**04. Analízis modell kidolgozása 2.**

12 - ch33kybr33ky\_labz

Konzulens:

Bodó Zsófia

Csapattagok

|  |  |  |
| --- | --- | --- |
| **Frontó András** | **G392CQ** | andrasfronto@gmail.com |
| **Czetter Gábor** | **CP6OG3** | cz3tt1@gmail.com |
| **Tomori Dániel** | **TDWSF5** | tomori.daniel@gmail.com |
| **Verasztó Bálint** | **VSDRBZ** | balintveraszto@gmail.com |

1. **Analízis modell kidolgozása**
   1. ***Objektum katalógus***
      1. **Player**

A Player-eket a felhasználók irányítják a pálya falakkal határolt mezőin. Képesek Box-okat tolni, de nem tudnak átmenni a Wall-okon. A Button-ok nem aktiválódnak, ha rájuk lépnek viszont a TrapDoor-ok és a Hole-ok képesek megölni őket. Továbbá lépés hiányába, ha rájuk tolódik egy doboz akkor meghalnak.

* + 1. **Box**

A Box-okat a Playerek tologatják a Map-en. Nem lehet Wall-ba tolni őket, de cserébe képesek aktiválni a Button-al ellátott Field-eket. Ha Hole-al vagy TrapDoor-al rendelkező Field-re tolódnak akkor meghalnak. Ha a Target mezőbe csúszik be egy Box, az pontot ad annak a Player-nek, aki utoljára hozzáért, majd eltűnik.

* + 1. **Field**

A játék mezői, amik a Wall-okkal határolva alkotják a játszható területet. Field-eken a játek elhelyez különböző tárgyakat, akadályokat, hogy izgalmasabbá tegye a felhasználói élményt. Ezek a tárgyak vagy akadályok, mind máshogy reagálnak arra, hogy Player, vagy Box került-e rájuk.

* + 1. **Wall**

Határolja a játszható terület. Se Player se Box nem képes átmenni rajta.

* + 1. **Hole**

Ez egy olyan mező, melyen egy hatalmas lyuk tátong. Ha Player vagy Box kerül rá eltűnik és törlődik a játékból. Ha Player esik bele, az addig megszerzett pontjai nullázódnak.

* + 1. **Button**

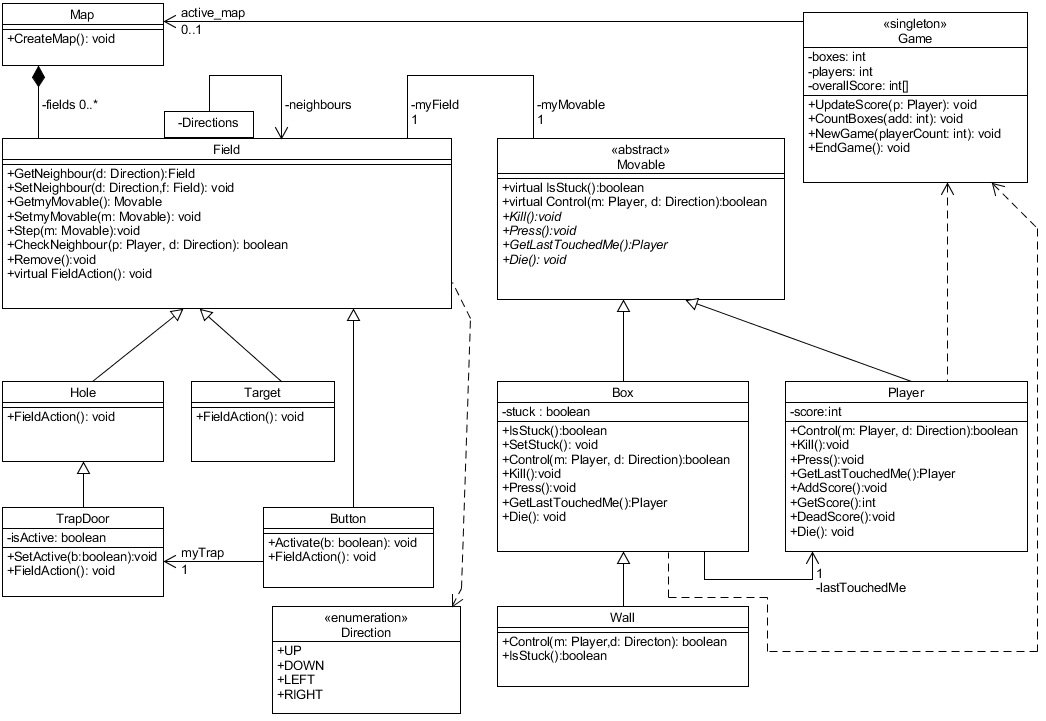
Egy mező, melyre a játék egy gombot helyezett el. Csak akkor tud benyomódni, aktiválódni, ha egy Box kerül fölé. Ekkor a hozzá tartozó TrapDoor, valahol egy másik mezőn kinyílik és ami rajta van eltűnik.

* + 1. **TrapDoor**

Ha a hozzá tartozó Button aktív, kinyílik és a rajta lévő Box, vagy Player beleesik. Ha a Button-ról eltűnik a Box, bezárul és újra egy egyszerű mező lesz.

* + 1. **Target**

A cél mező. Ha Box kerül rá eltűnik és pontot kap a végső lökést adó játékos. Ha játékos lép rá, nem történik semmi.

* 1. ***Statikus struktúra diagramok***
  2. ***Osztályok leírása***

Azért hogy leegyszerűsítsük az osztály leírásokat azokat a virtuális függvényeket amiket a leszármazott nem ír felül nem írtuk ki még egyszer (pl.: Button.Step,Button.Remove stb).

