Kukacfarm

49 – Tetragon

Konzulens:
Simon Balázs

Csapattagok

Deák László	FVI5WQ	dsurfingman jahúú ko uk
Katus Kristóf	HHD9YO	katusmeister gémél kom
Ofella Péter	L9VNPG	fussatok gímél kom
Paulik Tamás	KDH6U3	platina512 t-ónlájn hú

2008. május 14.


team
Tetragon4

Tartalomjegyzék

1. fejezet Team bejelentkezés

A csapat bejelentkezése Tetragon névvel megtörtént a Hercules rendszeren keresztül. Az egyes csapattagok elérhetőségei megtalálhatóak az előlapon. A csapat honlap fórum részében nagyjából nyomon követhető a tevékenységünk (igyekszünk azt használni a levelezés helyett):

- http://platina512.uw.hu/index.php

2. fejezet Követelmény, projekt, funkcionalitás

2.1. Követelmény definíció

2.1.1. A projekt célja

A projekt során egy szabványos Java nyelven írt, kis gépigényű játékot hozunk létre, UML segítségével. A program egy két felhasználó által irányított kígyós játék. A játék részletes leírását a 2.3.-as pontban.

2.1.2. A fejlesztőkörnyezet

A fejlesztéshez NetBeans 6.0-t használunk, ennek UML plugin-jával modellezünk, és ez alapján generáljuk a kód vázát. Verziókövetéshez a CVS és az SVN merült fel, de a többség leszavazta ezek használatát. Teszteléshez a JUnit-ot fogjuk használni. A dokumentumokat Microsoft Word-del készítjük.

2.1.3. A futtatáshoz szükséges környezet

A program várhatóan futni fog egy 466 MHz-es vagy gyorsabb gépen, 512 MB memóriával, amelyen fut a Java Runime Environment. Az élvezetes játékhoz természetesen szükség van egérre, billentyűzetre és grafikus monitorra is. További perifériás eszközökkel fokozható a játékélmény.

2.1.4. A felhasználói felület

A kész program grafikus felhasználói felülettel rendelkezik, amelyet az egér és a billentyűzet segítségével lehet vezérelni.

2.1.5. Minőségi tényezők

Teljesítmény: A cél az, hogy a játék élvezhetően játszható legyen a fentebb meghatározott minimális rendszeren. A grafikus felületnél törekedni fogunk a dupla pufferelésre, és hogy egyszerre két felhasználó is tudjon billentyűzeten játszani.
Újrafelhasználhatóság: A cél az, hogy a grafikus felhasználói felületet a program többi részétől amennyire csak lehet különválasszuk, így lehetővé téve azt, hogy később a grafikus felület egyszerűen és gyorsan változtatható legyen (MVC paradigma; ugyanez igaz a vezérlésre is).
Rugalmasság: A rugalmasságot a fejlesztőkörnyezet biztosítja, a játéknak ugyanis minden olyan PC környezetben futtathatónak kell lennie, melyben létezik megfelelő Java futtatókörnyezet.

Felhasználhatóság: A program használatához a felhasználói kézikönyv elolvasása nem szükséges. Bárki könnyedén tudja majd használni aki valaha játszott már a hagyományos kígyó játékkal. Az egyes gomboknak, menüpontoknak beszédes neveket adunk.

2.1.6. A szoftver minősítése

A szoftvernek teljesítenie kell a fent leírt követelményeket. Ennek biztosítására szigorúan a modell alapján implementáljuk a programot, és folyamatosan teszteljük.

2.1.7. A terjesztés

A kész programot a laborvezető fogja megkapni. A program a későbbiekben interneten lesz elérhető korlátozott ideig a projektünk menedzselésére használt honlapon: a http://platina512.uw.hu címen.

2.2. Projekt terv

2.2.1. A fejlesztői csapat

Név	*Feladatkör*
Katus Kristóf	csapatvezető; változó
Deák László	változó
Ofella Péter	változó
Paulik Tamás	honlap üzemeltetése; változó

2.2.2. Életciklus modell

Legelőször a modellt kell megtervezni, amely elsősorban az objektum és a dinamikus modelleket foglalja magában. Ez alapján a szkeletont lehet elkészíteni. Itt az objektumok kapcsolatainak helyes megvalósítását kell mindenre kiterjedően ellenőrizni.

Ezután következik a prototípus implementálása. Ezen a változaton vizsgáljuk az implementáció helyességét, ezért biztosítani kell a könnyű tesztelhetőséget.

A végső fázis a grafikus változat elkészítése. Itt a grafikus interfésznek a prototípushoz való helyes illeszkedését kell tesztelni.

2.2.3. Fejlesztési ütemterv

A fejlesztésnek három fő állomása van:
Szkeleton: minden objektum szerepel benne, de csak az interfészük definiált.
Prototípus: a grafikus felület nélküli, helyesen működő program, az összes megvalósított metódussal.
Grafikus: végleges program grafikus felülettel ellátva, csak a felhasználói interfészt illető változásokat tartalmaz.

2.2.4. Szervezési struktúra

A csapat 4 főből áll. A döntéseket közös megegyezés alapján, a csapatvezető jóváhagyásával hozzuk meg. Az adott heti feladaton előreláthatólag mindig az a két ember fog dolgozni, aki a legkevésbé elfoglalt. Az elkészült anyagokat közösen ellenőrizzük.

A fejlesztés során a kommunikációra és közös adattárolásra a következő eszközöket alkalmazzuk:

- *e-mail:* csatolt fájlok küldésére.
- *MSN:* a konzultáció után felmerült kérdések megvitatására.

- *FTP:* a projekt forrásfájljainak tárolására.
- *Fórum:* a naplók vezetésére, találkozók megbeszélésére és közös anyagok megőrzésére.

2.2.5. Határidők

febr. 15.	A csapat bejelentkezése
febr. 20.	A követelmény, a projekt, és a funkcionalitás leírása
febr. 27.	Az analízis modell kidolgozása 1.
márc. 5.	Az analízis modell kidolgozása 2.
márc. 12.	A szkeleton tervezése
márc. 19.	A szkeleton beadása
márc. 26.	A prototípus koncepciója
ápr. 2.	Részletes tervek kidolgozása
ápr. 16.	A prototípus beadása
ápr. 23.	A grafikus felület specifikációja
máj. 7.	A grafikus változat beadása
máj. 14.	Összefoglalás

2.2.6. Kockázatok

Megnevezés	*Valószínűség*	*Hatás*
csapattag kilépése	alacsony	közepesen súlyos
hardware hiba	közepes	nagyon súlyos
követelmény megváltozása	biztos	enyhe
késés	alacsony	közepesen súlyos
csapattag megbetegedése	alacsony	enyhe
kommunikációs csat. hiba	alacsony	enyhe

2.2.7. Szükséges dokumentációk

A fejlesztési dokumentáció alapján a projekt áttekinthetővé válik más fejlesztők számára is, ezen kívül a továbbfejlesztést és újrafelhasználást is jelentősen támogatja. A felhasználói dokumentáció a felhasználó számára tartalmazza a telepítési és használati útmutatásokat.

2.3. A követelmények részletes leírása

A játék a hagyományos kígyó játék egy-gépen-multiplayeres megvalósított változata. A kígyós játékot egyszerre két játékos tudja játszani. Mindkét játékos egy-egy kígyót irányít.

A program indításakor a játékosok egy felhasználói menüvel találkoznak először. Ebben a menüben a következő menüpontok közül választhatnak: új játék, help. Az új játék gomb megnyomása új játék elindulását eredményezi. A help menüpont alatt segítséget kaphatnak a játék használatához (felhasználó számára készített dokumentáció) és a készítők névlistáját és a program névjegyét tekinthetik meg.

A játék indítása a játéktér megjelenítését, a kígyók létrehozását, és a bogyó(k) megjelenését eredményezi. A pályát falak határolják. A falak elhelyezkedése olyan, hogy azok megakadályozzák a játéktér elhagyását. A pályán két kígyó van egyszerre. A játék indításakor mindkét kígyó egyforma, 3 testelem hosszú, amiből a legelső testelem a fej. A kígyók a pálya vízszintes középvonalával párhuzamosan, azzal egybeesve helyezkednek el, és ellentétes irányban állnak. A játék indításának pillanatában a kígyók e két irányban automatikusan elkezdenek mozogni. A mozgás sebessége megmarad, a játékosok mindössze

az irányát képesek változtatni. A kígyók két irányba kanyarodhatnak. Mindkét kígyóhoz tartozik egy-egy pontszámláló, amely mutatja a játékosok által összegyűjtött pontokat. A pontszámlálók alapértelmezett értéke 2. A játék indításának pillanatában legalább egy, de legfeljebb 3 bogyó jelenik meg a pálya területén random helyen. A kígyók egymástól megkülönböztethetőek.

A játék menete során a játékosok pontokat gyűjthetnek, és növelhetik kígyójuk méretét. Minden egyes bogyó bekapása növeli a testük hosszát, és a pontjaik számát. A játékosok pontjai, mindig az aktuális kígyójuk testelemeinek összességét reprezentálják numerikus formában. Egy testelem +1 ponttal ekvivalens. A kígyók 3 féle bogyót kaphatnak be. A bogyók bekapása azoknak az eltűnését jelenti. A pályán egyszerre legalább egy, de legfeljebb 3 (de akár azonos típusú) bogyó lehet. Egy bogyó random helyen jelenik meg a pályán, de nem jelenhet meg a kígyókkal fedésben lévő területen. Létezésük véges ideig tart. Ha egy bogyót nem kapnak be, akkor az eltűnhet, de helyette új jelenik meg. A legegyszerűbb, a mezei bogyó, amitől egységnyivel nő a testhosszuk, vagyis eggyel nő a testelemeik száma. Ha a kígyó fűrészbogyót kap be, akkor rövid ideig olyan erős lesz a harapása, hogy mikor másik kígyónak (vagy saját magának) nekimegy, áthalad rajta, levágva az ellenfél farkát. Ilyenkor a levágott darab megsemmisül, az érintett kígyó rövidebb lesz, de nem pusztul el. Ha az ellenfél kígyó fejét harapja meg fűrészbogyóhatás alatt, akkor a kígyó egységnyi fej hosszúságú lesz, vagyis a testelemeinek a száma 1 lesz (pontjai száma 0; a pontok száma csak természetes szám lehet). A fűrészbogyó ellen csak a kőbogyó hatékony, amely a kígyó testén infinitezimálisan rövid idő alatt végigvándorolva az érintett testszakasz fenti módon leírt megharapását akadályozza meg. A köves részre harapó kígyó meghal. A kőbogyók a kígyó testéből nem távoznak, hanem sorban, a farok végétől kezdve felhalmozódnak. A játékban kígyók küzdenek egymás ellen a győzelemért. A szabad mozgásukat falakkal való ütközés, a többi kígyó megharapása, és saját testük megharapása akadályozza.

A játék vége. A játék akkor ér véget, ha legalább az egyik kígyó meghal, vagy ha a két kígyó egymás fejét egyszerre harapja meg. Ha mindkét kígyó egyszerre hal meg (pl.: egyszerre ütköznek falnak) vagy egymás fejét harapják meg, akkor játékot az a kígyó nyeri, amelyik a játék végén hosszabb testtel rendelkezik. Egyenlő testméret esetén az aktuálisan kevesebb kőbogyót elfogyasztó kígyó nyer. Ha azonban csak az egyik kígyó hal meg, akkor a másik játékos nyer. Egy kígyó halálát több élettörténeti esemény is okozhatja: ha az egyik kígyó megharapja a másik kígyó farkát, vagy saját farkát, de nincs fűrészbogyóhatás, akkor a harapó kígyó; ha egy kígyó ütközik a fallal, akkor az ütköző kígyó hal meg. Ilyen esetben az a játékos nyer, akinek a kígyója tovább marad életben.

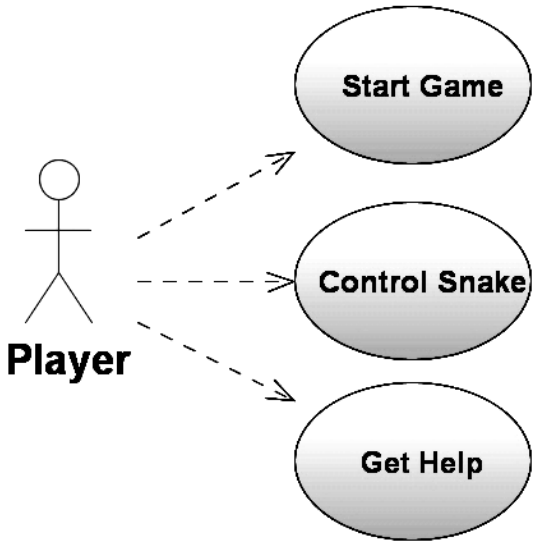
2.4. Szótár

bekap	a kígyó ezt teszi a bogyóval, ha nekimegy.
bogyó	az amit a kígyó bekaphat, alább 3 fajtája ismertetett.
bogyó eltűnik	egy adott bogyó egy bizonyos idő után a pályán eltűnhet.
bogyó megjelenik	amikor a pályán mezei-, fűrész-, kőbogyó jelenik meg.
élettörténet	a kígyóval történt események sorozata.
fal	a játéktér olyan entitása, mellyel való ütközés halált okoz.
farok	a kígyó fejen kívüli testelemeinek összessége.
fej	a kígyó egy speciális testeleme; a mozgás irányában a legelső; ezzel tud harapni, bekapni, levágni.
felhalmozódik	egy adott típusú bogyó a kígyó farkában összegyűlik.

fűrészbogyó	az a bogyó amelytől nő a kígyó hossza, és harapása egy adott ideig megerősödik.
fűrészbogyóhatás	az a tulajdonság, amellyel a fűrészbogyót bekapó kígyó rendelkezik.
harap	a kígyó ezt teszi, ha egy másik kígyónak nekimegy, a harapó kígyó meghal.
hossz	a kígyó testelemeinek összessége.
irányít	a játékos változtatja a kígyó irányát.
játék indítása	az egyik játékos elindítja a játékot.
játék menete	a játék indítása és a játék vége közti időszak.
játék vége	ha két kígyó feje egymásra harap, vagy egy kígyó meghal a játéknak vége.
játékos1	az a személy, aki az egyik kígyót irányítja
játékos2	az a személy, aki a másik kígyót irányítja
játéktér	az a képernyőrész, ahol a játékosok játszanak.
kígyó	a játékosok által irányított testelemekből álló azonosítható entitás.
kőbogyó	a kígyó testén végigvándorolva az érintett testszakasz megharapását akadályozza meg, a köves részre harapó kígyót elpusztítva. A kőbogyók a kígyó testéből nem távoznak, hanem sorban, a farok végétől kezdve felhalmozódnak.
levág	a kígyó ezt teszi, ha egy másik kígyónak nekimegy, és fűrészbogyóhatással rendelkezik.
meghal	amikor a játék véget ért, az egyik játékos nyer.
megsemmisül	ha egy kígyó farkát levágják, akkor az a része eltűnik.
menü	a játékosok által látott képernyő a játék vége és indítása között, vagyis amikor nem a játék menete zajlik.
mezei bogyó	az a bogyó amelytől nő a kígyó hossza.
mozgás sebessége	a mozgásnak van egy minimális sebessége, melyet csak az opciók menüben lehet megváltoztatni.
multiplayer	a játékot egyszerre több felhasználó játszhatja egy számítógépen.
nő	a kígyók hossza bizonyos bogyók bekapásától nő.
pálya	a játéktér olyan entitása, amelyen nincs fal.
pont	a pontszámláló által kijelzett decimális szám, mely a kígyó hosszának, és élettörténetének függvénye.
pontszámláló	a játékosok kígyóinak pontjait mutató játéktérbeli decimális számokat kijelző entitás.
random hely	a pálya egy véletlenszerűen választott helye.
testelem	a kígyó alapvető entitása; a kígyó egy testelemmel hosszabbodik meg, ha bekap egy bogyót.
ütközik	ha a kígyó a falnak nekimegy, akkor a kígyó meghal.

