4. Analízis modell kidolgozása 2

40 – *[scrum_that]*

Konzulens:
Szabó Ádám Imre

Csapattagok:

Kovács Levente Ákos	CM6UKU	vazul250@gmail.com
Lovász Attila Bence	INCMI7	attonet2@gmail.com
Graics Vince	HY9XQ6	wince17@gmail.com
Magyar Milán Bertalan	MCDNQL	milangfx@gmail.com
Tóth Krisztián Dávid	J38GIK	tht.krisztian@gmail.com

2015. március 8.

Tartalomjegyzék

Ábrák jegyzéke

4. Analízis modell kidolgozása 2

4.1. Objektum katalógus

4.1.1. Glue

A „Glue” objektum megvalósít egy adott tulajdonságú akadályt. Amely robot belemegy, annak a sebességét megfelezi.

4.1.2. GUI

A grafikus felületet megvalósító objektum. Ez az objektum maga a menü, ami a játék indítása után ugrik fel. Itt találhatóak a beállítások (mint például a gondolkodás idő és a maximális játék idő vagy a körök száma) és a játékmódok. Gombnyomásra fogja elindítani a játék működési szálát. Ez az objektum kezeli az ablak eseményeit és a játék bezárását.

4.1.3. HUD

Ez az objektum követi és nyilvántartja, hogy a robotok hány checkpoint-on mentek át, mennyi olaj és ragacs van náluk amit felhasználhatnak, illetve kiírja a képernyőre a hátramaradó időt és a megtett körök számát. Feladata, hogy minden körben megvizsgálja, hogy a robotok elérték-e a következő checkpointot.

4.1.4. MapBuilder

Fájlból beolvassa és létrehozza a memóriában a pályát, a kezdő pozíciókat és a checkpointokat reprezentáló objektumokat. Mivel a MapBuilder objektum tárolja a pályát így feladat, hogy vizsgálja a robotok azon belül tartózkodását.

4.1.5. Oil

Ez az objektum az Obstacle osztály leszármazottja. Hasonlóan a Glue objektumhoz, egy adott hatást valósít meg, ami letiltja a következő körben történő irányítását a robotnak, ami belelépett.

4.1.6. Phoebe

A játék logikát megvalósító objektum. Listában tárolja a pályán tartózkodó robotokat, akadályokat és figyeli, hogy mikor ér véget a játék. A „Phoebe” objektum rajzolja ki az objektumokat a pályán és szálként indítható osztályt, melyben maga a játék fut. Játékindításkor berakja a pályára a robotokat és az akadályokat a kezdő pozíciókba. Ebben az objektumban történnek az ellenőrzések (akadályba vagy robotba ütközések, pályáról leesés).

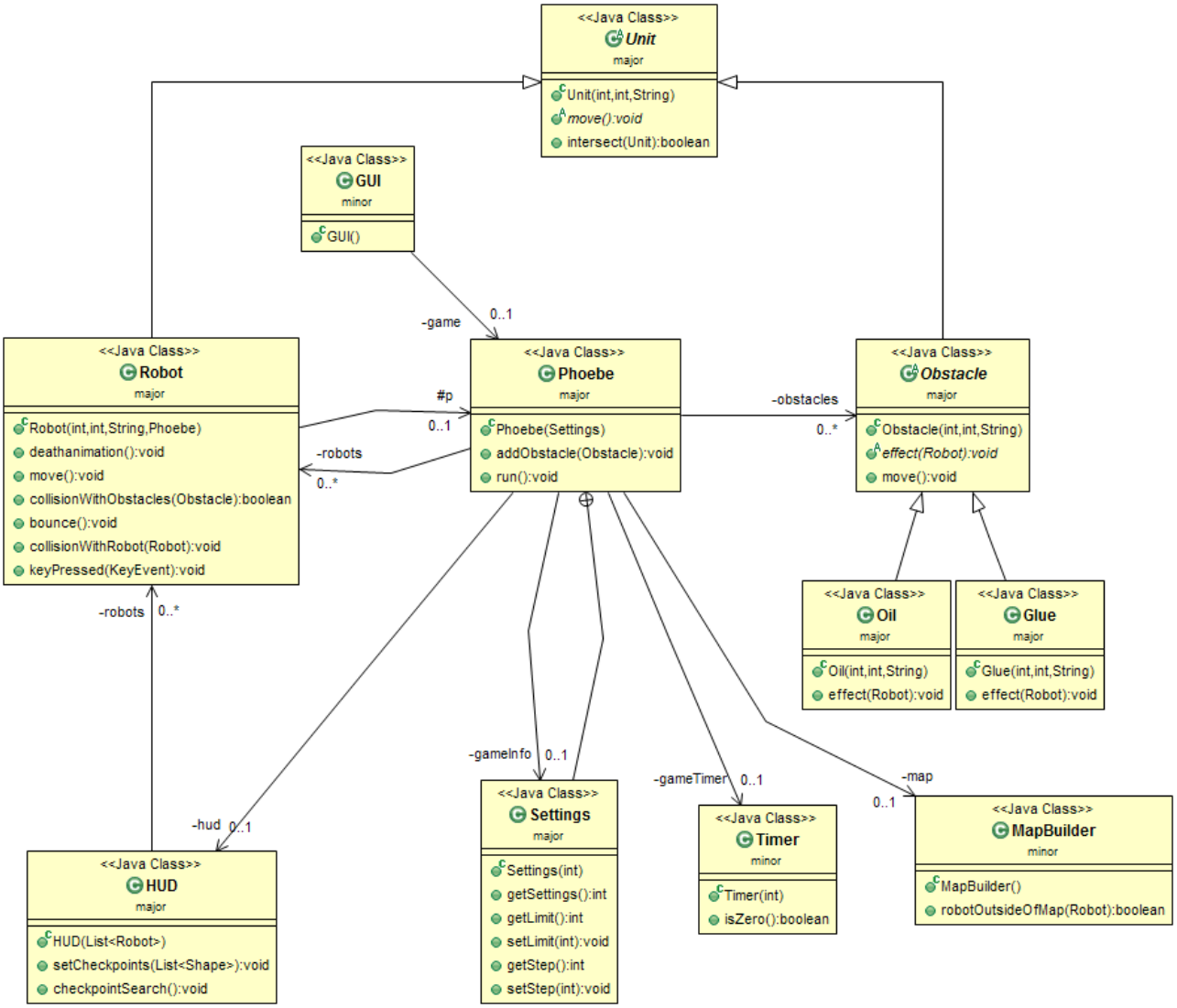
4.1.7. Robot

Olyan objektum, mely a pályán található robotokat valósítja meg. Leírja a viselkedésüket és a kezelésüket. A „Robot” osztály a Unit-ból származik le, ezáltal van pozíciója és az ütközés is le van kezelve. Felelős a mozgásért, megállapítja egy adott akadállyal vagy robottal ütközött-e és kezeli a felhasználó által leütött gombokat.

