10. Grafikus felület specifikációja

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

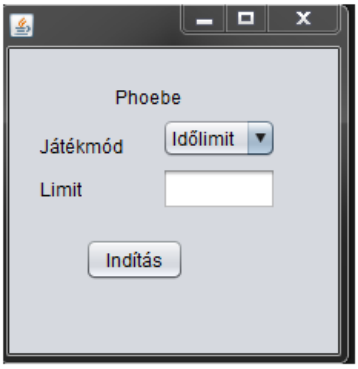
2015. április 26.

Tartalomjegyzék

10. Grafikus felület specifikációja

10.1. A grafikus interfész

A programot elindítva megjelenik egy menü ablak:

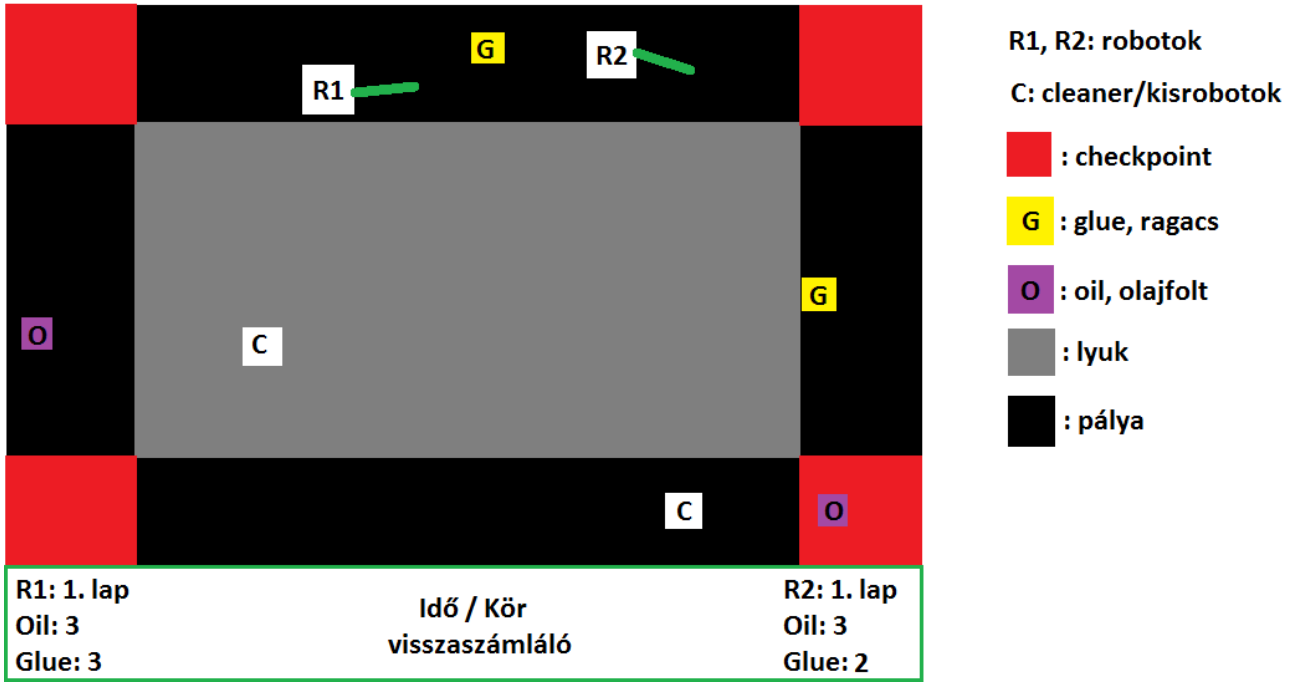


10.1. ábra. GUI

Kétféle játékmód közül választhat :

- Idő játékmód: megadott ideig versenyeznek a robotok
- Kör játékmód: megadott számú kör elérésig versenyeznek a robotok

A játék grafikus felülete:



10.2. ábra. Pálya szerkezete

A játék egy adott pillanata van felvázolva.