2.5. Essential use-case-ek és diagramjaik

2.5.1. Diagramok



2.5.2. Use Case leírások

Use Case: Start Game
Actor: Player
Leírás: A játék elindítása

Use Case: Control Snake
Actor: Player
Leírás: A kígyó irányítása

Use Case: Get Help
Actor: Player
Leírás: Segítség kérése a játékhoz

2.6. Forráskód formátum

```
/*
Fájl neve: code_format.java
Csapat: Tetragon Team
Készítő: Paulik Tamás (KDH6U3)
Kezdés: 2008.02.17.
Befejezés : 2008.02.17.
A fájl tartalma: Ebben a részben szerepeljen a fájl tartalmának rövid leírása,
      hogy könnyebben be lehessen azonosítani, hogy mi mire jó. A továbbiakban
      a sor végi kommentekben fogom jelölni, hogy mi az amire kódformátum
      szempontjából figyeljünk oda. Általánosan ügyeljünk arra, hogy egy sor
      ne legyen szélesebb egy A4-es lapnál. Nyilvan mindezeken felül legyenek
      beszédesek a változónevek és a működést magyarázó kommentek se
      maradjanak le.
Verzió: 1.0
*/

import java.lang.*; // importok előtt legyen egy üres sor
import java.lang.*;
import java.lang.*;
```

```
import java.lang.*; //importok után legyen egy üres sor

class Format { // A blokknyitó { a fejléccel egy sorban
	/* Alapvetően a valamely fejléchez tartozó blokk teljes egészében egy
	TAB-bal legyen beljebb, mint a fejléc, a záró } meg a fejléc kezdeti
	vonala alatt */
	int a, b = 0, c = b;
	char d, e, f; /* az azonos típusú változók egy sorban, ha nem férnek ki,
			akkor az új sor elején mindenképp új int,
			char stb. megnevezés */
		/* Kerüljük a helyileg deklarált változókat, kivéve ciklusok
		fejlécében, ha mégis kényelmesebb lenne valahol helyi deklarációt
		használnunk, azokat a változókat itt mindenképpen jelezzük */
	/*
	Lokákisan deklarált változók:
	int a; 15.sor
	char b; 22.sor
	*/ // tehát a lokálisan deklarált változókat egy kommentben jelezzük,
	   // a többivel egy helyen a deklarálás helyét megnevezve
	if (cond) {
		for(;;) {
			// lépések
		}
	}
	else {
		switch (cond) {
			case 1:
				// lépések
				break;
			case 2:
				// lépések
				break;
			case 3:
				// lépések
				break;
		}
	}
} // Format vége
	/* A blokkzáró külön sorban, megnevezve, hogy minek a vége így
	elkerülhető egy esetleges copy-paste esetén az, hogy lemaradjon a blokk
	vége */

class Format2 { // Osztályok között is legyen egy sor szünet
	// ...
} // Format2 vége
// code_format.java vége
/* A blokkvége komment mintájára fájlvége komment */
```

2.7. Felmerült eszközök

2.7.1. Integrált fejlesztő környezetek

NetBeans IDE 6.0: ingyenes, a Sun által támogatott környezet; tengernyi plugin-t lehet hozzá letölteni, többek között UML ábrák szerkesztéséhez, verziókövetéshez (CVS, SVN), teszteléshez (JUnit):

- http://www.netbeans.org/features/uml/index.html
- http://www.netbeans.org/features/ide/collaboration.html
- http://junit.netbeans.org/

Visual Studio: robosztus fejlesztőkörnyezet nagyvállalati igényekre szabva; az MSDN által jelentős helppel és supporttal rendelkezik, sok felhasználóbarát funkcióval. Az UML rajzolás viszont nem teljesen szabványos, ahogyan a rendelkezésre álló J# nyelv sem teljesen kompatibilis a Javával; cserébe gyors kódot fordít; az UML és a kód oda-vissza dinamikusan frissül
Eclipse: az IBM által támogatott nyílt forráskódú fejlesztő platform; különös hangsúlyt fektet a nagyfokú bővíthetőségre; még lassabb, mint a NetBeans

2.7.2. Kapcsolattartás, weboldal

VIK wiki-n létrehozható saját oldal; erre egy példa:

- https://wiki.sch.bme.hu/bin/view/Sandbox/KreativeSzglab4

A nyílt forráskódú szoftver projektek weboldala (CVS-t kezel):

- http://*sourceforge*.net/

Webes tartalomkezelő rendszer:

- http://*drupal*.hu/

Szintén egy nyílt forráskódú tartalomkezelő rendszer:

- http://www.*php-nuke*.hu/

Kifejezetten kollaborációhoz találták ki (SVN, Trac, Wiki, Task management stb.):

- http://*www.assembla.com/*

Google & egyéb:

- http://docs.google.com/
- http://www.google.com/calendar/
- http://www.google.com/notebook/
- http://code.google.com/opensource/
- http://*twiki*.org/
- https://*launchpad*.net/

2.7.3. UML és egyéb ábrák

StarUML

- http://www.staruml.com/

ArgoUML

- http://argouml.tigris.org/

További *UML eszközök* listája:

- http://en.wikipedia.org/wiki/List_of_UML_tools
- http://en.wikipedia.org/wiki/Category:Free_UML_tools

FreeMind: ingyenes mind mapping szoftver

- http://freemind.sourceforge.net/

Visio: a Microsoft Office család része, diagramrajzoló szoftver

- http://office.microsoft.com/hu-hu/visio/default.aspx

dia: ingyenes diagramrajzoló szoftver (UML-t is tud)

- http://live.gnome.org/Dia

2.7.4. Projekt menedzsment; tervezés, ütemezés és naplóvezetés

Open Workbench: nyílt forráskodú projekt ütemező program Windows platformra

- http://www.openworkbench.org/

További *projekt menedzsment eszközök*:

- http://en.wikipedia.org/wiki/Project_management_software
- http://en.wikipedia.org/wiki/List_of_project_management_software

3. fejezet Analízis modell kidolgozása I.

- A javított analízis modell a következő fejezetben található meg.

4. fejezet Analízis modell kidolgozása II.

4.1. Objektumlista és osztályok leírása

4.1.1. Class Alapelem

Az alapelem az objektumrendszer egy alapvető, absztrakt eleme, amelyből minden olyan osztály leszármazik, amely részt vesz a játéktér kialakításában. Minden elemnek lehet helye mérete és azonosítója.

All Known Subclasses:

Szelveny, Tegla, Bogyo

All Enclosing Diagrams:

 Kukac_jatek_osztalydiagram

---

public class **Alapelem**

Properties	
Alias	Alapelem
Visibility	public
Final	☐
Transient	☑
Abstract	☑
Leaf	☐

Attribute Summary	
Pozicio	**pozicio** minden elem pozícióját tartalmazza egy pozíció objektum
Dimension	**meret** az alapelem méretei

Constructor Summary
Alapelem(Pozicio pozicio, Dimension meret)

Operation Summary	
Pozicio	**getPozicio**() visszaadja az alapelem helyét
void	**setPozicio**(Pozicio val) beállítja az alapelem helyét
Dimension	**getMeret**() visszaadja az alapelem méreteit
void	**setMeret**(Dimension val) beállítja az alapelem méretei
void	**megharapva**(Alapelem a) látogató modell első hívott fv.-e, kreál egy látogatót

4.1.2. Class Bogyo

A bogyó egy absztrakt osztály, belőle származnak le az egyes bogyófajták. A bogyó osztály az alapelemből származik le.

Alapelem
└ **Bogyo**

All Known Subclasses:

KoBogyo, FureszBogyo, FoldiBogyo

All Known Associations:

Composition: Bokor

All Enclosing Diagrams:

Kukac_jatek_osztalydiagram

---

public class **Bogyo**
extends Alapelem

Properties	
Alias	Bogyo
Visibility	public
Final	☐
Transient	☑
Abstract	☑
Leaf	☐

Attribute Summary	
Bokor	**bokra** a bogyókat a bokor objektum tárolja és kezeli, ezért a bogyók ismerik a bokrot

Constructor Summary
Bogyo(Pozicio pozicio, Dimension meret, Bokor bokra)

Operation Summary	
void	**megharapva**(Alapelem a) látogató modell első hívott fv.-e, kreál egy látogatót
Bokor	**getBokra**() visszaadja, hogy a bogyó melyik bokor objektumhoz tartozik
void	**setBokra**(Bokor val) beállítja, hogy a bogyó melyik bokorhoz tartozzon
void	**visszaharap**(Visszaharapas visszaharapo) a látogató modellben a látogató átadására szolgáló fv.

4.1.3. Class Bokor

A bokor az az osztály, amely egy listát tartalmaz a pályán jelenlévő bogyókról. Tárolja, kezeli és menedzseli a pályán lévő bogyókat. Felel a bogyók létrehozásáért és törléséért. Egy példány létezik belőle.

Bokor
All Known Associations:

Composition: Bogyo
Composition: Motor

All Enclosing Diagrams:

Kukac_jatek_osztalydiagram

---

public class **Bokor**

Properties	
Alias	Bokor
Visibility	public
Final	☐
Transient	☑
Abstract	☐
Leaf	☐

Attribute Summary	
int	**bogyokMaxSzama** ahány bogyót egyszerre kezelni kell
Bogyo	**bogyok**[*] bogyók listája
int	**idoHatraAKovetkezoBogyoMeghalasaig** hátralévő idő a következő bogyó meghalásáig

Constructor Summary	
Bokor()	

Operation Summary	
int	**getBogyokMaxSzama**(int hanybogyo) visszaadja az egyszerre kezelendő maximális bogyók számát
void	**setBogyokMaxSzama**(int val) beállítja az egyszerre kezelendő maximális bogyók számát
int	**getIdoHatraAKovetkezoBogyoMeghalasaig**()visszaadja a hátralévő időt a következő bogyó meghalásáig
void	**setIdoHatraAKovetkezoBogyoMeghalasaig**(int val) beállítja a hátralévő időt a következő bogyó meghalásáig
void	**leptet**() ellenőrzi a pályán lévő bogyókat, végrehajtja rajtuk a kívánt műveleteket
void	**torolBogyo**(Bogyo b) töröl egy bogyót a pályán

4.1.4. Class FoldiBogyo

A bogyó osztály leszármazottja, a játékspecifikációban leírt mezei bogyót valósítja meg.

Bogyo
└ **FoldiBogyo**

All Enclosing Diagrams:

 Kukac_jatek_osztalydiagram

---

public class **FoldiBogyo**
extends Bogyo

Properties	
Alias	FoldiBogyo
Visibility	public
Final	☐
Transient	☑
Abstract	☐
Leaf	☐

Constructor Summary
FoldiBogyo(Pozicio pozicio, Dimension meret, Bokor bokra)

Operation Summary	
void	**megharapva**(Alapelem a) látogató modell első hívott fv.-e, kreál egy látogatót
void	**visszaharap**(Visszaharapas visszaharapo) a látogató modellben a látogató átadására szolgáló fv.

4.1.5. Class FoldiBogyoVisszaharapas

Ha a megharapott alapelem földibogyó, akkor az földibogyóvisszaharapás objektumot hoz létre, és átadja azt a kölcsönhatásban résztvevő félnek, a látogató modell szerint. A visszaharapás interface-t valósítja meg.

FoldiBogyoVisszaharapas
All Implemented Interfaces:
Visszaharapas
All Dependency Suppliers:
Implementation Visszaharapas
All Enclosing Diagrams:
Kukac_jatek_osztalydiagram

---

public class **FoldiBogyoVisszaharapas**
implements Visszaharapas

Properties	
Alias	FoldiBogyoVisszaharapas
Visibility	public
Final	☐
Transient	☑
Abstract	☐
Leaf	☐

Attribute Summary	
FoldiBogyo	**fb** az objektumot létrehozó objektumra hivatkozás

Constructor Summary
FoldiBogyoVisszaharapas(FoldiBogyo fb) paramétere a létrehozó objektum

Operation Summary	
FoldiBogyo	**getFb**() visszaadja az őt létrehozó objektumot
void	**setFb**(FoldiBogyo val) beállítja, hogy melyik objektumra hivatkozzon készítőként
void	**SzelvenyHarap**(Szelveny sz) ha szelvénnyel ütközik, akkor ezt a fv.-t hívja meg a szelvény
void	**TeglaHarap**(Tegla t) ha téglával ütközik, akkor ezt a fv.-t hívja meg a tégla
void	**FoldiBogyoHarap**(FoldiBogyo fb) ha földibogyóval ütközik, akkor ezt a fv.-t hívja meg a földibogyó
void	**KoBogyoHarap**(KoBogyo kb) ha kőbogyóval ütközik, akkor ezt a fv.-t hívja meg a kőbogyó
void	**FureszBogyoHarap**(FureszBogyo fb) ha fűrészbogyóval ütközik, akkor ezt a fv.-t hívja meg a fűrészbogyó

4.1.6. Class FureszBogyo

A bogyó osztály leszármazottja, a játékspecifikációban leírt fűrészbogyót valósítja meg.

Bogyo
└ **FureszBogyo**

All Enclosing Diagrams:

 Kukac_jatek_osztalydiagram

---

public class **FureszBogyo**
extends Bogyo

Properties	
Alias	FureszBogyo
Visibility	public
Final	☐
Transient	☑
Abstract	☐
Leaf	☐

Constructor Summary
FureszBogyo(Pozicio pozicio, Dimension meret, Bokor bokra)

Operation Summary	
void	**megharapva**(Alapelem a) látogató modell első hívott fv.-e, kreál egy látogatót
void	**visszaharap**(Visszaharapas visszaharapo) a látogató modellben a látogató átadására szolgáló fv.

4.1.7. Class FureszBogyoVisszaharapas

Ha a megharapott alapelem fűrészbogyó, akkor az fűrészbogyóvisszaharapás objektumot hoz létre, és átadja azt a kölcsönhatásban résztvevő félnek, a látogató modell szerint. A visszaharapás interface-t valósítja meg.

FureszBogyoVisszaharapas
All Implemented Interfaces:

Visszaharapas

All Dependency Suppliers:

Implementation Visszaharapas

All Enclosing Diagrams:

Kukac_jatek_osztalydiagram

---

public class **FureszBogyoVisszaharapas**
implements Visszaharapas

Properties	
Alias	FureszBogyoVisszaharapas
Visibility	public
Final	☐
Transient	☑
Abstract	☐
Leaf	☐

Attribute Summary	
FureszBogyo	**fb** az objektumot létrehozó objektumra hivatkozás

Constructor Summary
FureszBogyoVisszaharapas(FureszBogyo fb) paramétere a létrehozó objektum

Operation Summary	
FureszBogyo	**getFb**() visszaadja az őt létrehozó objektumot
void	**setFb**(FureszBogyo val) beállítja, hogy melyik objektumra hivatkozzon készítőként
void	**SzelvenyHarap**(Szelveny sz) ha szelvénnyel ütközik, akkor ezt a fv.-t hívja meg a szelvény
void	**TeglaHarap**(Tegla t) ha téglával ütközik, akkor ezt a fv.-t hívja meg a tégla
void	**FoldiBogyoHarap**(FoldiBogyo fb) ha földibogyóval ütközik, akkor ezt a fv.-t hívja meg a földibogyó
void	**KoBogyoHarap**(KoBogyo kb) ha kőbogyóval ütközik, akkor ezt a fv.-t hívja meg a kőbogyó
void	**FureszBogyoHarap**(FureszBogyo fb) ha fűrészbogyóval ütközik, akkor ezt a fv.-t hívja meg a fűrészbogyó

4.1.8. Class JatekTerep

A játékterep az az osztály, amely egy listát tartalmaz a pályán jelenlévő fal objektumokról. Tárolja, kezeli és menedzseli őket. Felel a falak létrehozásáért. Egy példány létezik belőle.

JatekTerep
All Known Associations:

Composition: Tegla
Composition: Motor

All Enclosing Diagrams:

Kukac_jatek_osztalydiagram

---

public class **JatekTerep**

Properties	
Alias	JatekTerep
Visibility	public
Final	☐
Transient	☑
Abstract	☐
Leaf	☐

Attribute Summary	
Tegla	**teglak**[*] téglák (fal elemek) listája

Constructor Summary
JatekTerep(Dimension meret) kezdetben beállítható a játéktér mérete

Operation Summary	
void	**addTegla**(Tegla ujtegla) új tégla hozzáadását teszi lehetővé
void	**keretez**(Dimension teglameret) leteszi a téglákat a megfelelő helyre, ezzel kialakítva a játékterepet

4.1.9. Class KoBogyo

A bogyó osztály leszármazottja, a játékspecifikációban leírt kőbogyót valósítja meg.

