**3. Analízis modell kidolgozása**

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 4.

# Analízis modell kidolgozása

## Objektum katalógus

### Box

A pályán található egyes dobozokat képviselik. Amennyiben egy dobozt a célmezőre tolnak ellehetetleníti önmaga mozgatását. **Wall**-on nem tolható át. **Worker** tudja tolni.

### EndField

A célmezőt képviseli, amire a **Box-**t kell tolni, hogy pontot kapjunk. Amennyiben egy **Box** rátolódik, az ott mozgathatatlanná válik, és a **Worker,** aki az eltolást végeztepontot kap érte.

### HoleField

Egy lyukat jelenít meg. Amennyiben egy **Box**, vagy **Worker** áll rajta, megsemmisíti a rajta álló objektumot. Tartozhat hozzá kapcsoló. Amikor a hozzá tartozó kapcsoló nem aktív akkor nem semmisíti meg a rajta álló említett objektumokat.

### SimpleField

A pálya egyes mezőit képviselik a játékban. Lekérdezhető tőle a megfelelő irányban lévő szomszédos **Field**. Egy cellán egyidőben állhat egy **Box,** vagy egy **Player**, valamint tartalmazhat referenciát egy speciális elemre, amely lehet **Hole** vagy **Switch**.

### SwitchField

Egy a pályán található kapcsolót valósít meg. Minden kapcsolóhoz tartozik egy referencia egy **Hole** osztályra. Amennyiben egy **Box** áll a **Switchen**, akkor aktiválódik, más esetben nem. A hozzá tartozó **Hole**-t szabályozza, az állásával, melyet képes megjegyezni.

### WallField

Egy oszlopot, vagy falszeletet ábrázol a pályán. Megakadályozza, hogy dobozt toljanak rá, vagy hogy rálépjen egy játékos. Ismeri a szomszédait (mezőket), irányokkal együtt.

### Worker

Egy játékost képvisel, aki pálya területén belül szabadon mozoghat. Segítségével lehetséges a ládák tologatása és tárolja, hogy hány ládát tolt már a helyére. A játékos őt irányítja.

## Statikus struktúra diagramok

## Osztályok leírása

### Box

#### Felelősség

Ennek az osztálynak az egyes példányai fogják reprezentálni a dobozokat a játékban. Őket kell majd a célmezőre eljuttatni.

#### Ősosztályok

Movable

#### Attribútumok

* **Field field:** az a mező, ahol éppen tartózkodik a láda.

#### Metódusok

* **void die():** Ez a függvény felel az entitás elpusztításáért, illetve csökkenti a szabad dobozok számát.
* **void pushBack(Direction d):** Ennek a függvénynek a segítségével a tolás irányába visszaléptethetjük a dobozt. (Erre szükség van pl. amikor a falba bele akarnánk tolni a dobozt.)
* **void pushByBox(Box b, Direction d):** Függvény arra az esetre, amennyiben a ládánkat (**Box**) egy másik doboz tolja meg, intézi, hogy a láda a megfelelő helyre kerüljön.
* **void pushByWorker(Worker w, Direction d):** A láda (**Box**) egy játékos által való megtolásákor hívjuk meg, intézi hogy a láda a megfelelő helyre kerüljön.
* **void scorePoint(Direction d):** Amennyiben a célmezőre tolódott a láda, ennek a függvénynek a segítségével végezhetjük el a megfelelő játékos pontjainak növelését, így a hívást mindig továbbadja a megfelelő irányban álló **Movable**-nek (meghívja rajta önmagát).

### BoxRecorder

#### Felelősség

Nyilvántartás vezetése a mozdíthatatlan mezőkről, amiken beragadt doboz van. Segítségével lehetséges a beragadt (pl. sarokban lévő) dobozok detektálása.

#### Attribútumok

* **ArrayList<Field> fixFields:** referenciák azokra a mezőkre, amikre nem lehet rálépni.

#### Metódusok

* **void checkRecordWith(Field f):** Ezzel a függvénnyel lehetséges annak az ellenőrzése, hogy a paraméterben kapott mező benne-e van már a nyilvántartásban. Ha nincs akkor csökkenti a szabad dobozok számát a **Game**-ben.
* **void endFieldOccupied(Field f):** Ennek a függvénynek a segítségével jelezhető, ha egy célmezőre (**EndField**) beérkezett egy láda (**Box**). A paraméter átadásával jelezzük, hogy melyik mezőről van szó, majd ez a mező bekerül a nyilvántartásba, hiszen onnan a doboz nem mozdítható el.
* **void update(Field f):** Ezt a függvényt mindig egy doboz (**Box**) mozgatása után hívjuk meg, miután elhelyeződött a mezőn (*Field.arrived()*). A paraméterben átadott mező az a mező, ahová a doboz újonnan került. A nyilvántartás alapján pedig el tudja dönteni, hogy a dobozunk beszorult-e (a szomszédai alapján). Amennyiben igen, akkor azt is felveszi a nyilvántartásba.
* **void init(ArrayList<Field> walls):** A **Map**-től megkapja a falakat, erre azért van szükség mert kezdetben ezeket tárolja el a fixFields-be.

### EndField

#### Felelősség

Célmező, amire a dobozokat (**Box**)juttatni kell. Amennyiben rálép egy doboz, akkor az mozdíthatatlanná válik ott.

#### Ősosztályok

Field

#### Attribútumok

-

#### Metódusok

* **Movable boxEnters(Box b, Direction d):** Fogadja a mezőre érkező **Box**-t, beállítja annak a mező referenciáját, aztán pedig kitörli a régi mezőjén lévő referenciáját (ahonnan jött, null-ra állítja). Végezetül pedig mozdíthatatlanná teszi a mezőre érkezett **Box-**t. Továbbá elindítja a pontszerzési folyamatot a megfelelő munkás számára.
* **Movable workerEnters(Worker w, Direction d):** Fogadja a mezőre érkező **Worker**-t, beállítja annak a mező referenciáját, aztán pedig kitörli a régi mezőjén lévő referenciáját (ahonnan jött, null-ra állítja).
* **void workerArrived(Worker w):** Elvégzi a kapott objektum mezőre való elhelyezését.
* **void boxArrived(Box x):** Elvégzi a kapott objektum mezőre való elhelyezését.

### Field

#### Felelősség

Absztrakt ősosztály, amiből a mező objektumok származnak. Tárolja a szomszédait, az irányokkal együtt. Továbbá tárolja a rajta lévő **Movable** objektumot.

#### Attribútumok

* **Field neighbor[Direction]:** tárolja a szomszédos mezőket a lehetséges irányokban.
* **Movable movable:** referencia egy **Movable** leszármazottra.

#### Metódusok

* **Field getNeighbor(Direction d):** Visszaadja az átadott irányban lévő szomszédos mezőt.
* **Movable boxEnters(Box b, Direction d):** Absztrakt függvény, ami a leszármazottakban kerül implementálásra.
* **Movable workerEnters(Worker w, Direction d):** Absztrakt függvény, ami a leszármazottakban kerül implementálásra.
* **Movable getMovable():** Visszaadja a mezőn lévő **Movable** leszármazott referenciáját.
* **void remove():** A mezőhöz tartozó (tárolt) **Movable** referenciát eltávolítja.
* **void workerArrived(Worker w):** Elvégzi a kapott objektum mezőre való elhelyezését.
* **void boxArrived(Box x):** Elvégzi a kapott objektum mezőre való elhelyezését.
* **void updateRecorder():** A **BoxRecorder** frissítéséért felelős függvény.

### Game

#### Felelősség

A játék alapvető mechanizmusaiért felel és tárolja a nyeréshez, vagy vesztéshez szükséges adatokat. Továbbá tárolja a még életben lévő **Worker**-öket, és a **Map**-et.