A játéktért később egy háttérkép fogja alkotni, melyen egy kijelölt téglalap alakú pályán fognak ugrálni a robotok.
A robotoknak két kép fog megfelelni:

- ugrás előtt: ülő/nyugvó béka
- ugráskor: ugró béka

A többi elemnek (kisrobotok, akadályok) is meg lesz a saját ikonja.
A játékos a robotok irányának meghatározásakor megjelenik egy zöld nyíl, amely segítségével be tudja állítani az ugrás irányát.
A robotok ragacs- és olajkészletei a pálya alatti alsó sávban vannak feltűntetve. Amennyiben az egyik robot akadályt helyezz le a pályára, azt az ugrás követően a készleteket frissíti a HUD.
A robotoknak az úton kell maradniuk és a checkpointokba bele kel ugraniuk, hogy körbe haladjanak a pályán.
A lyukba ugráskor a robot meghal, a cleaner nem. A robot meghalásakor egy robbanásos animáció jön létre.
Az alsó sávban megjelenik a HUD, amelyben a robotok adatai vannak kiírva, illetve a játékmód választásának függvényében egy visszaszámláló.

A verseny végén a győztes robot nevét kiírja a program és ezzel bezárul.

10.2. A grafikus rendszer architektúrája

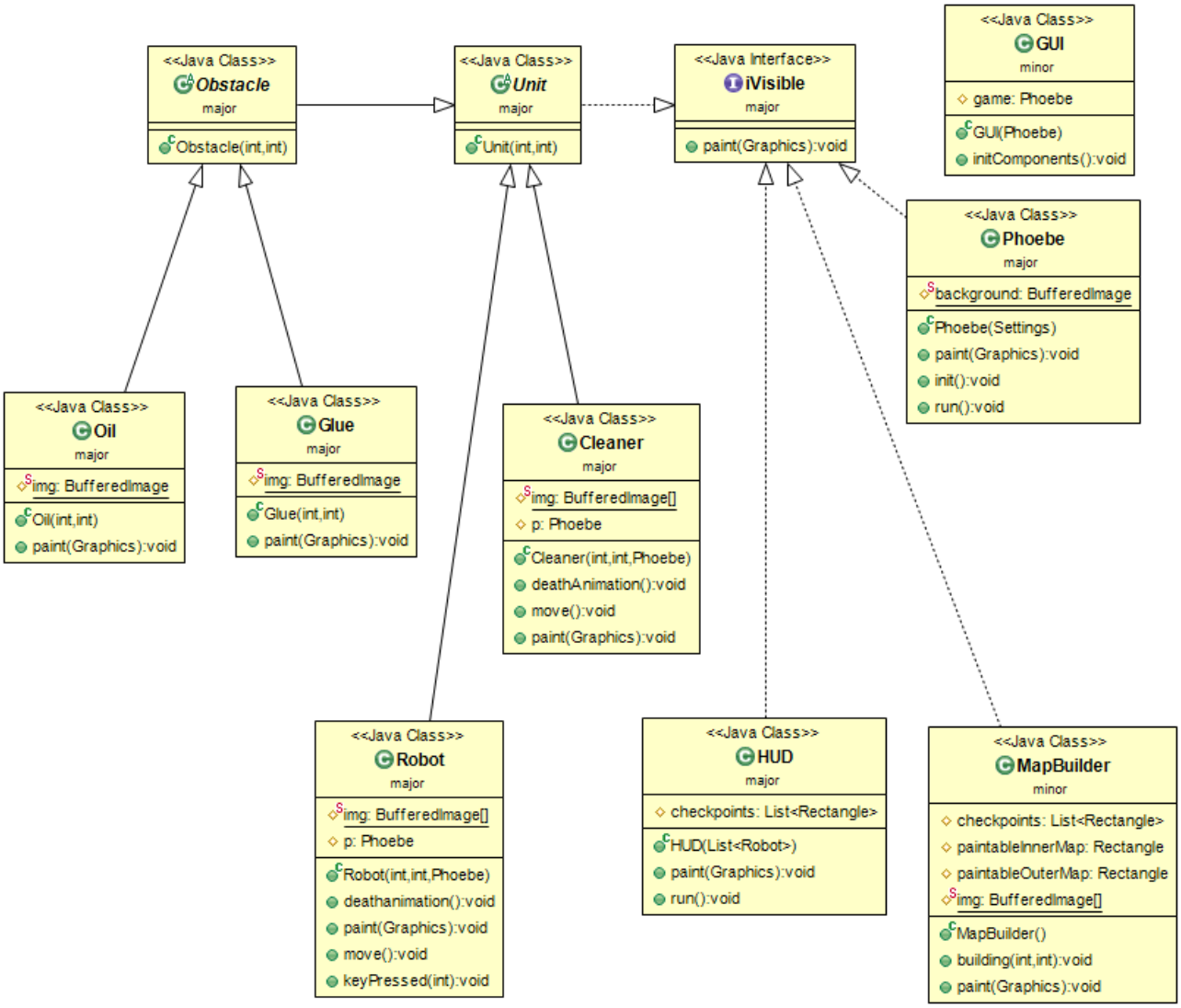
10.2.1. A felület működési elve

A grafikus felületet a java swing segítségével készítettük el. Két részből és a kezdeti menüből áll.

Az első rész magának a játéknak a megjelenítéséért felel. Minden, a képernyőn megjelenítendő objektumnak van egy iVisible interfésze. A Phoebe osztály ezeknek az objektumoknak a Paint függvényét hívja meg sorban kirajzolásnál, a saját paint függvényében, ami akkor hívódik ha invalidálják(repaintet hívnak) a képet. A megjelenítés push alapú. Ha a játékban történik bármi olyan változás, ami a grafikus felületre is kihathat, akkor az aktuális függvény meghívja a Phoebe repaint metódusát, ami ennek hatására újra rajzoltatja a grafikus felületet. A újra rajzolást az alábbi függvények kérhetik: Robot: deathAnimation, move, keyPressed, Cleaner: deathAnimation, move. A kirajzolás a swing Graphics függvényeire épül. Minden iVisible osztályhoz tartozik statikus BufferedImage változó ami paint hívására kirajzolódik az aktuális objektum x,y koordinátáin az objektumhoz tartozó szélességi és magassági értékkel.