4.1.8. Timer

Az eltelt időt és a fennmaradt idő nyilvántartásáért felelős. Ilyen például a játék elején a három másodperces visszaszámlálás vagy az időlimites játékmód esetén, amikor a maximális időtől számol visszafelé.

4.2. Statikus struktúra diagramok



4.1. ábra. Statikus struktúra diagram

4.3. Osztályok leírása

4.3.1. Glue

- Felelősség
  A játékban szereplő Ragacs foltok viselkedését leíró osztály
- Ősosztályok
  Unit→Obstacle
- Metódusok
  - **Glue**(int x, int y): Amikor létrehozunk valahol egy ragacs objektumot, akkor ez a konstruktor hívódik meg. Meghívja az ősosztály konstruktorát.
  - void **effect**(Robot r): Ütközéskor hívja meg az ütközést vizsgáló függvénye a Robot osztálynak. Módosítja a robot slowed értékét 50%-ra a robot slowed attribútum setterének meghívásával.

4.3.2. GUI

- Felelősség
  A grafikus felületért felelős osztály, amely a menüt és a játékot jeleníti meg.
- Attribútumok
  - **Phoebe** game: referencia a játékra
- Metódusok
  - **GUI**(): Konstruktor. Beállítja az ablak nevét, létrehozza az ablak elemeit, elrendezi őket és beállítja a figyelőket(ActionListener).

4.3.3. HUD

- Felelősség
  A robotok ragacs- és olajkészletét, illetve megtett köreit és checkpontjait számolja. Megvalósítja a checkpoint ellenőrzést.
- Attribútumok
  - **int[]** checkpointReached: Minden robothoz külön tárolja a legutoljára érintett checkpoint sorszámát.
  - **int[]** lap: Minden robothoz tárolja a megtett körök számát.
  - **int[]** numGlue: Minden robothoz tárolja a ragacsok számát.
  - **int[]** numOil: Minden robothoz tárolja az olajok számát.
  - **List** checkpoints: Tárolja a checkpointokat reprezentáló objektumokat List adatszerkezetben. A checkpointSearch függvény kérdezi le ebből a következő checkpoint helyzetét.
  - **List<Robot>** robots: A robotokat tároló List adatszerkezet. A checkpointSearch függvény kérdezi le ebből a robotokat, majd azok helyzetét.
- Metódusok
  - **HUD**(List<Robot> robs): Konstruktor, inicializálja a köröket számláló változót, az érintett checkpointokat, a ragacs és olajkészleteket.
  - void **checkpointSearch**(): Ellenőrzi hogy a robotok teljesítették-e a következő checkpointot.
  - void **setCheckpoints**(List checkObj): Checkpointokat reprezentáló adatszerkezet betöltése.

4.3.4. List

- Felelősség
  - Objektumok tárolása, ezt az interfészt megvalósító osztályban.
  - `https://docs.oracle.com/javase/6/docs/api/java/util/List.html`

4.3.5. MapBuilder

- Felelősség
  A pálya felépítéséért, a checkpointok tárolásáért és a robot pályán tartózkodásának vizsgálatáért felelős osztály.
- Attribútumok
  - **List** checkpoints: Tárolja a checkpointokat reprezentáló objektumokat List adatszerkezetben.
  - **int[]** startPosPlayerOne: Meghatároz egy (x,y) koordinátát, ahol az első játékos kezd.
  - **int[]** startPosPlayerTwo: Meghatároz egy (x,y) koordinátát, ahol az második játékos kezd.
- Metódusok
  - **MapBuilder**(): Konstruktor, a pálya beolvasása fájlból, majd létrehozása.
  - boolean **robotOutsideOfMap**(Robot r): Igaz értéket ad vissza, ha a robot leesett a pályáról, hamisat ha még rajta van.

4.3.6. Obstacle

- Felelősség
  A pályán/játékosoknál lévő különböző akadályokat (ragacs,olaj) összefogó ősosztály.
- Ősosztályok
  Unit
- Attribútumok
  - **int** WIDTH: Az akadályokat jellemző szélesség. Szükség van rá, hogy létrehozzuk a leszármazottak hitbox-át(sokszög pályaelem).
  - **int** HEIGHT: Az akadályokat jellemző hosszúság. Szükség van rá, hogy létrehozzuk a leszármazottak hitbox-át(sokszög pályaelem).
- Metódusok
  - **Obstacle**(int x, int y, String imagelocation): meghívja a Unit konstruktorát a megadott adatokkal és létrehoz egy sokszög elemet ami reprezentálja a pályán majd.
  - void **effect**(Robot r): Meghatározza, milyen hatással van a robotra, ha érintkezik egy Obstacle-lel. Absztrakt.
  - void **move**(): Mozgatásért felelős függvény. Ősosztályból öröklött, felülírt függvény. Mivel nem lehetséges az akadályok mozgása, ezért üres a függvény törzse.

4.3.7. Oil

- Felelősség
  A pályára lerakható olaj megvalósítása. Ha belelép egy játékos egy ilyen olajfoltba, az effect függvény letiltja a mozgatást az adott roboton a következő ugrásig.

- Ősosztályok
  Unit → Obstacle

- Metódusok
  - **Oil**(int x, int y, String imagelocation): Egy Oil elem létrehozásáért felelős.
  - void **effect**(Robot r): Meghatározza, milyen hatással van a robotra, ha beleugrik egy olajfoltba. Ebben az esetben letiltja a játékost, hogy irányt váltson.

4.3.8. Phoebe

- Felelősség
  A játék motorját képviselő osztály. A robotok pozíciójáért, az akadályok elhelyezéséért és a játék végéért felel.

- Attribútumok
  - **boolean** ended: Állapot változó, ha vége a játéknak, akkor true. Ha beteljesül egy játék végét jelentő esemény, akkor ezen a változón keresztül leáll a játék és megállapítódik a nyertes.
  - **List<Robot>** robots: A játékban szereplő robotok listája.
  - **List<Obstacle>** obstacles: A játékban szereplő akadályok listája.
  - **HUD** hud: A játékosok előrehaladását, ragacs és olajkészleteit tartja számon
  - **MapBuilder** map: A pályát reprezentáló objektum. Tárolja még a checkpointokat és a robotok kezdő koordinátáit.

- Metódusok
  - **Phoebe**(Setting set): A játék felépítése, a robotok lista, az akadályok lista létrehozása.
  - void **addObstacle**(Obstacle ob): Az obstacles tárolóba helyez egy akadályt.
  - void **run**(): Ez a metódus futtatja a főciklust, amelyben maga a játék működik.

4.3.9. Robot

- Felelősség
  A játékban résztvevő robotok viselkedését és kezelését leíró osztály.