#### Attribútumok

* **int freeBoxCounter:** a még mozgatható dobozok számát tárolja.
* **int workerCounter:** a még játékban lévő munkások számát tárolja.
* **Map map:** referencia a pályát tartalmazó osztályra.
* **ArrayList<Worker> workers:** a pályán lévő, még élő munkások.

#### Metódusok

* **void chooseMap(int n):** Meghívja az i-edik pálya betöltését.
* **void decreaseBoxes(Box b):** Abban az esetben hívódik meg, amennyiben egy doboz megsemmisül. Ha ez a doboz nem volt beragadva, akkor csökkenti a még mozgatható dobozok számát egyel.
* **void decreaseFreeBoxes():** Csökkenti a még mozgatható dobozok számát 1-el, amennyiben egy doboz beragad.
* **void endGame():** Akkor hívódik meg, ha a mozgatható dobozok száma nullára csökken. Ekkor a legtöbb pontot elért játékos nyer (maximumkeresés a még élő játékosok pont attribútumán).
* **void init(ArrayList<Worker> workers):** Eltárolja a munkásokat amelyet a **Map**-től kap.
* **void gameOver():** Akkor hívódik meg, ha minden játékos meghal. Ebben az esetben senki sem kerül ki nyertesként a játékból.
* **void workerDied(Worker w):** Amennyiben meghal a **Worker,** csökkenti a játékban lévő munkások számát, és kiveszi az adott munkás referenciáját.

### HoleField

#### Felelősség

Megsemmisíti a rajta álló **Movable** osztályt. Tartozhat hozzá kapcsoló, ennek állapotát figyelembe veszi.

#### Ősosztályok

Field

#### Attribútumok

* **SwitchField switch:** referencia a hozzá tartozó kapcsolóhoz.

#### Metódusok

* **Movable boxEnters(Box b, Direction d):** Fogadja a mezőre érkező **Box**-t, beállítja annak a mező referenciáját, aztán pedig kitörli a régi mezőjén lévő referenciáját (ahonnan jött). Majd pedig végül megnézi, hogy van e rajta **Movable**,és visszaadja annak a referenciáját (ha nincs akkor null).
* **Movable workerEnters(Worker w, Direction d):** Fogadja a mezőre érkező **Worker**-t, beállítja annak a mező referenciáját, aztán pedig kitörli a régi mezőjén lévő referenciáját (ahonnan jött). Majd pedig végül megnézi, hogy van e rajta **Movable**,és visszaadja annak a referenciáját (ha nincs akkor null).
* **void changeToActive():** A kapcsoló megváltoztatása esetén hívjuk meg, hogy ha áll valami abban a pillanatban a lyukon, akkor az megsemmisül.
* **void execute(Movable m):** Függvény arra, ha meg akarjuk semmisíteni a mezőn álló **Movable** objektumot. Meghívja a kellő metódusokat az objektum megsemmisítéséhez.
* **void interactWithBox(Box b):** Egy a mezőn lévő lyukkal való interakció (rálépés) esetén hívjuk meg. Ellenőrzi, hogy tartozik-e hozzá kapcsoló, továbbá meghívja a reagáláshoz tartozó további metódusokat.
* **void interactWithWorker(Worker w):** A működése megegyezik az előző függvényével.
* **erease():** Objektum lekerülésekor elvégzendő műveletek helye.

### Map

#### Felelősség

Szerepe a pálya betöltése. Ismeri az összes mezőt.

#### Attribútumok

* **ArrayList<Field>fields:** referenciák az egyes mezőkre.

#### Metódusok

* **void loadMap(int n):** A pálya betöltését intézi.

### Movable

#### Felelősség

Egy absztrakt ősosztály, amiből a mozgatható objektumok származnak. Tárolja azt, hogy melyik mezőn helyezkedik el.

#### Attribútumok

* **Field field:** a mező, amin tartózkodik.

#### Metódusok

* **void die():** Absztrakt függvény, leszármazottakban kerül megvalósításra.
* **void finalizeStep():** Ez az a függvény, ami az egyes lépések véglegesítésért felel.
* **void place(Field f):** Ez a függvény átállítja a **Movable** mező referenciáját a kapott mezőre.
* **void pushBack(Direction d):** Absztrakt függvény, leszármazottakban kerül megvalósításra.
* **void pushByBox(Box b, Direction d):** Absztrakt függvény, leszármazottakban kerül megvalósításra.
* **void pushByWorker(Worker w, Direction d):** Absztrakt függvény, leszármazottakban kerül megvalósításra.
* **void scorePoint(Direction d):** Absztrakt függvény, leszármazottakban kerül megvalósításra.

### SimpleField

#### Felelősség

Közönséges mező, amire szabadon rá lehet lépni. Ezeken közlekednek a munkások és a ládák.

#### Ősosztályok

Field

#### Metódusok

* **Movable boxEnters(Box b, Direction d):** Fogadja a mezőre érkező **Box**-t, beállítja annak a mező referenciáját, aztán pedig kitörli a régi mezőjén lévő referenciáját (ahonnan jött). Majd pedig végül megnézi, hogy van e rajta **Movable**,és visszaadja annak a referenciáját (ha nincs akkor null).
* **Movable workerEnters(Worker w, Direction d):** Fogadja a mezőre érkező **Worker**-t, beállítja annak a mező referenciáját, aztán pedig kitörli a régi mezőjén lévő referenciáját (ahonnan jött). Majd pedig végül megnézi, hogy van e rajta **Movable**,és visszaadja annak a referenciáját (ha nincs akkor null).
* **void boxArrived(Box b):** A *finalizeStep(),* azaz a lépés véglegesítésnél van szerepe ennek a függvénynek. Átadja az adott mezőre lépő **Box** referenciát. Továbbá frissíti a **BoxRecorder**-t.
* **void workerArrived(Worker w)**: Működése az előző függvénnyel megegyező, azonban nem frissíti a **BoxRecorder**-t.

### SwitchField

#### Felelősség

Egy kapcsolót és hozzá tartozó állapotot reprezentáló mező objektum. Ismeri a hozzá tartozó **HoleField**-t.

#### Ősosztályok

Field

#### Attribútumok

* **HoleField hole:** referencia a kapcsolóhoz tartozó lyukhoz.
* **boolean active:** változó, hogy a kapcsoló jelen pillanatban aktiválva van-e.

#### Metódusok

* **Movable boxEnters(Box b, Direction d):** Fogadja a mezőre érkező **Box**-t, beállítja annak a mező referenciáját, aztán pedig kitörli a régi mezőjén lévő referenciáját (ahonnan jött). Majd pedig végül megnézi, hogy van e rajta **Movable**,és visszaadja annak a referenciáját (ha nincs akkor null).
* **Movable workerEnters(Worker w, Direction d):** Fogadja a mezőre érkező **Worker**-t, beállítja annak a mező referenciáját, aztán pedig kitörli a régi mezőjén lévő referenciáját (ahonnan jött). Majd pedig végül megnézi, hogy van e rajta **Movable**,és visszaadja annak a referenciáját (ha nincs akkor null).
* **void activate():** Aktiválja a kapcsolót.
* **void holeInteracted(Movable m):** Akkor hívódik meg, ha a kapcsolóhoz tartozó mezőre valaki rálép. Eldönti, hogy a kapcsoló éppen aktív-e és ennek függvényében meghívja a szükséges metódusokat.
* **void interactWithBox(Box b):** Lekezeli, ha doboz érkezik a mezőre. Aktiválja a kapcsolót és szól a lyuknak, hogy megváltozott az állapota.
* **void interactWithWorker(Worker w):** Lekezeli, ha munkás érkezik a mezőre (nem aktiválódik).
* **void remove():** Az ős függvényének bővítése, azzal, hogy **false** értékre állítja az **active** változót.