* + 1. **Box**
       - **Felelősség**

A Moveable ősosztályból származó osztály. A Player-ek Box-okat tolnak a Field-eken keresztül. Box-ok képesek gombokat benyomni, és a Target-be lökésükért pont jár a végső lökést adó Player-nek, amit a lastTouchedMe változójában tárol. A stuck változója pedig azt jelzi ha a doboz végleg mozdíthatatlanná vált.

* + - * **Ősosztályok**

Moveable → Box

* + - * **Interfészek**

Nincs

* + - * **Attribútumok**
* **Player lastTouchedMe**
* **boolean stuck**
  + - * **Metódusok**
* **Player GetLastTouchedMe:**.Visszaadja azt a player-t aki legutoljára ért hozzá.
* **boolean Controll(Player p, Direction d):** Mozdulási szándékot intéz egy Field felé, és visszaadja, hogy sikerült-e a mozdulás. Első paramétere megmondja, hogy ki tolta, a második pedig, hogy milyen irányba. Annyiban tér el az ős Controll függvényétől, hogy ha nem tud menni egy Field-re, mert azon egy Moveable helyezkedik el akkor Kill-t hív rá ami, ha Player volt akkor megöli.
* **boolean IsStuck():** Visszaadja, hogy be van-e szorulva a doboz.
* **void SetStuck():** A stuck változót állítja, ha a doboz beszorult, vagy tolhatóvá vált.
* **void Kill():** Ha Kill-t hívnak rá nem csinál semmit mert a doboz nem össze nyomható.
* **voidPress():** Ha gombra tolódik, aktiválja azt.
* **void Die():** Amikor törlik akkor fut le, el intéz mindent ami a doboz játékból való kivételéhez kell.
  + 1. **Button**
       - **Felelősség**

A Field-ből öröklődik. Felelős azért, hogy ha egy Box kerül rá, ami képes benyomni, akkor a hozzá tartozó TrapDoor kinyíljon, továbbá ha az lekerül róla, bezáródjon. Ha egy Player lép rá, nem történik semmi, a csapó ajtó csukva marad.

* + - * **Ősosztályok**

Field → Button

* + - * **Interfészek**

Nincs

* + - * **Attribútumok**
* **TrapDoor myTrap**
* **Field[] neighbours**
* **Moveable myMoveable**
  + - * **Metódusok**
* **void Activate(Boolean b)**: Box hatására aktiválja a gombot, Player hatására nem történik semmi.
* **void FieldAction():** A gomb aktiválódását segíti elő. Figyeli, hogy beszorul e a láda.
  + 1. **Field**
       - **Felelősség**

Ezekből épül fel a Map. Egy Field mező felelős a rajta lévő Moveable-ért. Ha az a külső hatás miatt el szeretne mozdulni a Field felelős az átlépésért. A szomszédos mezőket is ő tartja számon.

* + - * **Ősosztályok**

Nincs

* + - * **Interfészek**

Nincs

* + - * **Attribútumok**
* **Field[] neighbours**
* **Moveable myMoveable**
  + - * **Metódusok**
* **Field GetNeighbour(Direction d)**: A megadott irányba lévő Field-et adja vissza.
* **void SetNeighbour(Direction d, Field f):** Beállítja az adott irányba a paraméterül kapott Field-et.
* **Moveable GetmyMoveable()**: Visszaadja a rajta lévő Moveable objektumot. Ha nincs rajta, a visszatérési értéke null.
* **void SetmyMoveable(Moveable m)**: Beállít az adott Field rá egy Moveable objektumot.
* **boolean CheckNeighbour(Player p, Direction d)**: Tovább lépési szándékot jelez a megadott Player-től az adott irányba. Ha a visszatérési értéke igaz, akkor szabadon léphet tovább, egyébként marad az adott Field-en.
* **void Step(Moveable m)**: A megadott Moveable objektummal elvégzi a lépést az adott Field-re.
* **void Remove():** Az épp rajta álló Moveablet eltünteti.
* **virtual void FieldAction():** Minden Field-hez az egyedi tulajdonságot hajtja végre ami kötelezően meghívódik, ha valaki rálép vagy rátolódik. Figyeli, hogy beszorul e a láda.
  + 1. **Direction**

Egy enumeratio, ami a lehetséges irányokat tartalmazza. (UP, DOWN, LEFT, RIGHT)

* + 1. **Game**
       - **Felelősség**

Egy singleton osztály. Feladata a játék elkezdése és lezárása, továbbá ezeknek a feltételeknek az ellenőrzése. Folyamatosan számon tartja hány Box mozgatható még, így ha ez a szám nullára csökken a játék autómatikusan véget ér, valamint a Player-ek pontjait és számát is ő tartja nyilván.

* + - * **Ősosztályok**

Nincs

* + - * **Interfészek**

Nincs

* + - * **Attribútumok**
* **int boxes**
* **int players**
* **int[] overallScore**
  + - * **Metódusok**
* **void UpdateScore(Player p):** Frissíti az adott Player pontszámát. Vagy növeli, vagy nulláza abban az esetben ha meghalt. Ha kiesik 3 játékos befejezi a játékot.
* **void CountBoxes(int add):** Számlálja a mozgatható Box-okat a Map-en. Amikor elfogynak befejezi a játékot.
* **void NewGame(int playerCount):** Új játékot indít a megadott számú játékossal.
* **void EndGame():** Véget ér a játék, ha vagy mozgatható Box, vagy 1 nél több irányítható Player nincs már a Map-en.
  + 1. **Hole**
       - **Felelősség**

A Field-ből öröklődik, ha bármi rákerül, eltűnik és törlődik a játékból. Ha Box kerül bele nem jár érte senkinek pont, ha Player, meghal és elveszíti az eddig összegyűjtött pontjait.

* + - * **Ősosztályok**

Field → Hole

* + - * **Interfészek**

Nincs

* + - * **Attribútumok**
* **Field[] neighbours**
* **Moveable myMoveable**
  + - * **Metódusok**
* **void FieldAction():** Ebben az esetben, bármi ami rákerül, leesik, tehát törlődik a játékból.
  + 1. **Map**
       - **Felelősség**

A pálya szabályos kialakításáért felelős, ezzel együtt raktározza a Field-eket.