- Ősosztályok
  Unit

- Attribútumok
  - **Phoebe** p: Referencia a játékmotorra.
  - **int** staticID: Az osztályhoz tartozó statikus azonosító, a példány azonosítójának(id) meghatározásához szükséges.
  - **int** HEIGHT: A robot képének magassága, collision detektálásnál, továbbá az irányítást segítő nyíl kezdő koordinátájának meghatározásánál szükséges.
  - **int** WIDTH: A robot képének szélessége, funkcionalitásban hasonló a WIDTH-hez.
  - **int** ID: A robot példányának egyedi azonosítója, a keyconfig sorának indexelésére szükséges.
  - **double** slowed: A sebesség módosításáért felel, default értéke 1.0. Amennyiben ragacsba lép a robot ez 0.5-re módosul és minden ugrás végén visszaáll az eredeti értékére. Ugrásnál ezzel szorozzuk be a végkordinátát kiszámító sugár hosszát.

  - **boolean** oiled: Azt jelzi, hogy olajba lépett-e. Ennek hatására a mozgás iránya módosíthatatlanná válik egy kis időre.
  - **int** arrowendx: A robot irányítását segítő nyílnak az x koordinátája, a nyíl kirajzolásánál van szerepe.
  - **int** arrowendy: A robot irányítását segítő nyílnak az y koordinátája, a nyíl kirajzolásánál van szerepe.
  - **double** alpha: A robot irányítását segítő nyíl vízszintessel bezárt szöge. A nyil kirajzolásánál, az ugrás végpontjának meghatározánál van szerepe.
  - **boolean** moved: Azt jelöli, hogy lépett-e már a robot az aktuális körben. A megjelenítésnél (a nyilat ugrás közben nem jelenítjük meg), illetve az irányítás letiltásánál van szerepe (olajba lépés esetén).

- Metódusok
  - **Robot**(int x,int y,String imagelocation,Phoebe p): Létrehoz egy robotot a megadott x,y koordinátákon, betölti a képét az imagelocation cím alapján, továbbá eltárolja a játékmotor referenciáját.
  - **void deathanimation**(): A Robot halálának grafikus megjelenítéséért felelős függvény.
  - boolean **collisionWithObstacle**(Obstacle o): Ellenőrzi hogy a robot ütközött-e az akadállyal. Igazzal tér vissza ha igen, hamissal ha nem.
  - void **collisionWithRobot**(Robot r): Ellenőrzi, hogy a robot ütközött-e másik robottal. Ha igen, akkor gondoskodik róla, hogy a robotok a megfelelő szögben pattanjanak le egymásról.
  - void **bounce**(): Két robot ütközése után meghívódó függvény. Az ütközés függvényében kiszámolja a lepattanás irányát és letiltja egy körre az irány változtatást.
  - void **move**(): A robot mozgását megvalósító függvény.
  - void **keyPressed**(): A robot irányítását megvalósító függvény, a játékmotor keylistener-e által hívódik meg, a lenyomott billentyű keyevent-jére. A következő ugrás beállítása, a ragacs/olaj lerakása történhet itt.

4.3.10. Timer

- Felelősség
  A játék elején a kezdésig visszaszámol. Játéktípustól függően felfelé vagy visszafelé számol. Ez az osztály felelős, hogy ha lejárt az idő, akkor legyen vége a játéknak.

- Metódusok
  - **Timer**(int i): Konstruktor; inicializál egy viszaszámláló órát.
  - boolean **isZero**(): Igazzal tér vissza, ha a megadott idő lejárt.

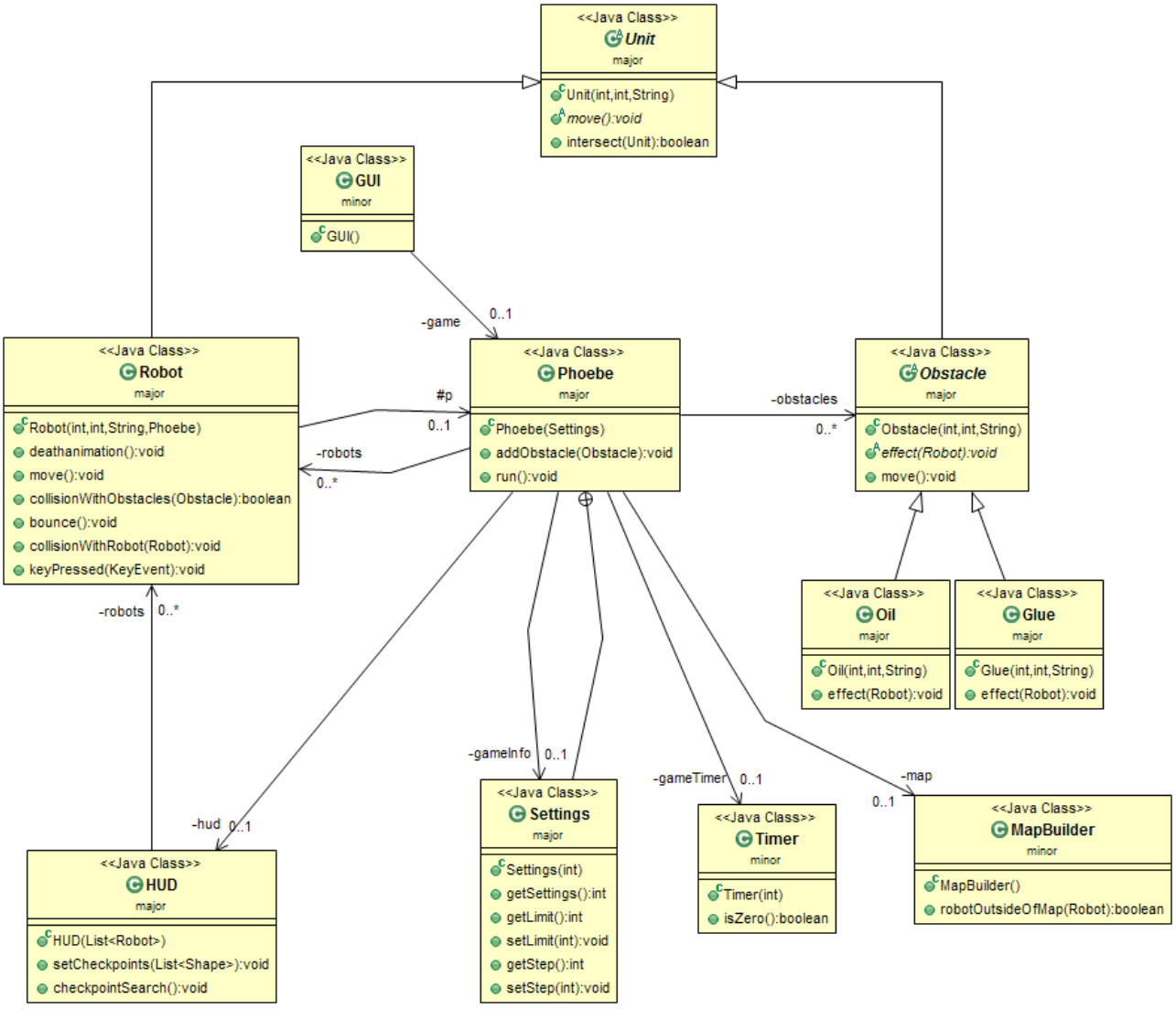
4.3.11. Unit

- Felelősség
  A pályán található objektumokért felel és azok viszonyáról (például ütközésükről).