### WallField

#### Felelősség

Olyan mező objektum, mely egy falat reprezentál. Nem lehet rajta átmenni, és ismeri a szomszédait.

#### Ősosztályok

Field

#### Metódusok

* **Movable boxEnters(Box b, Direction d):** Megakadályozza, hogy egy **Box** rálépjen a mezőre. Ezt azzal éri el, hogy amikor rálép, akkor a paraméterben kapott objektum referenciáját adja vissza.
* **Movable workerEnters(Worker w, Direction d):** Megakadályozza, hogy egy **Worker** rálépjen a mezőre. Ezt azzal éri el, hogy amikor rálép, akkor a paraméterben kapott objektum referenciáját adja vissza.

### Worker

#### Felelősség

A játékosok által irányított munkást reprezentáló osztály. Ő tudja a ládákat (**Box**) eltolni, illetve a pontokat is ő kapja majd. Ismeri a mezőt ahol áll és tárolja a pontjait.

#### Ősosztályok

Movable

#### Attribútumok

* **Field field:** referencia a mezőre, amin a munkás épp tartózkodik.
* **int points:** a játékos pontjai, amiket az egyes ládák célmezőre juttatásáért kap.

#### Metódusok

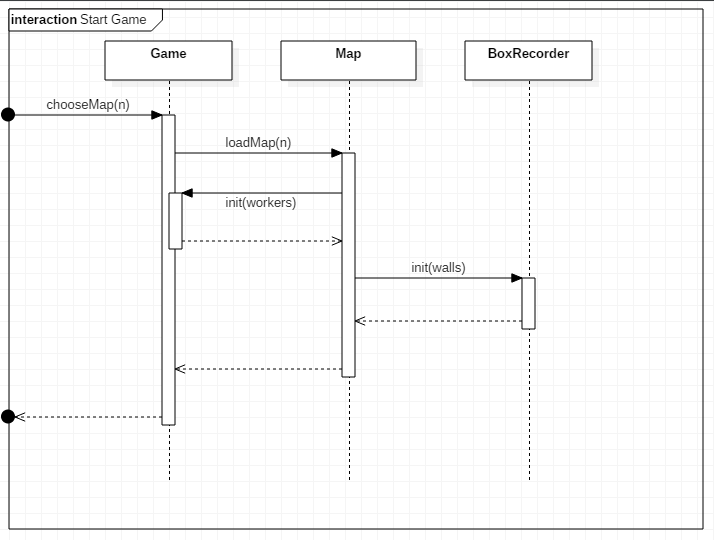
* **void control(Direction d):** A munkás (**Worker**) irányításáért felelős függvény. A paraméterben kapott irányban fogja megpróbálni elmozdítani a munkást a szomszédos mezők figyelembe vételével (Mindez önakaratból történik, tehát a felhasználó beavatkozására).
* **void die():** Ez a függvény felel az entitás elpusztításáért, illetve csökkenti a munkások számát.
* **void finalizeStep():** Ez az a függvény, ami az egyes lépések véglegesítésért felel.
* **void goBack(Direction d):** Szintén a munkás - munkás ütközés / eltolás megfelelő lekezelésében van szerepe, mégpedig elindít egy visszafele való lökést, hiszen nem szabad hagyni, hogy az egyik munkás a másikat közvetlenül eltolja, tehát úgymond visszarendezi a dolgokat.
* **void increasePoints():** A munkás pontjainak növelése.
* **void pushBack(Direction d):** Ennek a függvénynek ott van szerepe, amikor egy munkás a dobozt a falba bele akarná tolni. Ekkor ezt nem engedi majd, hanem az egész sor (pl.: munkás-doboz-fal) egyet visszafele fog lépni. Ekkor ha a munkás visszalépendő mezőjén tartózkodik valami, akkor a munkás meghal.  
  Ennek a megvalósításáért részben felelős ez a metódus is.
* **void pushByBox(Box b, Direction d):** Ez a függvény arra az esetre szolgál, hogyha a munkást egy dobozzal (**Box**) próbálnának meg eltolni.
* **void pushByWorker(Worker w, Direction d):** Ez a függvény arra az esetre szolgál, ha a munkást egy másik munkás próbálná meg eltolni.
* **void scorePoint(Direction d):** A pontok növeléséért felelős, méghozzá az *increasePoints()* függvény segítségével.

## Szekvencia diagramok

Megjegyzés: Ahol Movable szerepel a diagram nevében, ott azonos a működés Box és Worker esetén, csak nem akartuk megismételni a diagramot ugyanúgy.

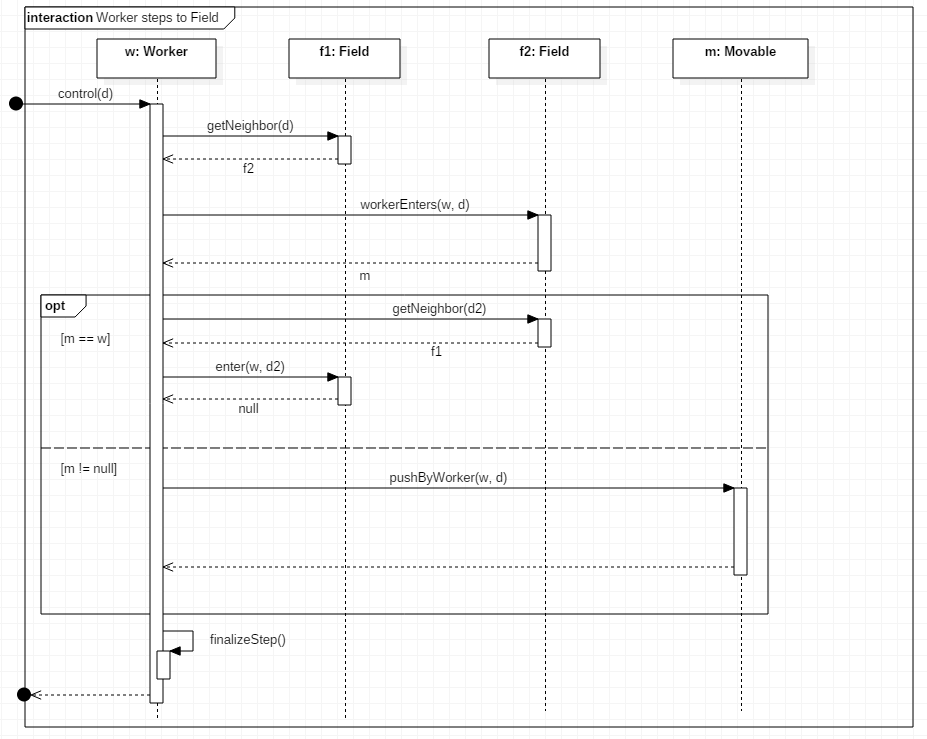
Megjegyzés: Singleton osztály esetén csak az osztálynév áll a lifeline-ban, mert a modellező program a kívánt jelölést nem engedte.