* + - * **Ősosztályok**

Nincs

* + - * **Interfészek**

Nincs

* + - * **Attribútumok**
* **Field[] fields**
* **Game active\_map**
  + - * **Metódusok**
* **void CreateMap()**: Megadott algoritmusok szerint alkottja meg random a pályát az egyszerű Field-ek és a speciális Field-ek használatával. Lerakja a Box-okat, Wall-okat és a Player-eket is.
  + 1. **Moveable**
       - **Felelősség**

A Moveable egy olyan abstract osztály melyből a Box és a Player is egyaránt származik. Egy Field csak egy Moveable-t taralmazhat és a Moveable is eltárolja a Field-et amin épp elhelyezkedik.

* + - * **Ősosztályok**

Nincs

* + - * **Interfészek**

Nincs

* + - * **Attribútumok**
* **Field myField**

* + - * **Metódusok**

(Az abstract függvényeket nem vettük fel a jobb átláthatóság miatt)

* **virtual boolean Controll(Player p, Direction d):** Mozdulást intéz egy Field felé, és visszaadja, hogy sikerült-e a mozdulás. Első paramétere megmondja, hogy ki tolta, a második pedig, hogy milyen irányba.
* **virtual boolean IsStuck():** Alapvetően false-val tér vissza.
  + 1. **Player**
       - **Felelősség**

A Moveable-ből származó osztály. Felelős a Playerek mozgatásáért és a pontjaiért.

* + - * **Ősosztályok**

Field → Player

* + - * **Interfészek**

Nincs

* + - * **Attribútumok**
* **int score**
  + - * **Metódusok**
* **void AddScore():** Egyet hozzáad a Player pontjaihoz.
* **int GetScore():** Lekérdezi az adott Player pontját.
* **boolean Controll(Player m, Direction d):** Mozdulást intéz egy Field felé, és visszaadja, hogy sikerült-e a mozdulás. Első paramétere megmondja, hogy ki tolta, a második pedig, hogy milyen irányba.
* **void DeadScore():** -1-re állítja az adott Player pontjait, majd a Game UpdateScrore() függvénye kiveteti a játékból.
* **void Kill():** Ha meghívják meghívja a saját Field-je Remove-jat így gyakorlatilag “meghal”.
* **void Press():** Ez a függvény false-val hívja a gomb Activate() függvényét, mert ha player lép a gombra ő nem aktiválja.
* **Player GetLastTouchedMe():** Saját magát adja vissza.
* **void Die():** Meghívja a saját DeathScore() függvényét így megindítja a halálával járó eseményeket.
  + 1. **Target**
       - **Felelősség**

A Fieldből származó osztály. Csak a Box-okra aktiválódik a Player-eket figyelmen kívül hagyja. Ha egy Box rákerül az eltűnik és az utolsó lökést adó játékosnak pontot ad. Természetesen a Target mezőn is ugyan úgy összenyomódhat a Player mint bárhol máshol.

* + - * **Ősosztályok**

Field → Target

* + - * **Interfészek**

Nincs

* + - * **Attribútumok**
* **Field[] neighbours**
* **Moveable myMoveable**
  + - * **Metódusok**
* **void FieldAction():** Rátoláskor a Box eltűnik és az utolsó Playernek aki hozzá ért pont írást kezdeményez.

**4.3.11 TrapDoor**

* + - * **Felelősség**

A Field leszármazottja. Egy hozzá tartozó gomb utasítására nyílik ki és viselkedik úgy, mint egy Hole, vagy csukódik be és egy sima Field-ként létezik.

* + - * **Ősosztályok**

Field → Hole → TrapDoor

* + - * **Interfészek**

Nincs

* + - * **Attribútumok**
* **boolean isActive**
* **Field[] neighbours**
* **Moveable myMoveable**
  + - * **Metódusok**
* **void SetActive(boolean: b):** Átállítja a TrapDoor állását. Nyitottból csukott, csukottból nyitott.
* **void FieldAction():** Aktiválódáskor kinyílik, és ha éppen egy Player vagy egy Box volt rajta, akkor azt törli. Ha a hozzá tartozó Button-ról letolják a Box-ot akkor bezárul és sima Field lesz amire midnen következmény nélkül rá lehet lépni vagy tolni.

**4.3.12 Wall**

* + - * **Felelősség**

A Box-ból öröklődik. Sem a Player-ek sem a Box-ok nem képesek áthaladni rajta. Áthatolhatatlanok a játék kezdetétől a végéig. Úgy viselkednek mint egy Box ami mindíg stuck.

* + - * **Ősosztályok**

Moveable → Box → Wall

* + - * **Interfészek**

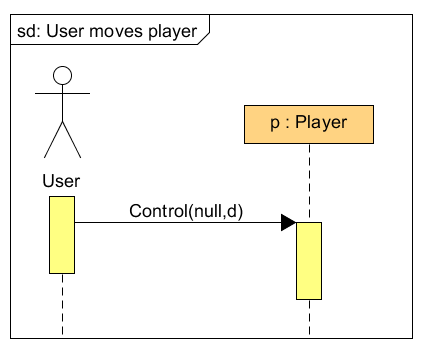
Nincs

* + - * **Attribútumok**
* **boolean stuck**
  + - * **Metódusok**
* **boolean Control(Player m, Direction d):** Mindig biztosan false-val tér vissza hiszen nem lehet rajta át menni és eltolhatatlan is.
* **boolean IsStuck():** Mindig biztosan true-val tér vissza hiszen folyton mozdíthatatlan.

