**6. Prototípus koncepciója**

15 – jgoldfisch

Konzulens:

Bodó Zsófia

Csapattagok

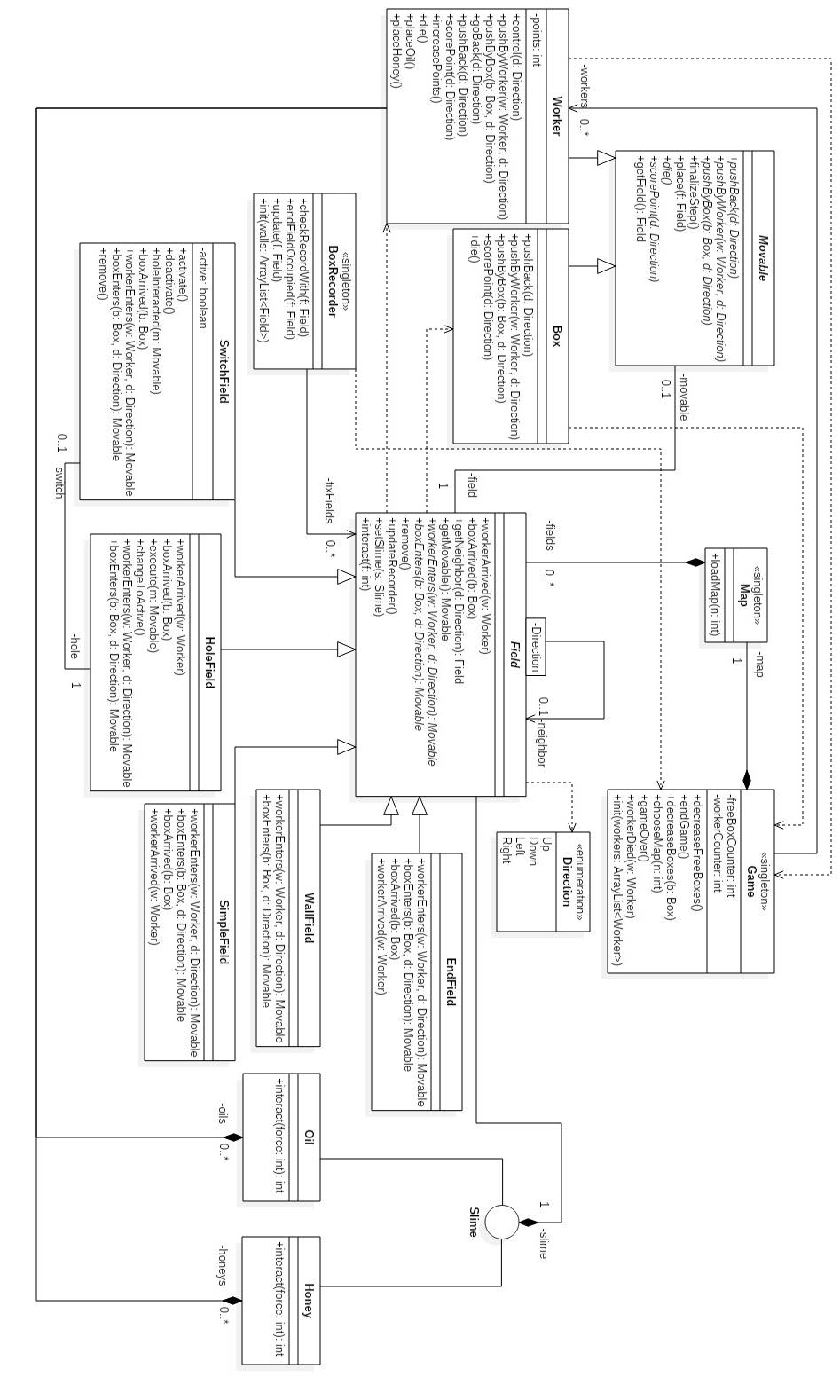
|  |  |  |
| --- | --- | --- |
| Mátyás Gergely | IL21NI | matyasg97@gmail.com |
| Horváth Ákos | DKILK6 | horvath.akos1997@gmail.com |
| Gurubi Barnabás | DXEXVR | gurubibarni@gmail.com |
| Tolnai Márk | ID61MK | tolesz11@windowslive.com |
| Bertalan Bálint | HNN9GA | blintber@gmail.com |

2018. március 25.