### Start Game



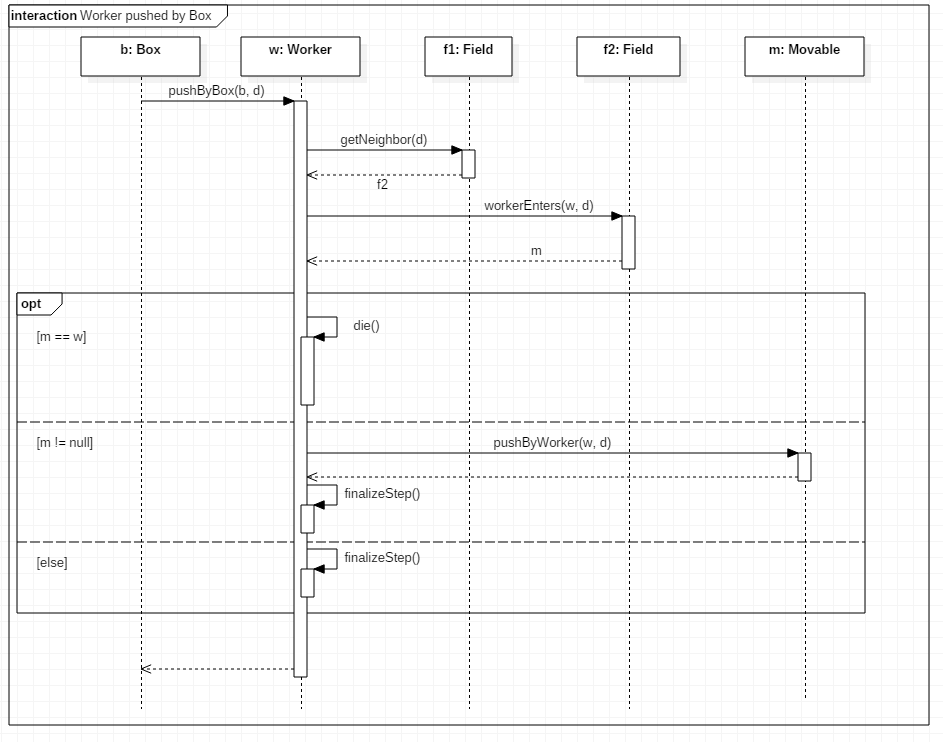
A játék betöltése és inicializálása a térkép alapján. A térkép adatai tartalmazzák a mezőket, a rajta található munkásokkal együtt.

### Worker steps to Field



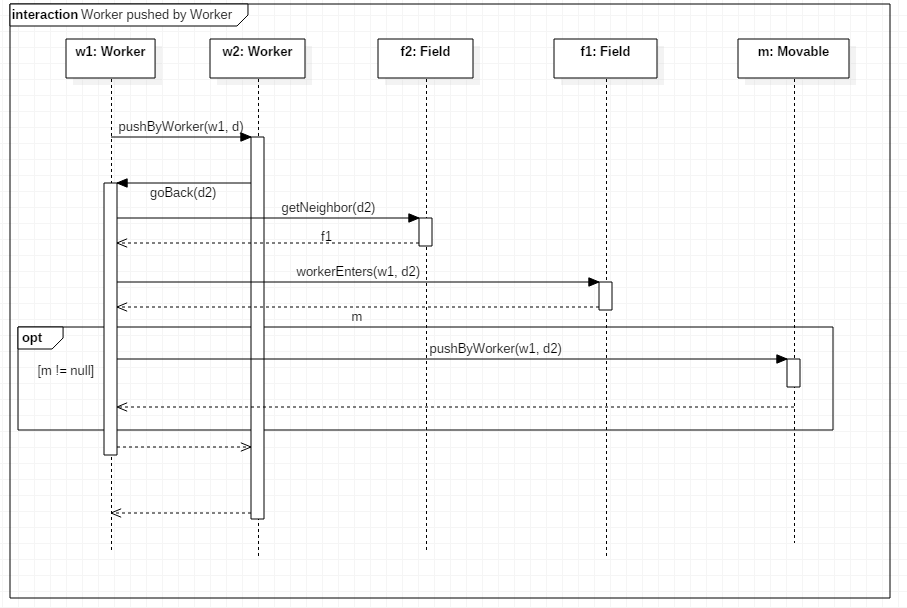
A munkás lépési folyamata, amely a felhasználótól érkező iránnyal indul. Az *enter* függvény visszatérési értéke alapján történik a folyamat további része. Legvégül a lépés véglegesítése jön.

### Worker pushed by Box



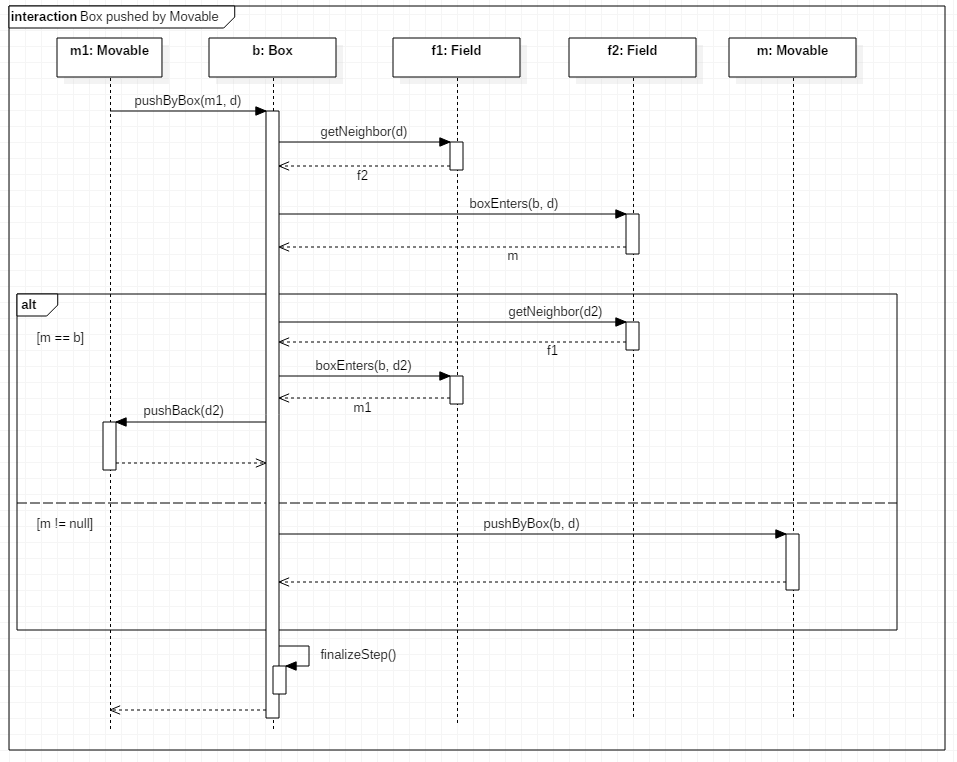
Egy doboz – munkás tolás lezajlása, szintén a szomszédos mezőn való elhelyezést kezdő *enter* függvény visszatérési értéke a fontos. Ez alapján dől el ha a munkás meghal (a szomszéd fal), tol (van valami a szomszédos mezőn), vagy csak odalép.

### Worker pushed by Worker



A munkás – munkás tolás lekezelése, a megtolt munkás „visszaküldi” a másikat az ellenkező irányban. Ekkor, ha már a régi helyén lett valami, akkor azt eltolja visszafele.