Bogyo
└ **KoBogyo**

All Enclosing Diagrams:

Kukac_jatek_osztalydiagram

---

public class **KoBogyo**
extends Bogyo

Properties	
Alias	KoBogyo
Visibility	public
Final	☐
Transient	☑
Abstract	☐
Leaf	☐

Constructor Summary
KoBogyo(Pozicio pozicio, Dimension meret, Bokor bokra)

Operation Summary	
void	**megharapva**(Alapelem a) látogató modell első hívott fv.-e, kreál egy látogatót
void	**visszaharap**(Visszaharapas visszaharapo) a látogató modellben a látogató átadására szolgáló fv.

4.1.10. Class KoBogyoVisszaharapas

Ha a megharapott alapelem kőbogyó, akkor az kőbogyóvisszaharapás objektumot hoz létre, és átadja azt a kölcsönhatásban résztvevő félnek, a látogató modell szerint. A visszaharapás interface-t valósítja meg.

KoBogyoVisszaharapas
All Implemented Interfaces:
Visszaharapas
All Dependency Suppliers:
Implementation Visszaharapas
All Enclosing Diagrams:
Kukac_jatek_osztalydiagram

---

public class **KoBogyoVisszaharapas**
implements Visszaharapas

Properties	
Alias	KoBogyoVisszaharapas
Visibility	public
Final	☐
Transient	☑
Abstract	☐
Leaf	☐

Attribute Summary	
KoBogyo	**kb** az objektumot létrehozó objektumra hivatkozás

Constructor Summary
KoBogyoVisszaharapas(KoBogyo kb) paramétere a létrehozó objektum

Operation Summary	
KoBogyo	**getKb**() visszaadja az őt létrehozó objektumot
void	**setKb**(KoBogyo val) beállítja, hogy melyik objektumra hivatkozzon készítőként
void	**SzelvenyHarap**(Szelveny sz) ha szelvénnyel ütközik, akkor ezt a fv.-t hívja meg a szelvény
void	**TeglaHarap**(Tegla t) ha téglával ütközik, akkor ezt a fv.-t hívja meg a tégla
void	**FoldiBogyoHarap**(FoldiBogyo fb) ha földibogyóval ütközik, akkor ezt a fv.-t hívja meg a földibogyó
void	**KoBogyoHarap**(KoBogyo kb) ha kőbogyóval ütközik, akkor ezt a fv.-t hívja meg a kőbogyó
void	**FureszBogyoHarap**(FureszBogyo fb) ha fűrészbogyóval ütközik, akkor ezt a fv.-t hívja meg a fűrészbogyó

4.1.11. Class Kukac

Egy kukacot reprezentáló osztály, rendelkezik attribútumként egy olyan változóval hogy életben van e, ismeri saját mozgásának irányát, képes saját hosszát megnövelni, megrövidíteni, lépni és tárolja szelvényeit.

Kukac
All Known Associations:

Composition: Szelveny
Composition: Kukacok

All Enclosing Diagrams:

Kukac_jatek_osztalydiagram

---

public class **Kukac**

Properties	
Alias	Kukac
Visibility	public
Final	☐
Transient	☑
Abstract	☐
Leaf	☐

Attribute Summary	
Szelveny	**szelvenyek**[*] Dinamikus homogén kollekció a szelvények tárolására
bool	**eleMeg** Bool változó ami azt tárolja, hogy a kukac él e még
double	**iranyszog** A kukac mozgásának irányát adja meg

Constructor Summary
Kukac(Point hova, double irany) Létrehozza a Kukac objektumot a megadott pozicióban a megadott irányban

Operation Summary	
void	**megno**() Megnöveli a kukac hosszát eggyel
void	**rovidul**(Szelveny honnantol) A honnantol szelvénnyel kezdve levágja a kukac szelvényeit.
Szelveny	**getSzelveny**(int melyiket) Visszaadja a kukac valamely elemét az elemre a szelvény kukacban elfoglalt sorszámával lehet hivatkozni
bool	**getEleMeg**() Az eleMeg változó lekérdezése
void	**setEleMeg**(bool val) Az eleMeg változó átállítása

bool	**elemee**(Szelveny sz) A fv igaz-at ad vissza, ha az adott szelvény része a kukacnak hamisat ha nem
void	**leptet**() Eggyel lépteti a kukacot, egyenként meghívva a szelvények leptet() fvét
double	**getIranyszog**() Az iranyszog lekérdezése
void	**setIranyszog**(double val) Az iranyszog beállítása

4.1.12. Class Kukacok

Egy homogén kollekció Kukacok tárolására és az azokon elvégzendő alapvető műveletek koordinálására.

Kukacok
All Known Associations:

Composition: Kukac
Composition: Motor

All Enclosing Diagrams:

Kukac_jatek_osztalydiagram

---

public class **Kukacok**

Properties	
Alias	Kukacok
Visibility	public
Final	☐
Transient	☑
Abstract	☐
Leaf	☐

Attribute Summary	
Kukac	**kukacok**[*] Dinamikus homogén Kukac kollekció
int	**kukacokSzama** A tárolt kukacok száma

Constructor Summary
Kukacok(int hanyat) A hanyat argumentumban megadott számú kukacot helyez el a pályán

Operation Summary	
Kukac	**getKukac**(int melyiket) Az argumentumban megadott sorszámú Kukacot adja vissza
void	**ellenoriz**() Ellenőrzi, hogy mely Kukacok élnek még és törli a halottakat
void	**getKukacokSzama**() Lekéri a kukacokSzama változót
void	**leptet**() Meghívja az összes tárolt Kukac leptet() függvényét

4.1.13. Class Motor

A játék léptetéséért az objektumok összehangolásával foglalkozó objektum. Az időzítéshez szálat használ, melyet majd a proto fázis után implementálunk.

Thread
└ **Motor**

All Known Associations:

- Composition: Poziciok
- Composition: Bokor
- Composition: JatekTerep
- Composition: Kukacok
- Composition: Tetragon_Team_Kukacfarm

All Enclosing Diagrams:

- Kukac_jatek_osztalydiagram

---

public class **Motor**
extends Thread

Properties	
Alias	Motor
Visibility	public
Final	☐
Transient	☑
Abstract	☐
Leaf	☐

Attribute Summary	
Kukacok	**kukacok** A játékban aktív Kukacok objektum
Poziciok	**poziciok** A játékban aktív Poziciok objektum
Bokor	**bokor** A játékban aktív bokor objektum
JatekTerep	**jatekTerep** A játékban aktív jatekTerep objektum

Constructor Summary
Motor(int kukacokSzama, int bogyokMaxSzama, Dimension jatekTerMeret) Felállítja a játék objektumait a megadott számú kukaccal, a bogyók maximális számával, illetve a játéktér méretével

Operation Summary	
Kukacok	**getKukacok**() A kukacok visszaadása
void	**setKukacok**(Kukacok val) A kukacok beállítása

Poziciok	**getPoziciok**() A poziciok lekérdezése
void	**setPoziciok**(Poziciok val) A poziciok beállítása
Bokor	**getBokor**() Bokor lekérdezése
void	**setBokor**(Bokor val) Bokor beállítása
JatekTerep	**getJatekTerep**() JatekTerep lekérdezése
void	**setJatekTerep**(JatekTerep val) JatekTerep beállítása
void	**run**() A szál futtatása

4.1.14. Class Pozicio

Az osztály Alapelem típusú objektumok térbeli pozíciójának tárolására szolgál.

Pozicio
All Known Associations:

Composition: Poziciok

All Enclosing Diagrams:

Kukac_jatek_osztalydiagram

---

public class **Pozicio**

Properties	
Alias	Pozicio
Visibility	public
Final	☐
Transient	☑
Abstract	☐
Leaf	☐

Attribute Summary	
Point	**hely** A térpeli hely
Alapelem	**alapelem** A hivatkozott alapelem

Constructor Summary
Pozicio(Point hely) Létrehozás egy pozíció alapján
Pozicio() Egy véletlenszerű szabad helyre teszi le a poziciot

Operation Summary	
Point	**getHely**() A hely lekérdezése
void	**setHely**(Point val) A hely beállítása
Alapelem	**getAlapelem**() Az alapelem lekérdezése
void	**setAlapelem**(Alapelem val) Az alapelem lekérde

4.1.15. Class Poziciok

A Pozicio-k tárolására szolgáló osztály.

Poziciok
All Known Associations:

Composition: Pozicio
Composition: Motor

All Enclosing Diagrams:

Kukac_jatek_osztalydiagram

---

public class **Poziciok**

Properties	
Alias	Poziciok
Visibility	public
Final	☐
Transient	☑
Abstract	☐
Leaf	☐

Attribute Summary	
Pozicio	**poziciok**[*] Homogén pozicio-kollekció

Constructor Summary	
Poziciok()	

Operation Summary	
void	**utkozes**(Alapelem a) Megvizsgálja, hogy adott alapelem ütközik e valamely már tárolt pozícióval
void	**addPozicio**(Pozicio p) pozíció felvétele
void	**deletiPozicio**(Pozicio p) Pozíció törlése

4.1.16. Class Szelveny

A Kukac alapvető eleme.

Alapelem
└ **Szelveny**

All Known Associations:

Composition: Kukac

All Enclosing Diagrams:

Kukac_jatek_osztalydiagram

---

public class **Szelveny**
extends Alapelem

Properties	
Alias	Szelveny
Visibility	public
Final	☐
Transient	☑
Abstract	☑
Leaf	☐

Attribute Summary	
Kukac	**kukaca** Hivatkozás a kukacra, melyhez a szelvény tartozik
int	**fureszHatas** A kukac fűrészhatásából hátralévő léptetések száma
bool	**koBogyoHatas** Azt jelzi, hogy van e kőbogyohatás az adott szelvényen

Constructor Summary
Szelveny(Pozicio pozicio, Dimension meret, Kukac kukaca) Létrehozza az adott szelvényt az adoot helyen, méretben, a megfelelő kukachoz

Operation Summary	
void	**megharapva**(Alapelem a) A visitor minta megvalósításához szükséges fv
Kukac	**getKukaca**() a kukaca lekérdezése
void	**setKukaca**(Kukac val) a kukaca beállítása
bool	**getFureszHatas**() a furészhatás lekérése
void	**setFureszHatas**(int val) a fureszhatas beállítása
bool	**getKoBogyoHatas**()

	A kobogyohatas lekérése
void	**setKoBogyoHatas**(bool val) A kobogyohatas beállítása
void	**leptet**() A szelvény önellenőrző funkciója, csökkenti a fűrészhatás időtartamát eggyel
void	**visszaharap**(Visszaharapas visszaharapo) A Visitor minta megvalósításához szükséges függvény

4.1.17. Class SzelvenyVisszaharapas

Ha a megharapott alapelem szelvény, akkor az szelvényvisszaharapás objektumot hoz létre, és átadja azt a kölcsönhatásban résztvevő félnek, a látogató modell szerint. A visszaharapás interface-t valósítja meg.

SzelvenyVisszaharapas
All Implemented Interfaces:
Visszaharapas
All Dependency Suppliers:
Implementation Visszaharapas
All Enclosing Diagrams:
Kukac_jatek_osztalydiagram

---

public class **SzelvenyVisszaharapas**
implements Visszaharapas

Properties	
Alias	SzelvenyVisszaharapas
Visibility	public
Final	☐
Transient	☑
Abstract	☐
Leaf	☐

Attribute Summary	
Szelveny	**sz** az objektumot létrehozó objektumra hivatkozás

Constructor Summary
SzelvenyVisszaharapas(Szelveny sz) paramétere a létrehozó objektum

Operation Summary	
Szelveny	**getSz**() visszaadja az őt létrehozó objektumot
void	**setSz**(Szelveny val) beállítja, hogy melyik objektumra hivatkozzon készítőként
void	**SzelvenyHarap**(Szelveny sz) ha szelvénnyel ütközik, akkor ezt a fv.-t hívja meg a szelvény
void	**TeglaHarap**(Tegla t) ha téglával ütközik, akkor ezt a fv.-t hívja meg a tégla
void	**FoldiBogyoHarap**(FoldiBogyo fb) ha földibogyóval ütközik, akkor ezt a fv.-t hívja meg a földibogyó
void	**KoBogyoHarap**(KoBogyo kb) ha kőbogyóval ütközik, akkor ezt a fv.-t hívja meg a kőbogyó
void	**FureszBogyoHarap**(FureszBogyo fb) ha fűrészbogyóval ütközik, akkor ezt a fv.-t hívja meg a fűrészbogyó

4.1.18. Class Tegla

Az alapelemből leszármazó osztály. A falat reprezentálja.

Alapelem
└ **Tegla**

All Known Associations:

Composition: JatekTerep

All Enclosing Diagrams:

Kukac_jatek_osztalydiagram

---

public class **Tegla**
extends Alapelem

Properties	
Alias	Tegla
Visibility	public
Final	☐
Transient	☑
Abstract	☑
Leaf	☐

Attribute Summary	
JatekTerep	**terepe** a falat a játékterep objektum tárolja és kezeli, ezért ismerik, az őt tároló objektumot

Constructor Summary
Tegla(Pozicio pozicio, Dimension meret, JatekTerep terepe)

Operation Summary	
void	**megharapva**(Alapelem a) látogató modell első hívott fv.-e, kreál egy látogatót
JatekTerep	**getTerepe**() visszaadja, hogy a tégla melyik játékterep objektumhoz tartozik
void	**setTerepe**(JatekTerep val) beállítja, hogy a tégla melyik játékterep objektumhoz tartozik
void	**visszaharap**(Visszaharapas visszaharapo) a látogató modellben a látogató átadására szolgáló fv.

4.1.19. Class TeglaVisszaharapas

Ha a megharapott alapelem tégla, akkor az téglavisszaharapás objektumot hoz létre, és átadja azt a kölcsönhatásban résztvevő félnek, a látogató modell szerint. A visszaharapás interface-t valósítja meg.

TeglaVisszaharapas
All Implemented Interfaces:
Visszaharapas
All Dependency Suppliers:
Implementation Visszaharapas
All Enclosing Diagrams:
Kukac_jatek_osztalydiagram

---

public class **TeglaVisszaharapas**
implements Visszaharapas

Properties	
Alias	TeglaVisszaharapas
Visibility	public
Final	☐
Transient	☑
Abstract	☐
Leaf	☐

Attribute Summary	
Tegla	**t** az objektumot létrehozó objektumra hivatkozás

Constructor Summary
TeglaVisszaharapas(Tegla t) paramétere a létrehozó objektum

Operation Summary	
Tegla	**getT**() visszaadja az őt létrehozó objektumot
void	**setT**(Tegla val) beállítja, hogy melyik objektumra hivatkozzon készítőként
void	**SzelvenyHarap**(Szelveny sz) ha szelvénnyel ütközik, akkor ezt a fv.-t hívja meg a szelvény
void	**TeglaHarap**(Tegla t) ha téglával ütközik, akkor ezt a fv.-t hívja meg a tégla
void	**FoldiBogyoHarap**(FoldiBogyo fb) ha földibogyóval ütközik, akkor ezt a fv.-t hívja meg a földibogyó
void	**KoBogyoHarap**(KoBogyo kb) ha kőbogyóval ütközik, akkor ezt a fv.-t hívja meg a kőbogyó
void	**FureszBogyoHarap**(FureszBogyo fb) ha fűrészbogyóval ütközik, akkor ezt a fv.-t hívja meg a fűrészbogyó

4.1.20. Class Tetragon_Team_Kukacfarm

A Tetragon_Team_Kukacfarm osztály felelős az inicializáció elindításáért. Ő hozza létre a Motor osztályt, amely elvégzi az inicializációt. Egy példány létezik belőle. Új játékot lehet vele indítani, és meglévőt lehet törölni.

Tetragon_Team_Kukacfarm
All Known Associations:

Composition: Motor

All Enclosing Diagrams:

Kukac_jatek_osztalydiagram

---

public class **Tetragon_Team_Kukacfarm**

Properties	
Alias	Tetragon_Team_Kukacfarm
Visibility	public
Final	☐
Transient	☑
Abstract	☐
Leaf	☐

Attribute Summary	
Motor	**motor** egy motor objektumot tartalmaz

Constructor Summary
Tetragon_Team_Kukacfarm()

Operation Summary	
void	**main**(srgs[*]) program belépési pont
Motor	**getMotor**() visszaadja a motort
void	**setMotor**(Motor val) beállítja a motort

4.1.21. Interface Visszaharapas

A visszaharapás interface a visszaharapások kohézív tulajdonságainak összessége. Leírja, hogyan kell kinéznie az őt megvalósító interfacek-nek.

