rajzolunk.

11.3.4. KeyDrawer

Felelőssége
Kirajzol egy kulcsot.

Ősosztályok
EntityDrawer

Interfészek
Nincs

Attribútumok

Nincs

Metódusok

+draw(Graphics2D g): void A kulcs állapotától függően kirajzol egy kulcsot (felvett/nem felvett). g paraméter: A kirajzolandó grafikus felület.

11.3.5. FilledBlockDrawer

Felelőssége
Egy FilledBlock objektumot kirajzol.

Ősosztályok
BlockDrawer

Interfészek
Nincs

Attribútumok

Nincs

Metódusok

+draw(Graphics2D g): void Végigiterál az entitásokon, és a játékosokon, és mindnek meghívja a draw metódusát. g paraméter: A grafikus felület, amire rajzolunk.

11.3.6. PlayerDrawer

Felelőssége
Egy játékost rajzol ki.

Ősosztályok
Nincs

Interfészek
Drawer

Attribútumok

-player: Player A kirajzolandó játékos.

Metódusok

+draw(Graphics2D g): void Kirajzolja a játékost. Natív AWT metódusokat használ. g paraméter: A grafikus felület, amire rajzolunk.

11.3.7. ExitGameMenuItem

Felelőssége
A MenuItem leszármazottja, a viselkedése az activate() metódusban található.

Ősosztályok
MenuItem

Interfészek
Nincs

Attribútumok

Nincs

Metódusok

+activate(): void Kilép a programból.

11.3.8. PlayGameMenuItem

Felelőssége
A MenuItem leszármazottja, viselkedése az activate() metódusban szerepel.

Ősosztályok
MenuItem

Interfészek
Nincs

Attribútumok

Nincs

Metódusok

+activate(): void Az első pályától indít egy játékot.

11.3.9. BlockDrawer (absztrakt)

Felelőssége
Absztrakt osztály, az egyes blokktípusok kiterjeszthetik.

Ősosztályok
Nincs

Interfészek
Drawer

Attribútumok

#block: Block A kirajzolandó blokk.

Metódusok

+draw(Graphics2D g): void A kirajzolást a leszármazottaknak kell definiálniuk. g paraméter: A grafikus felület, amire rajzolni kell.

11.3.10. MenuDrawer

Felelőssége
Kirajzol egy menüt.

Ősosztályok
Nincs

Interfészek
Drawer

Attribútumok

-menu: Menu A kirajzolandó menü.

Metódusok

+draw(Graphics2D g): void Végigiterál a menüpontokon, és azok rajzolóján meghívja a draw metódust. Közben a grafikus felületet úgy transzformálja, hogy a menüpontok egymás alá kerüljenek. g paraméter: A grafikus felület, amire rajzolunk.

11.3.11. PlayerContainerDrawer

Felelőssége
Egy játékos konténerét rajzolja ki.

Ősosztályok
Nincs

Interfészek
Drawer

Attribútumok

-container: PlayerContainer A kirajzolandó konténer.

Metódusok

+draw(Graphics2D g): void Eltranszformálja a grafikus felületet a játékos pozíciójára, majd meghívja a játékos kirajzolójának draw() metódusát. g paraméter: A grafikus felület, amire rajzolunk.

11.3.12. BlockInputHandler

Felelőssége
Egy blokk kontrollerhez irányítja az érkezett bemenetet, az aktuális konfiguráció függvényében.

Ősosztályok
InputHandler

Interfészek
Nincs

Attribútumok

-controller: BlockController A kontroller, ami irányítja a blokkot.

-config: BlockKeyConfiguration A tárolt billentyűkiosztás.

Metódusok

+getController(): BlockController Visszaadja a tárolt kontrollert. Visszatérési érték: A tárolt kontroller.

+getConfig(): BlockKeyConfiguration Visszaadja a tárolt konfigurációt. Visszatérési érték: A tárolt konfiguráció.

+setController(BlockController c): void Beállítja a kontrollert. `c` paraméter: A kontroller, amit beállítunk.

+setConfig(BlockKeyConfiguration c): void Beállítja a konfigurációt. `c` paraméter: A beállítandó konfiguráció.

+handleKeyPressed(int k): void Ha a leütött billentyű és a konfiguráció függvényében meghívja a kontroller egyes metódusait. `k` paraméter: A leütött billentyű kódja (KeyEvent konstansok).

+handleKeyReleased(int k): void Felengedett billentyű esetén nem történik semmi. `k` paraméter: A leütött billentyű kódja (KeyEvent konstansok).

11.3.13. GameFrame

Ősosztályok
Frame

Interfészek
Nincs

Attribútumok

-drawer: Drawer

Metódusok

+setDrawer(Drawer d): void A natív frame ezt a rajzolót fogja használni a rendereléshez.

+paint(Graphics g): void A rajzolás a natív Graphics objektumra történik, a beállított rajzoló objektummal. g paraméter: A natív Graphics objektum.

+update(Graphics g): void A kép villogásának elkerülése érdekében dupla bufferelést használunk, ezt a natív update metódus felüldefiniálásával tesszük meg. g paraméter: A natív Graphics objektum.

+clearKeyListeners(): void

11.3.14. PlayFieldController

Felelőssége
Egy játékteret irányít.

Ősosztályok
Nincs

Interfészek
Nincs

Attribútumok

-playField: PlayField Az irányítandó játéktér.

Metódusok

+toggleMode(): void Megváltoztatja a játékmódot a játéktéren.

11.3.15. InputDispatcher

Felelőssége
A beérkezett billentyűeseményeket üzenetszórással eljuttatja a feliratkozott InputHandlereknek.

Ősosztályok
KeyAdapter

Interfészek
Nincs

Attribútumok

-listeners: ArrayList<InputHandler> A feliratkozott InputHandlerek.

Metódusok

+attach(InputHandler handler): void Feliratkoztatja a megadott InputHandlert az eseményekre.

+keyPressed(KeyEvent e): void A billentyűzetlenyomásoknál szétszórja a kapott billentyűkódot a feliratkozottaknak. `e` paraméter: A kapott esemény.

+keyReleased(KeyEvent e): void A billentyűzet felengedésekkor szétszórja a kapott billentyűkódot a feliratkozottak között. `e` paraméter: A kapott esemény.

11.3.16. PlayerKeyConfiguration

Felelőssége
Egy játékos irányításához tartozó konfigurációt tárolja.

Ősosztályok
Nincs

Interfészek
Nincs

Attribútumok

-left: int A bal iránybillentyű kódja.

-right: int A jobb iránybillentyű kódja.

-jump: int Az ugrás kódja.

Metódusok

+getLeft(): int Visszaadja a balra mozgás kódját. Visszatérési érték: A balra mozgás kódja.

+getRight(): int Visszaadja a jobbra mozgás kódját. Visszatérési érték: A jobbra mozgás kódja.

+getJump(): int Visszaadja az ugrás kódját. Visszatérési érték: Az ugrás kódja.

+setLeft(int key): void Beállítja a balra mozgás kódját. `key` paraméter: A balra mozgás kódja.

+setRight(int key): void Beállítja a jobbra mozgás kódját. `key` paraméter: A jobbra mozgás kódja.

+setJump(int key): void Beállítja az ugrás kódját. `key` paraméter: Az ugrás kódja.

11.3.17. PlayFieldKeyConfiguration

Felelőssége
A játéktér irányításához tartozó billentyűzetkiosztást tárolja.

Ősosztályok
Nincs

Interfészek
Nincs

Attribútumok

-toggle: int A játékmódváltás kódja.

Metódusok

+getToggleMode(): int Visszaadja a játékmódváltás kódját. Visszatérési érték: A módváltás kódja.

+setToggleMode(int key): void Beállítja a módváltás kódját. `key` paraméter: A módváltás kódja.

11.3.18. ContinueGameMenuItem

Felelőssége
A MenuItem leszármazottja, viselkedése az activate() metódusban szerepel.

Ősosztályok
MenuItem

Interfészek
Nincs

Attribútumok

Nincs

Metódusok

+activate(): void A legutolsó (amire a játékos eljutott) pályától indít egy játékot.

11.3.19. PlayerInputHandler

Felelőssége
Egy játékost irányít (meghívja a kontroller megfelelő metódusait) az érkezett bemenet és a konfiguráció függvényében.

Ősosztályok
InputHandler

Interfészek
Nincs

Attribútumok

-controller: PlayerController A tárolt kontroller.

-config: PlayerKeyConfiguration A tárolt konfiguráció.

Metódusok

+getController(): PlayerController Visszaadja a tárolt kontrollert. Visszatérési érték: a tárolt kontroller.

+getConfig(): PlayerKeyConfiguration Visszaadja a tárolt konfigurációt. Visszatérési érték: A tárolt konfiguráció.

+setController(PlayerController c): void Beállítja a kontrollert. `c` paraméter: A beállítandó kontroller.

+setConfig(PlayerKeyConfiguration c): void Beállítja a konfigurációt. `c` paraméter: A beállítandó konfiguráció.

+handleKeyPressed(int k): void A megfelelő billentyű lenyomásakor a kontroller metódusait hívja. `k` paraméter: A lenyomott billentyű kódja (KeyEvent konstans).

+handleKeyReleased(int k): void A megfelelő billentyű felengedésekor meghívja a kontroller megfelelő metódusát (megállítja a játékost). `k` paraméter: A felengedett billentyű kódja (KeyEvent konstans).

11.3.20. EntityContainerDrawer

Felelőssége
Egy entitás konténerét rajzolja ki.

Ősosztályok
Nincs

Interfészek
Drawer

Attribútumok

-container: EntityContainer A kirajzolandó konténer.

Metódusok

+draw(Graphics2D g): void Eltranszformálja a grafikus felületet az entitás pozíciójába, majd meghívja az entitás rajzolójának draw metódusát. `g` paraméter: A grafikus felület, amire rajzolunk.

11.3.21. EntityDrawer (absztrakt)

Felelőssége
Az egyes entitások rajzoló osztályai ebből származnak.

Ősosztályok
Nincs

Interfészek
Drawer

Attribútumok

#entity: Entity A tárolt entitás.

Metódusok

+draw(Graphics2D g): void A leszármazottak saját maguk definiálják a kirajzolást. g paraméter: A grafikus felület, amire rajzolni kell.

11.3.22. BlockKeyConfiguration

Felelőssége
Egy blokk mozgatásához tartozó billentyűzetkiosztást tárolja.

Ősosztályok
Nincs

Interfészek
Nincs

Attribútumok

-north: int A felfelé mozgatás billentyűkódja.

-east: int A jobbra mozgatás billentyűkódja.

-south: int A lefelé mozgatás billentyűkódja.

-west: int A balra mozgatás billentyűkódja.

Metódusok

+getNorth(): int Visszaadja a felfelé mozgatás billentyűkódját. Visszatérési érték: A kért kód.

+getEast(): int Visszaadja a jobbra mozgatás billentyűkódját. Visszatérési érték: A kért kód.

+getSouth(): int Visszaadja a lefelé mozgatás billentyűkódját. Visszatérési érték: A kért kód.

+getWest(): int Visszaadja a balra mozgatás billentyűkódját. Visszatérési érték: A kért kód.

11.3.23. MenuItem (absztrakt)

Felelőssége
Egy menüpontot reprezentáló osztály.

Ősosztályok
Nincs

Interfészek
Nincs

Attribútumok

-title: String A menüpont felirata.

-active: boolean A menüpont állapota (aktív, inaktív).