***4.4 Szekvencia diagramok***

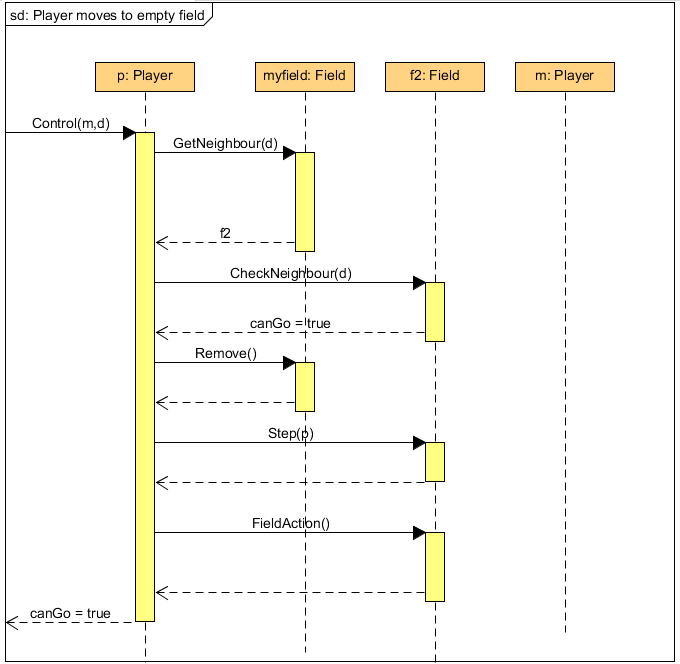
***4.4.1 User moves player***

A felhasználó mozgatja az egyik játékost. Fontos, hogy itt nem egy másik térképen lévő objektum “löki” meg a játékost - ezért a Controll első átvett értéke *null*.

******

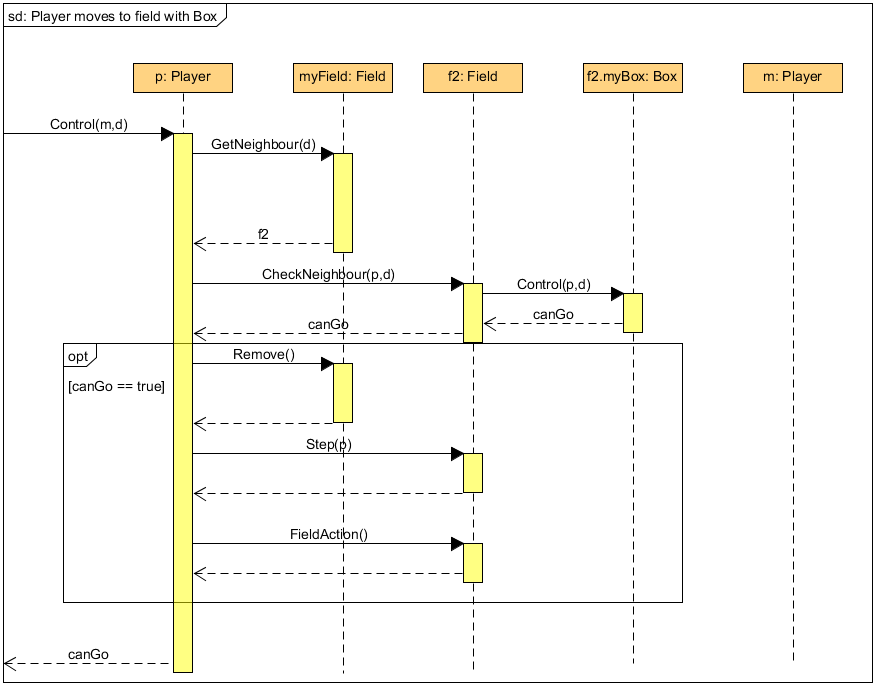
***4.4.2 Player moves to empty field***

A játékost valami (akár a user akár pl egy doboz) elmozgatja egy irányba, ahol üres mező taláható. Az ábrán nincs lekezelve az az eset, amikor van valami a szomszéd mezőn - ezt külön diagramon kezeljük. Az m értéke vagy a mozgást kezdeményező Player vagy null ha a játékos kezdte.

******

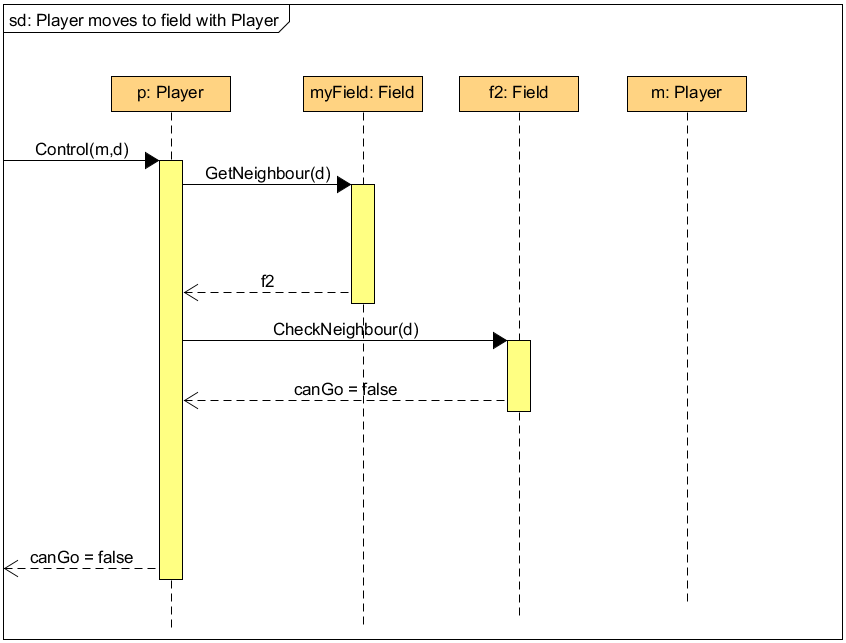
***4.4.3 Player moves to field with box***

A játékos olyan mezőre lép, amin van egy doboz. Ha a játékost a felhasználó mozgatta, akkor a Control első paramétere null, ha valamilyen másik objektum, akkor a mozgást kezdeményező Player. Ha a láda által visszaadott canGo érték igaz, akkor a játékos eltolhatja a dobozt és a lépés végbemegy. A doboznak saját magát adja a Controll-ban, így a dobozok mindig arról a Player-ről tudnak aki “utoljára ért bele” a mozgásba.