A második rész: a HUD megjelenítéséért felel. Ez a frame alsó sávját foglalja el és a játékosok számára írja ki a játék aktuális információit: robotok olaj/ragacs tartalékát, az eltelt időt, továbbá a megtett körök számát. Ez a rész pull alapú, a változásokat bizonyos időközönként mintavételezi (1 mp) és megjeleníti. A menü a játékmód kiválasztását és a játék indítását végzi.

10.2.2. A felület osztály-struktúrája



10.3. ábra. Grafikus osztály diagramm

10.3. A grafikus objektumok felsorolása

10.3.1. Cleaner

- Ősosztályok
  Unit
- Attribútumok
  - # **BufferedImage** img[]: A kis robotok képeit tartalmazza, az animáció miatt többet.
- Metódusok
  - + void **deathAnimation**(): A kis robot halálának animációja.
  - - void **clean**(Obstacle obst): A feltakarított akadály törlése az akadályok listájából.
  - + void **paint**(Graphics g): Kirajzolja a kis robot képét az x,y koordinátákon.
  - + void **die**(): Egy olajat hoz létre a robot halálakor, majd meghívja a deathAnimation()-t.
  - + void **setUnitImage**(): Beállítja a kis robot osztályhoz tartozó képet a user directoryban található cleaner.jpg-re

10.3.2. GUI

- Metódusok
  - + **GUI**(): Konstruktor. Beállítja az ablak nevét, létrehozza az ablak elemeit, elrendezi őket és beállítja a figyelőket(ActionListener).

10.3.3. HUD

- Interfészek:
  - Runnable
  - iVisible
- Attribútumok
  - - **List<Rectangle>** checkpoints: Tárolja a checkpointokat reprezentáló téglalapokat List adatszerkezetben.
- Metódusok
  - + void **paint**(Graphics g): Rajzolást elvégző metódus.

10.3.4. iVisible

- Felelősség
  A grafikus motorhoz szükséges interfész. Olyan osztályok, melyek kirajzolható elemeket tartalmaznak megvalósítják ezt az interfészt.
- Metódusok
  - + void **paint**(Graphics g): Rajzolást elvégző metódus.