Metódusok

+activate(): void A menüpont aktiválásakor történős eseményeket a leszármazottaknak kell definiálnia.

+getTitle(): String Visszaadja a menüpont feliratát. Visszatérési érték: A menüpont felirata.

+setActive(boolean state): void Beállítja a menüpont aktív/inaktív voltát. `state` paraméter: Az új állapot.

+isActive(): boolean Visszaadja a menüpont állapotát. Visszatérési érték: A menüpont állapota (igaz, ha aktív).

11.3.24. InputHandler (absztrakt)

Felelőssége
Az egyes inputkezelő osztályok innen származnak.

Ősosztályok
Nincs

Interfészek
Nincs

Attribútumok

Nincs

Metódusok

+handleKeyPressed(int key): void A leszármazottak saját maguk definiálják mit tesznek egy billentyű lenyomásakor. `key` paraméter: A lenyomott billentyű.

+handleKeyReleased(int key): void A leszármazottak saját maguk definiálják mit tesznek egy billentyű felengedésekor. `key` paraméter: A felengedett billentyű.

11.3.25. BlockController

Felelőssége
Egy blokkot irányít.

Ősosztályok
Nincs

Interfészek
Nincs

Attribútumok

-block: Block Az irányítandó blokk.

-playField: PlayField A blokk ezen a játéktéren van.

Metódusok

+moveNorth(): void Felfele mozgatja a blokkot.

+moveEast(): void Jobbra mozgatja a blokkot.

+moveSouth(): void Lefele mozgatja a blokkot.

+moveWest(): void Balra mozgatja a blokkot.

11.3.26. PlayFieldDrawer

Felelőssége
Kirajzolja a játékteret.

Ősosztályok
Nincs

Interfészek
Drawer

Attribútumok

-playField: PlayField A kirajzolandó játéktér.

Metódusok

+draw(Graphics2D g): void Végigiterál a blokkok konténerein, és meghívja a rajzolójuk draw() metódusát. g paraméter: A grafikus felület, amire rajzolunk.

11.3.27. WallDrawer

Felelőssége
Kirajzol egy falat.

Ősosztályok
EntityDrawer

Interfészek
Nincs

Attribútumok

Nincs

Metódusok

+draw(Graphics2D g): void Kirajzolja a falat annak paraméterei függvényében (szélesség, magasság). Natív AWT metódusokat használ. `g` paraméter: A grafikus felület, amire rajzolunk.

11.3.28. Menu

Felelőssége
Egy menüt reprezentáló osztály. Menüpontok tárolása és aktiválása/deaktiválása a feladata.

Ősosztályok
Nincs

Interfészek
Nincs

Attribútumok

-items: ArrayList<MenuItem> A menüben található menüpontok listája.

-selectedIndex: int A kiválasztott menüpont indexe.

Metódusok

+addItem(MenuItem item): void Hozzáad egy menüpontot a menühöz. `item` paraméter: Az új menüpont.

+getItems(): ArrayList<MenuItem> Visszaadja a menüben tárolt menüpontokat. Visszatérési érték: A tárolt menüpontok.

+activate(): void Aktiválja a kiválasztott menüpontot. Ha nincs kiválasztva semmi, nem történik semmi.

+setActive(int n): void Aktívra állítja a megadott indexű menüpontot, és inaktívvá teszi a többit. `n` paraméter: Az aktiválandó menüpont indexe.

+getSelectedIndex(): int Visszaadja a kiválasztott menüpont indexét. Visszatérési érték: A kiválasztott menüpont indexe.

11.3.29. PlayFieldInputHandler

Felelőssége
A játéktér kontrollerének metódusait hívja az érkező bemenet és a konfiguráció függvényében.

Ősosztályok
InputHandler

Interfészek
Nincs

Attribútumok

-controller: PlayFieldController A tárolt kontroller.

-config: PlayFieldKeyConfiguration A tárolt konfiguráció.

Metódusok

+setController(PlayFieldController con): void Beállítja a kontrollert. `con` paraméter: A beállítandó kontroller.

+setConfig(PlayFieldKeyConfiguration con): void Beállítja a konfigurációt. `con` paraméter: A beállítandó konfiguráció.

+handleKeyPressed(int k): void A megfelelő billentyű leütésekor meghívja a kontroller megfelelő metódusát (toggleMode()). `k` paraméter: A lenyomott billentyű kódja.

+handleKeyReleased(int k): void Billentyű felengedésekor semmi sem történik. `k` paraméter: A felengedett billentyű kódja.

11.3.30. PlayerController

Felelőssége
Egy játékost irányít.

Ősosztályok
Nincs

Interfészek
Nincs

Attribútumok

-player: Player Az irányítandó játékos.

-playField: PlayField A játékos ezen a játéktéren van.

Metódusok

+moveLeft(): void A játékost balra mozgatja, ha játékmódban vagyunk.

+moveRight(): void A játékost jobbra mozgatja, ha játékmódban vagyunk.

+jump(): void A játékos ugrik, ha játékmódban vagyunk, és talajon áll (nem esik).

+stopMove(): void Megállítja a játékost (vízszintes mozgásban csak), ha játékmódban vagyunk.

11.3.31. MenuItemDrawer

Felelőssége
Egy menüpontot rajzol ki.

Ősosztályok
Nincs

Interfészek
Drawer

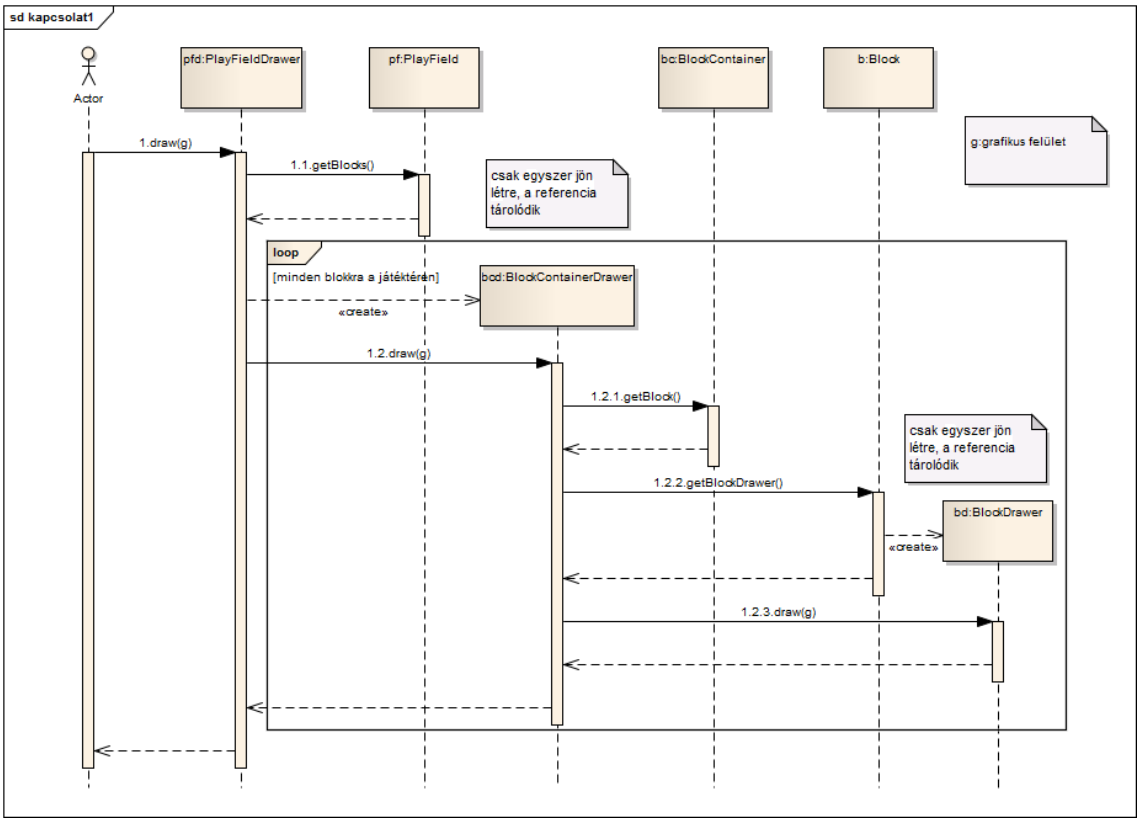
Attribútumok

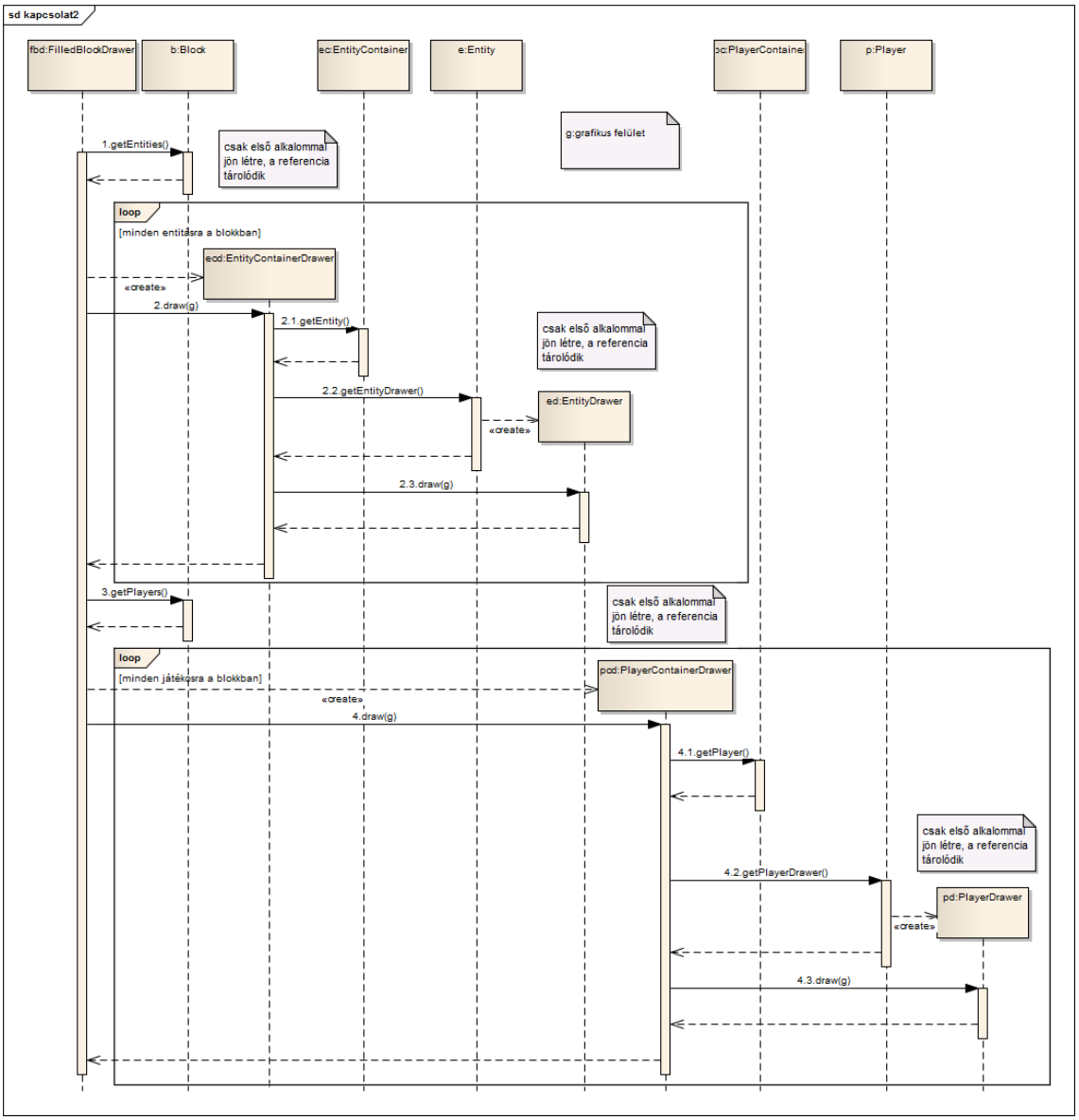
-item: MenuItem A kirajzolandó menüpont.