# Prototípus koncepciója

## Változás hatása a modellre

### Módosult osztálydiagram



### Új vagy megváltozó metódusok

#### Box

#### Metódusok

* **void pushByBox(Box b, Direction d, int f):** Függvény arra az esetre, amennyiben a ládánkat (**Box**) egy másik doboz tolja meg, intézi, hogy a láda a megfelelő helyre kerüljön. Továbbá ellenőrzi, hogy a kapott erővel van-e lehetőség eltolni a doboz-t, figyelembe véve a mezőn lévő anyagot.
* **void pushByWorker(Worker w, Direction d, int f):** A láda (**Box**) egy játékos által való megtolásákor hívjuk meg, intézi hogy a láda a megfelelő helyre kerüljön. Továbbá ellenőrzi, hogy a munkás által átadott erővel van-e lehetőség a dobozt eltolni a mezőn lévő anyag függvényében.

#### Field

#### Attribútumok

* **Slime slime:** referencia a hozzá tartozó **Slime-**hoz.

#### Metódusok

* **void setSlime(Slime s):** Beállítja a mezőn lévő **Slime** referenciáját a kapottra.
* **int interact(int f):** A kapott erőt a mezőn lévő anyag függvényében megváltoztatja.

#### HoleField

#### Metódusok

* **void changeToActive():** A kapcsoló megváltoztatása esetén hívjuk meg, hogy ha áll valami abban a pillanatban a lyukon, akkor az megsemmisül. Ezen felül megszünteti a lyukon lévő **Slime** referenciáját.

#### Honey

#### Felelősség

Növeli a mező súrlódását.

#### Ősosztályok

Slime

#### Metódusok

* **int interact(int f):** Növeli a kapott erőt.

#### Oil

#### Felelősség

Csökkenti a mező súrlódását.

#### Ősosztályok

Slime

#### Metódusok

* **int interact(int f):** Csökkenti a kapott erőt.

#### Slime

#### Felelősség

A mező surlódását módosítható objektumokhoz tartozó interface.

#### Metódusok

* **int interact(int f):** A kapott erőt módosítja. A megvalósítása az interface-t megvalósító osztályokban található.

#### Worker

#### Attribútumok

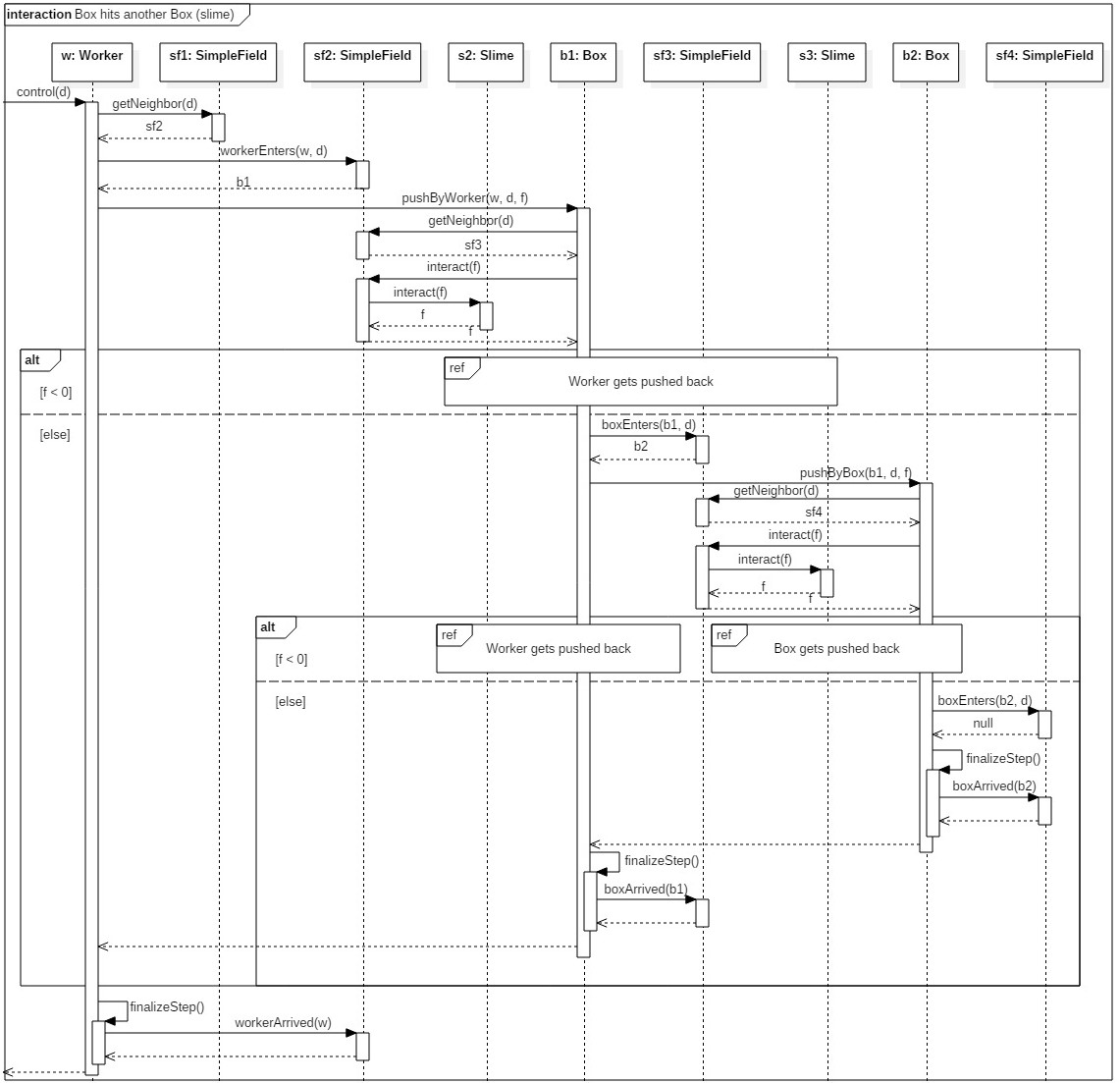
* **int force:** A játékos ereje, mely minden meghatározza, hogy el tudja-e tolni a dobozok sorát.
* **ArrayList<Oil> oils:** Referenciák a munkás által lerakható olaj **Slime**-okra.
* **ArrayList<Honey> honeys:** Referenciák a munkás által lerakható méz **Slime**-okra.