Visszaharapas
All Dependency Clients:

- Implementation FureszBogyoVisszaharapas
- Implementation SzelvenyVisszaharapas
- Implementation KoBogyoVisszaharapas
- Implementation FoldiBogyoVisszaharapas
- Implementation TeglaVisszaharapas

All Enclosing Diagrams:

- Kukac_jatek_osztalydiagram

---

public interface **Visszaharapas**

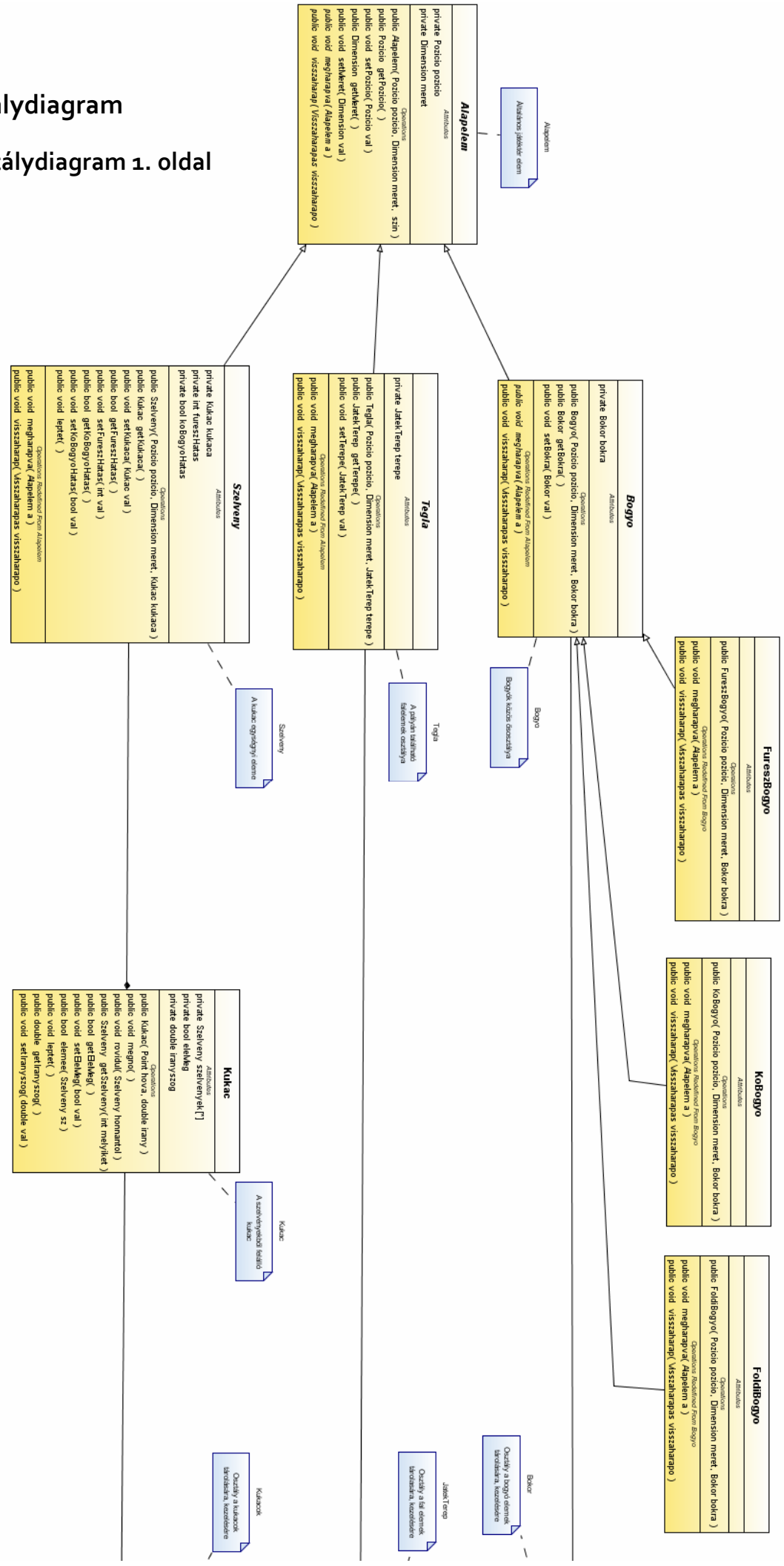
Properties	
Alias	Visszaharapas
Visibility	public
Final	☐
Transient	☑
Abstract	☑
Leaf	☐

Operation Summary	
void	**SzelvenyHarap**(Szelveny sz)
void	**TeglaHarap**(Tegla t)
void	**FoldiBogyoHarap**(FoldiBogyo fb)
void	**KoBogyoHarap**(KoBogyo kb)
void	**FureszBogyoHarap**(FureszBogyo fb)

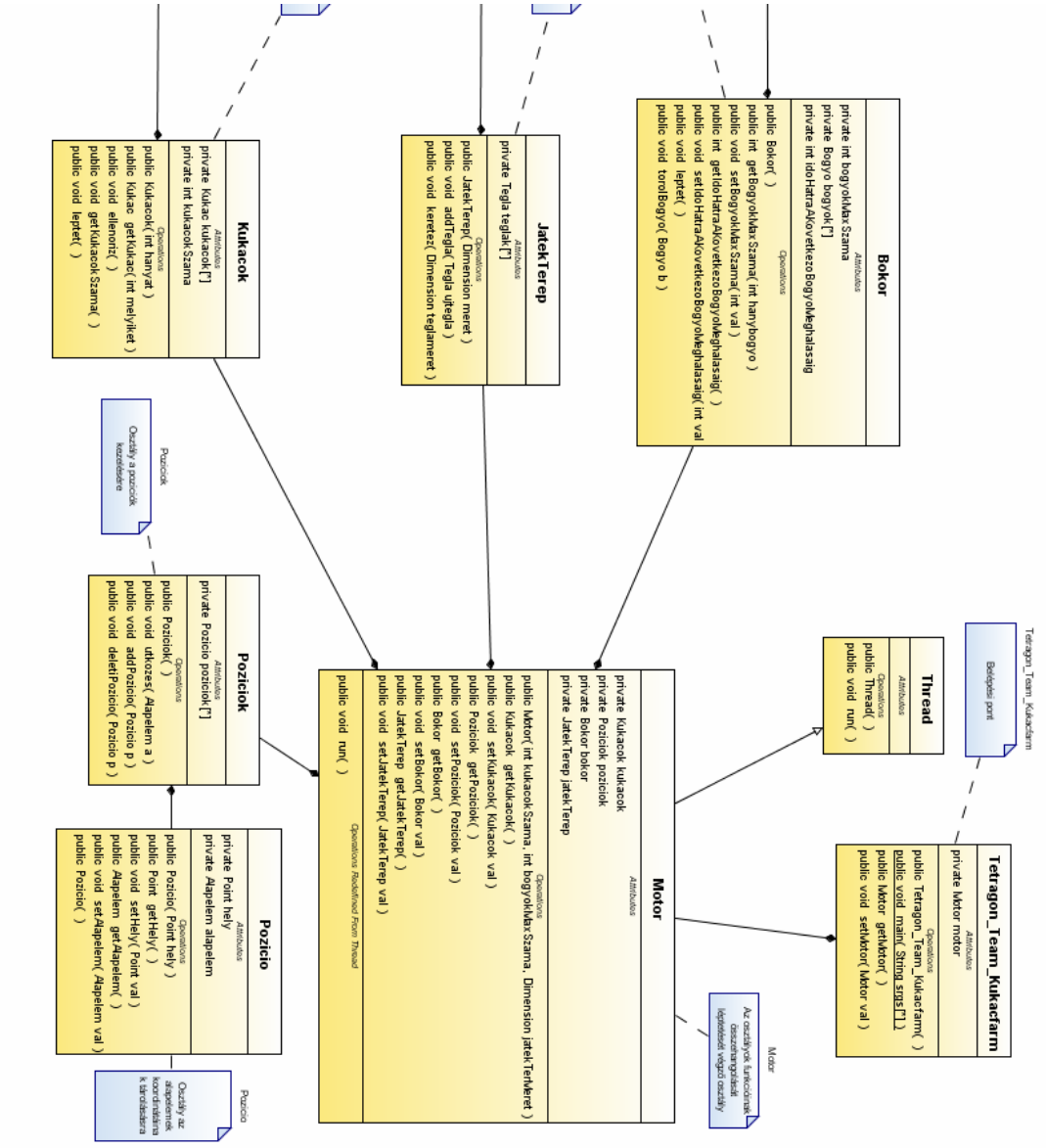
A függvények leírása az adott – interface-t megvalósító – osztály leírásában található.