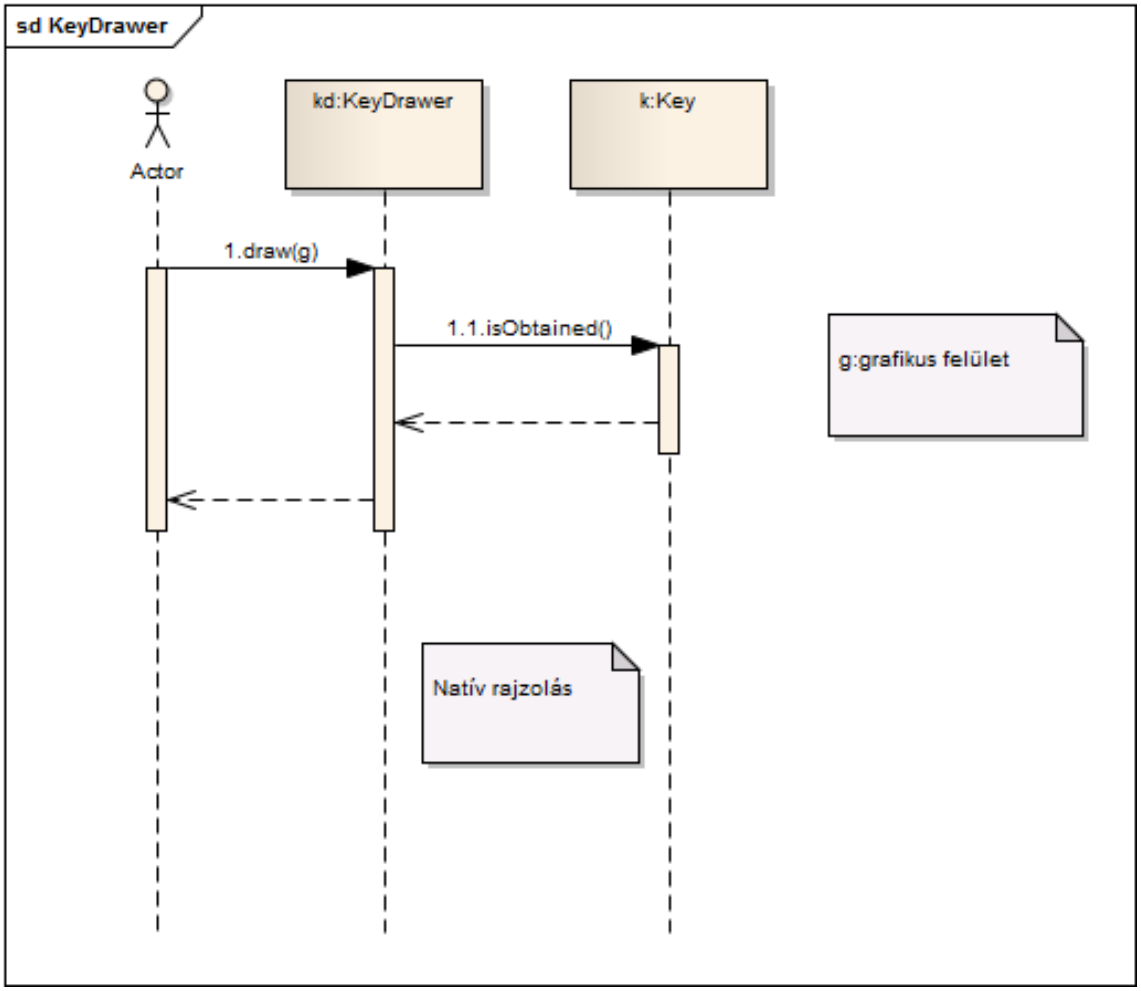
Metódusok

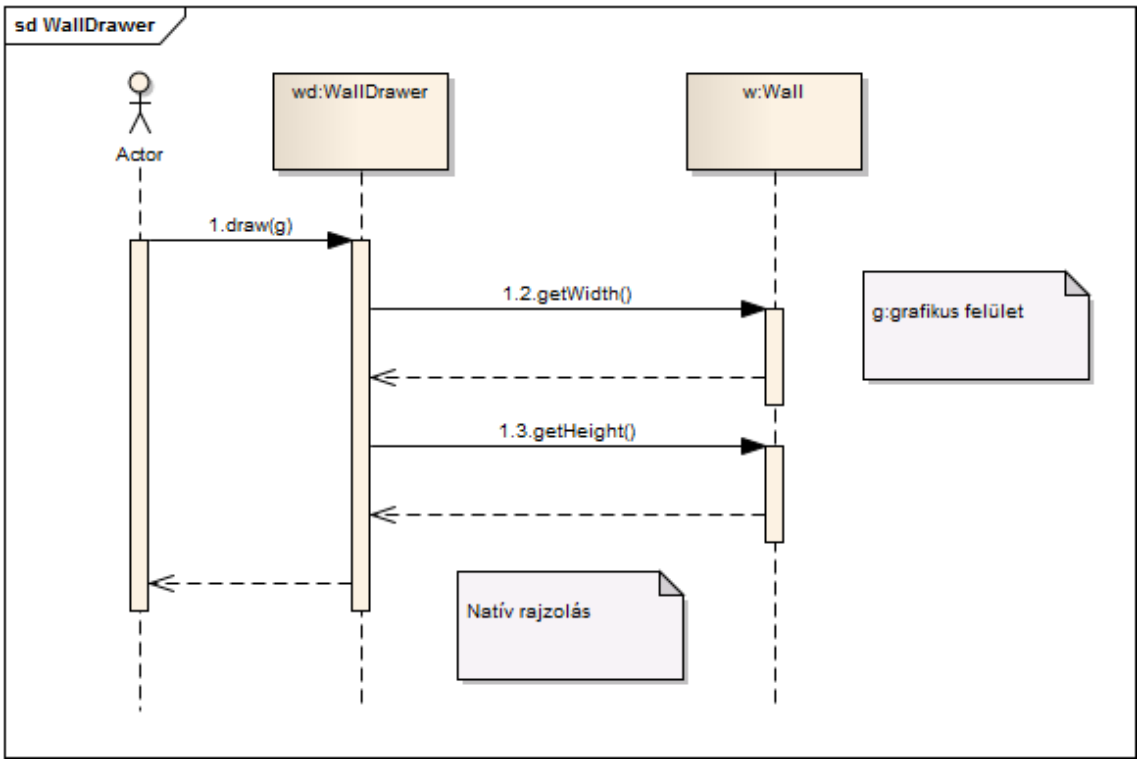
+draw(Graphics2D g): void A menüpont állapotától függően (aktív/inaktív) kirajzolja a menüpontot. Natív AWT metódusokat használ. g paraméter: A grafikus felület, amire rajzolunk.

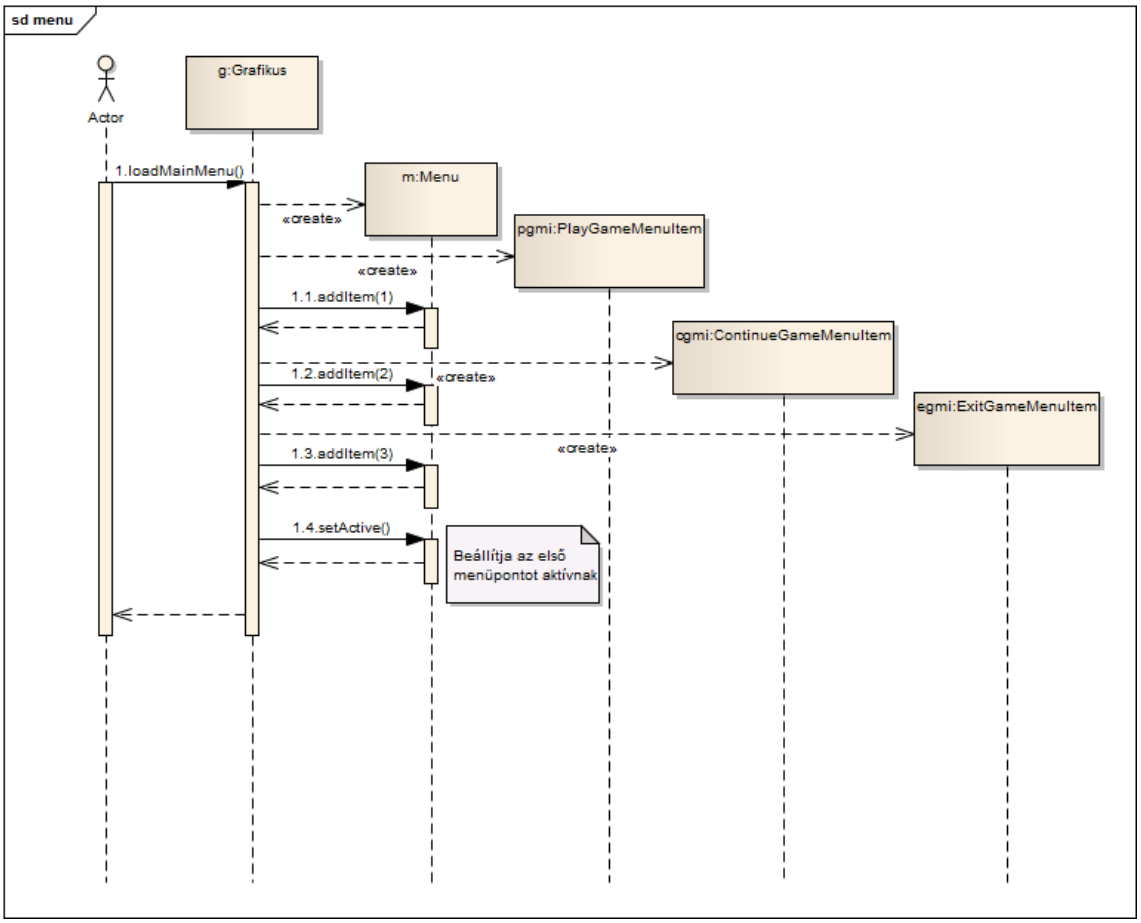
11.4. Kapcsolat az alkalmazói rendszerrel