- Attribútumok
  - **int** x: Az egység x koordinátája
  - **int** y: Az egység y koordinátája
  - **Object** hitbox: Az egységet a pályán reprezentáló sokszög.

- Metódusok

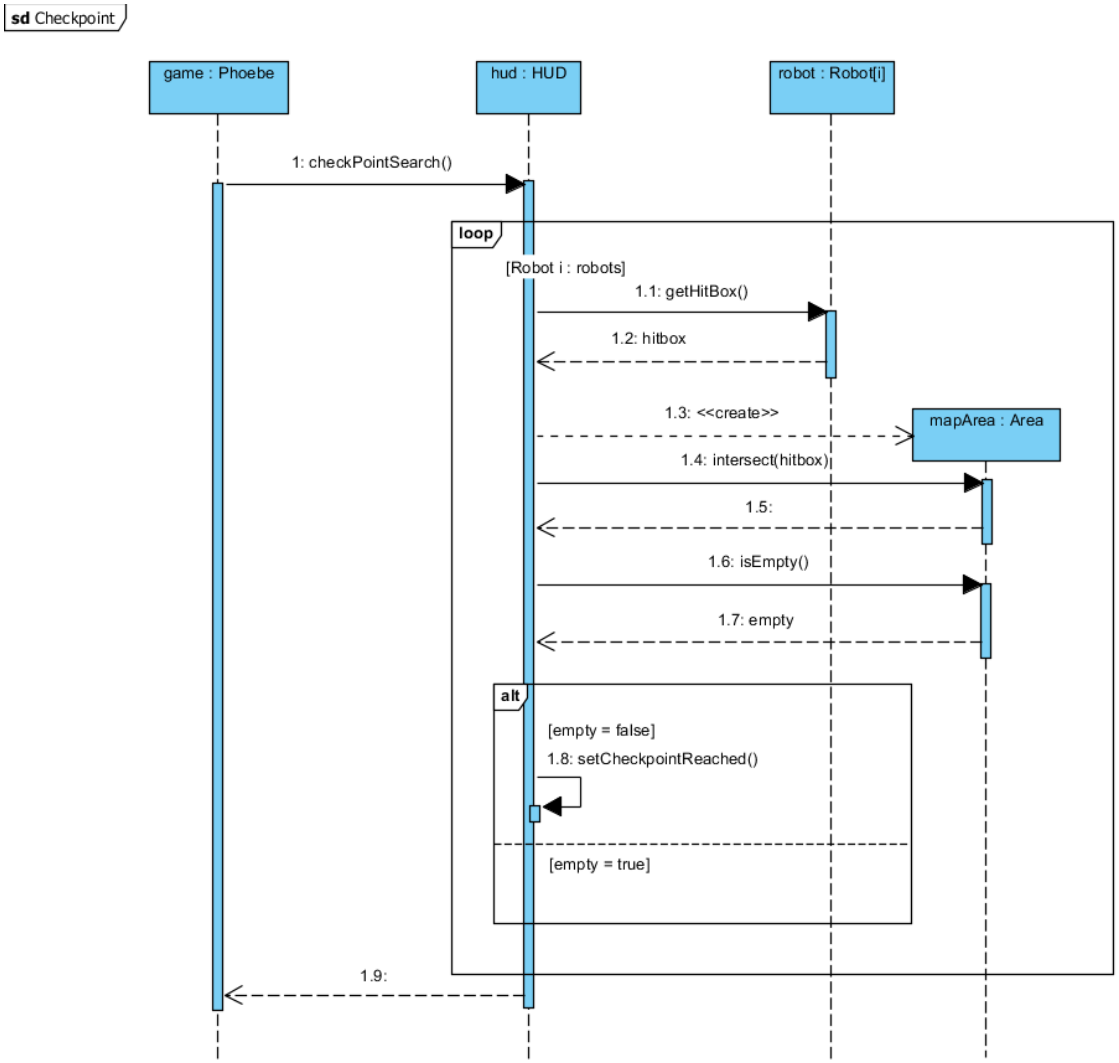
– **Unit**(): A Unit osztály konstruktora. Feladata, hogy eltárolja az x,y koordinátát.
– void **move**() : Absztrakt függvény, mely a leszármazottakban fog megvalósulni. Az egységek mozgásáért felelős.
– boolean **intersect**(Unit u): Két egység ütközését meghatározó függvény.