4.2. Osztálydiagram

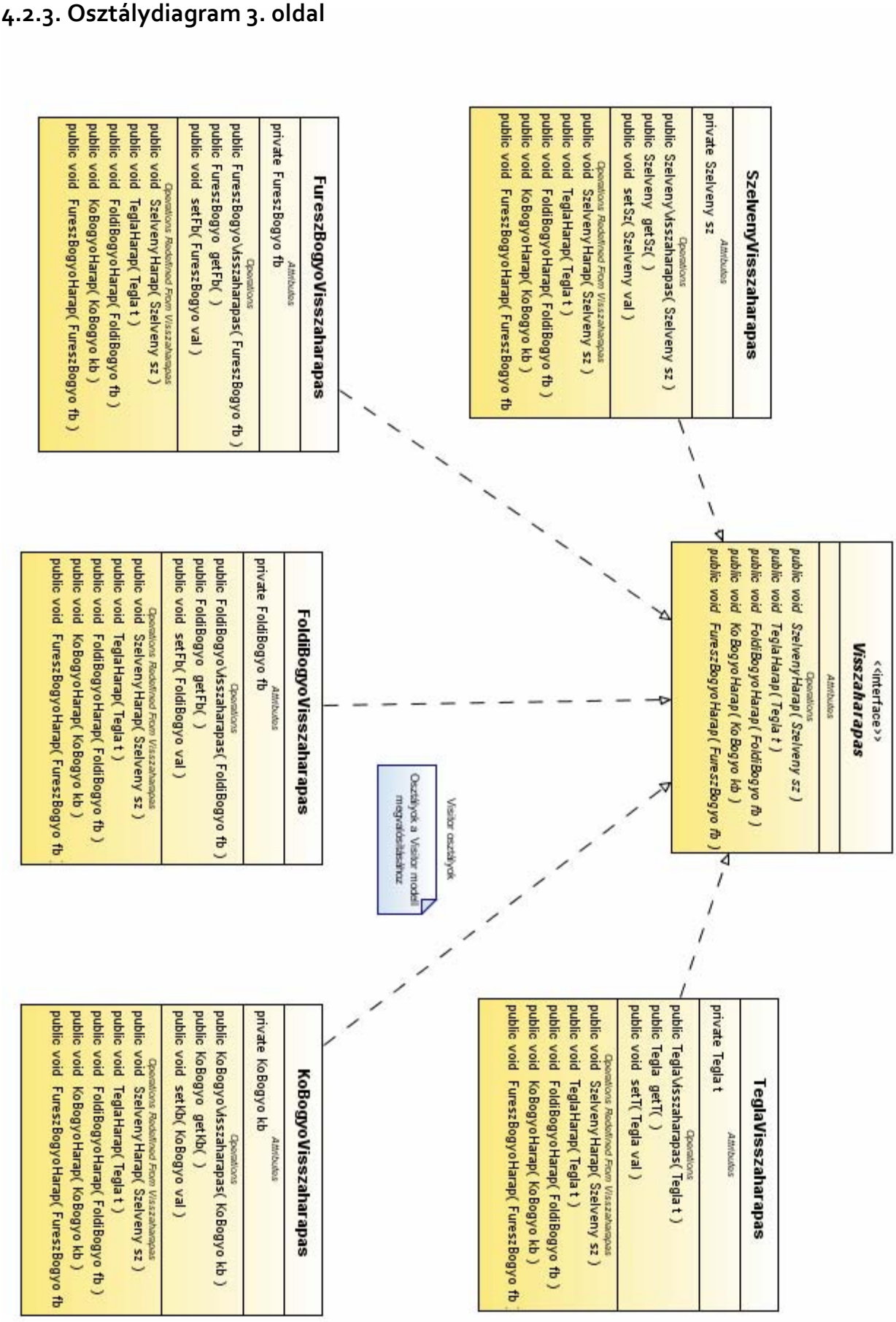
4.2.1. Osztálydiagram 1. oldal



4.2.2. Osztálydiagram 2. oldal

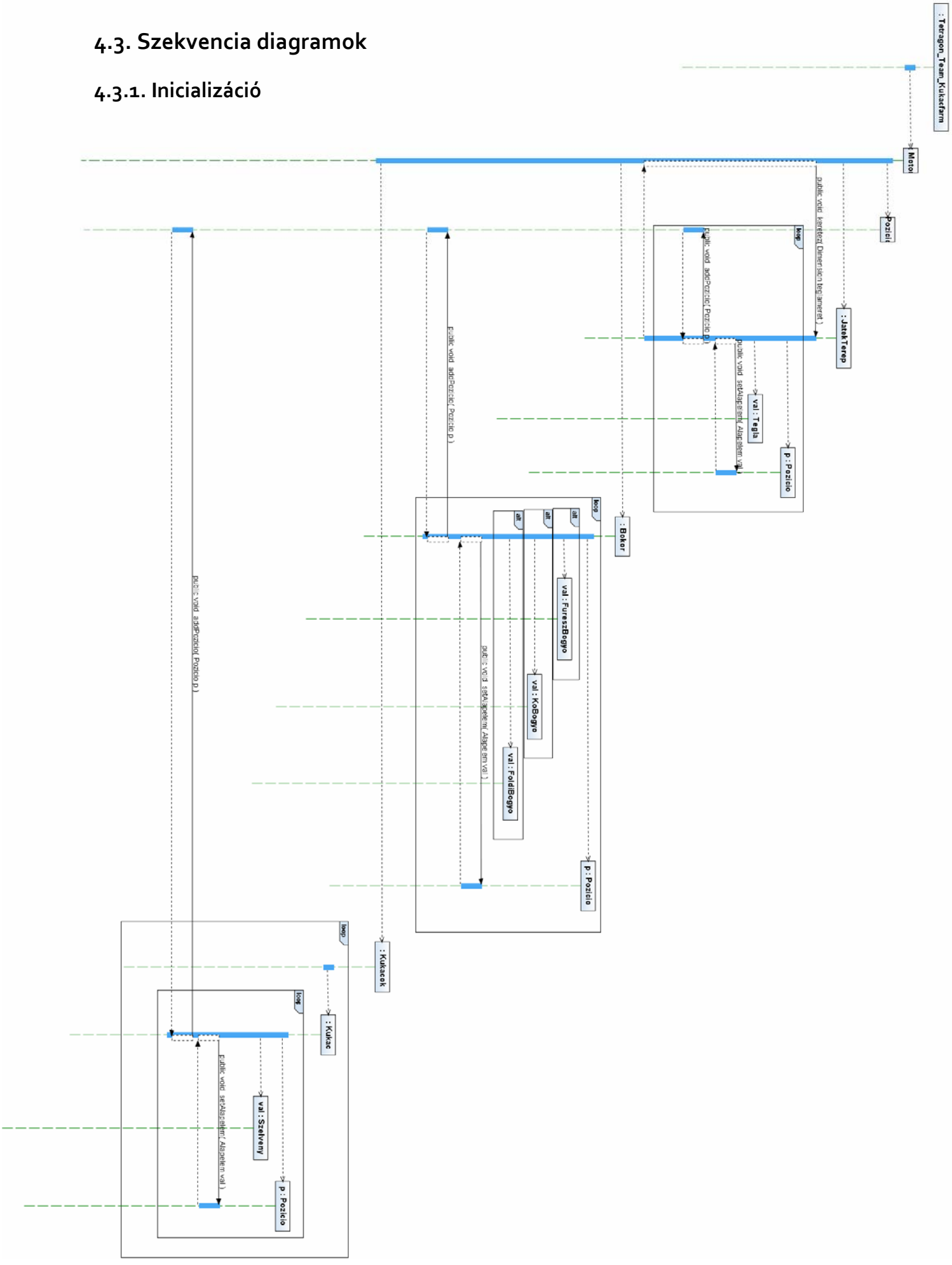


4.2.3. Osztálydiagram 3. oldal

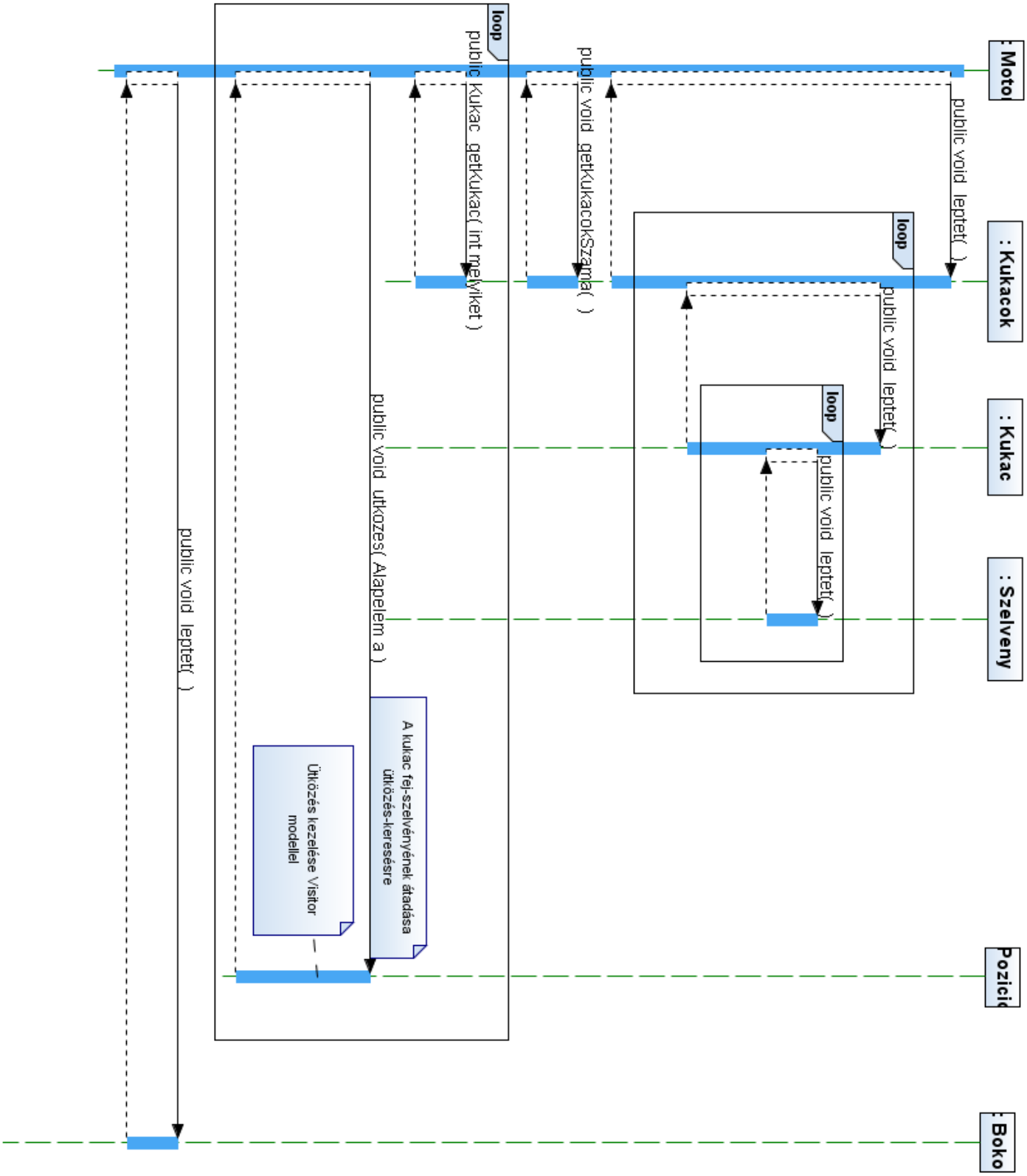


4.3. Szekvencia diagramok

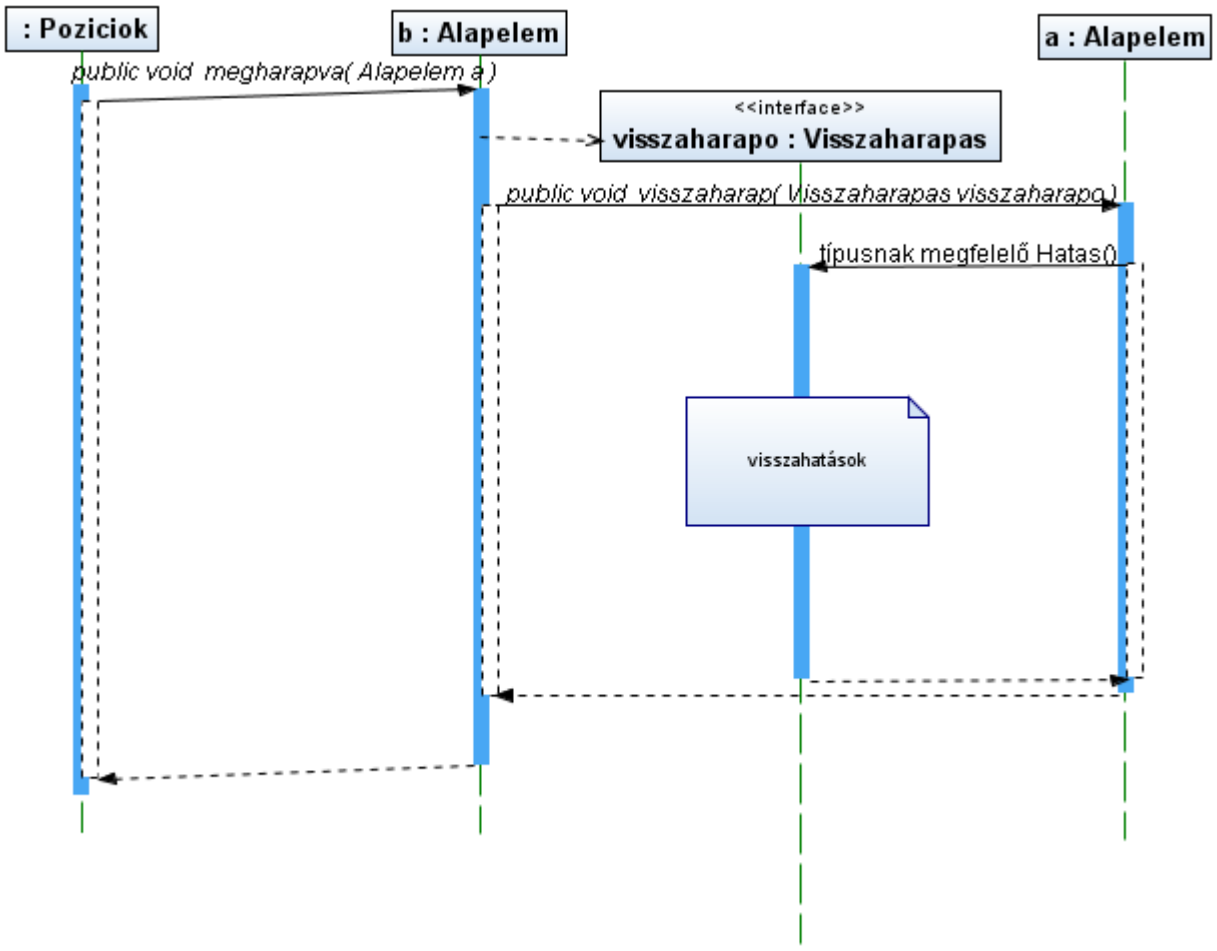
4.3.1. Inicializáció



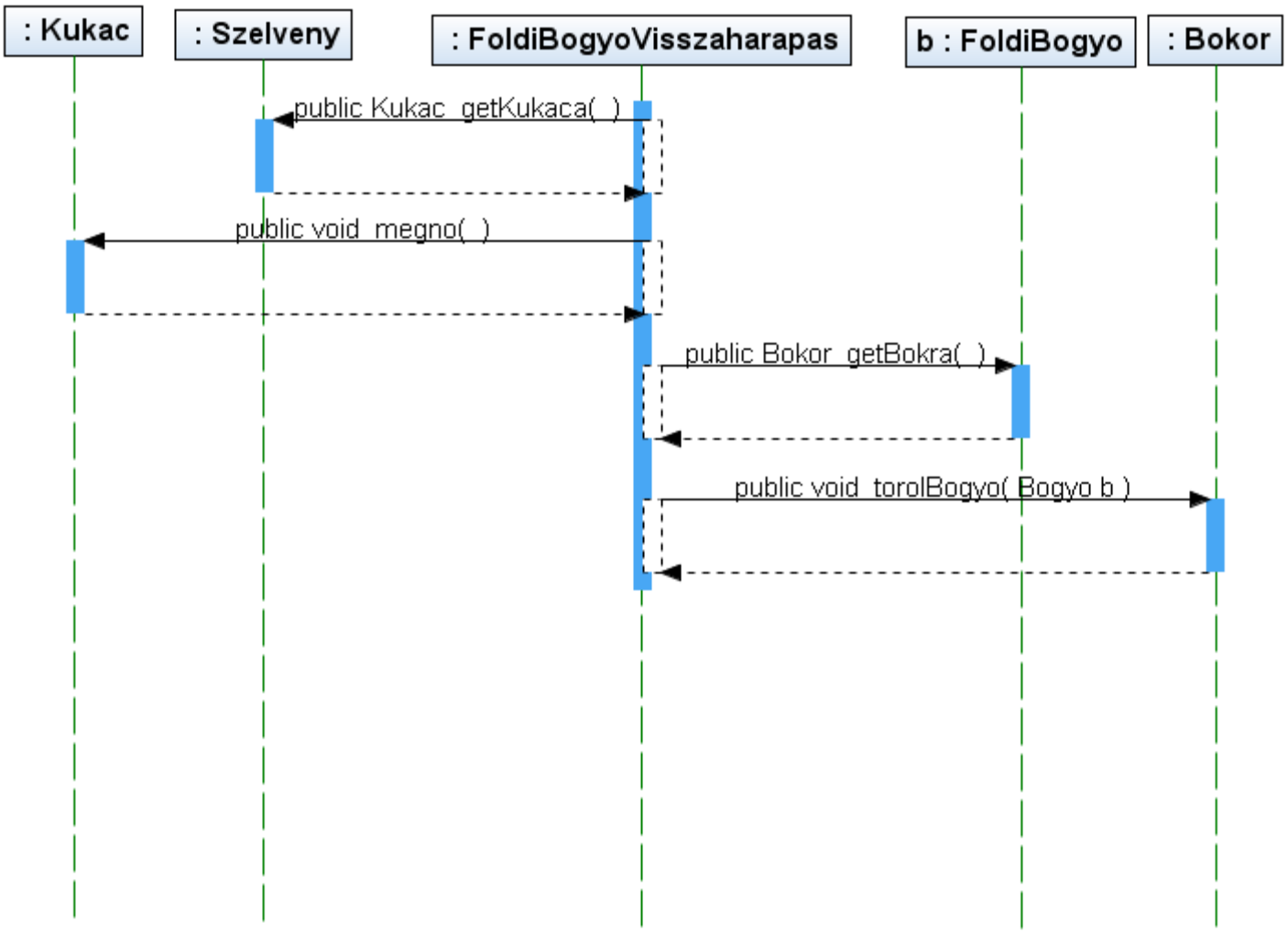
4.3.2. Motor step



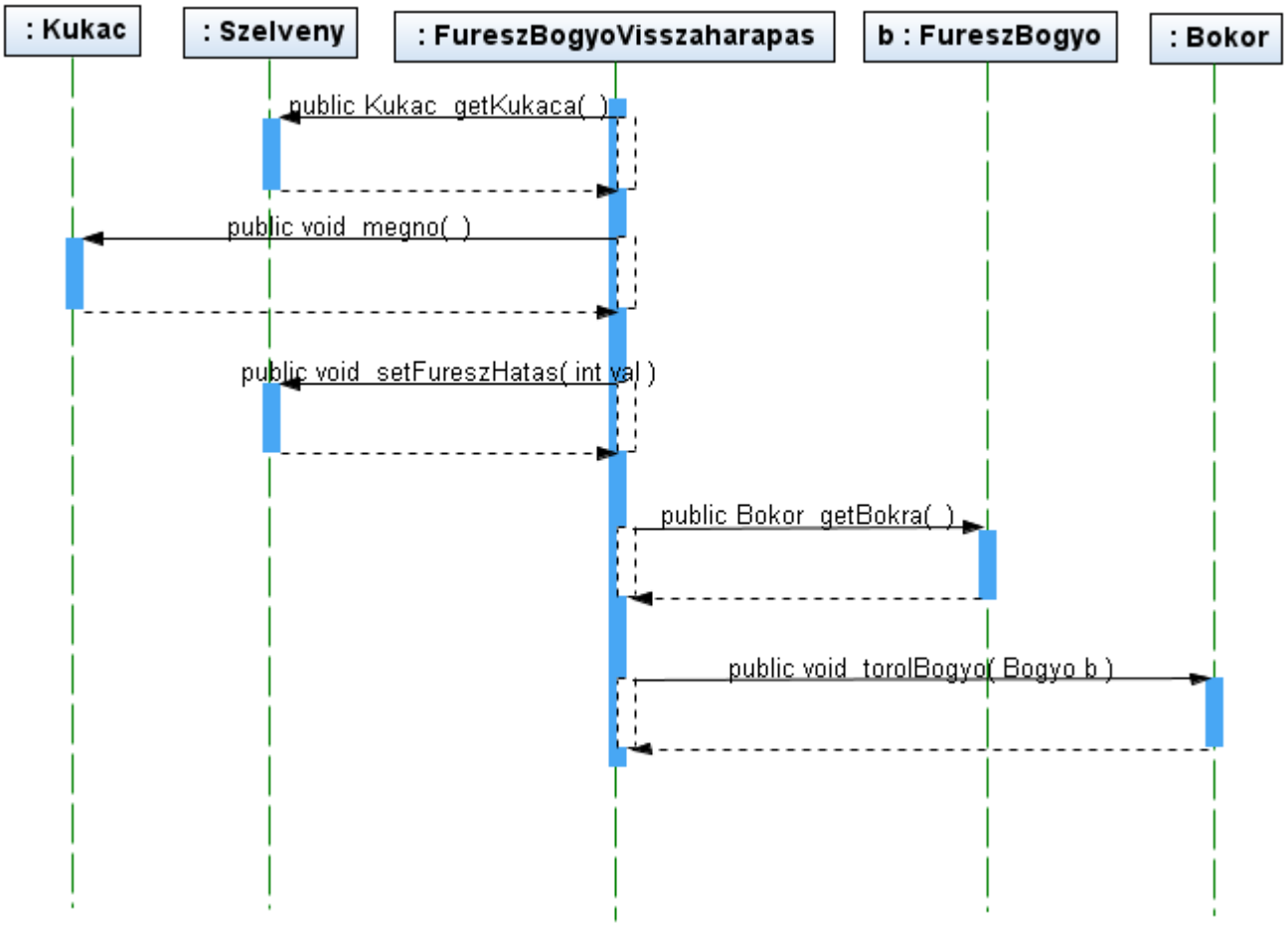
4.3.3. Ütközés


: Poziciok
b : Alapelem
a : Alapelem
public void megharapva(Alapelem a)
<<interface>>
visszaharapo : Visszaharapas
public void visszaharap(Visszaharapas visszaharapo)
típusnak megfelelő Hatas()
visszahatások