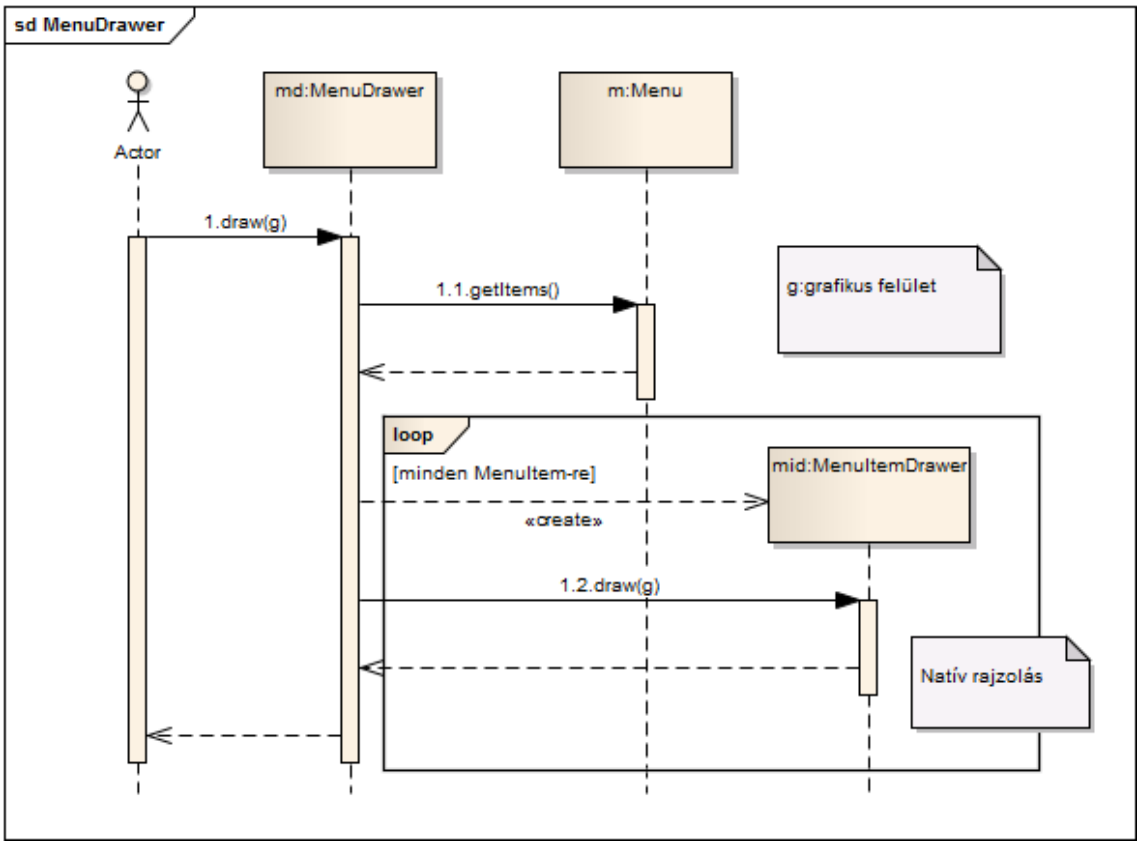


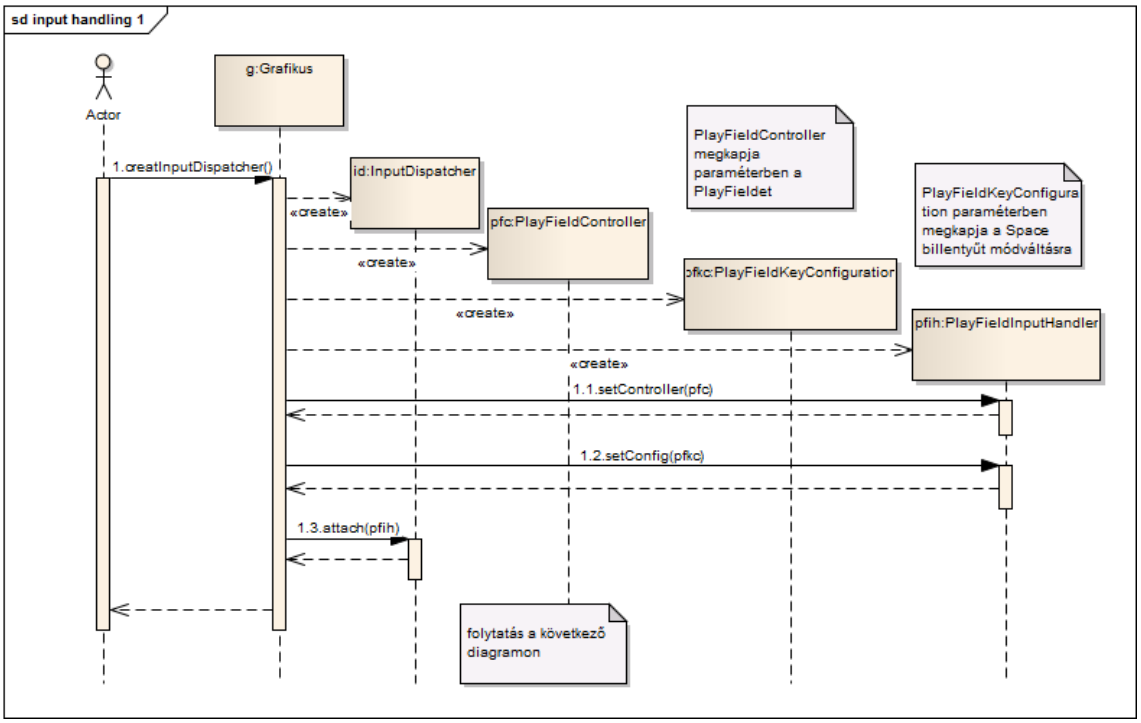

sd kapcsolat2
fbd:FilledBlockDrawer
b:Block
ec:EntityContainer
e:Entity
pc:PlayerContaine
p:Player
g:grafikus felület
1.getEntities()
csak első alkalommal jön létre, a referencia tárolódik
loop
[minden entitásra a blokkban]
ecd:EntityContainerDrawer
«create»
2.draw(g)
2.1.getEntity()
csak első alkalommal jön létre, a referencia tárolódik
2.2.getEntityDrawer()
ed:EntityDrawer
«create»
2.3.draw(g)
3.getPlayers()
csak első alkalommal jön létre, a referencia tárolódik
loop
[minden játékosra a blokkban]
pcd:PlayerContainerDrawer
«create»
4.draw(g)
4.1.getPlayer()
csak első alkalommal jön létre, a referencia tárolódik
4.2.getPlayerDrawer()
pd:PlayerDrawer
«create»
4.3.draw(g)

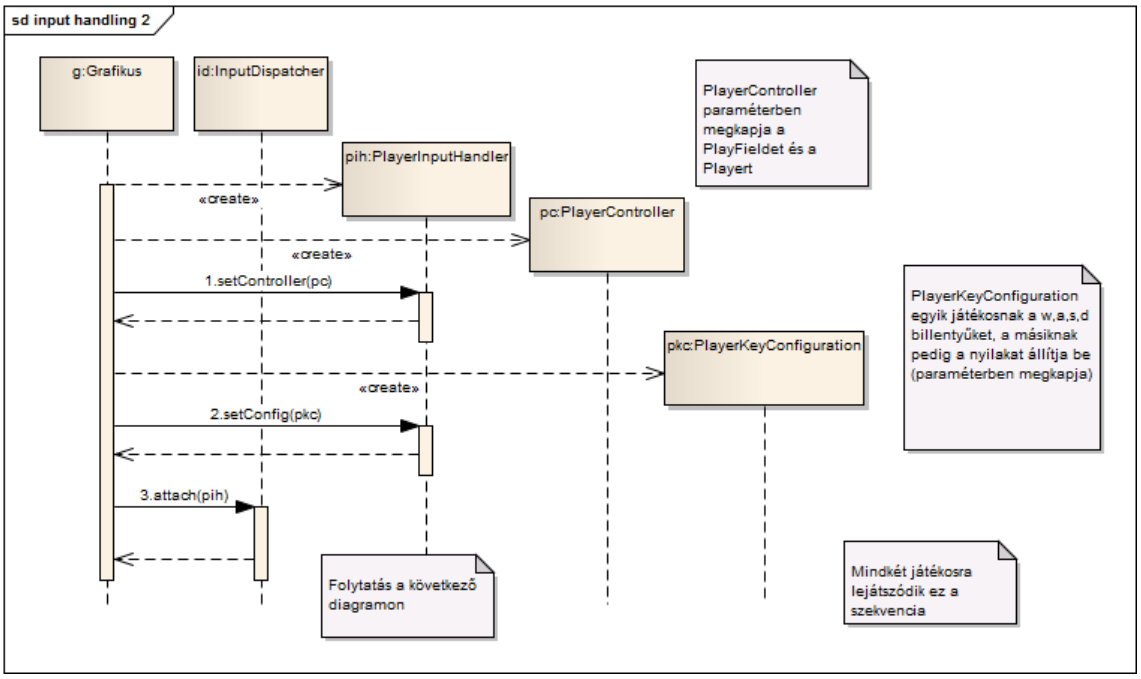

sd KeyDrawer
Actor
kd:KeyDrawer
k:Key
1.draw(g)
1.1.isObtained()
g:grafikus felület
Natív rajzolás

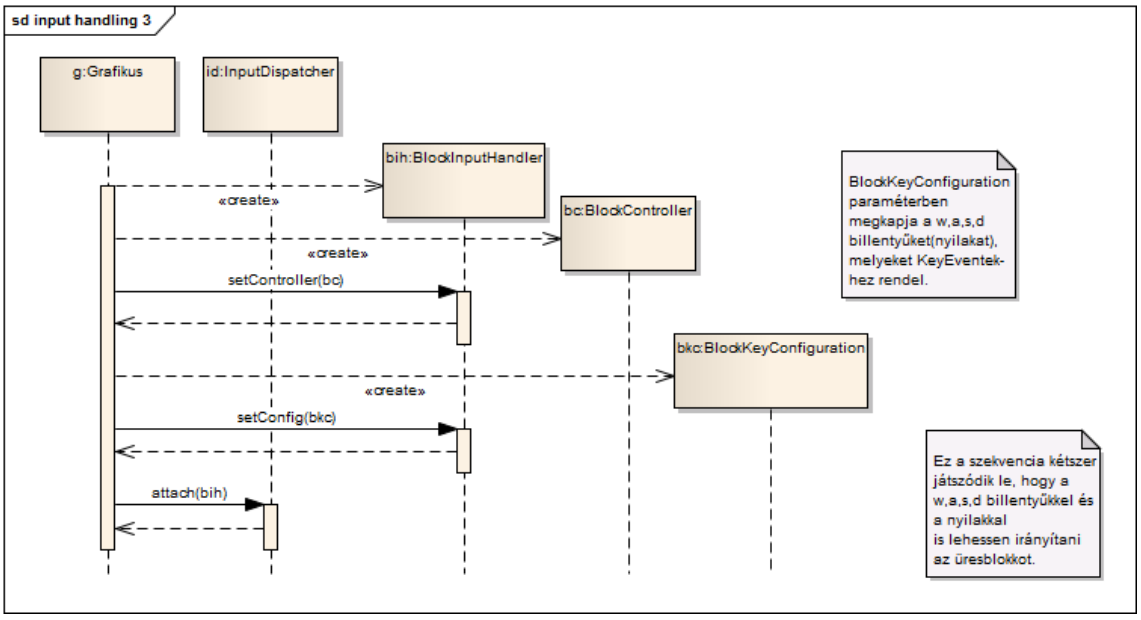

sd WallDrawer
Actor
wd:WallDrawer
w:Wall
1.draw(g)
1.2.getWidth()
g:grafikus felület
1.3.getHeight()
Natív rajzolás