4.2. ábra. Statikus struktúra diagram

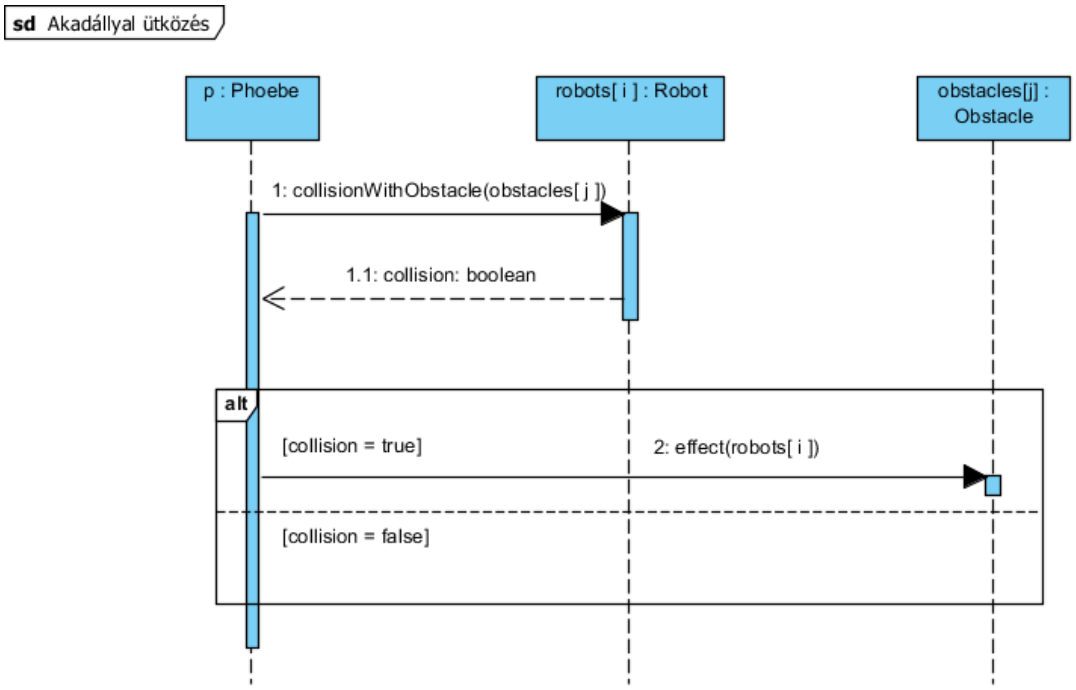
4.4. Szekvencia diagramok

4.4.1. Robot::CheckpointSearch



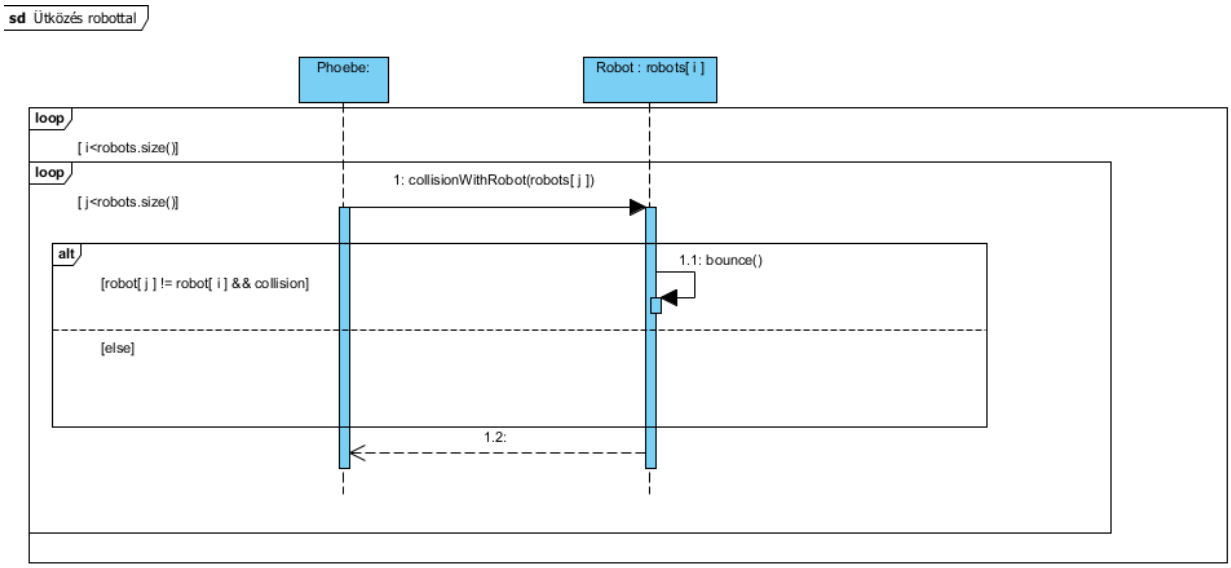
4.3. ábra. Következő checkpoint vizsgálata