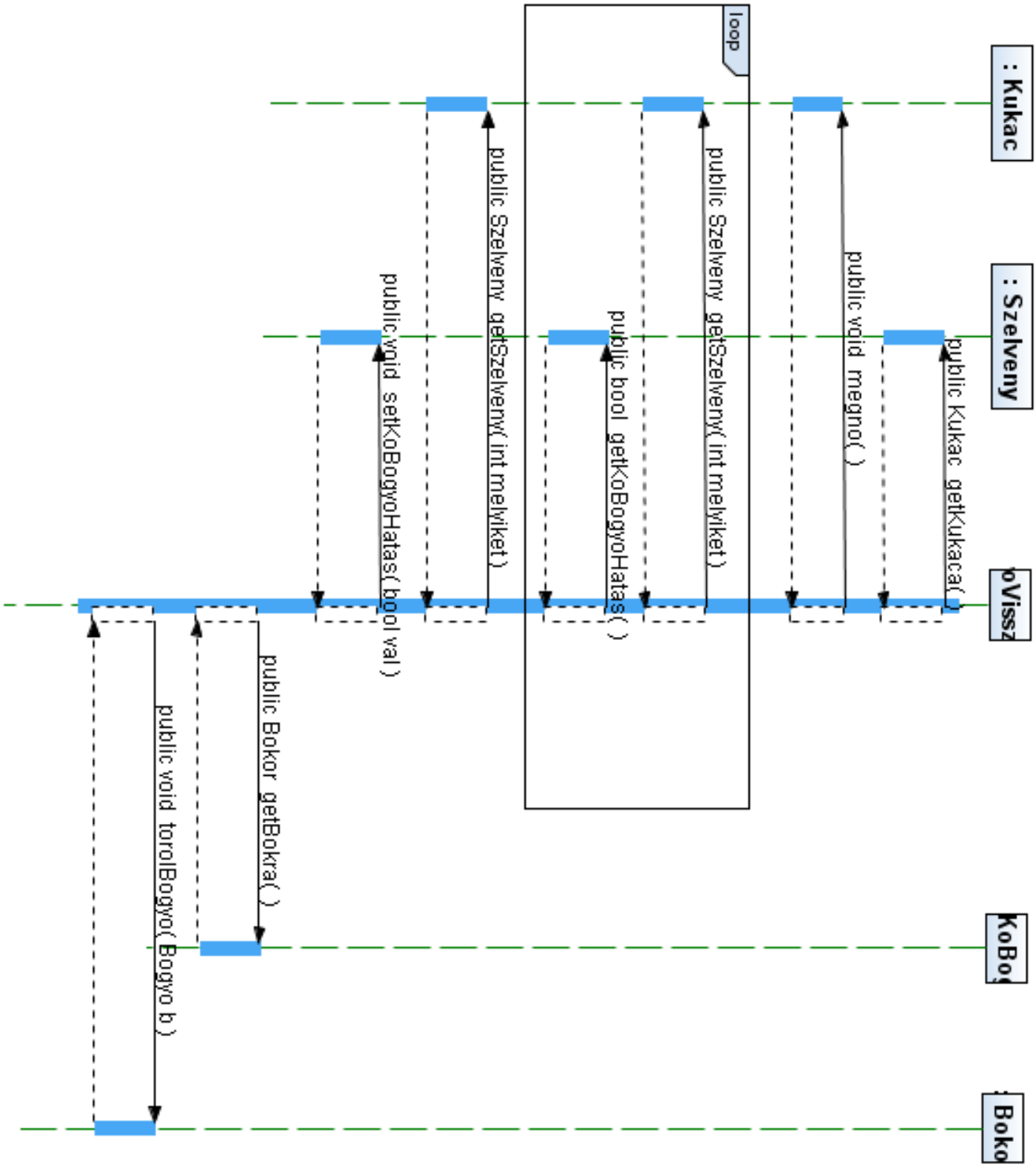
4.3.4. Földibogyóharapás



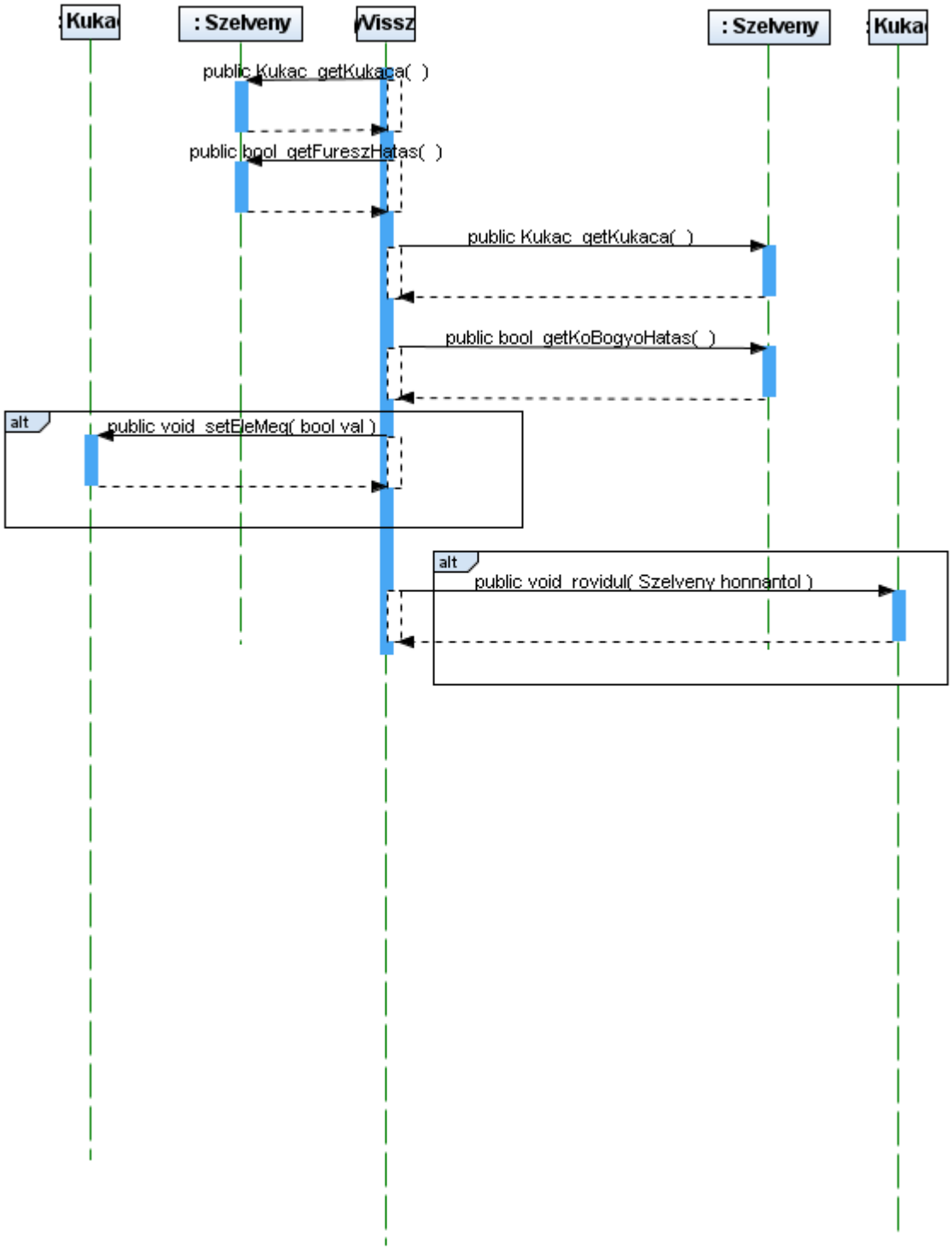
4.3.5. Fűrészbogyóharapás



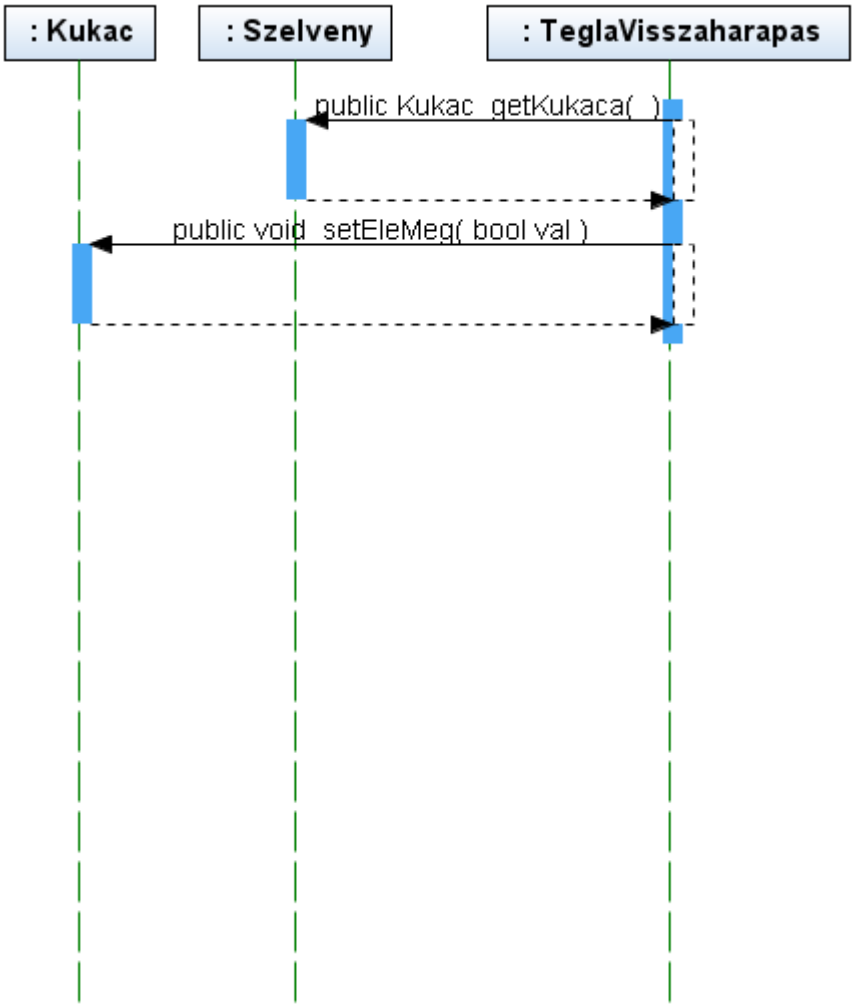
4.3.6. Kőbogyóharapás



4.3.7. Szelvényharapás

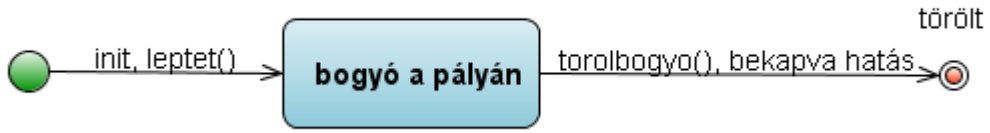


4.3.8. Téglaharapás

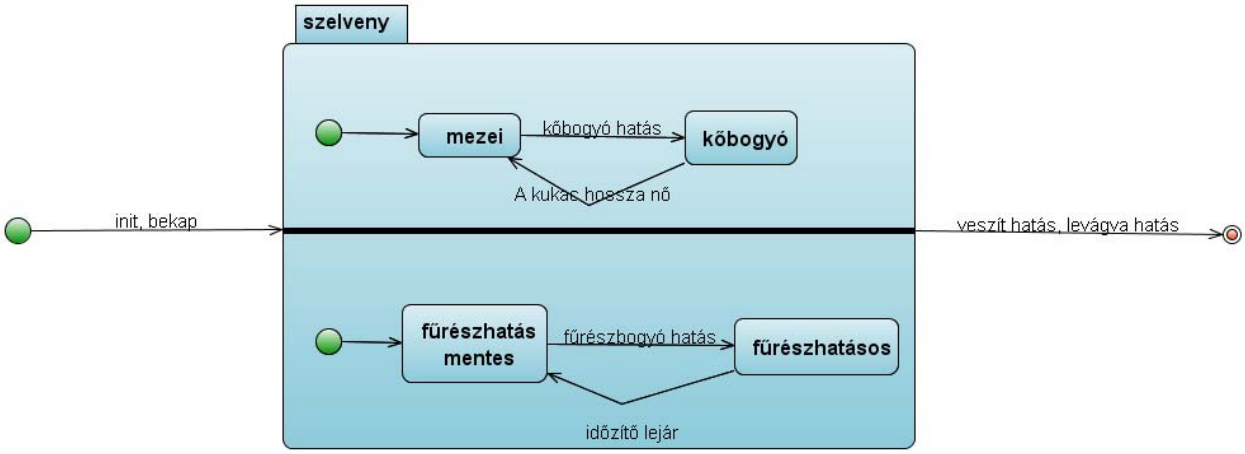


4.4. Állapotdiagramok

4.4.1. Bogyó



4.4.2. Szelvény

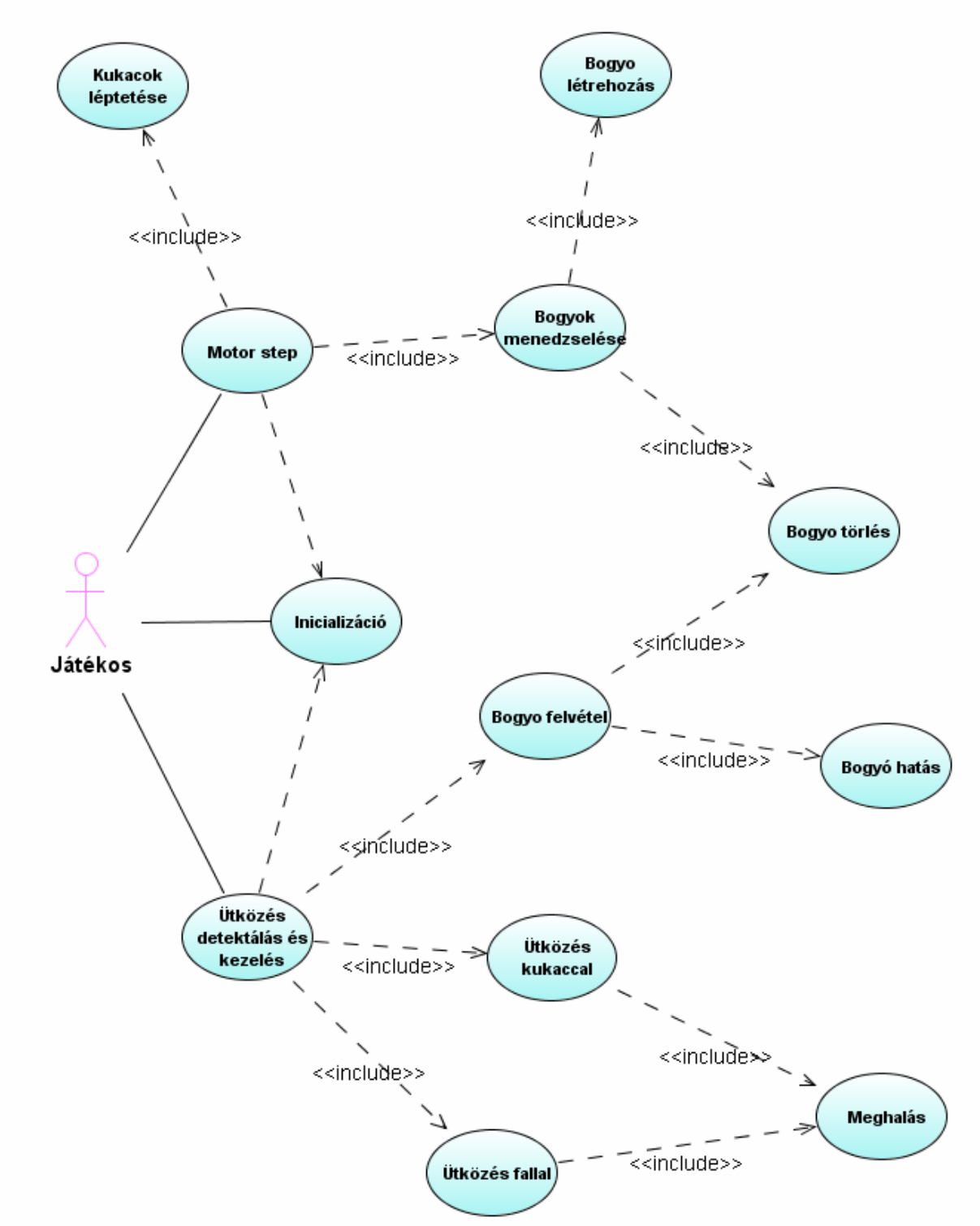


4.4.3. Kukac

5. fejezet Szkeleton tervezése

5.1. A szkeleton model valóságos use-case-ei

5.1.1. Use case diagram



5.1.2. Use case-k leírása

Use Case: Inicializáció
Actor: Játékos
Leírás: Felépíti a játékhoz szükséges kiinduló objektumhierarchiát

Use Case: Motor step
Actor: Játékos
Leírás: Végrehajt egy egységnyi lépést a fő játék hurokból

Use Case: Ütközés detektálás és kezelés
Actor: Játékos
Leírás: Ütközések lefolyása

Use Case: Kukacok léptetése
Actor: Játékos
Leírás: Kukac orientációjának beállítása

Use Case: Bogyok menedzselése
Actor: Játékos
Leírás: Bogyo létrehozás és törlés

Use Case: Bogyo létrehozás
Actor: Játékos
Leírás: Egy bogyó létrehozása

Use Case: Bogyo törlés
Actor: Játékos
Leírás: Egy bogyó törlése

Use Case: Bogyo felvétel
Actor: Játékos
Leírás: Kukac ütközése bogyóval és a bogyó törlése