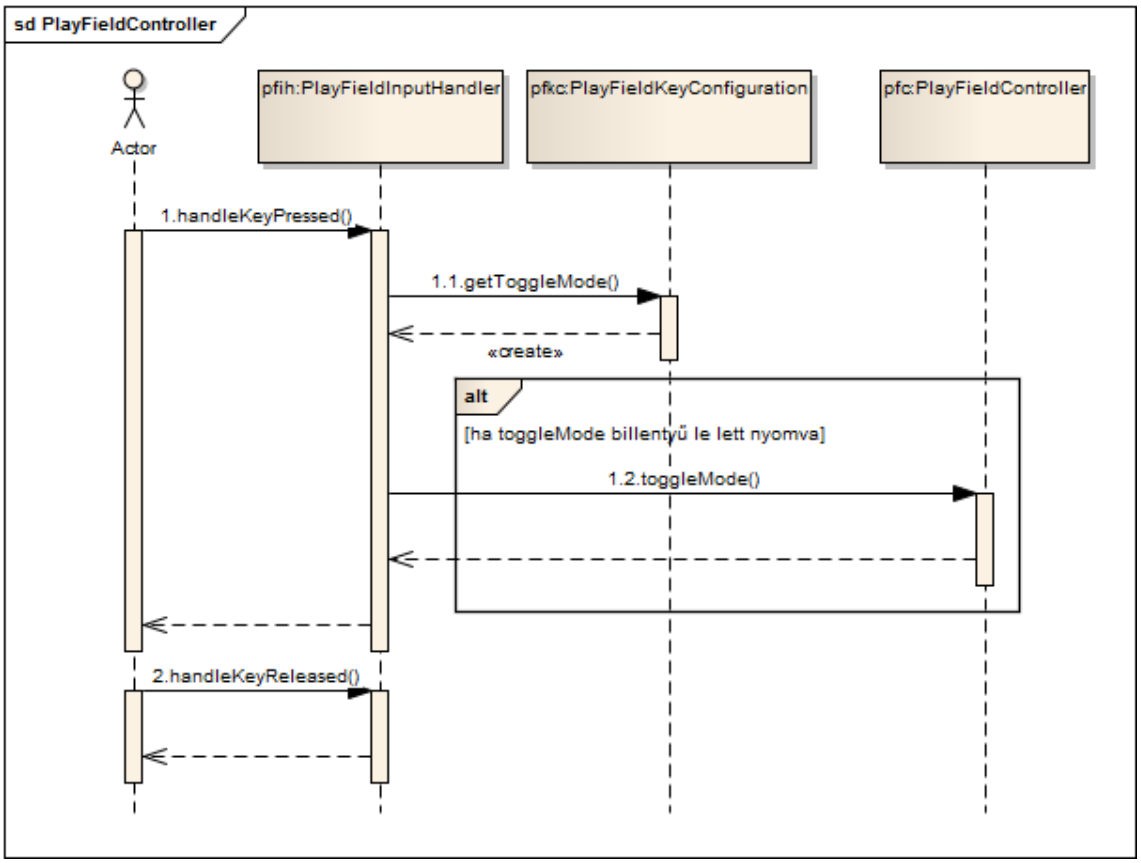

sd menu
Actor
g:Grafikus
1.loadMainMenu()
m:Menu
«create»
pgmi:PlayGameMenuItem
«create»
1.1.addItem(1)
cgmi:ContinueGameMenuItem
1.2.addItem(2)
«create»
egmi:ExitGameMenuItem
«create»
1.3.addItem(3)
1.4.setActive()
Beállítja az első menüpontot aktívnak

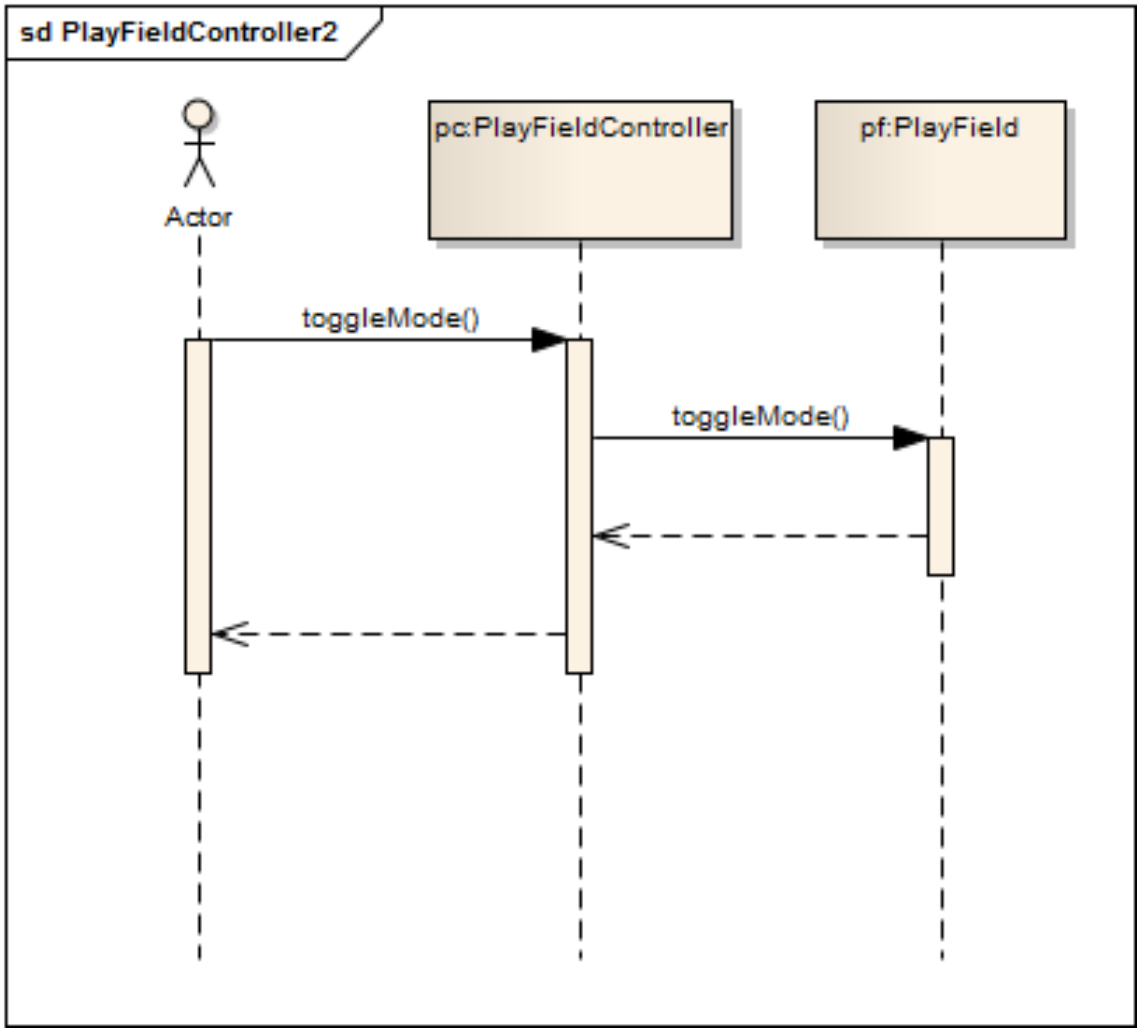

sd MenuDrawer
Actor
md:MenuDrawer
m:Menu
1.draw(g)
1.1.getItems()
g:grafikus felület
loop
[minden MenuItem-re]
mid:MenuItemDrawer
«create»
1.2.draw(g)
Natív rajzolás

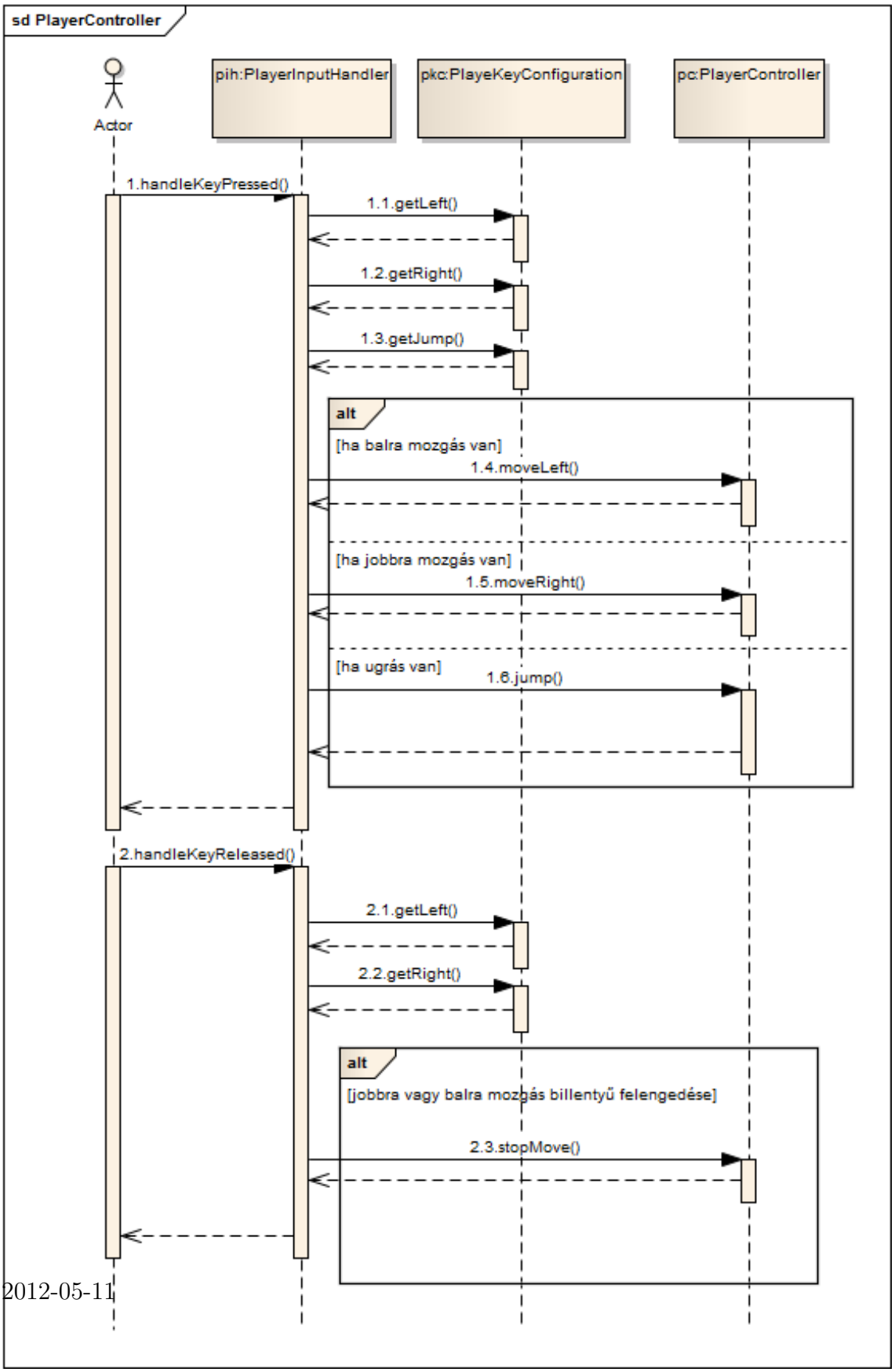

sd input handling 1
Actor
g:Grafikus
1.creatInputDispatcher()
id:InputDispatcher
«create»
pfc:PlayFieldController
«create»
PlayFieldController megkapja paraméterben a PlayFieldet
pfkc:PlayFieldKeyConfiguration
«create»
PlayFieldKeyConfiguration paraméterben megkapja a Space billentyűt módváltásra
pfih:PlayFieldInputHandler
«create»
1.1.setController(pfc)
1.2.setConfig(pfkc)
1.3.attach(pfih)
folytatás a következő diagramon