******

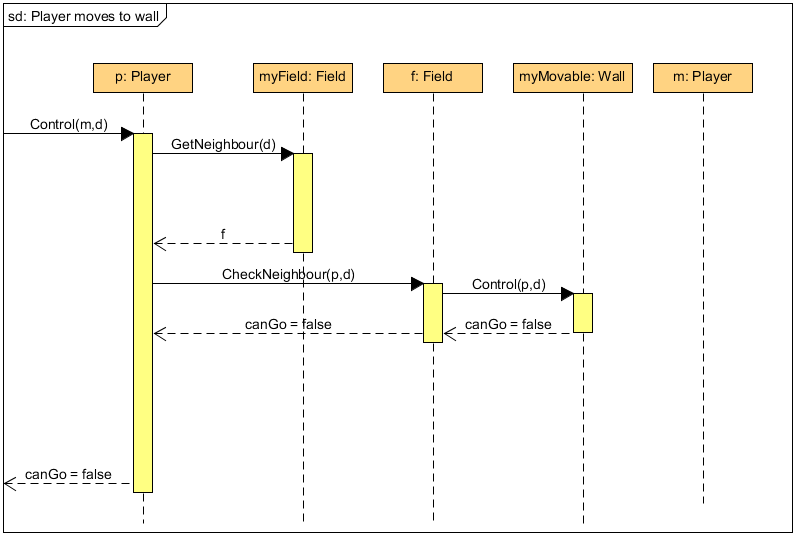
***4.4.4 Player moves to field with player***

A játékost a felhasználó vagy egy másik objektum egy olyan mezőre mozgatná, amin egy másik játékos áll. (Itt is a Controll függvény első paramétere ettől függően vagy null vagy a “megtoló” Player.)

******

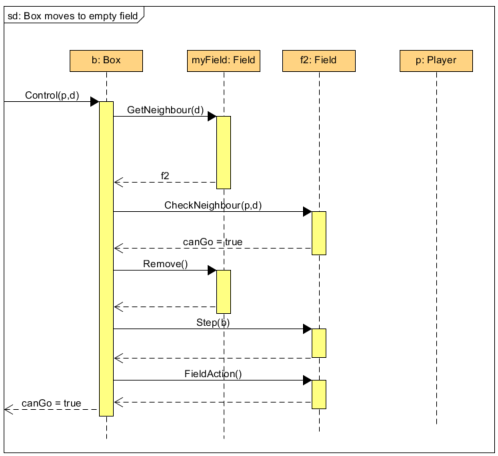
***4.4.5 Player moves to wall***

A Player egy falra akar lépni amit nem tehet, itt a Palyer lép Player-re esethez hasonlóannem fog oda lépni. (Itt is a Controll függvény első paramétere ettől függően vagy null vagy a “megtoló” Player.)



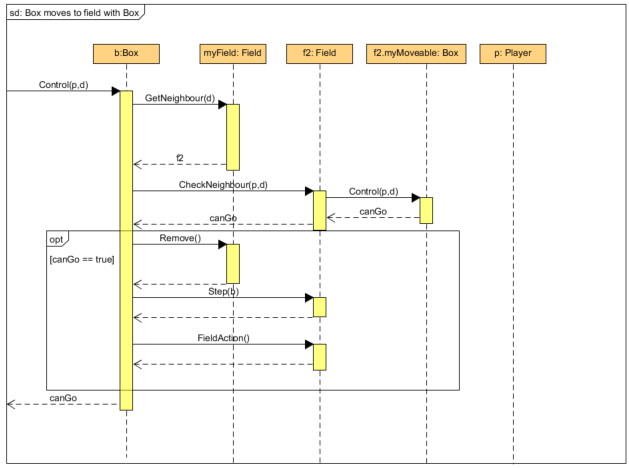
***4.4.6 Box moves to empty field***

Egy dobozt egy szomszédos üres mezőre tolják. Ebben a diagramban csak azt az esetet kezeljük, amikor a szomszéd mező üres. Mivel egy dobozt csak egy objektum tolhat el, ezért p értéke sose null.

******

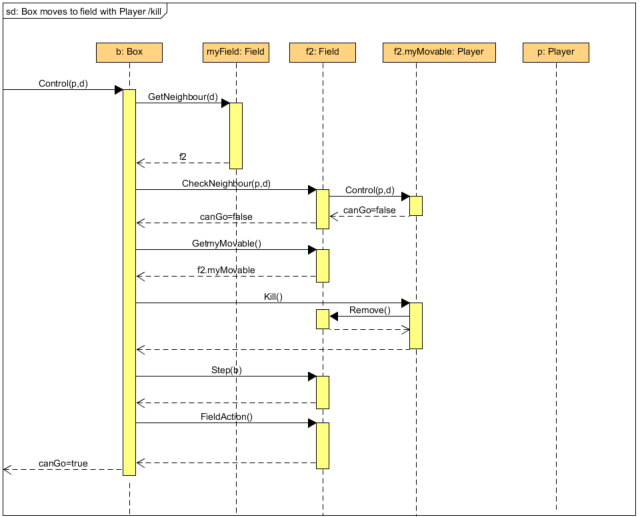
***4.4.7 Box moves to field with box***

A dobozt egy olyan mezőre tolják, amin szintén egy doboz van. Rekurzíve megnézi, hogy tud-e lépni az adott irányba. Ha igen, akkor lép. Mivel egy dobozt csak egy objektum tolhat el, ezért p értéke sose null.