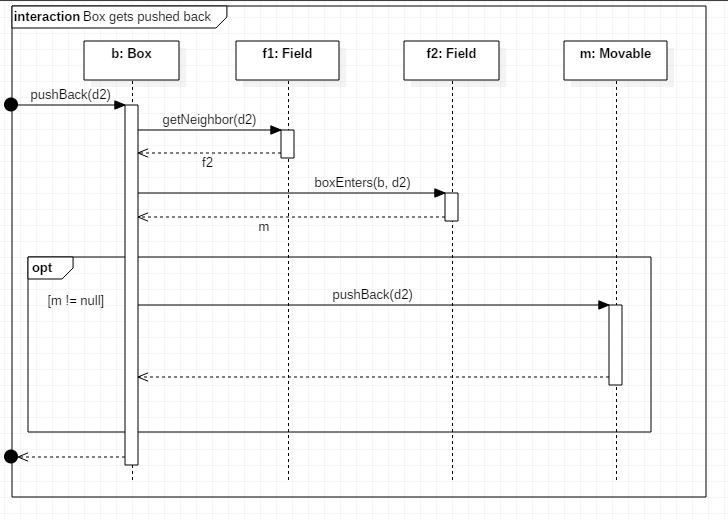
### Box pushed by Movable



Megjegyzés: Mivel az *m1* absztrakt osztály példányaként szerepel, így nem tudhatjuk, hogy doboz vagy munkás. Itt dobozként szerepel hiszen a *pushByBox* függvényhívás történik. Ez a folyamat munkás esetén is ugyanilyen (*pushByWorker*), azonban nem ábrázoltuk újra.

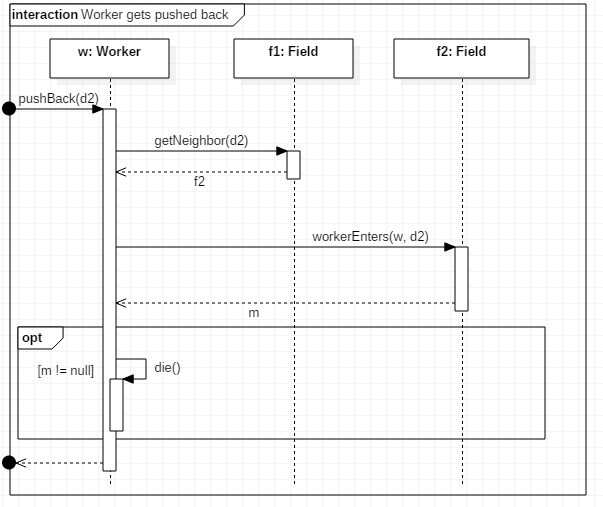
Doboz tolása, hasonló a folyamat, mint a már említett munkás – doboz esetben. Ha falba ütközik a doboz akkor elindul egy visszafele tolás az ellenkező irányba. Ha van mellette a szomszédon valami/valaki akkor azt továbbtolja. Fontos a lépés véglegesítése.

### Box gets pushed back



Az előbb említett falba ütközés esetén fellépő visszatolás. Ha van a szomszédos mezőn *Movable*, akkor azt is visszatolja.

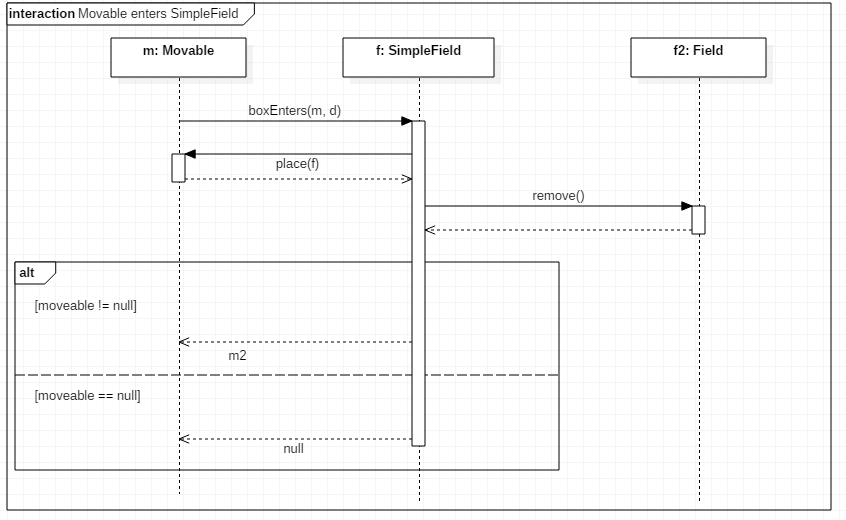
### Worker gets pushed back



Ha embert tolják visszafele és beleütközik valamibe, nem tolja tovább hanem meghal, hiszen az összenyomta.

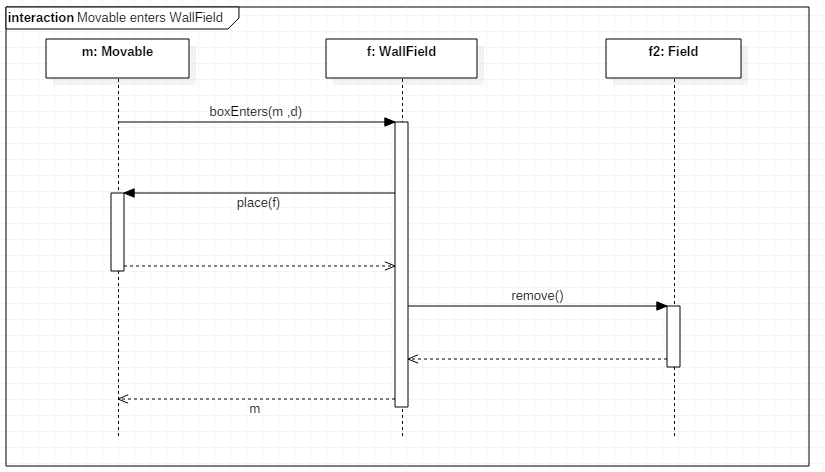
### Movable enters SimpleField

Megjegyzés: Vagy m2-vel, vagy null-al tér vissza. Máshol is megjelenik a jelölés így.



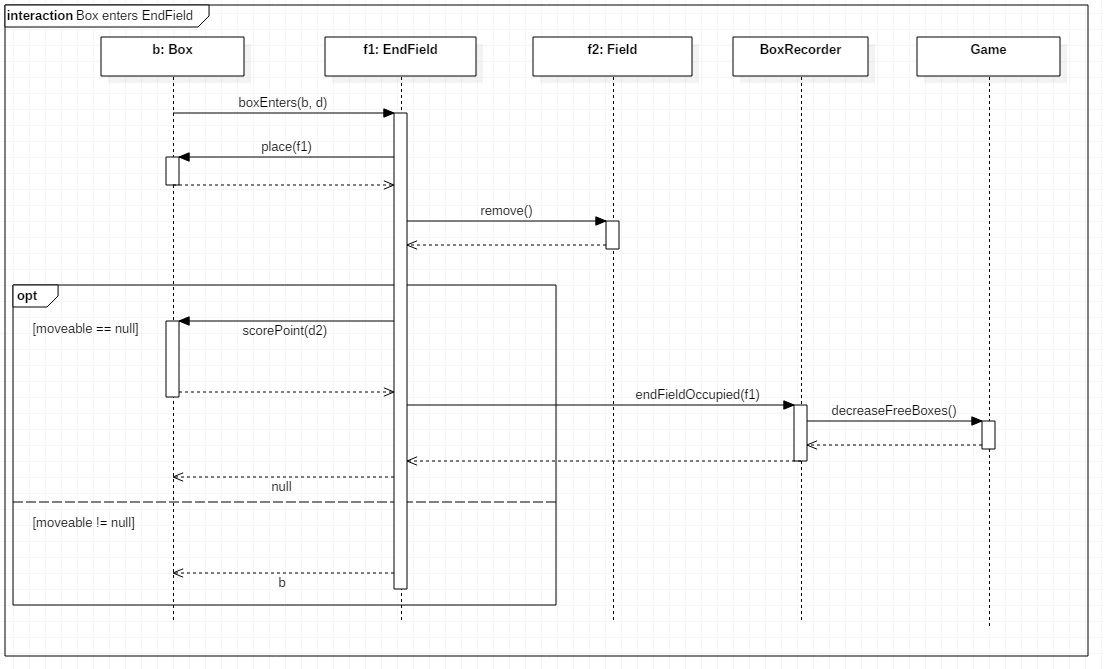
Elhelyezi a *Movable*-t a mezőn, úgy, hogy átírja a *Movable*-ben található mezőre való hivatkozást magára (de ő még nem tárolja el) és eltávolítja a szomszédból a hivatkozását. Ha van rajta valaki akkor azt visszadja.