#### Metódusok

* **void placeOil():** Függvény arra az esetre, ha a játékos egy olajmezőt helyez a pályára.
* **void placeHoney():** Függvény arra, ha a játékos egy mézet helyez a pályára.
* **void pushByBox(Box b, Direction d, int f):** Ez a függvény arra az esetre szolgál, hogyha a munkást egy dobozzal (**Box**) próbálnának meg eltolni. Továbbá továbbadja a kapott erőt a következőnek a sorban.
* **void pushByWorker(Worker w, Direction d, int f):** Ez a függvény arra az esetre szolgál, ha a munkást egy másik munkás próbálná meg eltolni. Továbbá továbbadja a kapott erőt a következőnek a sorban.

### Szekvencia-diagramok

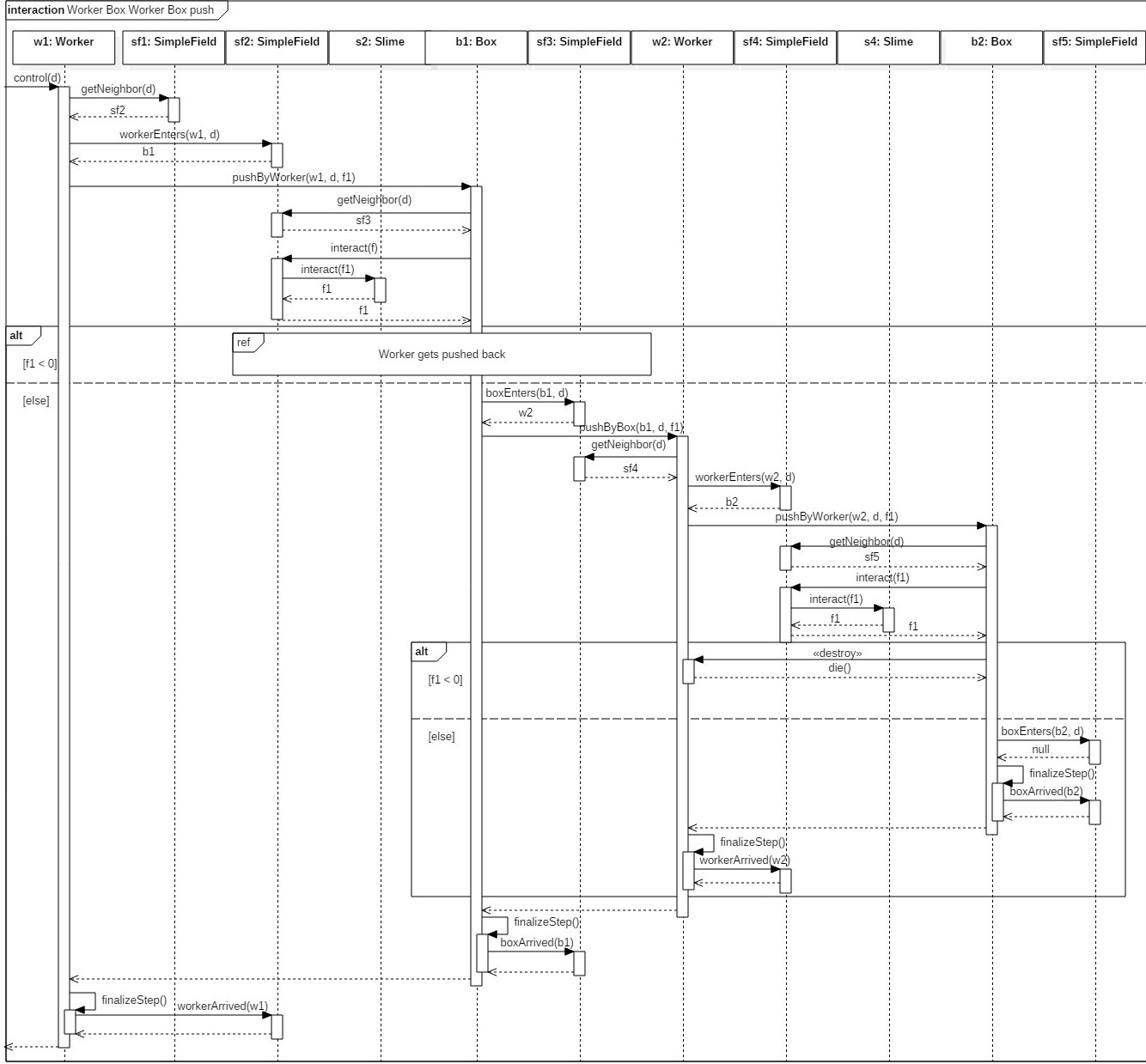
#### Worker 🡪 Box 🡪 Box push



Ezen a szekvencián a Worker 🡪 Box 🡪 Box eset van ábrázolva. A Worker erejéből mindig levonódik egy bizonyos mennyiség a mezőn található Slime függvényében. Amennyiben a munkás ereje elfogy, a Worker, illetve a doboz(ok) is visszakerülnek az eredeti helyükre.

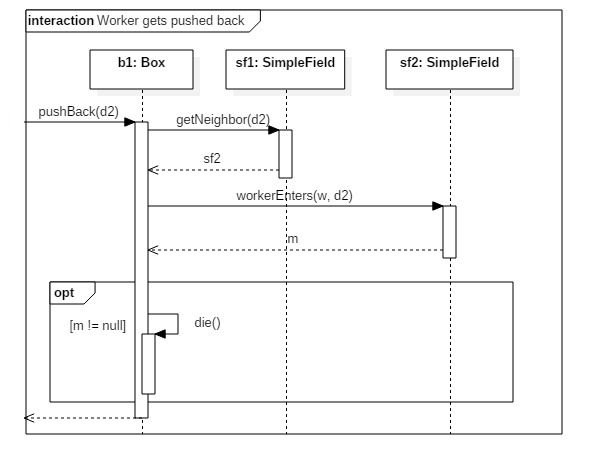
Ha nem fogy el az erő a második Box után sem, akkor az egész sor egyet jobbra lép.