4.4.2. Robot::CollisonWithObstacle



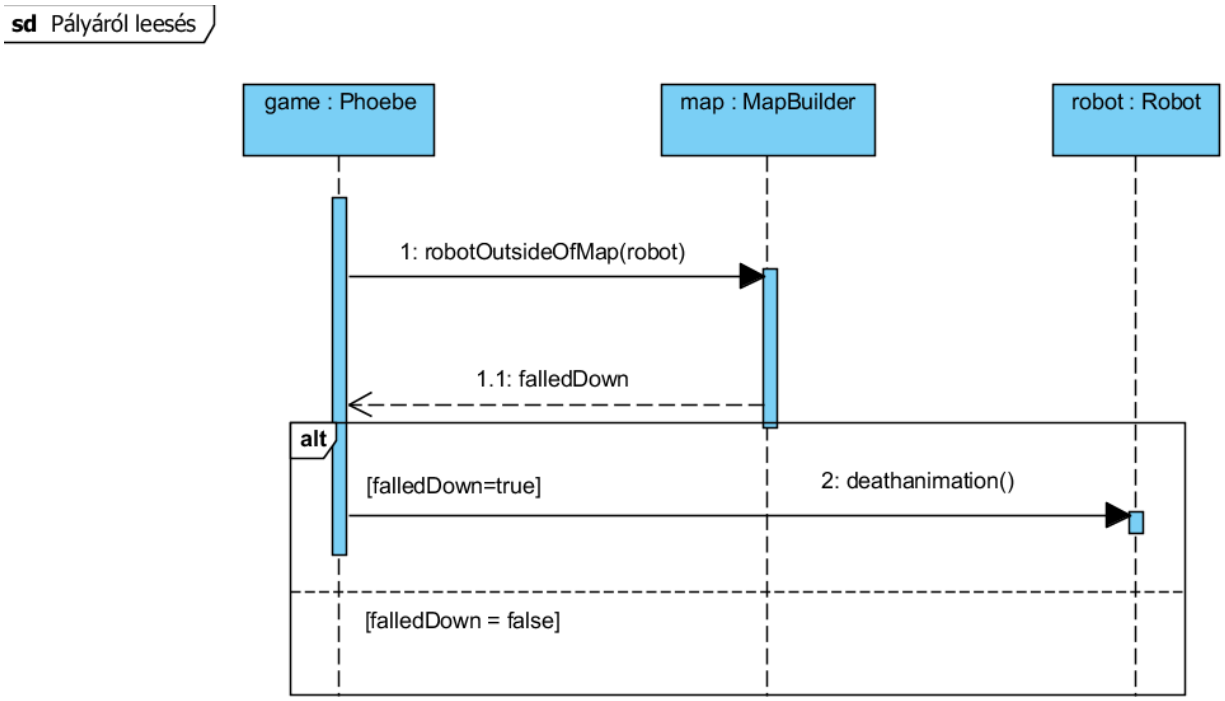
4.4. ábra. Robot ütközése akadállyal

4.4.3. Robot::CollisionWithRobot



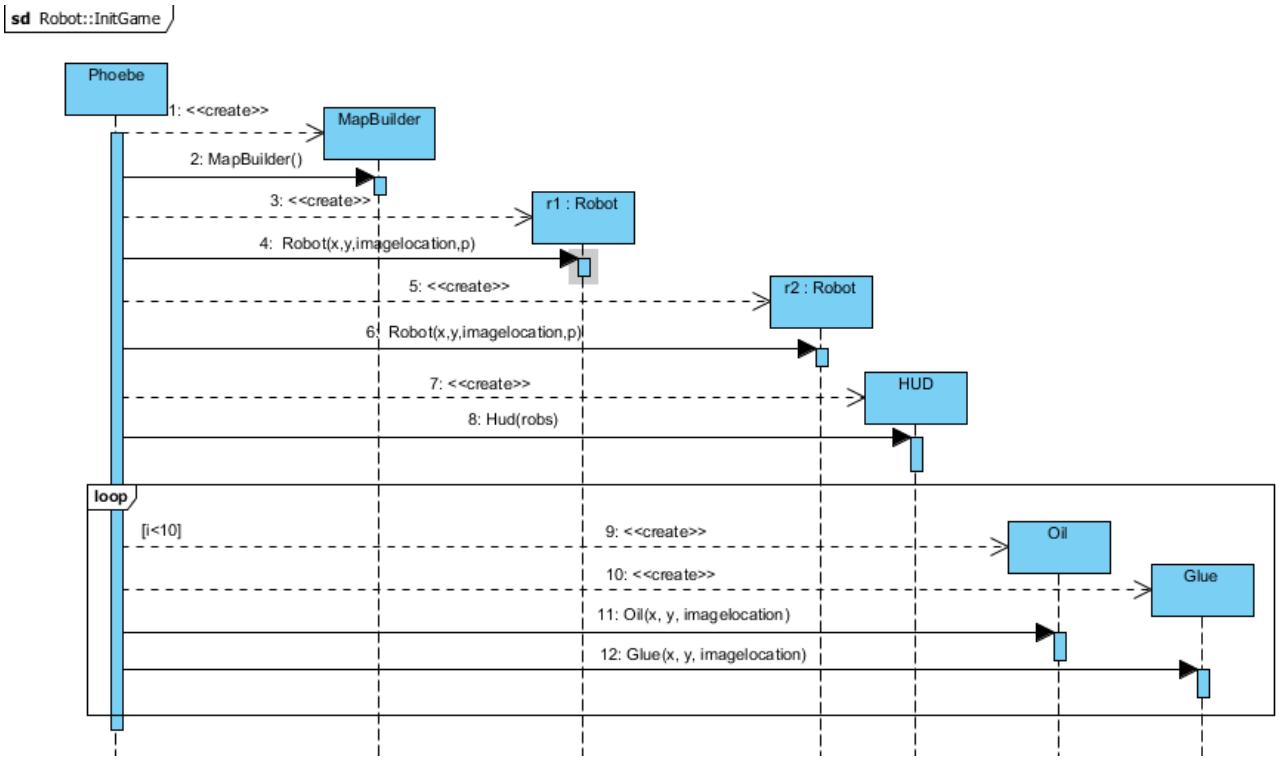
4.5. ábra. Robot ütközése másik robottal