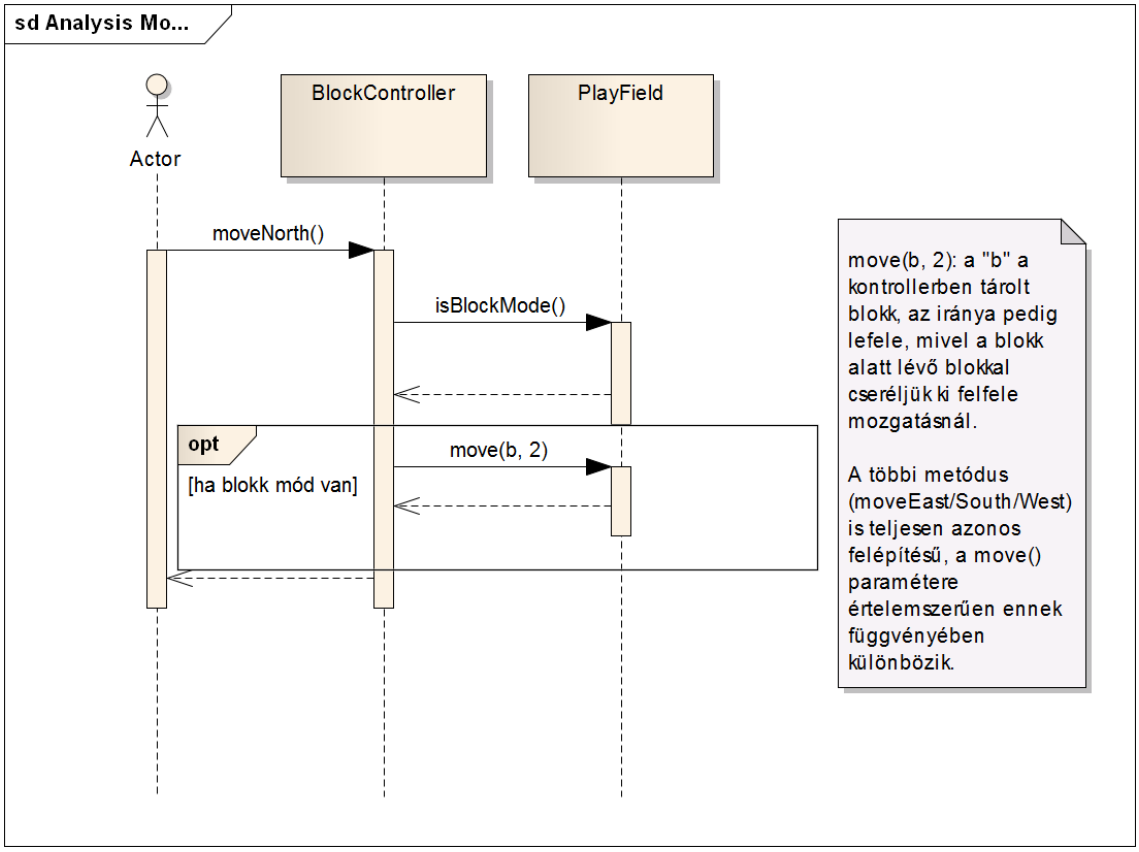

sd input handling 2
g:Grafikus
id:InputDispatcher
pih:PlayerInputHandler
pc:PlayerController
pkc:PlayerKeyConfiguration
«create»
«create»
«create»
1.setController(pc)
2.setConfig(pkc)
3.attach(pih)
PlayerController paraméterben megkapja a PlayFieldet és a Playert
PlayerKeyConfiguration egyik játékosnak a w,a,s,d billentyűket, a másiknak pedig a nyilakat állítja be (paraméterben megkapja)
Folytatás a következő diagramon
Mindkét játékosra lejátszódik ez a szekvencia

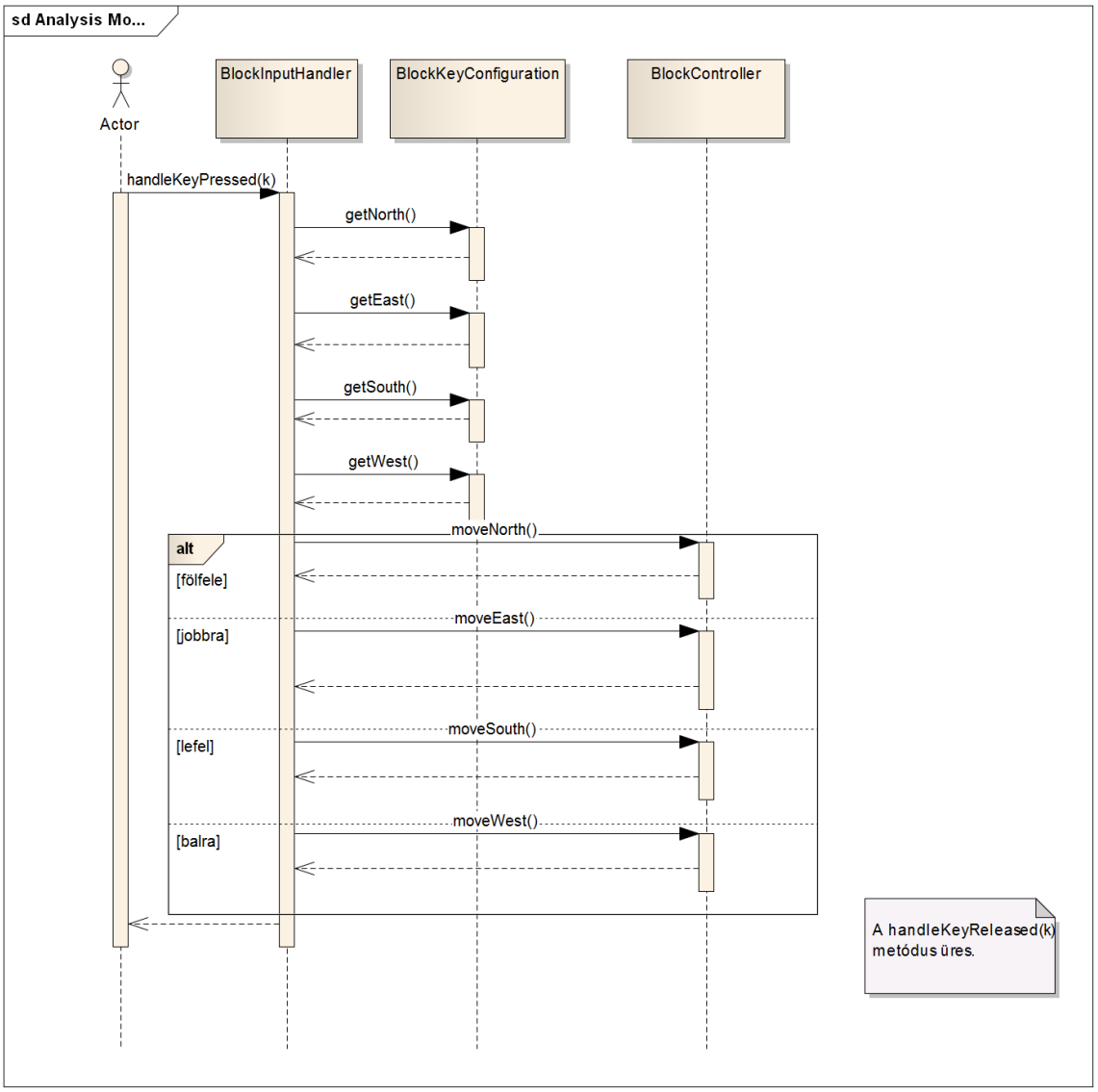

sd input handling 3
g:Grafikus
id:InputDispatcher
bih:BlockInputHandler
bc:BlockController
bkc:BlockKeyConfiguration
«create»
«create»
«create»
setController(bc)
setConfig(bkc)
attach(bih)
BlockKeyConfiguration paraméterben megkapja a w,a,s,d billentyűket(nyilakat), melyeket KeyEventekhez rendel.
Ez a szekvencia kétszer játszódik le, hogy a w,a,s,d billentyűkkel és a nyilakkal is lehessen irányítani az üresblokkot.

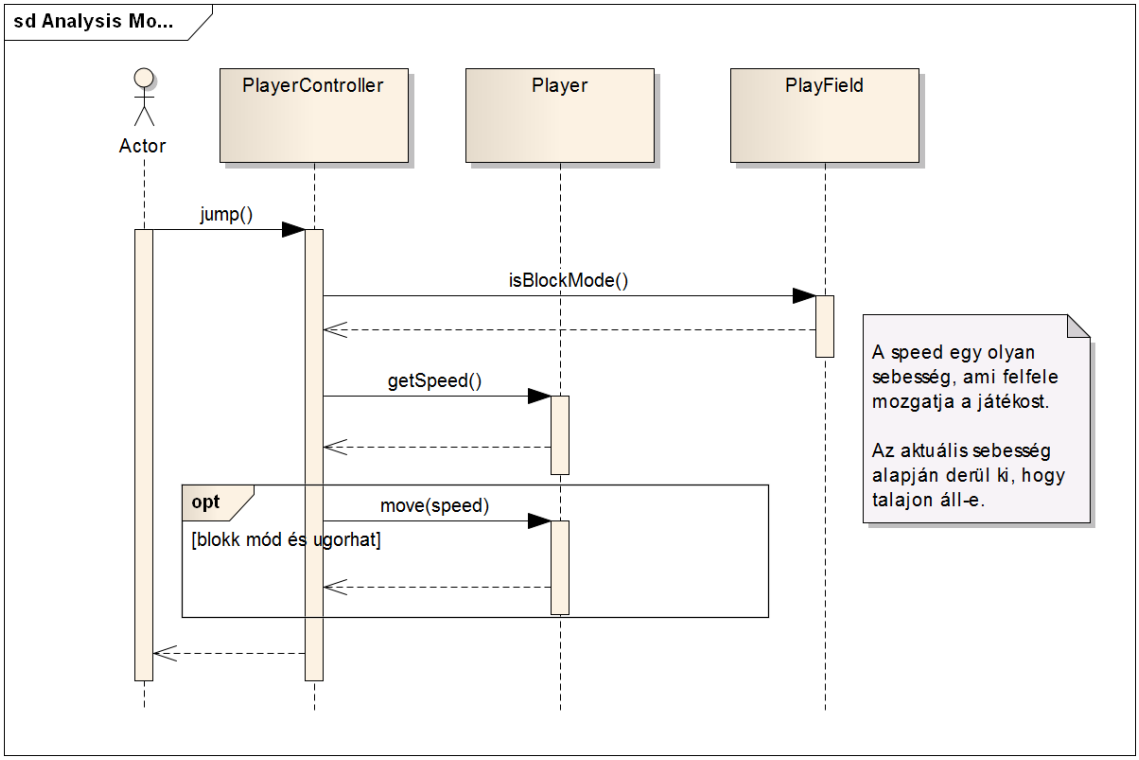

sd PlayFieldController
Actor
pfih:PlayFieldInputHandler
pfkc:PlayFieldKeyConfiguration
pfc:PlayFieldController
1.handleKeyPressed()
1.1.getToggleMode()
«create»
alt
[ha toggleMode billentyű le lett nyomva]
1.2.toggleMode()
2.handleKeyReleased()