#### Worker 🡪 Box 🡪 Worker 🡪 Box push



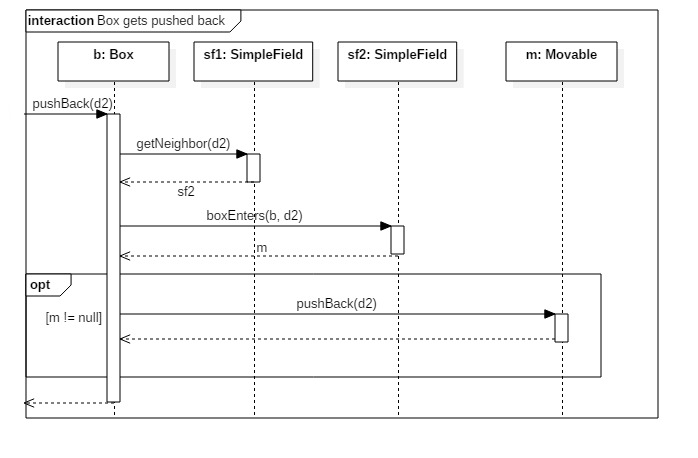
Ezen a szekvencián a Worker 🡪 Box 🡪 Worker 🡪Box eset van ábrázolva. A Worker erejéből mindig levonódik egy bizonyos mennyiség a mezőn található Slime függvényében. Ha egy Worker következik a sorban, akkor nem vonunk le a sor elején lévő Worker erejéből, hanem csak továbbadjuk az erőt. Amennyiben a Worker ereje elfogy az első Box-nál akkor a munkás visszakerül az eredeti helyére. Ha a második Boxnál fogy el az erő, akkor pedig a két Box között lévő Worker meghal, hiszen rátolódik a tőle balra lévő Box. Ha az erő a második Box után sem fogy el, akkor az egész sor egyet jobbra lép.

#### Worker gets pushed back



Ez a szekvencia a Worker visszalépését szemlélteti.

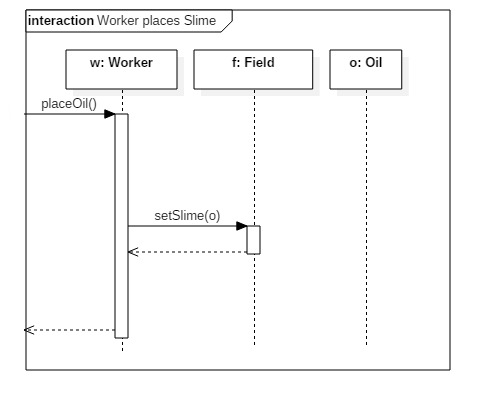
#### Box gets pushed back



Ez a szekvencia a Box visszalépését szemlélteti.

#### Worker places Slime

Ez a szekvencia az Oil, azaz az olaj elhelyezését szemlélteti. A placeOil() függvény a megfelelő billentyűlenyomásra hívódik meg. Amennyiben Honey objektumot szeretne elhelyezni a játékos, akkor a placeHoney() függvény hívódna meg a placeOil() függvény helyett, illetve Oil referencia helyett Honey referencia lenne átadva, de egyébként megegyező lenne a szekvenciája, ezért itt most csak ezt ábrázoltuk.



## Prototípus interface-definíciója

### Az interfész általános leírása

A program interfésze csak a szabványos bemenetről fogad parancsokat, ezzel lehetővé téve mind a terminálból való vezérlést, mind a fájlból való parancsfogadást. Kimenttét a szabványos kimentere írja, így támogatva a már említett terminálból történő használatot, valamint átirányítható fájlba.

A fájlból való parancs fogdásnak és kimenet átirányításnak köszönhetően mód van az előre elkészített automatikus tesztek futtatására, és ezek eredményének elmentésére. Ezzel lehetséges az automatikus tesztelés, mely sokkal hatékonyabb és eredményre vezetőbb, mint a hagyományos „kézi” tesztelés. Ezek a tesztesetek parancsok sorozatából állnak.

### Bemeneti nyelv

load-map <filename>

**Leírás:** Pályabetöltése fájlból.

**Opciók:**

* <filename> - Pályát tartalmazó fájl neve

control <worker-id> <direction>

**Leírás:** Adottmunkás léptetése egy irányba.

**Opciók:**

* <worker-id> - Munkás azonosítója
* <direction> - Irány [-u|-r|-d|-l] - fel (Up), jobbra (Right), le (Down), balra (Left)

drop-special <worker-id> <type>

**Leírás:** Adottmunkással méz vagy olaj elhelyezése az aktuális mezőjére.

**Opciók:**

* <worker-id> - Munkás azonosítója
* <type> - Anyag fajtája [-h|-o] - méz (Honey), olaj (Oil)

ls-workers

**Leírás:** Pályán lévő munkások listázása.

**Opciók:** -

ls-boxes

**Leírás:** Pályán lévő dobozok listázása.

**Opciók:** -

ls-fields

**Leírás:** Pályán lévő mezők listázása.

**Opciók:** -

log [<filename>]