10.3.5. MapBuilder

- Felelősség
  A pálya felépítéséért, a checkpointok tárolásáért és a robot pályán tartózkodásának vizsgálatáért felelős osztály.
- Interfészek
  iVisible
- Attribútumok
  - + **List<Rectangle>** checkpoints: Tárolja a checkpointokat reprezentáló téglalapokat List adatszerkezetben.
  - - **Area** map: A pályát reprezentáló objektum.
  - + **Rectangle** paintableInnerMap: A pálya belső területe.
  - + **Rectangle** paintableOuterMap: A pálya külső területe.
  - # **BufferedImage** img[]: A pályán található elemek, checkpointok képei.
- Metódusok
  - + **MapBuilder**(): Konstruktor, a pálya beolvas3ása fájlból, majd létrehozása.
  - + void **building**(int windowWidth, int windowHeight): Felépíti a pályán található téglalapokat. Ez a függvény nem hívódik meg a játék közben, csak akkor van rá szükség, ha a pálya szélességén, hosszúságán akarunk változtatni.
  - + void **paint**(Graphics g): Rajzolást elvégző metódus.

10.3.6. Obstacle

- Felelősség
  A pályán/játékosoknál lévő különböző akadályokat (ragacs,olaj) összefogó ősosztály.
- 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): meghívja a Unit konstruktorát a megadott adatokkal és létrehoz egy négyzet elemet ami reprezentálja a pályán majd.

10.3.7. Glue

- Attribútumok
  - # **BufferedImage** img: Ez a statikus atribútum a ragacs képét tárolja, a megjelenítésben van szerepe.
- Metódusok
  - + void **paint**(Graphics g):Kirajzolja a ragacs képét az x,y koordinátákon.
  - + void **setUnitImage**():Beállítja a ragacs osztályhoz tartozó képet a user directoryban található glue.jpg-re

10.3.8. Oil

- Attribútumok
  - # **BufferedImage** img: Ez a statikus atribútum az olaj képét tárolja, a megjelenítésben van szerepe.
- Metódusok
  - + void **paint**(Graphics g):Kirajzolja az olaj képét az x,y koordinátákon.
  - + void **setUnitImage**():Beállítja az olaj osztályhoz tartozó képet a user directoryban található oil.jpg-re

10.3.9. Phoebe

- Interfészek:
  - Runnable
  - iVisible
- Attribútumok
  - # **BufferedImage** background: A játék hátterét adó kép.
- Metódusok
  - + **Phoebe**(Setting set): A játék felépítése, a robotok, a tisztogató kisrobotok és az akadályok listáinak létrehozása.Az ended inicializálása, az init függvény meghívása és a grafikus felület felépítése történik itt.
  - + void **paint**(Graphics g): Kirajzolja a játék aktuális állását.
  - - void **init**(): Inicializálja a játékot a kezdeti beállításokra. Létrehozza az időzítőt, a mapot, a robotokat a kezdőpozíciók szerint, a hudot és beállítja a különböző osztályokhoz tartozó statikus képeket, továbbá a pálya alap ragacsait és olajait is szétszórja.

10.3.10. Robot

- Attribútumok
  - - **final int** ANIMATIONSPEED: Az ugrás animálásának részletessége(hányszor hívja meg a paintet).
  - # **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.
  - # **BufferedImage** img[]:A robotok képeit tartalmazza, az animáció miatt többet.
  - + **int** arrowendx: A robot irányítását segítő nyilnak az x koordinátája, a nyíl kirajzolásánál van szerepe.
  - + **int** arrowendy: A robot irányítását segítő nyilnak 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.
- Metódusok
  - + **Robot**(int x,int y,Phoebe p): Létrehoz egy robotot a megadott x,y kordinátákon, inicializálja a tagváltozóit és eltárolja a játékmotor referenciáját.