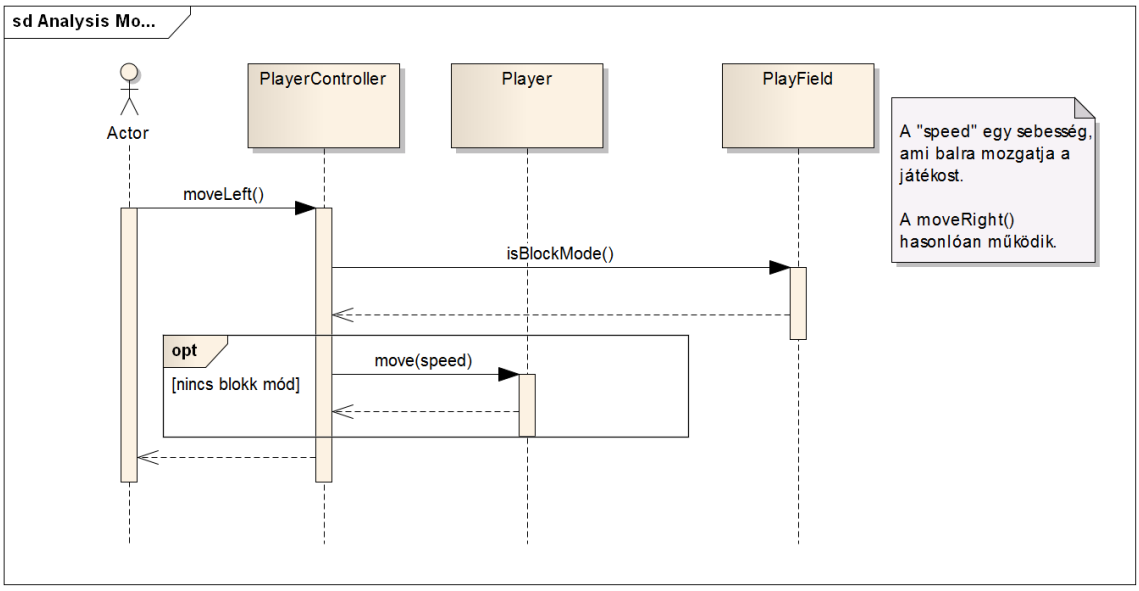

sd PlayFieldController2
Actor
pc:PlayFieldController
pf:PlayField
toggleMode()
toggleMode()


sd PlayerController
Actor
pih:PlayerInputHandler
pkc:PlayeKeyConfiguration
pc:PlayerController
1.handleKeyPressed()
1.1.getLeft()
1.2.getRight()
1.3.getJump()
alt
[ha balra mozgás van]
1.4.moveLeft()
[ha jobbra mozgás van]
1.5.moveRight()
[ha ugrás van]
1.6.jump()
2.handleKeyReleased()
2.1.getLeft()
2.2.getRight()
alt
[jobbra vagy balra mozgás billentyű felengedése]
2.3.stopMove()


sd Analysis Mo...
Actor
BlockController
PlayField
moveNorth()
isBlockMode()
opt
[ha blokk mód van]
move(b, 2)
move(b, 2): a "b" a kontrollerben tárolt blokk, az iránya pedig lefele, mivel a blokk alatt lévő blokkal cseréljük ki felfele mozgatásnál.
A többi metódus (moveEast/South/West) is teljesen azonos felépítésű, a move() paramétere értelemszerűen ennek függvényében különbözik.


sd Analysis Mo...
Actor
BlockInputHandler
BlockKeyConfiguration
BlockController
handleKeyPressed(k)
getNorth()
getEast()
getSouth()
getWest()
moveNorth()
alt
[fölfele]
moveEast()
[jobbra]
moveSouth()
[lefel]
moveWest()
[balra]
A handleKeyReleased(k) metódus üres.


sd Analysis Mo...
Actor
PlayerController
Player
PlayField
jump()
isBlockMode()
A speed egy olyan sebesség, ami felfele mozgatja a játékost.
Az aktuális sebesség alapján derül ki, hogy talajon áll-e.
getSpeed()
opt
move(speed)
[blokk mód és ugorhat]


sd Analysis Mo...
Actor
PlayerController
Player
PlayField
A "speed" egy sebesség, ami balra mozgatja a játékost.
A moveRight() hasonlóan működik.
moveLeft()
isBlockMode()
opt
move(speed)
[nincs blokk mód]

11.5. Napló

Kezdet	Időtartam	Résztvevők	Leírás
2012.04.18. 12:00	2 óra	Dányi	Grafikus felület architektúrájának tervezése.
2012.04.18. 20:00	1 óra	Dányi Kern Kolozsi	Megbeszélés, feladatok kiosztása. Kolozsi elkészíti a képernyőterveket, Kern a diagramokért felelős, Dányi kidolgozza a fennmaradó feladatokat.
2012.04.20. 09:00	1 óra	Kolozsi	Nézetek elkészítése
2012.04.21. 15:00	3 óra	Dányi	IO és grafikus rendszer tervezése.
2012.04.22. 09:00	2 óra	Dányi	IO és grafikus rendszer tervezése, javítása.
2012.04.22. 10:00	2 óra	Kern	Szekvencia diagramok
2012.04.22. 14:30	3 óra	Dányi	Menürendszer tervezése.
2012.04.22. 21:00	1 óra	Dányi	Menürendszer tervezése, dokumentáció írása.
2012.04.23. 10:00	4 óra	Dányi Kern	Szekvenciák,kommentezés+véglegesítés.

13. Grafikus változat

13.1. Fordítási és futtatási útmutató

13.1.1. Fájllista

Fájl neve	Méret	Keletkezés ideje	Tartalom
compile&run.bat	11 byte	2012.05.06	Fordítás és futtatás
compile.bat	363 byte	2012.05.06	Fordítás
default.json	2516 byte	2012.05.06	Alapértelmezett pálya
level1.json	2122 byte	2012.05.06	Pálya fájl
level2.json	2446 byte	2012.05.06	Pálya fájl
level0.json	2122 byte	2012.05.06	Pálya fájl
level1.json	2787 byte	2012.05.06	Pálya fájl
level2.json	2625 byte	2012.05.06	Pálya fájl
level3.json	2623 byte	2012.05.06	Pálya fájl
level4.json	2700 byte	2012.05.06	Pálya fájl
level5.json	2653 byte	2012.05.06	Pálya fájl
level6.json	3695 byte	2012.05.06	Pálya fájl
level7.json	3123 byte	2012.05.06	Pálya fájl
run.bat	50 byte	2012.05.06	Futtatás
save.dat	1 byte	2012.05.07	Mentést tartalmazó fájl
Grafikus.java	9975 byte	2012.05.06	A program fő osztálya
BlockController.java	1128 byte	2012.05.06	Blokk irányító
PlayerController.java	1948 byte	2012.05.06	Játékos irányító
PlayFieldController.java	492 byte	2012.05.06	Játéktér irányító
BlockContainerDrawer.java	1406 byte	2012.05.06	Blokk konténer rajzoló
BlockDrawer.java	585 byte	2012.05.06	Block rajzoló
DoorDrawer.java	1014 byte	2012.05.06	Ajtó rajzoló
Drawer.java	309 byte	2012.05.06	Rajzoló interfész
EntityContainerDrawer.java	948 byte	2012.05.06	Rajzoló osztály
EntityDrawer.java	612 byte	2012.05.06	Rajzoló osztály
FilledBlockDrawer.java	1710 byte	2012.05.06	Rajzoló osztály
GameFrame.java	2488 byte	2012.05.06	Rajzoló osztály
block_bg.png	9571 byte	2012.05.06	Grafikus elem
door.png	3611 byte	2012.05.06	Grafikus elem
game_won.png	191642 byte	2012.05.06	Grafikus elem
key_default.png	3365 byte	2012.05.06	Grafikus elem
key_obtained.png	3078 byte	2012.05.06	Grafikus elem

level_won.png	197099 byte	2012.05.06	Grafikus elem
cont.png	12181 byte	2012.05.06	Grafikus elem
cont_active.png	11633 byte	2012.05.06	Grafikus elem
exit.png	11631 byte	2012.05.06	Grafikus elem
exit_active.png	11247 byte	2012.05.06	Grafikus elem
new.png	11622 byte	2012.05.06	Grafikus elem
new_active.png	11168 byte	2012.05.06	Grafikus elem
menu_bg.png	200490 byte	2012.05.06	Grafikus elem
player_red_fall_left.png	3428 byte	2012.05.06	Grafikus elem
player_red_fall_right.png	3439 byte	2012.05.06	Grafikus elem
player_red_left.png	3388 byte	2012.05.06	Grafikus elem
player_red_left_move_0.png	3476 byte	2012.05.06	Grafikus elem
player_red_left_move_1.png	3473 byte	2012.05.06	Grafikus elem
player_red_left_move_2.png	3490 byte	2012.05.06	Grafikus elem
player_red_left_move_3.png	3452 byte	2012.05.06	Grafikus elem
player_red_right.png	3388 byte	2012.05.06	Grafikus elem
player_red_right_move_0.png	3482 byte	2012.05.06	Grafikus elem
player_red_right_move_1.png	3478 byte	2012.05.06	Grafikus elem
player_red_right_move_2.png	3467 byte	2012.05.06	Grafikus elem
player_red_right_move_3.png	3449 byte	2012.05.06	Grafikus elem
player_white_fall_left.png	3458 byte	2012.05.06	Grafikus elem
player_white_fall_right.png	3455 byte	2012.05.06	Grafikus elem
player_white_left.png	3368 byte	2012.05.06	Grafikus elem
player_white_left_move_0.png	3472 byte	2012.05.06	Grafikus elem
player_white_left_move_1.png	3483 byte	2012.05.06	Grafikus elem
player_white_left_move_2.png	3493 byte	2012.05.06	Grafikus elem
player_white_left_move_3.png	3455 byte	2012.05.06	Grafikus elem
player_white_right.png	3368 byte	2012.05.06	Grafikus elem
player_white_right_move_0.png	3483 byte	2012.05.06	Grafikus elem
player_white_right_move_1.png	3505 byte	2012.05.06	Grafikus elem
player_white_right_move_2.png	3490 byte	2012.05.06	Grafikus elem
player_white_right_move_3.png	3461 byte	2012.05.06	Grafikus elem
wall.png	39289 byte	2012.05.06	Grafikus elem
window_bg.png	67814 byte	2012.05.06	Grafikus elem
KeyDrawer.java	1027 byte	2012.05.06	Rajzoló osztály
MenuDrawer.java	1364 byte	2012.05.06	Rajzoló osztály
MenuItemDrawer.java	1289 byte	2012.05.06	Rajzoló osztály
PlayerContainerDrawer.java	976 byte	2012.05.06	Rajzoló osztály