**Leírás:** Metódus hívások naplózása, ha meg van adva opcionális fájlnév akkor fájlba, ha nincs, akkor a képernyőre.

**Opciók:**

* <filename> - Opcionális. Ha nincs meg adva akkor a képernyőre naplóz, ha meg van adva akkor az adott fájlba.

show-log

**Leírás:** Kiírja a naplózások helyét. (pl. scrn (képernyő), logfile.txt)

**Opciók:** -

log-off [<filename> | -all]

**Leírás:** Leállítja az adott egység naplózását.

**Opciók:** [<filename> | -all] – Opcionális. Kihagyva a képernyőre való naplózást állítja le. Fájlnevet megadva az adott fájlba való naplózást. -all kapcsolóval pedig minden naplózást.

drop-map

**Leírás:** Az aktuális pálya eldobása (például rossz betöltése esetén).

**Opciók:** -

show-map

**Leírás:** Az aktuális pálya kiírása (fájlnév).

**Opciók:** -

run-test <filename>

**Leírás:** Előre megírt teszteset futtatása.

**Opciók:** <filename> - A tesztesetet tartalmazó fájl.

exit

**Leírás:** Kilépés a programból.

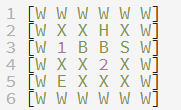
**Opciók:** -

**Konfigurációs fájlok nyelvtana**

**Pályát megadó fájl**

Annak érdekében, hogy minél átláthatóbban, egyszerűbben megadhassa a felhasználó a saját pályáját, egy saját, letisztultságra törekvő nyelvtant dolgoztunk ki. Formátuma egyszerű txt.

* **Először egy példa, utána magyarázat:**



Ez egy 6x6-os pálya, fallal körülvéve, két munkással, két dobozzal, lyukkal, kapcsolóval és egy célmezővel.

**Nyelvtan:**

A pálya méretét a sorok és oszlopok száma határozza meg, az oszlopok szóközzel elválasztottak. Minden sor *[* szimbólummal kezdődik és *]* szimbólummal végződik (sortörés, a jobb áttekinthetőség érdekében ajánlott. A sor nyitó és sor záró szimbólumok között különböző elemek szerepelhetnek (lsd. lentebb). A mezők alapból sima mezők, melyeken állhatnak entitások, ha nem sima mezőt szeretnénk módosítót kell elhelyezni.

***Megkötések*:** A soroknak ugyanolyan hosszúnak kell lenniük, különben érvénytelen a pálya.

**Elemek:**

* *W – Fal elem (Wall)*
* *H – Lyuk (mező módosító, Hole)*
* *S – Kapcsoló (mező módosító, Switch)*
* *E – Célmező (mező módosító, Endzone)*

Az előzőekben említett módosítók nélkül egy adott mező sima mező lesz, ekkor helyezkedhet rajta inicializáláskor entitás, vagy lehet üres.

* *X – Az adott sima mezőn nincs semmi.*
* *B – Az adott sima mezőn egy doboz található (Box)*
* *0-9 – Az adott mezőn egy munkás található, aminek az azonosítója (worker-id) az adott egyjegyű szám.*

Egyéb felmerülő kérdés, hogy melyik lyukhoz, melyik kapcsoló tartozik. Ezt a kérdést az inicializálás egy megadott módon dönti el, ami a következő:

Az egyes elemek inicializálása balról-jobbra, fentről-le történik, ha kapcsolóhoz, illetve lyukhoz ér, akkor annak a párja az inicializálást folytatva az első megfelelő elem. (A példában a 2. sor 4. helyén álló lyuk és a 3. sor 5. helyén álló kapcsoló tehát össze van kötve). Amennyiben nem jut az adott elemnek „pár”, akkor az annak megfelelően viselkedik.

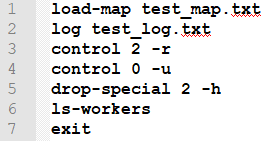
Ez a saját nyelvtan jelentősen leegyszerűsíti a pályák megadását, ami más általános formátumban (pl. XML, JSON) leírva sokkal kevésbé lenne áttekinthető. Jó példa a struktúra által nyújtott segítésére, hogy a megadás szemantikájában benne vannak a mező szomszéd kapcsolatok, így azokat explicit nem szükséges megadni. (Ez a példában szereplő 6x6-os pályán 120 szomszéd kapcsolat „megspórolása”, hiszen ezeket az inicializálás automatikusan felderíti és beállítja).