******

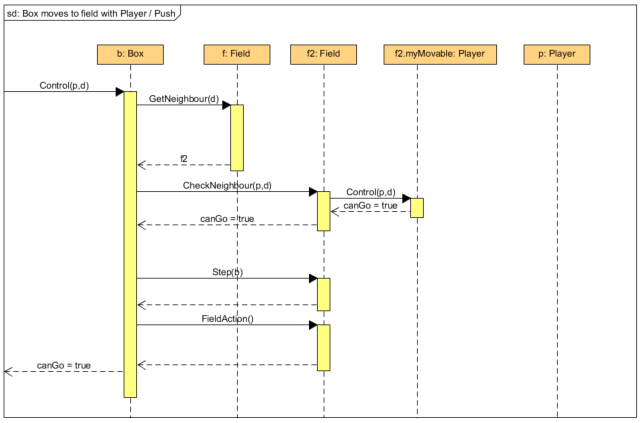
***4.4.8 Box moves to field with player that can not move on***

A dobozt egy olyan mezőre tolják, ahol egy másik játékos van. Az adott játékos, viszont nem tud mozogni arra, amerre tolják, ezért a láda összenyomja és meghal.

******

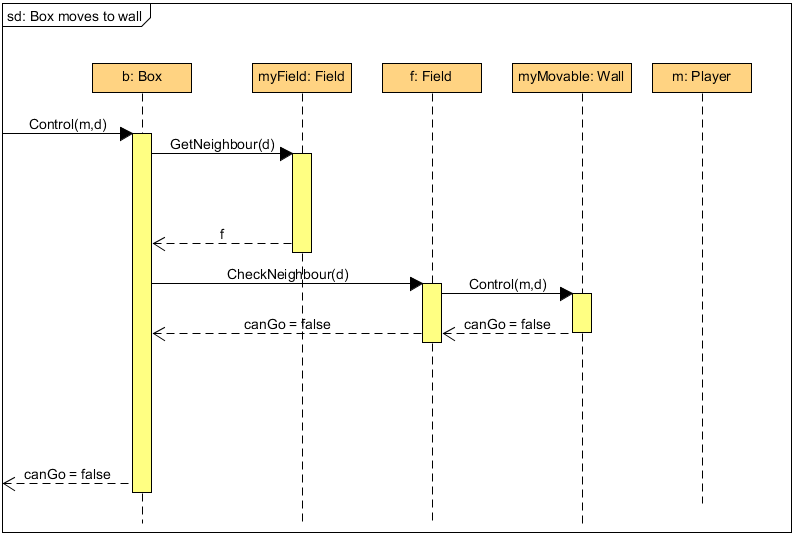
***4.4.9 Box moves to field with Player that can move on***

A Box-ot egy olyan Field-re tolják, amin van egy játékos. A játékos az aktuális irányba tovább tud mozogni, így a doboz eltolja.



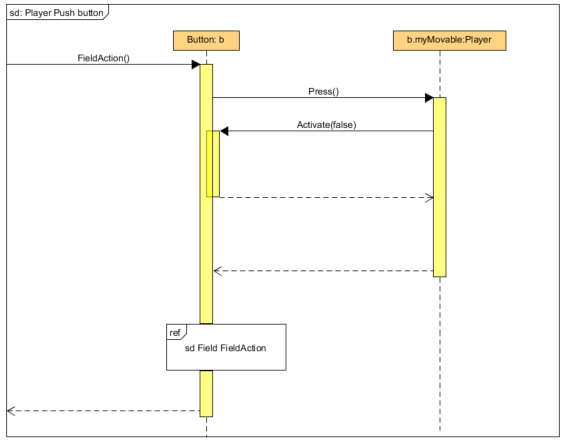
***4.4.8 Box moves to wall***

A dobozt egy falra akarják tolni - amit nem tehet - tehát itt a canGo false-al jön vissza és a lépés nem megy végbe.



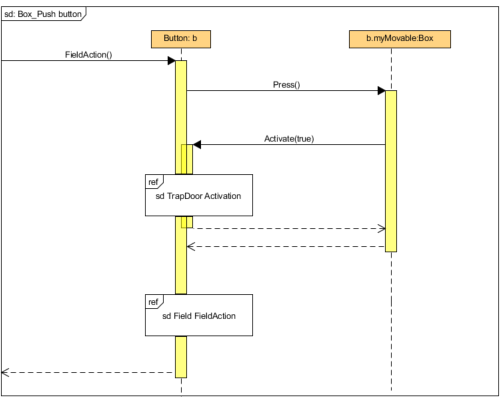
***4.4.9 Player pushes button***

Egy Player lép egy Button-ra. Meghívja a Button a Player press függvényét. Mivel egy Player nem tud megnyomni egy Buttont ezért “nem történik semmi”, nem aktiválódik a gomb (az Activatet false paraméterrel hívja).



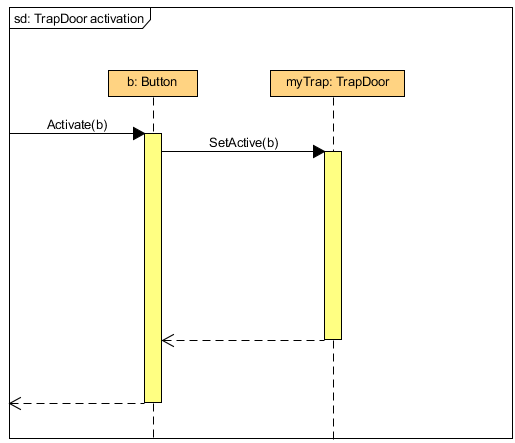
***4.4.10 Box pushes button***

A doboz egy olyan mezőre kerül, amin egy gomb van. Meghívódik a Button Activate függvénye a rajta lévő objektumnak megfelelő boolean értékkel. (a Box megnyomja a Player nem.)