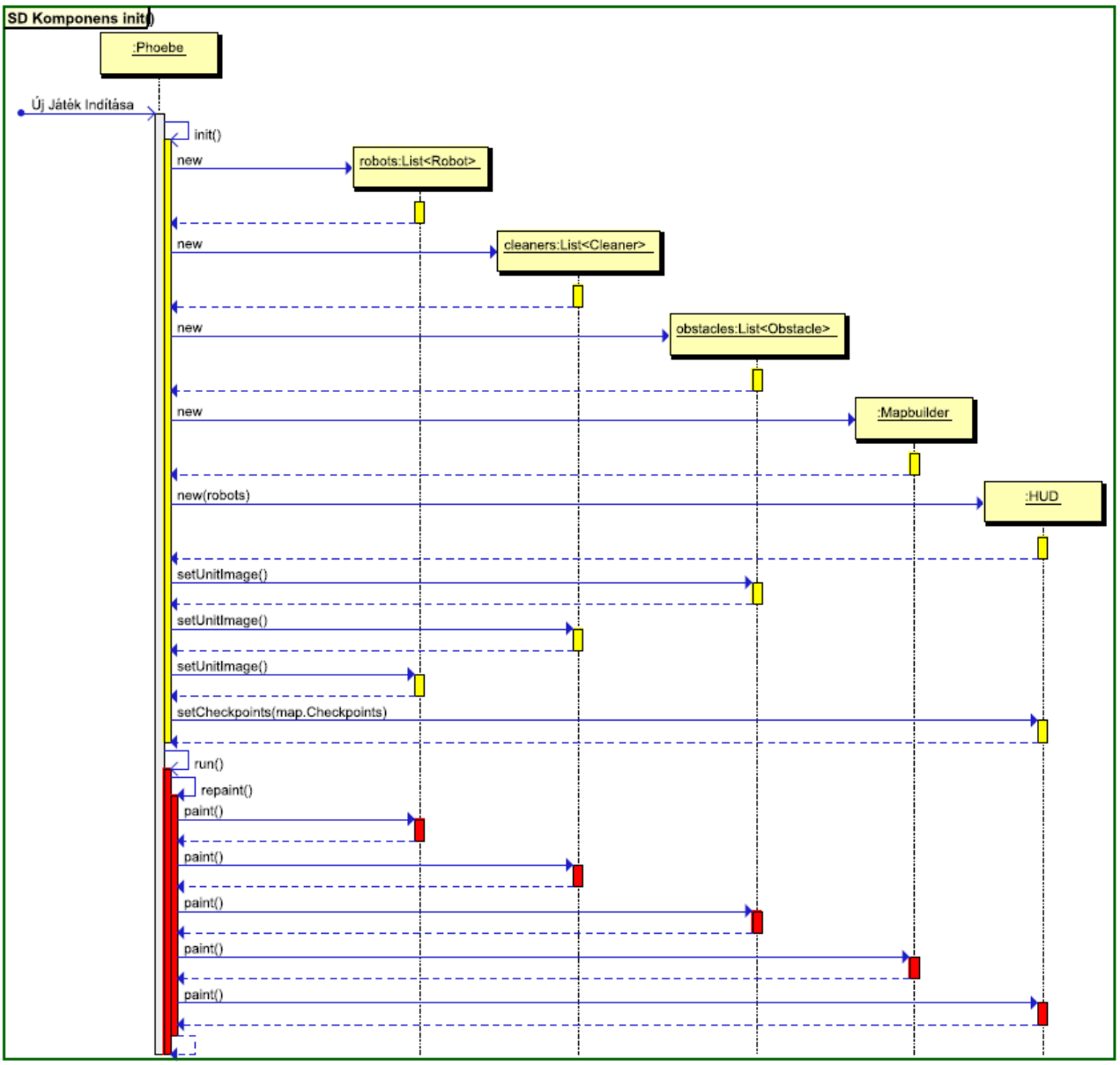
  - + void **deathanimation**():A Robot halálának gafikus megjelenítéséért felelős függvény.
  - + void **paint**(Graphics g):kirajzolja a robotot a saját koordinátáin, ha nem lép éppen akkor az irányítást segítő nyilat is.
  - + void **setUnitImage**():beállítja a robot osztályhoz tartozó képeket.

10.3.11. Unit

- Interfészek
  iVisible
- Attribútumok
  - # **Rectangle** hitbox: Az egységet a pályán reprezentáló téglalap.