### Movable enters WallField



Fal esetén ez a procedúra hasonló, de itt a hívót adja vissza, így tudva később majd a hívó hogy falnál van.

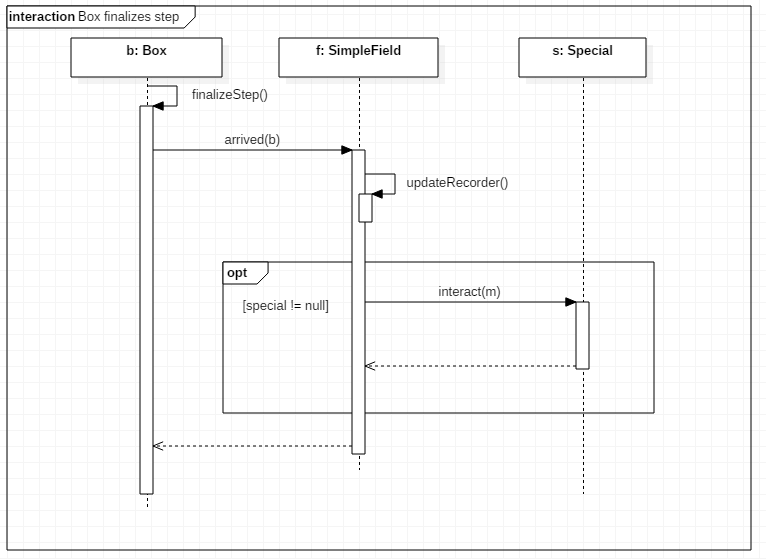
### Box enters EndField



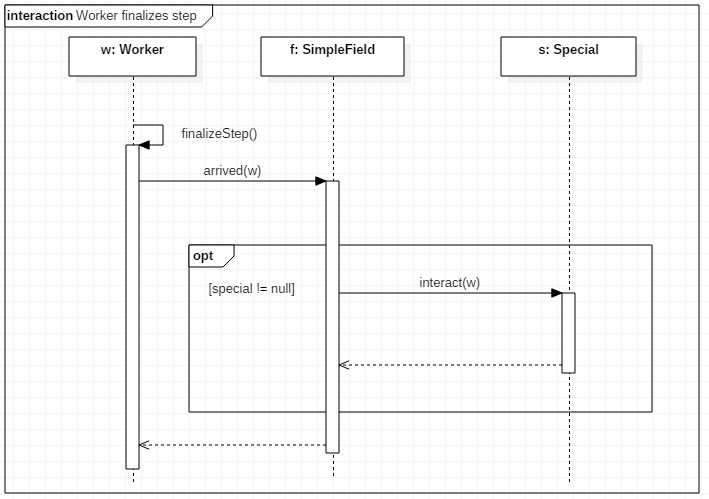
Doboz célmezőre érkezéskor annyi plusz teendőnk van, hogy a fix dobozok számát módosítani kell, hiszen az a doboz fix lesz, ezután úgy viselkedik mintha fal lenne. Továbbá a játékos pontozását is elindítja.

Megjegyzés: SwitchField, HoleField „mindkét” *enter-* függvényeés az EndField munkás érkezését kezelő függvénye is úgy viselkedik mint a SimpleField esetében bemutatott folyamat.

### Box finalizes step (depr)

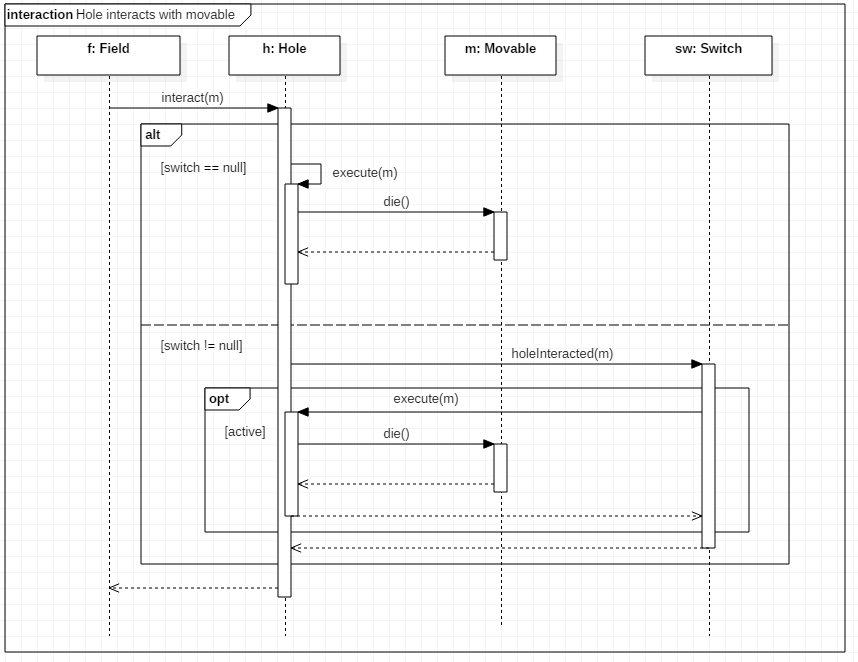


### Worker finalizes step (depr)

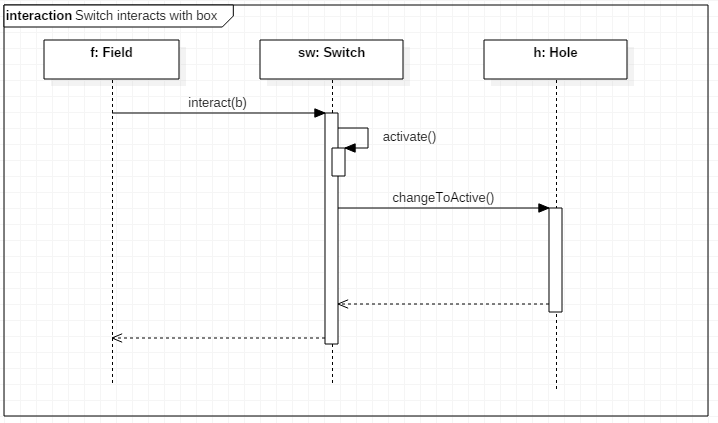


### Hole interacts with movable (depr)

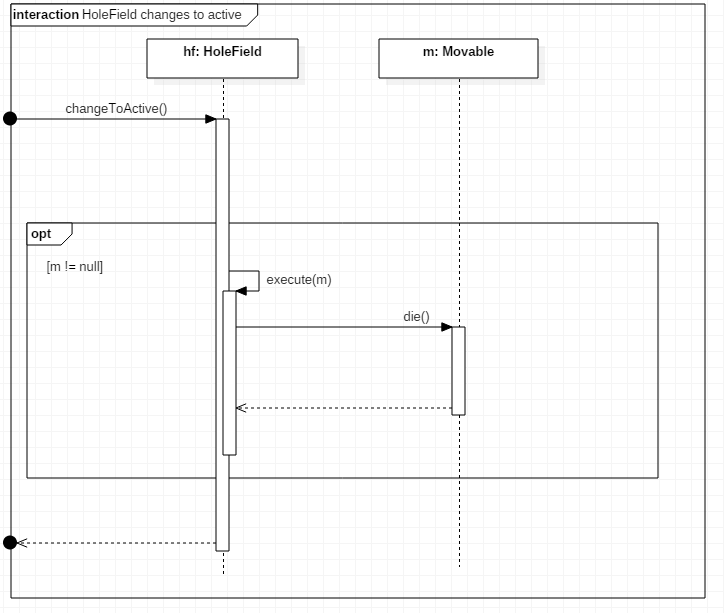
Megjegyzés: Mivel az *active* a Switch osztály boolean típusú változója, így az *opt* feltételébe csak röviden került bele (és nem *active == true -*ként*)*