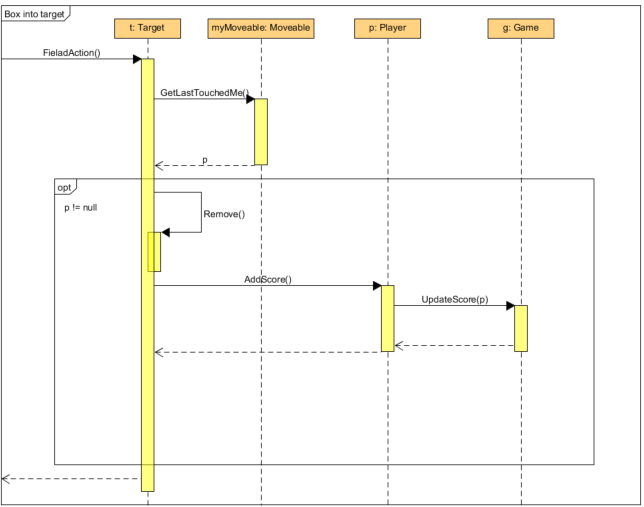
***4.4.11 TrapDoor activation***

Ha egy gombra rátolnak egy dobozt, az aktivál egy TrapDoort. Beállítja a gombon az aktivitást, ami meghívja az általa tartalmazott TrapDoor aktiváló függvényét, ami ez után kinyílik és Hole-ként viselkedik. Ha a doboz le kerül a csapó ajtó be zárul.

******

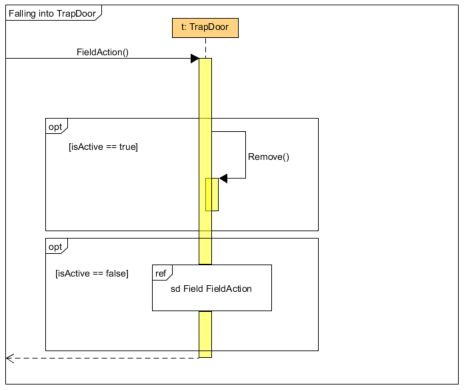
***4.4.12 Move box into target***

Egy doboz célba tolása. Megnézi a Field ki tolta meg az adott objektumot. Ha ez az objektum egy doboz, akkor azt eltünteti a pályáról, illetve az azt megtoló játékosnak jóváírja a pontot.

******

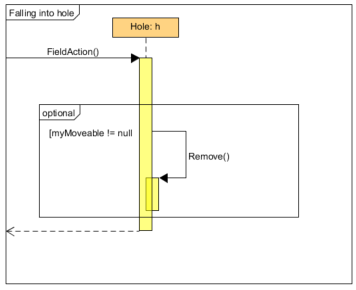
***4.4.13 Falling into TrapDoor***

Amikor egy objektum rákerül egy TrapDoor-ra, akkor annak aktiváltságától függően vagy beleesik és eltűnik, vagy nem.

******

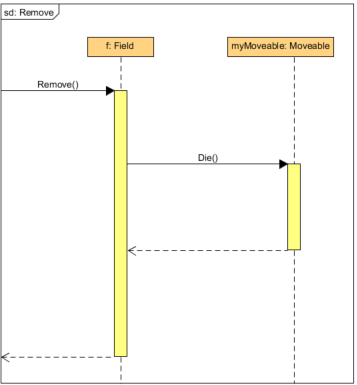
***4.4.14 Falling into hole***

Mikor egy objektum egy Hole típusú Fieldre lép, akkor ”leesik” és eltűnik a pályáról.

******

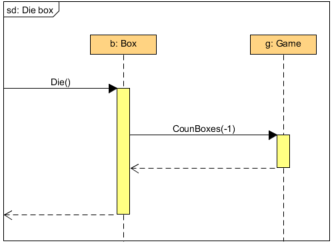
***4.4.15 Objektum eltüntetése Fieldről***

A Fieldre meghívjuk a Remove() függvényt, ami a rajta tartózkodó Moveable objektum Die függvényét hívja meg és kiveszi azt a játékból.

******

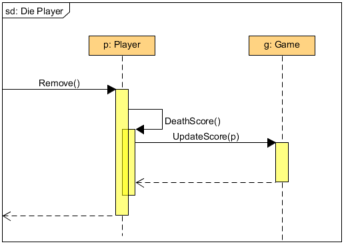
***4.4.16 Box’s Die function***

Csökkentjük az adott játékban lévő dobozok számát ha a doboz “meghal”.



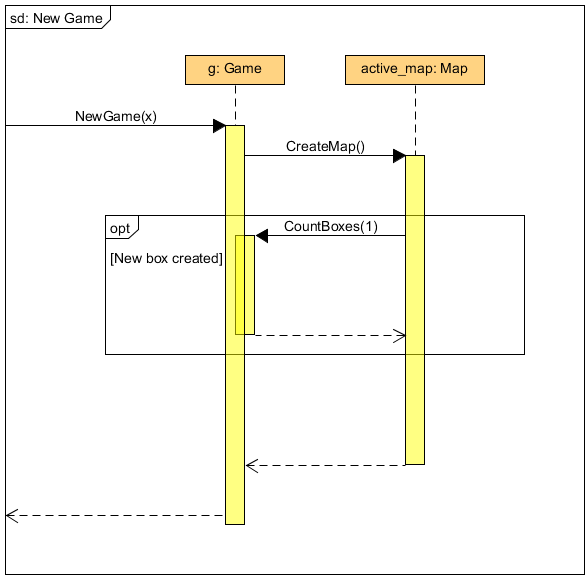
***4.4.17 Player’s Die function***

A játékosnak meghívjuk a DeathScore függvényét ezzel a pontszámát -1 -re állítva. Az UpdateScore függvény ekkor törli a játékost az adott játékból.