4.4.4. Robot::FallDown



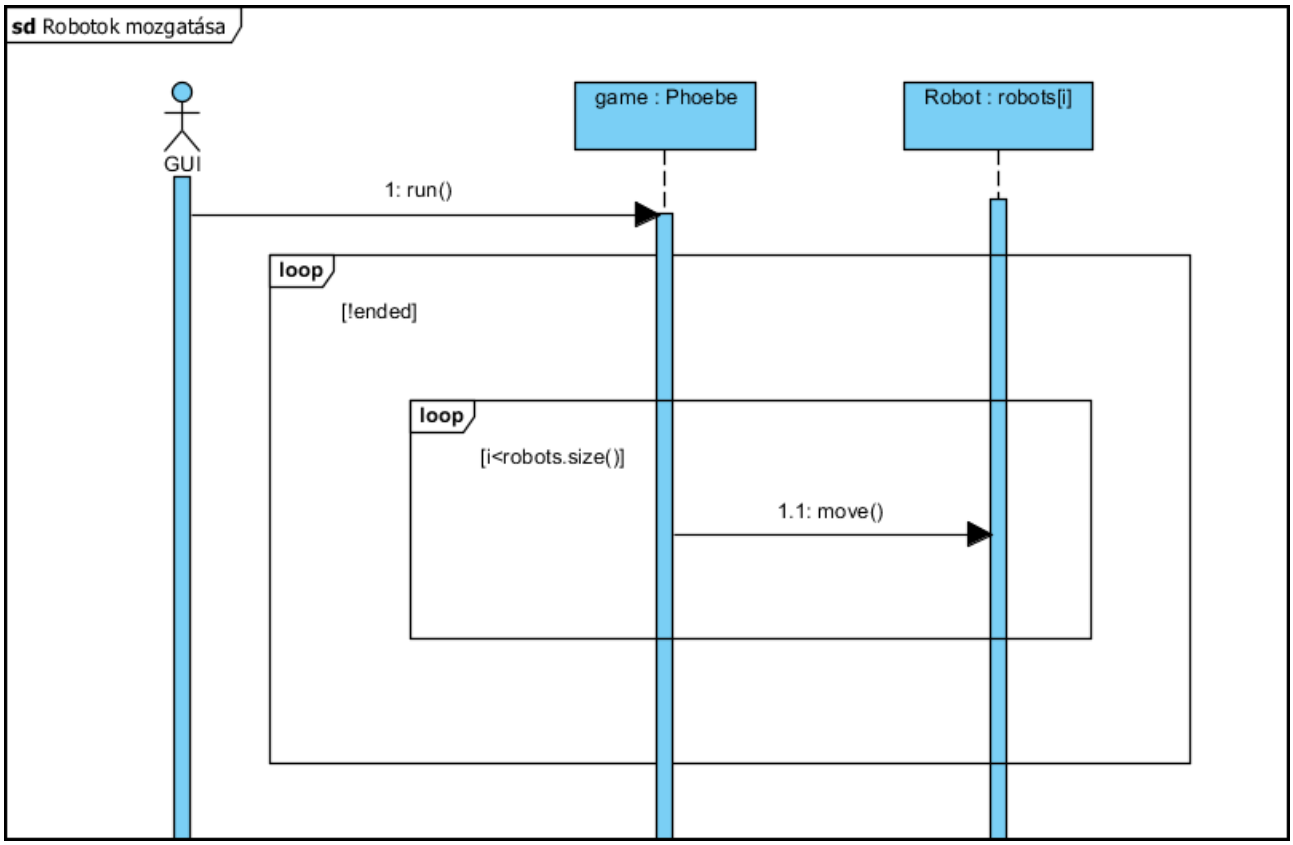
4.6. ábra. Robot leesése a pályáról

4.4.5. Robot::InitGame



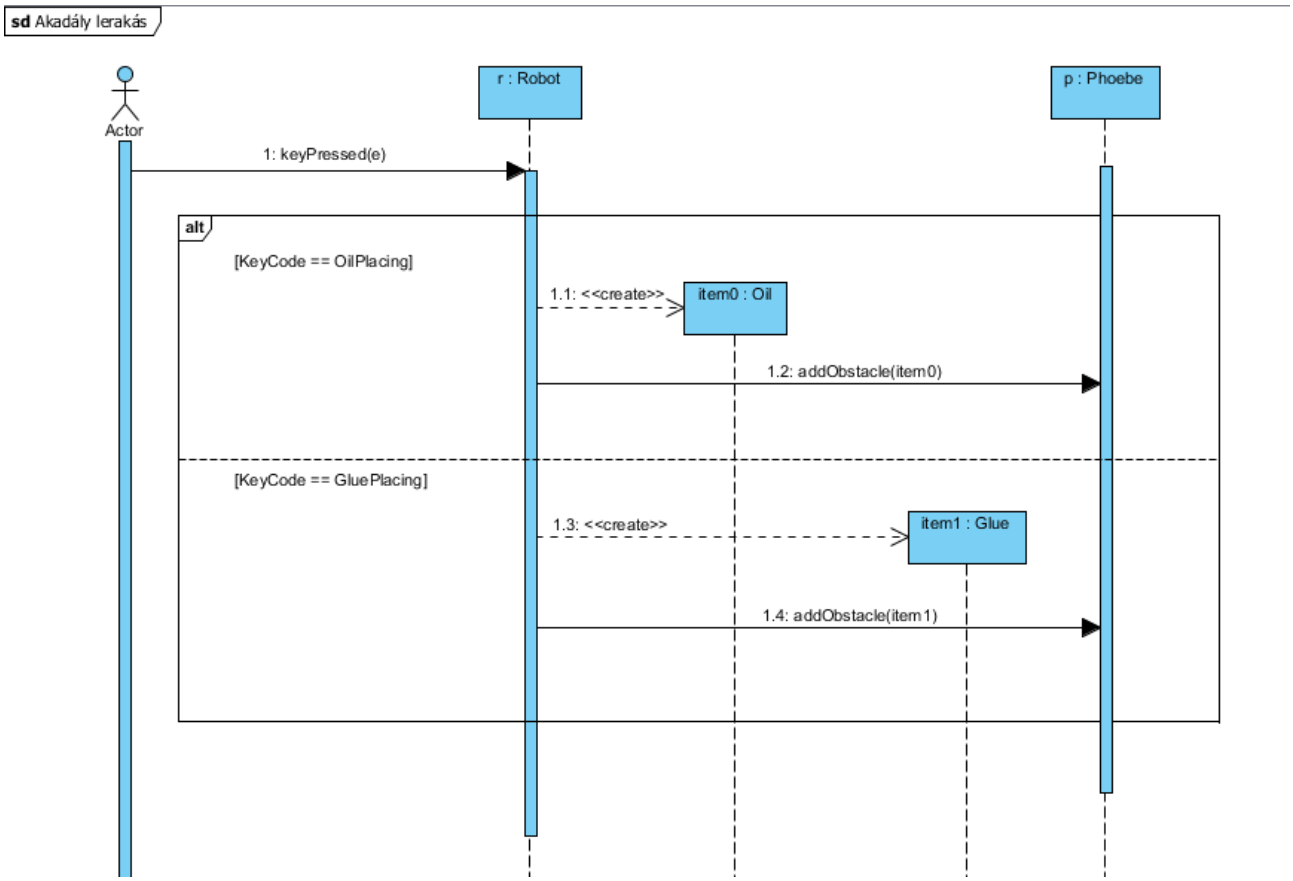
4.7. ábra. A játék inicializálása

4.4.6. Robot::Move



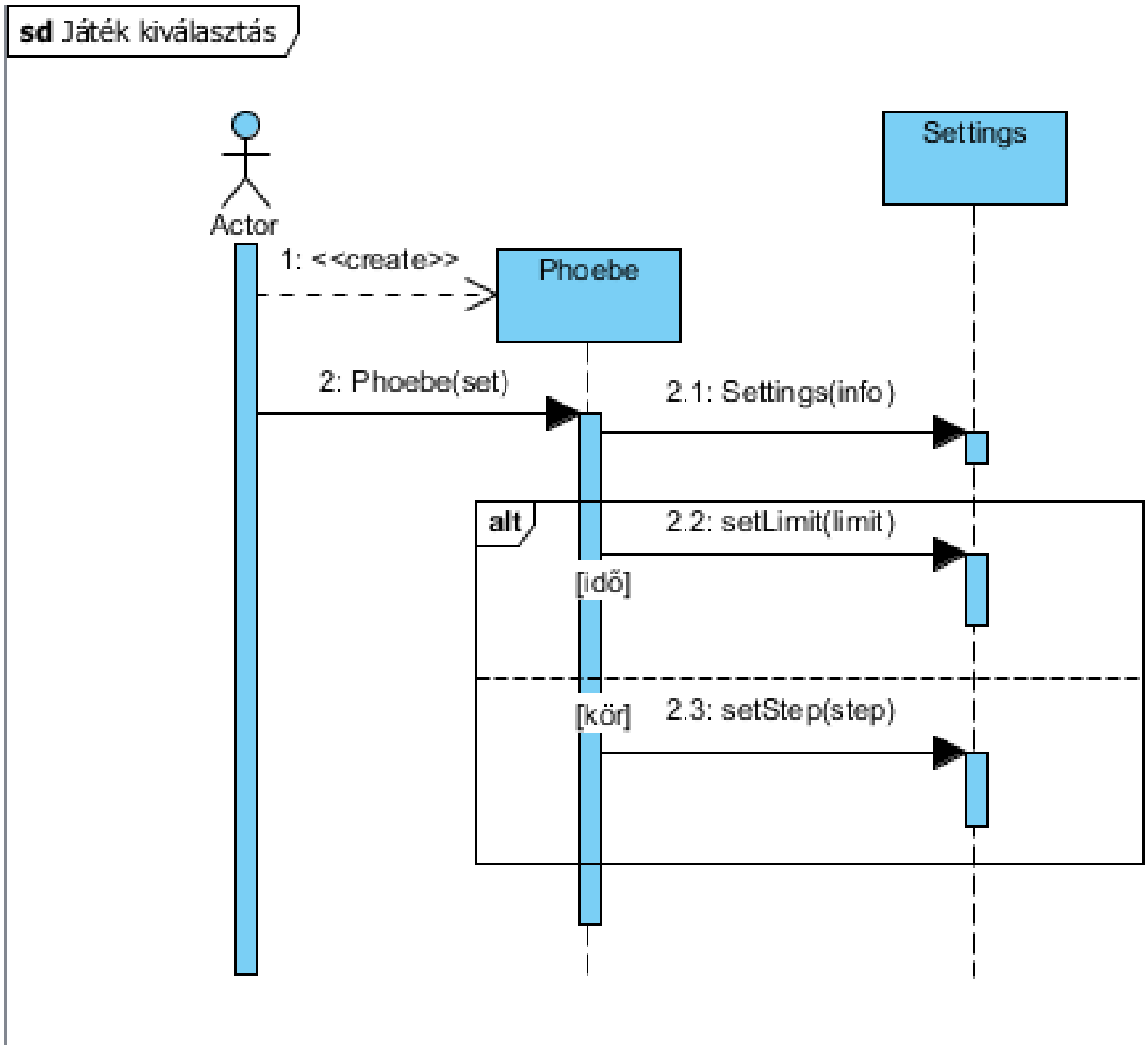
4.8. ábra. A robot mozgatása

4.4.7. Robot::NewObstacle



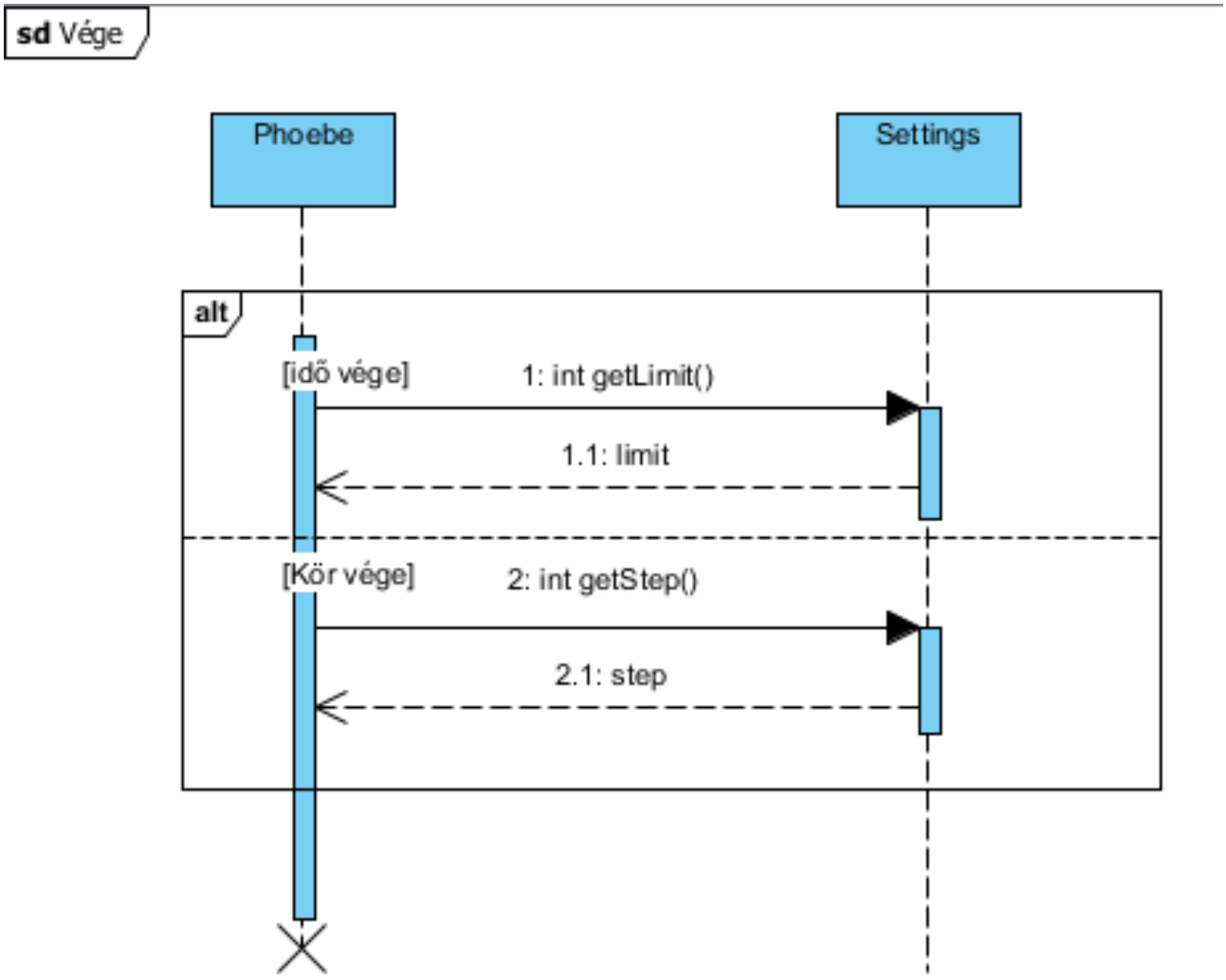
4.9. ábra. Akadály lerakása

4.4.8. Robot::Settings