### Switch interacts with box (depr)

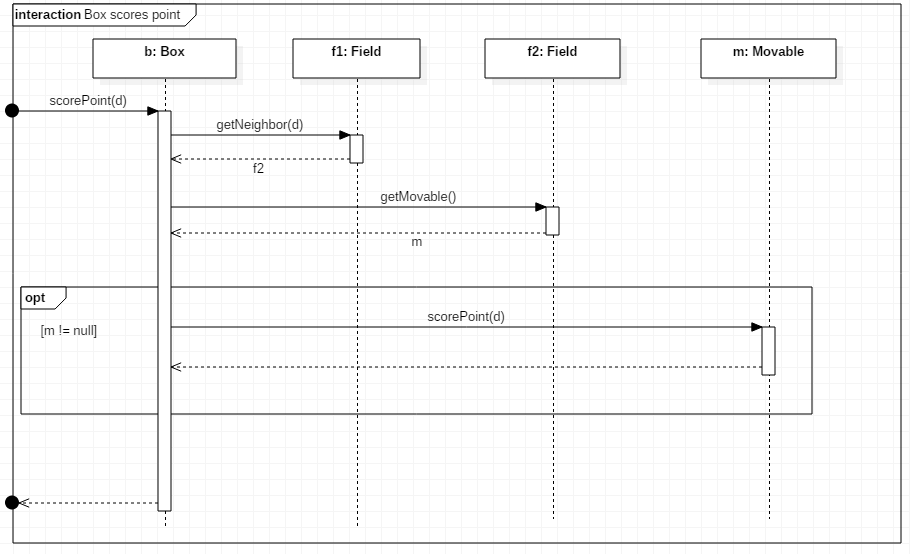


### HoleField chnages to active



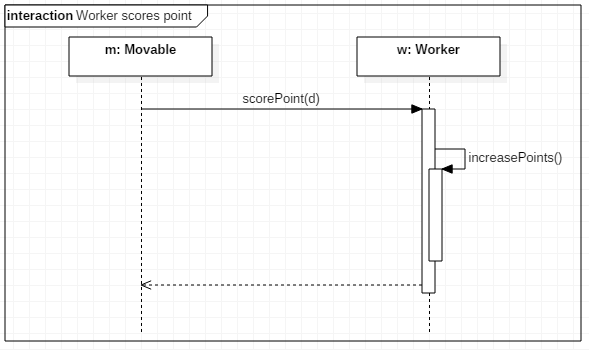
A lyukhoz tartozó kapcsoló aktívvá válása esetén, a lyuk a rajta álló entitást elpusztítja (ha van ilyen).

### Box „scores point”



A pontozási folyamatot vezértlő hívást továbbadja a megfelelő irányban lévő *Movable*-nek

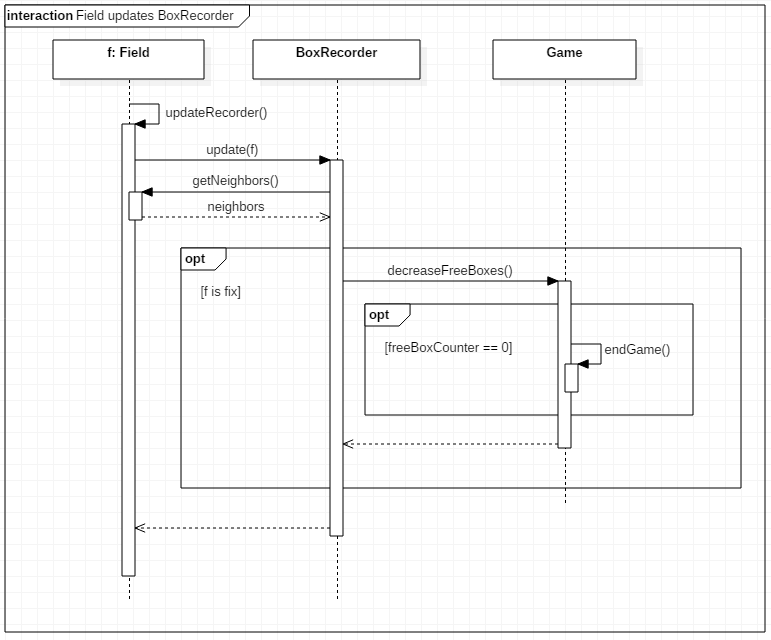
### Worker scores point



Pontot ér el a munkás, ha a pontozási folyamat elér hozzá.

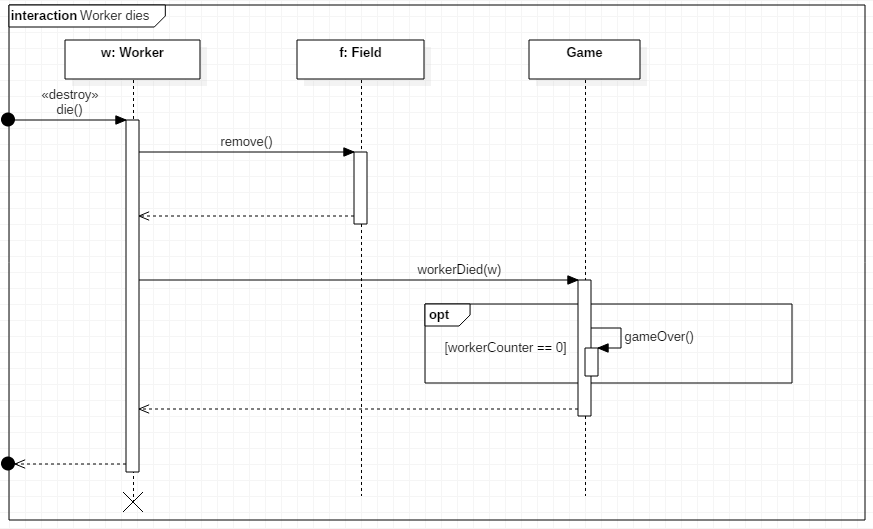
### Field updates BoxRecorder

Megjegyzés: Az „*f is fix*” feltétel annyit jelent, hogy az f a szomszédai alapján fix-nek tekinthető e.



A *BoxRecorder*-nek odaadja magát egy mező, amit az a szomszédai alapján lehet, hogy beletesz a nyilvántartásba, ekkor a szabad dobozok száma csökken, és ha nincs ilyen akkor a játék végetér.