******

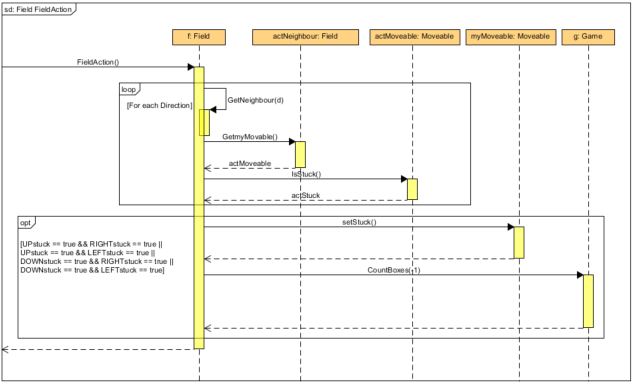
***4.4.18 New Game***

Létrehoz egy játékot, legenerálja a Map elemeit.

******

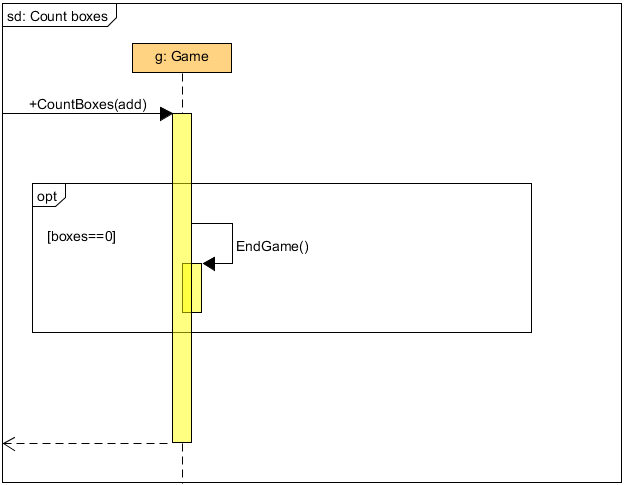
***4.4.19 Field Field Action***

Minden irányba megnézzük hogy tolható-e. Ha a doboz “beszorult” akkor csökkenti a pályán lévő dobozok számát. Amikor a dobozok elfogynak a játéknak vége.

******

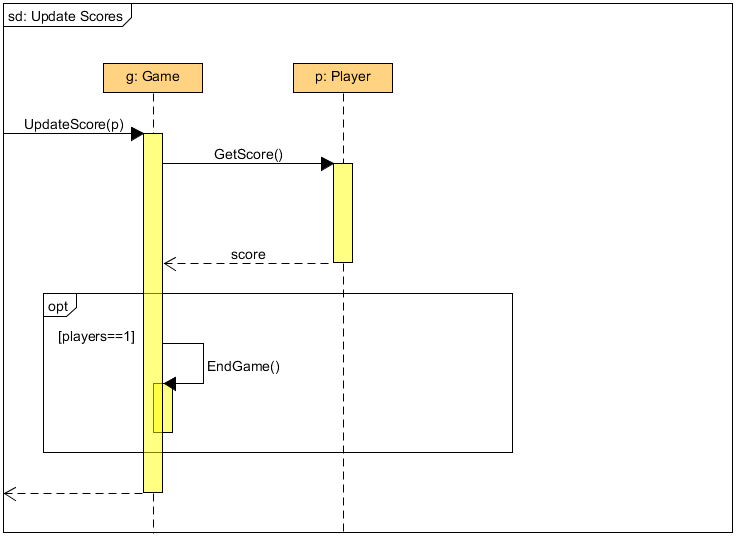
***4.4.20 Count Boxes***

A dobozok számát mindig a megadott értékkel “növeli” ha a dobozok elfogynak tehát nincs több mozgatható akkor a játék véget ér.

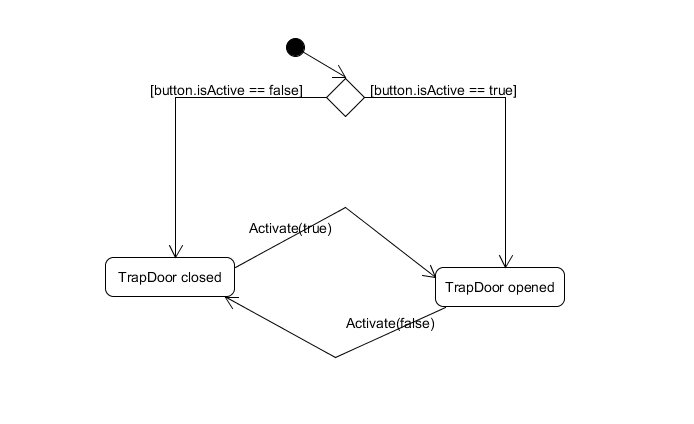
******

***4.4.21 Update Scores***

A játékosok pontját a DeathScore() -1-re állítja, ha ezt érzékeli az UpdateScore akkor csökkenti a Playerek számát. Ha a playerek száma 1 akkor az utolsó játékos nyer és a játék véget ér.

******

* 1. ***State-chartok***



***4.5 Napló***

|  |  |  |  |
| --- | --- | --- | --- |
| **Kezdet** | **Időtartam** | **Résztvevők** | **Leírás** |
| 2018.03.01 10:00 | 1 óra | Czetter  Frontó  Tomori  Verasztó | Megbeszélés:  Objektumkatalógus megvitatása. Feladatok kiosztása |
| 2018.03.01 11:00 | 3 óra | Czetter  Frontó  Tomori  Verasztó | Osztályok és szekvencia diagrammok átgondolása, újratervezése. |
| 2018.03.02 12:00 | 2 óra | Czetter  Frontó  Tomori  Verasztó | Osztályok és szekvencia diagrammok összehangolása, javítása. |
| 2018.03.04 15:00 | 5 óra | Czetter  Frontó  Tomori  Verasztó | Ellenőrzés, javítás dokumentáció írása és lektorálása. |