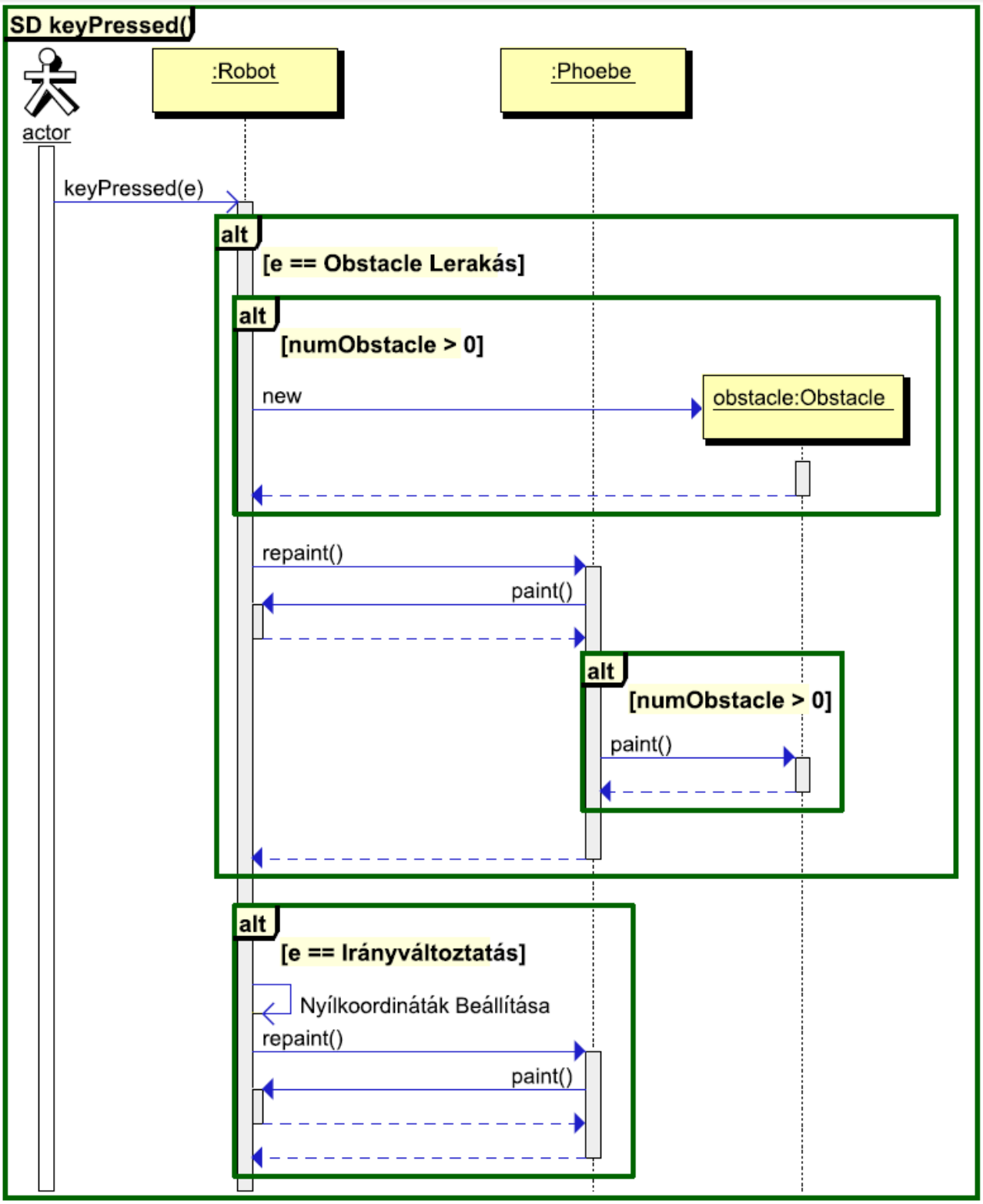
10.4. Kapcsolat az alkalmazói rendszerrel