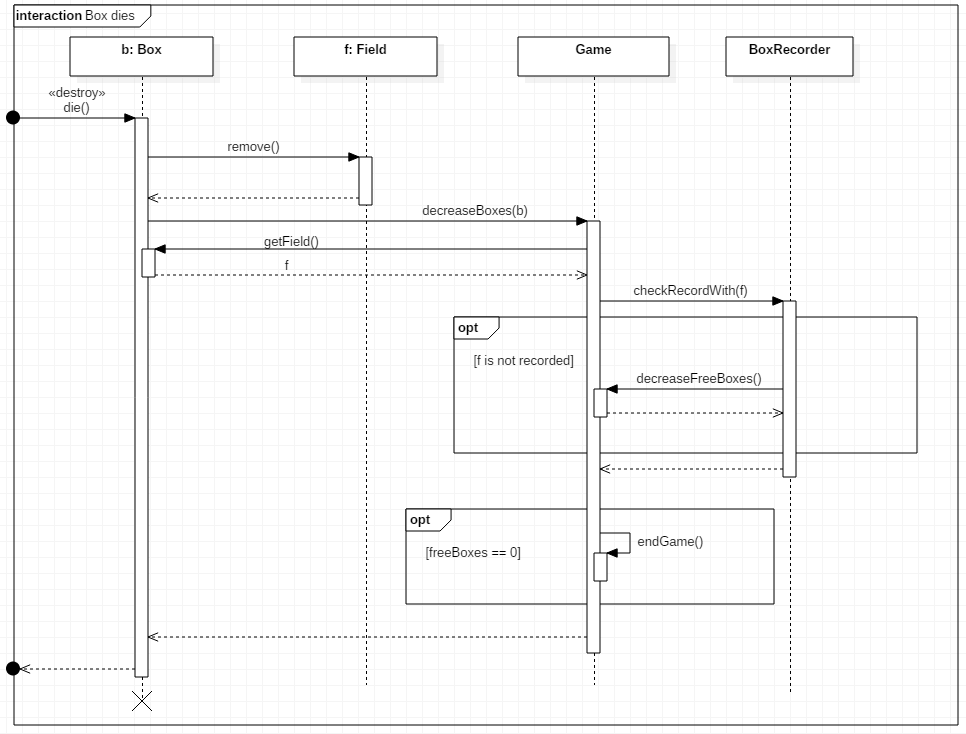
### Worker dies



A munkás a halálakor eltávolítja a referenciáját a mezőről, majd értesíti a játékot, hogy meghal, ha mindenki meghalt, akkor vége a játéknak.

### Box dies

Megjegyzés: Az „*f is not recorded*” feltétel annyit jelent, hogy az f nincs benne a *BoxRecorder* nyilvántartásában, tehát nem fix.



Doboz halálakor fontos teendő, hogy ha szabad doboz volt, vagyis nem volt beragadva, akkor a *BoxRecorder*-t és a *Game*-et értesíteni kell a változásról. Ha nincs több szabad doboz akkor a játék véget ér.

## State-chartok

A jelen modellben nincs szükség állapotgépre.

## Napló

|  |  |  |  |
| --- | --- | --- | --- |
| **Kezdet** | **Időtartam** | **Résztvevők** | **Leírás** |
| 2018.02.20. 18:00 | 2 óra | Horváth  Tolnai  Bertalan  Mátyás  Gurubi | **Értekezlet**.  **Döntés**: Alapvető működés első átgondolása. Mátyás elkészíti 02.21-re, a konzultációra a próba szekvencia diagramot. |
| 2018.02.21. 09:00 | 2 óra | Mátyás | **Tevékenység**: Mátyás elkészíti a próba szekvencia diagramot. |
| 2018.02.21 19:00 | 2 óra | Horváth  Tolnai  Bertalan  Mátyás  Gurubi | **Értekezlet**.  **Döntés**: Közös csapatmegbeszélés, részek kiosztása. Határidő: 02.24. 18:00. Tolnai, Bertalan: 3.1 – 3.3. Horváth, Mátyás, Gurubi: 3.4. Mátyás: 3.6 |
| 2018.02.22. 19:00 | 2 óra | Bertalan | **Tevékenység**: 3.1 Kidolgozása 3.2 elkezdése. |
| 2018.02.22. 20:00 | 1 óra | Tolnai | **Tevékenység**: 3.1 ellenőrzése és 3.2 kidolgozásának átgondolása. |
| 2018.02.23. 13:00 | 2 óra | Horváth | **Tevékenység**: Szekvencia diagramok alap tervezése |
| 2018:02:23. 15:00 | 2 óra | Gurubi | **Tevékenység**: Szekvencia diagramok rajzolása: 3.4.1 – 3.4.5 |
| 2018.02.23. 16:00 | 2 óra | Mátyás | **Tevékenység**: Szekvencia diagramok rajzolása: 3.4.8-3.4.12 |

|  |  |  |  |
| --- | --- | --- | --- |
| 2018. 02. 23. 18:00 | 2 óra | Tolnai | **Tevékenység**: Tolnai kidolgozza a 3.3.1 – 3.3.4 |
| 2018.02.23. 21:00 | 1 óra | Bertalan | **Tevékenység**: Bertalan kidolgozza a 3.3.5 – 3.3.7 |
| 2018. 02. 24. 13:00 | 2 óra | Bertalan | **Tevékenység**: Bertalan kidolgozza a 3.3.8 – 3.3.11 |
| 2018. 02. 24. 14:00 | 2 óra | Tolnai | **Tevékenység**: Tolnai kidolgozza a 3.3.12 – 13. Továbbá javít az eddig elkészült Osztály leírásokban. (3.3) |
| 2018. 02. 24. 16:00 | 2 óra | Hotváth | **Tevékenység**: Szekvencia diagramok rajzolása: 3.4.6 – 3.4.7 és javítás az eddigi elkészült szekvencia diagramokon. |
| 2018.02.24. 18:00 | 2 óra | Gurubi | **Tevékenység**: Szekvencia diagramok rajzolása: 3.4.13 – 3.4.16 |
| 2018.02.24. 19:00 | 1 óra | Mátyás | **Tevékenység**: Szekvencia diagramok rajzolása: 3.4.17 – 3.4.18 |
| 2018.02.25. 09:00 | 2 óra | Tolnai | **Tevékenység**: Osztály diagram (3.2) kiegészítése, befejezése (első) |
| 2018.02.25. 10:00 | 2 óra | Bertalan | **Tevékenység**: Osztály leírások (3.3) javítása, összehangolása az elkészült diagramokkal. |
| 2018.02.25. 13:00 | 2 óra | Horváth | **Tevékenység**: Szekvencia diagramok rajzolása: 3.4.19 – 3.4.21 |
| 2018.02.25 15:00 | 2 óra | Gurubi | **Tevékenység**: Javítás a szekvencia diagramokon. Főleg a végén lévőkőn. |
| 2018.02.25. 16:00 | 2 óra | Tolnai | **Tevékenység**: Osztály diagram (3.2) kiegészítése, javítása |
| 2018.02.25. 17:00 | 2 óra | Bertalan | **Tevékenység**: Osztály leírások (3.3) javítása, összehangolása a diagramokkal. |
| 2018.02.25. 18:00 | 2 óra | Horváth | **Tevékenység**: Horváth hibákat, ellentmondásokat keres és javít az egész dokumentumban és a hozzá tartozó egyéb anyagokban, továbbá egyeztet a többiekkel és megcsinálja a javasolt dolgokat. |
| 2018.02.25. 20:00 | 1 óra | Horváth | **Tevékenység**: Horváth beleszerkeszti a szekvencia diagramokat a dokumentumba. |
| 2018.02.25. 18:00 | 2 óra | Gurubi | **Tevékenyég**: Gurubi véglegesíti a szekvencia diagramokat. |
| 2018.02.25. 20:00 | 1 óra | Gurubi | **Tevékenység**: Gurubi átnézi a 3.1, 3.2, 3.3 részeket a dokumentumban |
| 2018.02.25. 21:00 | 2 óra | Mátyás | **Tevékenység**: Dokumentum végleges átnézése és ellenőrzése. |
| 2018.02.25. 23:00 | 1 óra | Mátyás | **Tevékenység**: Dokumentum szerkesztése |
| 2018.02.26. 09:00 | 1 óra | Mátyás | **Tevékenység**: Last minute hibajavítás. Dokumentum nyomtatása. |