PlayerDrawer.java	7061 byte	2012.05.06	Rajzoló osztály
PlayFieldDrawer.java	1460 byte	2012.05.06	Rajzoló osztály
StaticScreen.java	425 byte	2012.05.06	Rajzoló osztály
WallDrawer.java	1460 byte	2012.05.06	Rajzoló osztály
BlockInputHandler.java	1703 byte	2012.05.06	Inputkezelő osztály
BlockKeyConfiguration.java	1312 byte	2012.05.06	Inputkezelő osztály
GameSaver.java	1252 byte	2012.05.06	Inputkezelő osztály
InputDispatcher.java	1162 byte	2012.05.06	Inputkezelő osztály
InputHandler.java	546 byte	2012.05.06	Inputkezelő osztály
PlayerInputHandler.java	1801 byte	2012.05.06	Inputkezelő osztály
PlayerKeyConfiguration.java	1394 byte	2012.05.06	Inputkezelő osztály
PlayFieldInputHandler.java	1154 byte	2012.05.06	Inputkezelő osztály
PlayFieldKeyConfiguration.java	690 byte	2012.05.06	Inputkezelő osztály
ContinueGameMenuItem.java	507 byte	2012.05.06	Menüosztály
ExitGameMenuItem.java	390 byte	2012.05.06	Menüosztály
Menu.java	1792 byte	2012.05.06	Menüosztály
MenuItem.java	984 byte	2012.05.06	Menüosztály
PlayGameMenuItem.java	433 byte	2012.05.06	Menüosztály
Block.java	6004 byte	2012.05.06	Blokk alaposztály
BlockMatcher.java	5370 byte	2012.05.06	Blokk illesztő
EmptyBlock.java	2162 byte	2012.05.06	Üres blokk
FilledBlock.java	9122 byte	2012.05.06	Sima blokk
BlockContainer.java	1745 byte	2012.05.06	Konténer osztály
EntityContainer.java	1883 byte	2012.05.06	Konténer osztály
PlayerContainer.java	1610 byte	2012.05.06	Konténer osztály
Game.java	1235 byte	2012.05.06	Játék osztály
PlayField.java	11002 byte	2012.05.06	Pálya osztály
Door.java	2593 byte	2012.05.06	Ajtó osztály
Entity.java	2406 byte	2012.05.06	Entity osztály
Key.java	2808 byte	2012.05.06	Kulcs osztály
SpawnPoint.java	1823 byte	2012.05.06	Spawnpoint osztály
Wall.java	3553 byte	2012.05.06	Fal osztály
CommandException.java	280 byte	2012.05.06	Kivétel osztály
BlockInfo.java	473 byte	2012.05.06	Debug osztály
EntityPosition.java	1005 byte	2012.05.06	Pozíció segédosztály
Info.java	501 byte	2012.05.06	Debug interfész
PlayerSpawnPoint.java	1227 byte	2012.05.06	Konténer osztály
Position.java	678 byte	2012.05.06	Pozíció segédosztály

Speed.java	764 byte	2012.05.06	Sebesség segédosztály
Collideable.java	219 byte	2012.05.06	Ütközés osztály
CollisionProcesser.java	2861 byte	2012.05.06	Ütközés osztály
Rectangle.java	1098 byte	2012.05.06	Ütközés osztály
Shape.java	133 byte	2012.05.06	Ütközés osztály
Player.java	5833 byte	2012.05.06	Játékos osztály
MapFromJson.java	11622 byte	2012.05.06	Pályaépítő
PlayFieldBuilder.java	3926 byte	2012.05.06	Pályaépítő
Diff.java	3475 byte	2012.05.06	Segédeszköz
Logger.java	1306 byte	2012.05.06	Segédeszköz
JSONArray.java	30073 byte	2012.05.06	JSON osztály
JSONException.java	706 byte	2012.05.06	JSON osztály
JSONObject.java	55756 byte	2012.05.06	JSON osztály
JSONString.java	733 byte	2012.05.06	JSON osztály
JSONTokener.java	12945 byte	2012.05.06	JSON osztály
test.bat	1434 byte	2012.05.06	teszt

13.1.2. Fordítás és telepítés

A program fordításához elengedhetetlen a Java SDK 6-os verziója, ez a HSZK számítógépein elérhető. A parancssorba a `javac -version` parancsot írva meg kell jelennie a fordító verziójának (ha nem elérhető a parancs, hibaüzenet keletkezik).

A fordítás egyszerűen elvégezhető, ha a gyökérkönyvtárban kiadjuk a `compile` parancsot. Ekkor a Java fordító a „src” könyvtárban található forrásfájlokat lefordítja, majd a „build” könyvtárba helyezi az elkészített binárisokat.

A fordítás manuálisan is megtehető az alábbi parancs kiadásával:

```
javac -classpath src/ -encoding UTF-8 -d build/ src/fearlesscode/app/Grafikus.java
```

Egyéb teendő nincs, a program lefordított állapotban nem igényel külön telepítést. Futtatási útmutató a 13.1.3 fejezetben.

13.1.3. Futtatás

A futtatás szükséges feltétele a Java 6-os verziója, ez a HSZK számítógépein elérhető. A parancssorból ennek meglétét a `java -version` parancs kiadásával ellenőrizhetjük.

A futtatáshoz szükség van a már lefordított binárisokra is, ennek előállítása a 13.1.2 fejezetben található.

A futtatáshoz adjuk ki egyszerűen a `run` parancsot.
Manuálisan indításhoz az alábbi parancsot kell kiadni:

```
java -classpath build/ fearlesscode.app.Grafikus
```

A játék menüjében az iránygombokkal navigálhatunk (majd az enter billentyűvel kiválaszthatjuk a menüpontot). A játékban az iránygombokkal, valamint a WASD billentyűkkel irányíthatjuk a játékosokat.

13.2. Értékelés

Tag neve	**Munka százalékban**
Dányi	38%
Dudás	16%
Kern	24%
Kolozsi	22%

13.3. Napló

Kezdet	Időtartam	Résztvevők	Leírás
2012.04.25. 13:00	2 óra	Dányi	Felmerült bugok rendszerezése, javítása, naplók összesítése.
2012.04.26. 00:00	0.5 óra	Dányi	Grafikus rendszer kódolása.
2012.04.27. 23:00	2 óra	Dányi	Dokumentáció rendbeszedése.
2012.04.28. 12:00	1.5 óra	Dányi	Drawerek optimalizálása, mentés/betöltés implementálása, bugok javítása.
2012.04.28. 17:00	2 óra	Dányi	Blokkátmenet közben fellépő bugok javítása, blokk mozgatással kapcsolatban bugok javítása.
2012.04.29. 14:00	3 óra	Kern	Dokumentáció
2012.04.29. 22:00	3 óra	Kolozsi	Mapok json kódolása.
2012.05.03. 21:00	1 óra	Dudás	Textúrázás.
2012.05.04. 00:00	2 óra	Dudás	Textúrázás, kód implementálás.
2012.05.04. 22:00	5.5 óra	Dudás	Textúrázás, kód implementálás, animációk, menü grafikus megjelenítése.
2012.05.05. 15:00	2 óra	Dányi	Statikus képernyők programozása, bugok javítása.
2012.05.05. 17:00	1 óra	Kolozsi	Tesztelés, konfiguráció, blokkváltásnál player ne mozogjon
2012.05.06. 15:00	3 óra	Dudás	Bugfix, kép cache, statikus képernyők, funkciók implementálása
2012.05.06. 18:00	0.5 óra	Kern	Jegyzőkönyv,tesztelés
2012.05.07. 10:00	0.5 óra	Dányi	Dokumentáció véglegesítés

14. Összefoglaló

14.1. A projectre fordított idő

A projectre összesen 310.5 munkaórát fordítottunk, ennek megoszlása az alábbi:

Csapattag	Szkeleton	Proto	Grafikus	Összesen
Dányi	43	47.5	26.5	117
Dudás	15.5	23	11.5	50
Kern	34.5	29	10.5	74
Kolozsi	27.5	36	6	69.5

Az összesített százalékos értékek a 13. fejezetben („Grafikus változat”) találhatóak.

14.2. A project során elkészült program statisztikái

Fázis	Fájlok	Sorok	Program	Komment
Szkeleton	20	980	636	242
Proto	31	3577	2045	1214
Grafikus	65	6493	3679	2185

A „Sorok” oszlop nem feltétlenül egyenlő a „Program”+„Komment” oszlopokkal, a különbözet az olyan sorokat jelöli, amik sem programkódot, sem kommentet nem tartalmaznak.

A grafikus változatban a forrássorok 57%-a volt programkód, 34%-a komment, 10%-a pedig whitespace, az üres sorok nélkül pedi a kommentek arány 37% a forrássorokhoz képest.

14.3. Személyes tapasztalatok

14.3.1. Dányi

Tanulságok A félév során számomra a legnagyobb tanulságokat a csapatmunka jelentette, ilyenkor szembesül vele az ember, hogy az egyéni fejlesztéshez képest a csoportmunka kezdetben nagy overheadet jelent munka szempontjából, hiszen figyelmet kell fordítani a folyamatos kommunikációra, a csapat által bevezetett normákat betartani.

Legnehezebb és legkönnyebb rész A legkönnyebb rész számomra a Szkeleton elkészítése jelentette, főleg a sok kreativitást igénylő modellezés fejezetek. A legnehezebb pedig a Proto elkészítése, hisz ilyenkor bukkant föl a legtöbb nem várt akadály.

Arányok Az egyes részekre fordított idő, és a kapható pontok szerintem megfelelő arányban álltak egymással, egyedül a prototípust érzem túl hangsúlyosnak a szkeletonhoz képest.

Nehézségek A legnagyobb nehézséget a protó előkészítése jelentette, itt szerintem a megoldandó feladatok alulspecifikáltak voltak.

Változások A protóra szánt idő lehetne hosszabb, a szkeleton rovására.

Ajánlott feladatok A tárgy keretein belül egy „tower defense" típusú játék elkészítése során az objektum-orientáltságot nagyon szemléletesen be lehetne mutatni, az ilyen fajta játékoknál a modell szinte adja magát, ám megvalósítása mégsem triviális. Az elkészítéséhez nem feltétlenül van szükség komolyabb algoritmusokra (egy gonddal kevesebb), így magára a modellre megfelelő időt lehetne szánni, ezen felül jól tesztelhető vizuális eszközök nélkül.

14.3.2. Dudás

Tanulságok Láttuk realizálódni az egész folyamatot, ami eddig kicsit megfoghatatlan volt, mivel csak tudásként volt meg bennünk. Ennek következtében sokkal jobban elmélyült bennünk egy szoftverfejlesztés lépései, hiszen nem csak tankönyvekben láttuk milyen lépést mi követ, hanem a saját bőrünkön is megtapasztaltuk.

Változások A heti bontás teljesen jó, de el lehetne csúsztani, hogy ne hétfőn, hanem pénteken legyen a beadás, mert így sokkal hatékonyabban lehetne dolgozni.

Ajánlott feladat Angry Birds!

14.3.3. Kern

Tanulságok Összességében nagyon hasznosnak találtam a tárgyat, kár hogy nem tudtam vele eleget foglalkozni. A csapatban való dolgozás és egy konkrét fejlesztési folyamat megismerése természetesen mindenkinek csak előnyére válhat.

Arányok Munka/kredit arányt nem találtam helyénvalónak. Ráadásul a leadási idő egybeesett a zh időponttal.Eleinte nagyon nem tetszett, hogy ennyi diagramot kell rajzolni, de aztán később nagyon jól jöttek kódolásnál.

Változások A megvalósítandó program szerintem megfelelő volt, mert kellő komplexitású és nem unalmas. Későbbi években is jól járnának ehhez hasonló platform játékokkal.

14.3.4. Kolozsi

Tanulságok Megtapasztaltuk a csapatmunka előnyeit és hátrányait.

Legnehezebb és legkönnyebb rész Legnehezebbek az első részek voltak, ahol meg kellett hozni azokat a döntéseket, amik a project egészét meghatározták.

Legkönnyebbek a végső részek, ahol már lényegi döntéseket nem kellett hozni, csak meg kellett valósítani azt amit már elötte kidolgoztunk.

Arányok Jó a beosztás.

Változtatás A jövőben odafigyelhetnének arra, hogy ne a ZH napra rakják a beadást, mivel igy választani kell hogy melyikre fordít az ember több időt.