4.10. ábra. A játék beállításainak kiválasztása

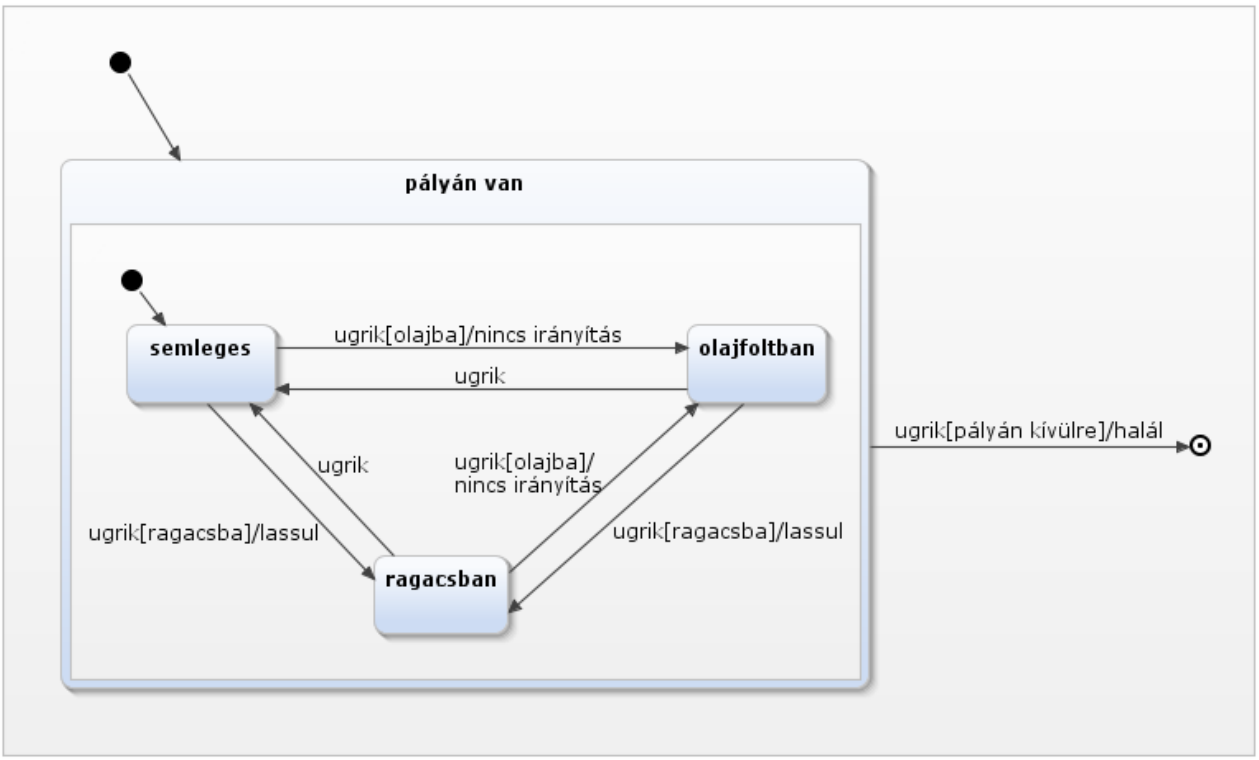
4.4.9. Robot::End



4.11. ábra. Játék vége

4.5. State-chartok

4.5.1. Robot::States



4.12. ábra. A robot állapotai

4.6. Napló

Kezdet	Időtartam	Résztvevők	Leírás
2015.03.04. 8:15	2 óra	**Kovács** **Lovász** **Tóth** **Graics** **Magyar**	Konzultáció, részfeladatok kiosztása
2015.03.06. 17:00	3 óra	**Magyar** **Tóth**	Analízis modell finomítása, hibáinak kijavítása