10.4.1. Komponensek inicializálása



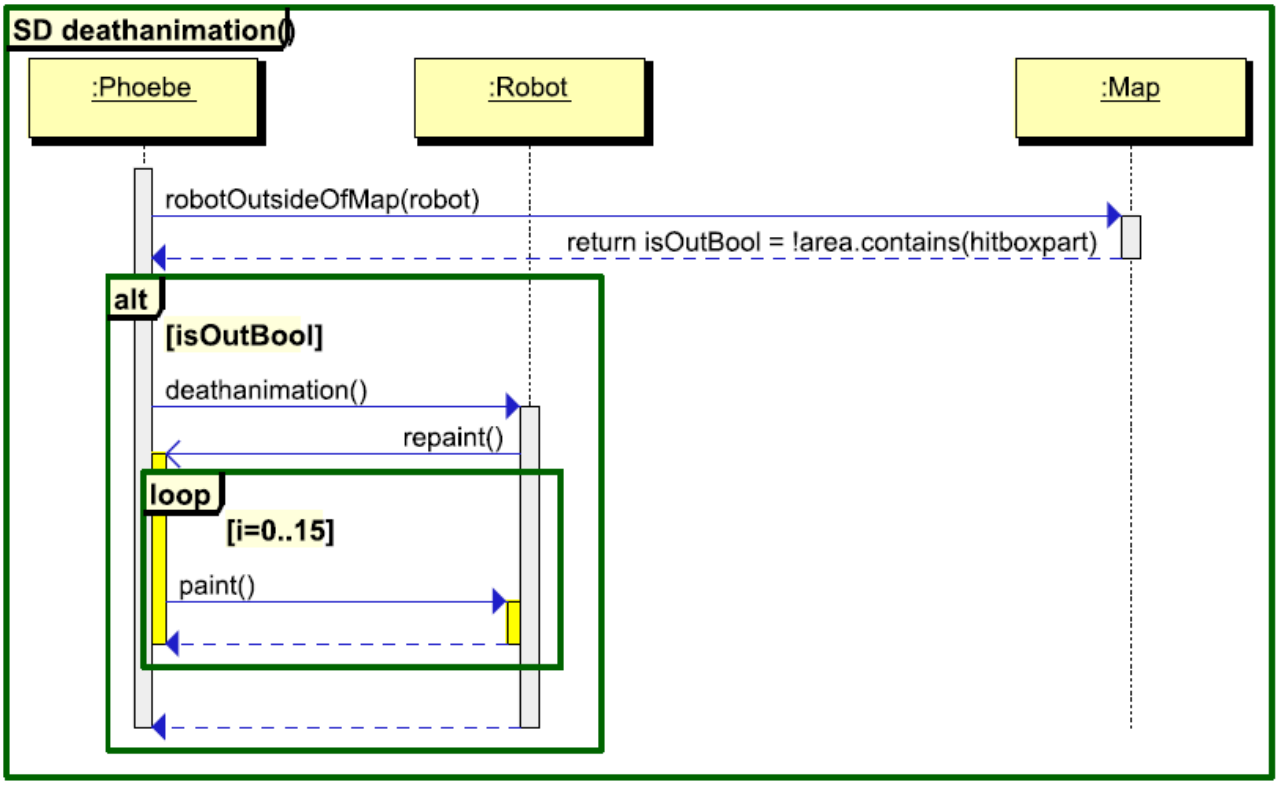
10.4. ábra. Komponensek inicializálásának szekvencia diagramja

10.4.2. Billentyű lenyomása



10.5. ábra. Gomblenyomás szekvencia diagramja

10.4.3. Robot megsemmisülése



10.6. ábra. Robot pályáról való leesésének a szekvencia diagramja

10.5. Napló

Kezdet	Időtartam	Résztvevők	Leírás
2010.03.22. 10:00	2 óra	**Kovács** **Lovász** **Tóth** **Graics** **Magyar**	Konzultáció, részfeladatok kiosztása
2015.04.24. 12:00	2 óra	**Graics**	Szekvencia diagramok elkészítése
2015.04.25. 13:00	3 óra	**Magyar**	A grafikus objektumok felsorolása
2015.04.25. 16:00	1,5 óra	**Tóth**	Grafikus osztály diagram elkészítése, Konzultáció
2015.04.26. 10:00	2 óra	**Lovász**	Grafikus interfész dokumentálása