Use Case: Bogyo hatás
Actor: Játékos
Leírás: Kukac állapotának meghatározása (bogyó hatások)

Use Case: Ütközés kukaccal
Actor: Játékos
Leírás: Két kukac ütközése

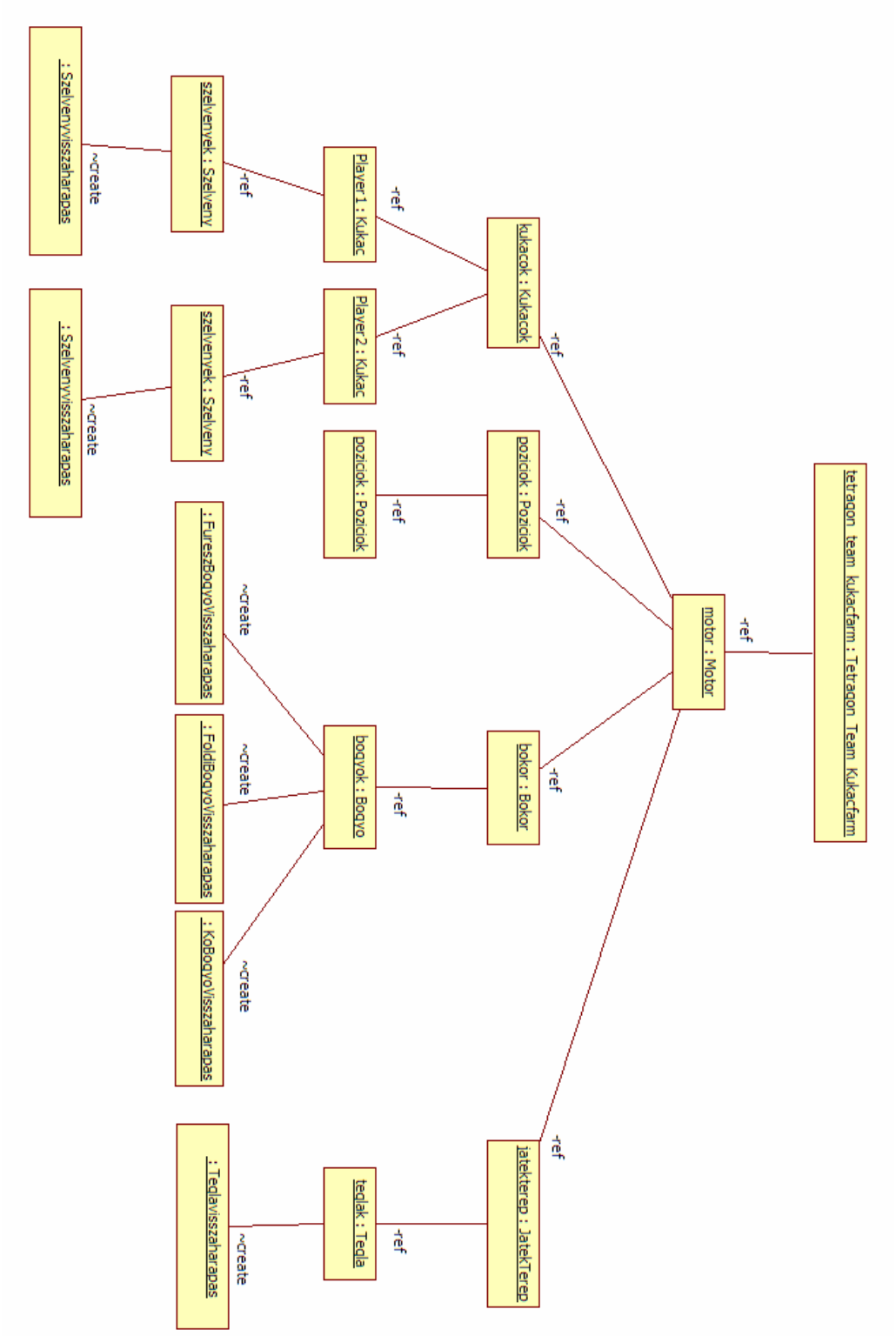
Use Case: Ütközés fallal
Actor: Játékos
Leírás: Kukac és fal ütközése

Use Case: Meghalás
Actor: Játékos
Leírás: Ha egy kukac egy másikkal vagy egy fallal ütközik, akkor meghal

5.2. Architektúra

5.2.1. Architektúra

Az alábbi objektumdiagram mutatja általános esetben, az inicializáció után kialakult objektumhivatkozásokat. Jelöltük azt is, hogy az egyes visitorok létrehozásáért ki a felelős.



5.2.2. Szkenáriók

Inicializáció

3.3.1.-es fejezet szekvencia diagramja alapján látható, hogy ez az eset nem igényel túlzott interakciót a felhasználó részéről, ennek lefutása viszonylag megkötött. Egyetlen közbeavatkozás csak a loop-ok esetén lehetséges, itt megkérdezzük, hogy akarunk-e új Alapelem típusú objektumot felvenni az adott típusból.

Motor step

Itt minden egyes Alapelem objektum esetén feltesszük a kérdést, hogy akarjuk-e azt léptetni, Kukac típusú objektum esetén az egyes Szelvény típusú objektumokra ezt a kérdést nem tesszük fel, léptetődnek. Az ütközéseket külön esetben lehet tesztelni, itt azokkal nem foglalkozunk. Végül megkérdezzük, hogy végre akarunk-e hajtani még egy step-et a motorban.

Ütközések (harapások)

Ebben az esetben megnézzük az egyes objektumok közötti ütközések lefolyását. A két ütköző objektum típusát a felhasználótól kérdezzük meg. Az ütközés utóhatásait az egyes visitor objektumok valósítják meg, az egyes ütközések között lényegében csak az lesz a különbség, hogy melyik visitor jön létre a szükséges műveletek elvégzésére.

5.2.3. Ütemezés

A szkeleton modellben nem használunk szálakat, hogy a működés determinisztikus legyen; ez szükséges feltétele annak, hogy az egyes szekvencia diagramok a szkeleton alapján ellenőrizhetőek. A szálkezelést később, a prototípusban vezetjük be.

5.3. A szkeleton kezelői felületének terve, dialógusok

A szkeleton, mint program célja annak bizonyítása, hogy a programunk statikus és dinamikus modelljeiről leképzett programváz képes-e az elvárt működést produkálni. A szkeletonban minden objektum szerepel, de azoknak csak az interfésze definiált.

Minden egyes metódus az indulás pillanatában a képernyőre kiírja a saját nevét, majd meghívja azokat a metódusokat, amelyeket a szolgáltatás végrehajtása érdekében meg kell hívnia. Ha a metódusból valamely feltétel fennállása esetén hívunk meg más metódusokat, akkor a feltételre vonatkozó kérdést interaktívan a képernyőn fel kell tenni és a kapott választ felhasználva kell tovább lépni.

A szkeletonnak alkalmasnak kell lenni arra, hogy a különböző forgatókönyvek és szekvencia diagramok ellenőrizhetőek legyenek. Csak karakteres képernyőkezelés fogadható el, mert ez biztosítja a rendszer egyszerűségét.

A valóságos use case, a korábbi szekvencia diagramok és a következő pontban szereplő kommunikációs diagramok alapján elkészíthető az ellenőrzést szolgáló dialógussorozat:

- Az indulás után a teljes inicializáció eljátszása; több elem fölvétele esetén rákérdez, hogy mennyit vegyen föl.
- Az inicializáció után a léptetések következnek, ha egy ciklusnak vége, az kiíródik a konzolra és rákérdez, hogy jöhet-e a következő iteráció, ekkor lehet kilépni is ebből

az esetből. Rákérdez arra is, hogy az adott körben melyik kukac milyen irányba forduljon. Ütközést nem vizsgálunk, a hangsúly az irányításon és a bogyók menedzselésén van (letelt-e az idő – erre is rákérdezünk).

- A léptetések rész után előre definiált esetek alapján rendre megfigyeljük minden ütközéspáros lefolyását (előfeltételek és utóhatások). Figyelembe kell venni, hogy a kukac állapotától függően egy vele való ütközés máshogy is lefolyhat.
- Amennyiben a fenti ellenőrzések valamelyike sikertelen, újra át kell gondolni a modellt (esetleg lehetséges, hogy a modell jó, de a szkeletonban valamilyen szemantikai hiba van, nem teljesen a modellt tükrözi, és ezt nem vettük észre).

Függvényhívás esetén a kiírás a következő formában történik (rendre egymás után egy sorban):

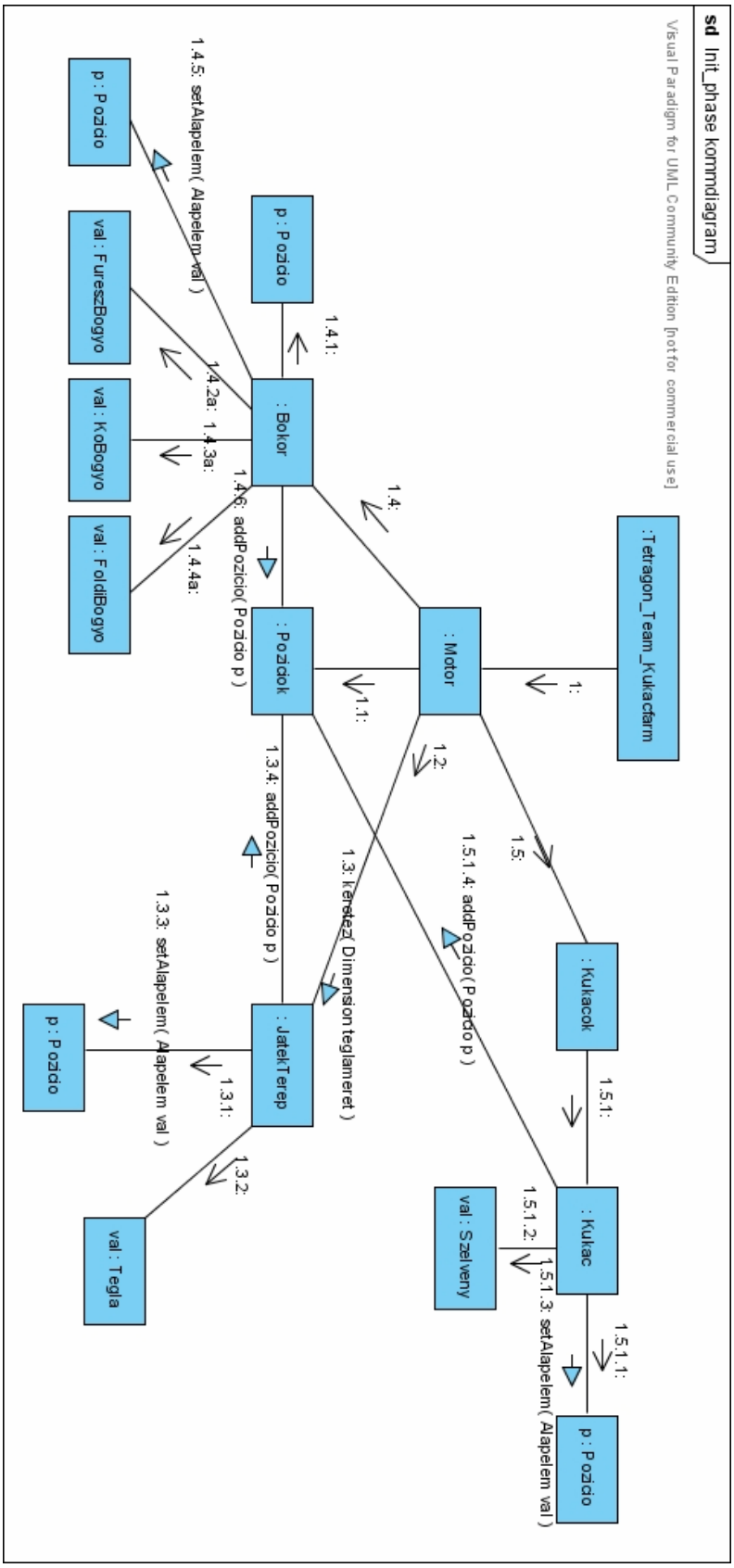
- Osztály neve
- Utána az objektum neve is jöhetne, de egyszerűbb a hashcode (Java specifikus; egy int szám) kiírása, és ezzel ráadásul egyértelműen azonosítható az objektum
- Függvény neve és paraméterszignatúra (és esetleg az átadott objektumok hashcode-ja – a paraméter típusa után, mintha egy függvénydefiníció headerjét néznénk, csak a változónév helyett a hashcode szerepel)
- Hívás esetén call, visszatérés esetén return

Egy konkrét példa a konzol kimenetére a 3.3.8.-as pont szekvencia diagramja (Téglaharapás) alapján a következőképpen nézhet ki a szkeleton megfelelő részének futtatása után (az ütközésdetektálás nem szerepel):

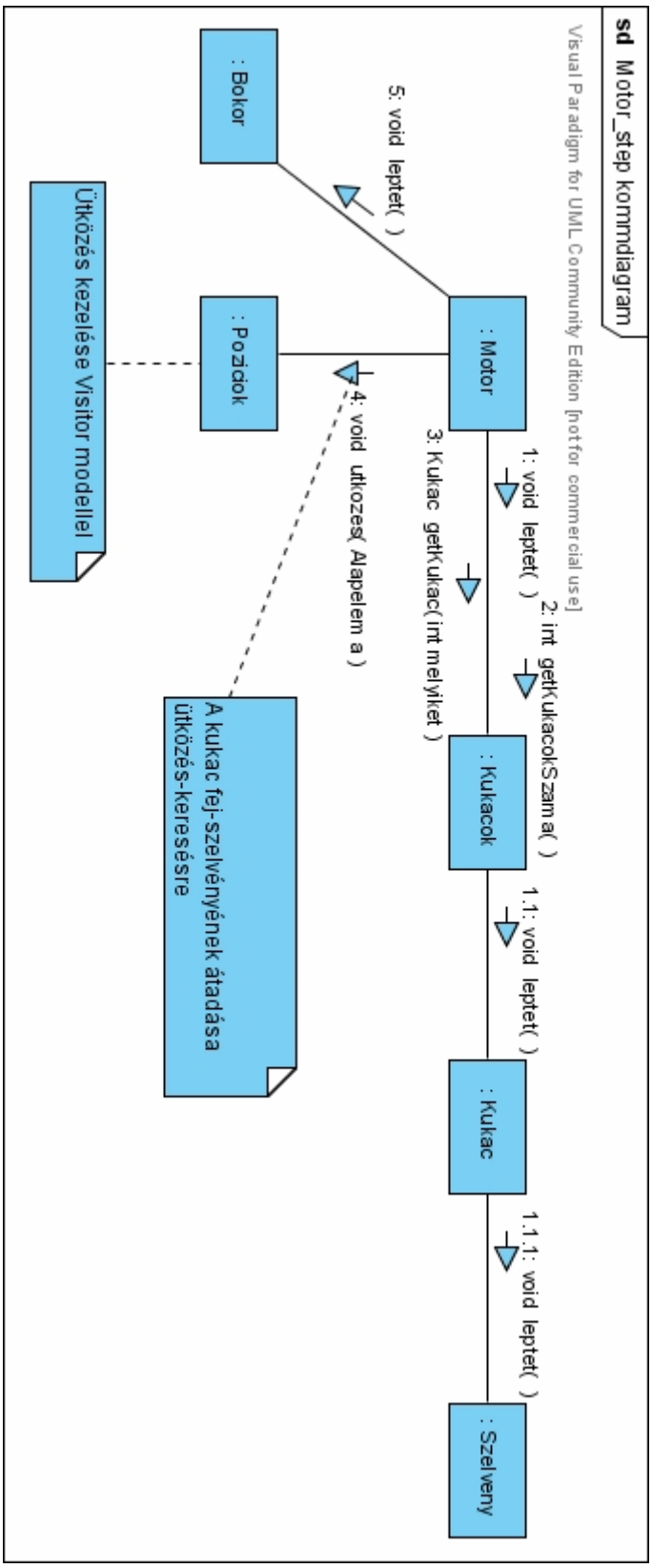
```
…
[:Szelveny|1235|getKukaca()|CALL]
[:Szelveny|1235|getKukaca()|RETURN]
[:Kukac|869|setEleMeg(bool)|CALL]
[:Kukac|869|setEleMeg(bool)|RETURN]
…
```