**Tesztesetet megadó fájl (parancs szkript)**

Lehetőség van atesztesetek előre való összeállítására, ezzel automatikusan futtatva őket, így sokkal hatékonyabb tesztelést elérve.

Ennek módja, hogy parancsokat előre leírunk megfelelő sorrendben, megfelelően paraméterezve egy egyszerű txt fájlba. A formátum választása itt is azért esett erre az egyszerű esetre, mert így lehet a lehető legegyszerűbben összeállítani egy ilyen fájlt, szinte magától értetődően, hiszen igazából egy parancssorozat.

* **Először egy példa, utána magyarázat (***run***-***test test.txt***) :**



Ez a teszteset betölt egy pályát, beállítja a naplózás helyét, irányítja a 2-es azonosítójú munkást jobbra, a 0-s azonosítójút felfele. Ezután a 2-es azonosítójú munkással lerak egy adag mézet a mezőjére. Végezetül kilistázza a munkásokat. Utána a tesztesetből való kilépés következik.

**Nyelvtan:**

A tesztesetek leírása tulajdonképpen parancsok sorozata, amiket a terminálban is ugyanígy adnánk ki. Fontos, hogy minden parancs külön sorban kell, hogy elhelyezkedjen.

***Tudnivalók:*** Minden tesztesetbe való belépéskor a terminálban beállított naplózási módok törlődnek, tehát azok nem befolyásolják a tesztesetben való naplózást (kilépéskor visszaállítódnak a teszteset előtti naplózási módok, de a tesztesetben beállítottak nem lesznek érvényben.)

Az egyes tesztesetekben ugyanazok a parancsok érhetőek el, ugyanolyan formában, mint a program átlagos futtatásánál. Tesztesetet a ***run-test <filename>*** paranccsal indíthatunk. Minden tesztesetet az ***exit*** parancsnak kell zárnia.

### Kimeneti nyelv

Kimenetet csak a program, valamint az objektumok állapotát lekérdező parancsok állítanak elő, ezek a következőek (kimenettel együtt):

**Program állapotát lekérdező parancsok:**

* show-log

**Kimenet**: Egy lista a naplózások helyéről

|  |  |
| --- | --- |
| **Minta** | **Példa** |
| <log-destination> | scrn |
| <log-destination> | log1.txt |
| <log-destination> | method\_log.txt |

* show-map

**Kimenet**: Aktuális pályafájl

|  |  |
| --- | --- |
| **Minta** | **Példa** |
| <map-file> | small\_map.txt |

**Objektumok állapotát lekérdező parancsok**

* ls-workers

**Kimenet**: Egy lista a pályán lévő munkásokról, állapotokkal együtt (azonosító, pozíció a pályán (vízszintes, függőleges, mindegyik 1-től indul), tolóerő).

|  |  |
| --- | --- |
| **Minta** | **Példa** |
| <worker-id> <x-pos> <y-pos> <force> | ID:2 X:3 Y:5 F:6 |
| <worker-id> <x-pos> <y-pos> <force> | ID:0 X:1 Y:2 F:8 |

* ls-boxes

**Kimenet**: Egy lista a pályán lévő dobozokról, állapotokkal együtt (azonosító, pozíció a pályán (vízszintes, függőleges, mindegyik 1-től indul), beszorult-e).

|  |  |
| --- | --- |
| **Minta** | **Példa** |
| <box-id> <x-pos> <y-pos> <stuck> | ID:b2 X:6 Y:3 false |
| <box-id> <x-pos> <y-pos> <stuck> | ID:b0 X:1 Y:1 true |

* ls-fields

**Kimenet**: Egy lista a pályán mezőiről, állapotokkal együtt (pozíció a pályán (vízszintes, függőleges, mindegyik 1-től indul), fajta) ezek az általános információk, a mezőspecifikusak:

Kapcsoló: be/ki, méz/olaj/semmi, doboz/munkás/semmi; Lyuk: nyitva/csukva, méz/olaj/semmi, doboz/munkás/semmi; Sima mező: méz/olaj/semmi, doboz/munkás/semmi; Célmező: méz/olaj/semmi, doboz/munkás/semmi

|  |  |
| --- | --- |
| **Minta** | **Példa** |
| <x-pos> <y-pos> wall | X:3 Y:5 wall |
| <x-pos> <y-pos> simple <spec> <mov-id> | X:1 Y:2 simple h 1 |
| <x-pos> <y-pos> hole <state> <spec> <mov-id> | X:2 Y:6 hole 1 x x |
| <x-pos> <y-pos> switch <state> <spec> <mov-id> | X:4 Y:4 switch 1 x b1 |
| <x-pos> <y-pos> endz <spec> <mov-id> | X:5 Y:6 endz h b3 |

## Összes részletes use-case

[A use-case-eknek a részletezettsége feleljen meg a kezelői felületnek, azaz a felület elemeire kell hivatkozniuk a bemeneti nyelv parancsai alapján.

Alábbi táblázat minden use-case-hez külön-külön.]

|  |  |
| --- | --- |
| **Use-case neve** |  |
| **Rövid leírás** |  |
| **Aktorok** |  |
| **Forgatókönyv** |  |

## Tesztelési terv

[A tesztelési tervben definiálni kell, hogy a be- és kimeneti fájlok egybevetésével miként végezhető el a program tesztelése. Meg kell adni magas szintű teszt forgatókönyveket. Az egyes teszteket elég informálisan, szabad szövegként leírni, tesztesetenként egy-öt mondatban. Minden teszthez meg kell adni, hogy mi a célja, a proto mely funkcionalitását, osztályait stb. teszteli. Az alábbi táblázat minden teszt-esethez külön-külön elkészítendő.]

|  |  |
| --- | --- |
| **Teszt-eset neve** |  |
| **Rövid leírás** |  |
| **Teszt célja** |  |

## Tesztelést támogató segéd- és fordítóprogramok specifikálása

[Rövid bemutatással (elvárt funkcionalitás) specifikálni kell a tesztelést támogató segédprogramokat.]

## Napló

|  |  |  |  |
| --- | --- | --- | --- |
| **Kezdet** | **Időtartam** | **Résztvevők** | **Leírás** |
| 2018.03.21. 18:00 | 2 óra | Bertalan  Tolnai  Horváth  Mátyás  Gurubi | **Értekezlet**.  **Döntés**: Horváth elkészíti a tesztelési tervet (7.3) és 7.4-et  Bertalan elkészíti a 7.0.1-7.0.2  Tolnai elkészíti a 7.0.3-at  Gurubi elkészíti a 7.1-et  Mátyás elkészíti a 7.2-őt  Határidő vasárnap délelőtt. |
| 2018.03.22. 18:00 | 2 óra | Bertalan | **Tevékenység**: Bertalan kidolgozza a változtatások hatását a metódusokra (7.0.2) továbbá elkezdi az osztálydiagramot (7.0.1). |
| 2018.03.23 17:00 | 2 óra | Tolnai | **Tevékenység**: Tolnai elkezdi a megváltozott szekvenciadiagramok elkészítését. |
| 2018.03.23 19:00 | 1 óra | Mátyás | **Tevékenység**: use-case-ek elkezdése |
| 2018.03.24 10:00 | 2 óra | Hotváth | **Tevékenység**: Tesztesetek átgondolása, elkezdése |
| 2018.03.24 11:00 | 2 óra | Gurubi | **Tevékenység**: Bemeneti nyelv terve, megfogalmazása. |
| 2018.03.24 15:00 | 1 óra | Tolnai | **Tevékenység**: Szekvenciadiagramok folytatása, majdnem összes elkészítése. |
| 2018.03.24 19:00 | 1 óra | Bertalan | **Tevékenység**: Bertalan befejezi az osztálydiagramot és javítja a felmerülő változásokat a metódusokban. |
| 2018.03.25 09:00 | 1 óra | Horváth | **Tevékenység**: Tesztesetek befejezése. |
| 2018.03.25 13:00 | 1 óra | Mátyás | **Tevékenység**: use-case-ek befejezése. |
| 2018.03.25 15:00 | 1 óra | Gurubi | **Tevékenység**: Kimeneti nyelv terve, megfogalmazása. |
| 2018.03.25 19:00 | 1 óra | Mátyás | **Tevékenység**: Dokumentum összeállítása, felmerülő hibák, problémák javítása. |