5.4. Kommunikációs diagramok a belső működésre

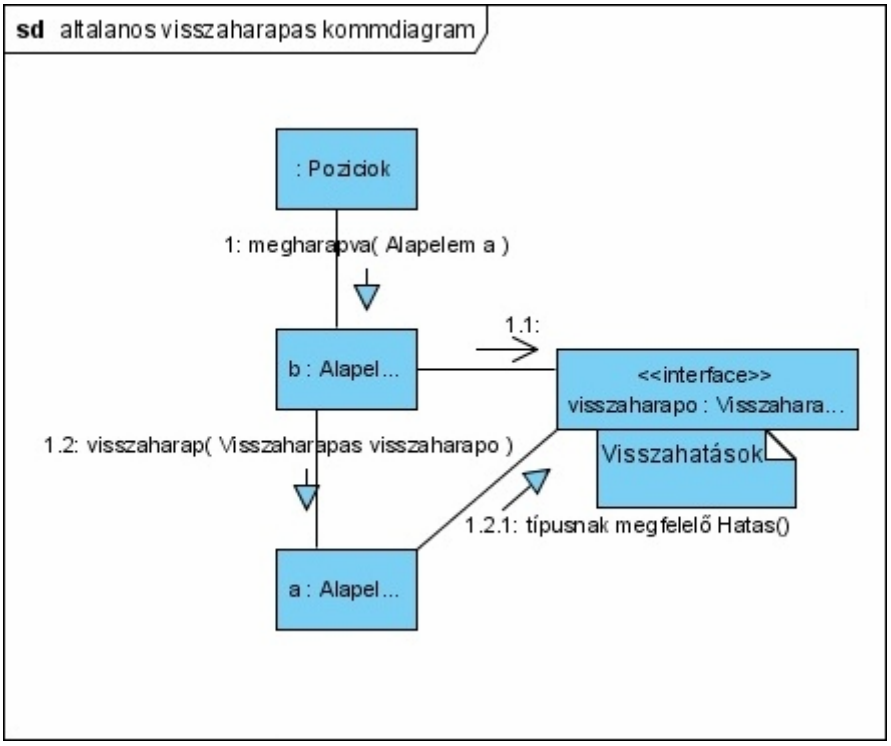
5.4.1. Inicializáció



5.4.2. Motor step



5.4.3. Ütközés



5.4.4. Megjegyzések

Ezek a kommunikációs diagramok a korábbi, 3.-as fejezetben lévő szekvencia diagramok alapján készültek. Az egyes speciális ütközések kommunikációs diagramját nem közöltük, lényegében nincs különbség az általános ütközéshez képest, csak a résztvevő osztályok mások (ezek szekvencia diagramjai viszont megtalálhatóak a korábbi fejezetben).

NetBeans-ben szekvencia diagramot át lehet konvertálni kommunikációssá (ő még a régi nevet használja: kollaborációs diagram), de a nyilak és a rajtuk lévő feliratok illetve a számozás teljesen össze vannak keveredve, és kézi szerkesztés sem segít rajtuk, így végül más eszközt, Visual Paradigm-et használtunk.

6. fejezet Szkeleton beadása

6.1. Útmutatás a szkeletonhoz

6.1.1. Fájllista

Fájl neve	*Fájl mérete*	*Keletkezési idő*	*Leírás*
Tetragon_Szkeleton\javadoc	819 585 bájt	2008. március 18., 23:09:32	Ez a könyvtár tartalmazza a Javadoc speciális kommentekből és a forrásfájlokból generált fejlesztői referenciát.
Tetragon_Szkeleton\src\tetragonteamkukacfarm	106 573 bájt	2008. március 18., 23:09:55	Ez a könyvtár tartalmazza a forrásfájlokat, a továbbiakban nem írjuk ki a teljes elérési útvonalat.
%\Alapelem.java	4 670 bájt	2008. március 18., 23:09:55	Az alapelem az objektumrendszer egy alapvető, absztrakt eleme, amelyből minden olyan osztály leszármazik, amely részt vesz a játéktér kialakításában. Minden elemnek lehet helye mérete és azonosítója.
%\Bogyo.java	3 346 bájt	2008. március 18., 23:09:55	A bogyó egy absztrakt osztály, belőle származnak le az egyes bogyófajták. A bogyó osztály az alapelemből származik le.
%\Bokor.java	6 734 bájt	2008. március 18., 23:09:55	A bokor az az osztály, amely egy listát tartalmaz a pályán jelenlévő bogyókról. Tárolja, kezeli és menedzseli a pályán lévő bogyókat. Felel a bogyók létrehozásáért és törléséért. Egy példány létezik belőle.
%\FoldiBogyo.java	2 521 bájt	2008. március 18., 23:09:55	A bogyó osztály leszármazottja, a játékspecifikációban leírt mezei bogyót valósítja meg.
%\FoldiBogyoVisszaharapas.java	5 808 bájt	2008. március 18., 23:09:55	Ha a megharapott alapelem földibogyó, akkor az földibogyóvisszaharapás objektumot hoz létre, és átadja azt a kölcsönhatásban résztvevő félnek, a látogató modell szerint. A visszaharapás interface-t valósítja meg.
%\FureszBogyo.java	2 518 bájt	2008. március 18., 23:09:55	A bogyó osztály leszármazottja, a játékspecifikációban leírt fűrészbogyót valósítja meg.
%\FureszBogyoVisszaharapas.java	6 019 bájt	2008. március 18., 23:09:55	Ha a megharapott alapelem fűrészbogyó, akkor az fűrészbogyóvisszaharapás objektumot hoz létre, és átadja azt a kölcsönhatásban résztvevő félnek, a látogató modell szerint. A visszaharapás interface-t valósítja meg.
%\JatekTerep.java	3 008 bájt	2008. március 18., 23:09:55	A játékterep az az osztály, amely egy listát tartalmaz a pályán jelenlévő fal objektumokról. Tárolja, kezeli és menedzseli őket. Felel a falak létrehozásáért. Egy példány létezik belőle.
%\KoBogyo.java	2 474 bájt	2008. március 18., 23:09:55	A bogyó osztály leszármazottja, a játékspecifikációban leírt kőbogyót valósítja meg.
%\KoBogyoVisszaharapas.java	5 849 bájt	2008. március 18., 23:09:55	Ha a megharapott alapelem kőbogyó, akkor az kőbogyóvisszaharapás objektumot hoz létre, és átadja azt a kölcsönhatásban résztvevő félnek, a látogató modell szerint. A visszaharapás interface-t valósítja meg.
%\Kukac.java	8 158 bájt	2008. március 18., 23:09:55	Egy kukacot reprezentáló osztály, rendelkezik attribútumként egy olyan változóval hogy életben van e, ismeri saját mozgásának irányát, képes saját hosszát megnövelni, megrövidíteni, lépni és tárolja szelvényeit.
%\Kukacok.java	4 355 bájt	2008. március 18., 23:09:55	Egy homogén kollekció Kukacok tárolására és az azokon elvégzendő alapvető műveletek koordinálására.
%\Motor.java	9 416	2008. március	A játék léptetéséért az objektumok összehangolásával

	bájt	18., 23:09:55	foglalkozó objektum. Az időzítéshez szálat használ, melyet majd a proto fázis után implementálunk.
%\Pozicio.java	4 211 bájt	2008. március 18., 23:09:55	Az osztály Alapelem típusú objektumok térbeli pozíciójának tárolására szolgál.
%\Poziciok.java	2 803 bájt	2008. március 18., 23:09:55	A Pozicio-k tárolására szolgáló osztály.
%\SkeletonAux.java	4 731 bájt	2008. március 18., 23:09:55	A szkeletonhoz az egyes függvényhívások kiírását megkönnyítő osztály.
%\SkeletonDialog.java	1 137 bájt	2008. március 18., 23:09:55	A szkeletonhoz segítség, egyszerűen lehet vele int és boolean bemenetet beolvasni, hibakezelés nincs, a bemenet helyes megadása a mi felelősségünk. Csak fejlesztőknek.
%\Szelveny.java	7 823 bájt	2008. március 18., 23:09:55	A Kukac alapvető eleme.
%\SzelvenyVisszaharapas.java	6 023 bájt	2008. március 18., 23:09:55	Ha a megharapott alapelem szelvény, akkor az szelvényvisszaharapás objektumot hoz létre, és átadja azt a kölcsönhatásban résztvevő félnek, a látogató modell szerint. A visszaharapás interface-t valósítja meg.
%\Tegla.java	4 051 bájt	2008. március 18., 23:09:55	Az alapelemből leszármazó osztály. A falat reprezentálja.
%\TeglaVisszaharaps.java	5 705 bájt	2008. március 18., 23:09:55	Ha a megharapott alapelem tégla, akkor az téglavisszaharapás objektumot hoz létre, és átadja azt a kölcsönhatásban résztvevő félnek, a látogató modell szerint. A visszaharapás interface-t valósítja meg.
%\Tetragon_Team_Kukacfarm.java	3 731 bájt	2008. március 18., 23:09:55	A Tetragon_Team_Kukacfarm osztály felelős az inicializáció elindításáért. Ő hozza létre a Motor osztályt, amely elvégzi az inicializációt. Egy példány létezik belőle. Új játékot lehet vele indítani, és meglévőt lehet törölni.
%\Visszaharapas.java	1 482 bájt	2008. március 18., 23:09:55	A visszaharapás interface a visszaharapások kohézív tulajdonságainak összessége. Leírja, hogyan kell kinéznie az őt megvalósító interfacek-nek. A függvények leírása az adott – interface-t megvalósító – osztály leírásában található.

6.1.2. Útmutatás

Fordítás

A fordítás legegyszerűbben parancssorból történhet. Amennyiben a java fordító és futtató fájl elérési helye környezeti változó, a fordítás könnyen megy. Ha nem, akkor kénytelenek vagyunk begépelni a fordító teljes elérési útját. A következő parancsot kell kiadnunk (Tetragon_Szkeleton\src legyen az aktuális könyvtár; a class-ok package-ben vannak, ezért kell innen kiadni a parancsot):

```
X:\...\Tetragon_Szkeleton\src>"Y:\...\Java\jdk...\bin\javac.exe" -encoding utf8
tetragonteamkukacfarm\Tetragon_Team_Kukacfarm.java
```

A karakterkódolással is problémák lehetnek, ugyanis a konzolra való kiírások ill. a kommentek ékezetes karaktereket is tartalmaznak és a szöveges forrásfájlok UTF-8-ban lettek kódolva, így ezt is meg kell adni.

Futtatás

Hasonlóképpen járunk el, mint az előző esetben, csak most a futtatót indítjuk el a fő osztállyal paraméterezve:

```
X:\...\Tetragon_Szkeleton\src>"Y:\...\Java\jdk...\bin\java.exe"
```

tetragonteamkukacfarm.Tetragon_Team_Kukacfarm

Ha a program elindul kétfajta kérdést tehet fel:

- eldöntendőt, ilyenkor „true”-val vagy „false”-al válaszolhatunk
- egy pozitív egész számot kér be

Hibás bemenet esetén a program futása azon nyomban megakad, erre vigyázzunk. Ha nem veszünk fel legalább egy kukacot legalább egy szelvénnyel, akkor az ütközések rész nem fut le, pontosabban elszáll.

6.1.3. A szkeleton egy kimenete

```
[3.3.1. Inicializáció]
14978587:Tetragon_Team_Kukacfarm.Tetragon_Team_Kukacfarm () | CALL
   30518483:Motor.Motor (2:Integer, 20:Integer, 7320:Dimension) | CALL
      8918249:Poziciok.Poziciok () | CALL
      8918249:Poziciok.Poziciok () | RETURN
      7615385:JatekTerep.JatekTerep (7320:Dimension) | CALL
      Hány téglát vegyek fel?
1
         33219526:Pozicio.Pozicio () | CALL
         33219526:Pozicio.Pozicio () | RETURN
         8388097:Tegla.Alapelem (33219526:Pozicio, 0:Dimension) | CALL
         8388097:Tegla.Alapelem (33219526:Pozicio, 0:Dimension) | RETURN
         8388097:Tegla.Tegla (33219526:Pozicio, 0:Dimension, 7615385:JatekTerep) | CALL
         8388097:Tegla.Tegla (33219526:Pozicio, 0:Dimension, 7615385:JatekTerep) | RETURN
         7615385:JatekTerep.addTegla (8388097:Tegla) | CALL
            8388097:Tegla.getPozicio () | CALL
            8388097:Tegla.getPozicio () | RETURN
            33219526:Pozicio.setAlapelem (8388097:Tegla) | CALL
            33219526:Pozicio.setAlapelem (8388097:Tegla) | RETURN
            8388097:Tegla.getPozicio () | CALL
            8388097:Tegla.getPozicio () | RETURN
            8918249:Poziciok.addPozicio (33219526:Pozicio) | CALL
            8918249:Poziciok.addPozicio (33219526:Pozicio) | RETURN
         7615385:JatekTerep.addTegla (8388097:Tegla) | RETURN
      7615385:JatekTerep.JatekTerep (7320:Dimension) | RETURN
      7615385:JatekTerep.keretez (7320:Dimension) | CALL
      7615385:JatekTerep.keretez (7320:Dimension) | RETURN
      4121220:Bokor.Bokor () | CALL
      Hány fűrészbogyót vegyek fel?
1
         8814217:Pozicio.Pozicio () | CALL
         8814217:Pozicio.Pozicio () | RETURN
         31289430:FureszBogyo.Alapelem (8814217:Pozicio, 0:Dimension) | CALL
         31289430:FureszBogyo.Alapelem (8814217:Pozicio, 0:Dimension) | RETURN
         31289430:FureszBogyo.Bogyo (8814217:Pozicio, 0:Dimension, 4121220:Bokor) | CALL
         31289430:FureszBogyo.Bogyo (8814217:Pozicio, 0:Dimension, 4121220:Bokor) | RETURN
         31289430:FureszBogyo.FureszBogyo (8814217:Pozicio, 0:Dimension, 4121220:Bokor) | CALL
         31289430:FureszBogyo.FureszBogyo (8814217:Pozicio, 0:Dimension, 4121220:Bokor) | RETURN
         8814217:Pozicio.setAlapelem (31289430:FureszBogyo) | CALL
         8814217:Pozicio.setAlapelem (31289430:FureszBogyo) | RETURN
         8918249:Poziciok.addPozicio (8814217:Pozicio) | CALL
         8918249:Poziciok.addPozicio (8814217:Pozicio) | RETURN
      Hány kőbogyót vegyek fel?
1
         16399041:Pozicio.Pozicio () | CALL
         16399041:Pozicio.Pozicio () | RETURN
         2583282:KoBogyo.Alapelem (16399041:Pozicio, 0:Dimension) | CALL
         2583282:KoBogyo.Alapelem (16399041:Pozicio, 0:Dimension) | RETURN
         2583282:KoBogyo.Bogyo (16399041:Pozicio, 0:Dimension, 4121220:Bokor) | CALL
         2583282:KoBogyo.Bogyo (16399041:Pozicio, 0:Dimension, 4121220:Bokor) | RETURN
         2583282:KoBogyo.KoBogyo (16399041:Pozicio, 0:Dimension, 4121220:Bokor) | CALL
         2583282:KoBogyo.KoBogyo (16399041:Pozicio, 0:Dimension, 4121220:Bokor) | RETURN
         16399041:Pozicio.setAlapelem (2583282:KoBogyo) | CALL
         16399041:Pozicio.setAlapelem (2583282:KoBogyo) | RETURN
         8918249:Poziciok.addPozicio (16399041:Pozicio) | CALL
         8918249:Poziciok.addPozicio (16399041:Pozicio) | RETURN
      Hány földibogyót vegyek fel?
1
         31342381:Pozicio.Pozicio () | CALL
```

```
         31342381:Pozicio.Pozicio () | RETURN
         381299:FoldiBogyo.Alapelem (31342381:Pozicio, 0:Dimension) | CALL
         381299:FoldiBogyo.Alapelem (31342381:Pozicio, 0:Dimension) | RETURN
         381299:FoldiBogyo.Bogyo (31342381:Pozicio, 0:Dimension, 4121220:Bokor) | CALL
         381299:FoldiBogyo.Bogyo (31342381:Pozicio, 0:Dimension, 4121220:Bokor) | RETURN
         381299:FoldiBogyo.FoldiBogyo (31342381:Pozicio, 0:Dimension, 4121220:Bokor) | CALL
         381299:FoldiBogyo.FoldiBogyo (31342381:Pozicio, 0:Dimension, 4121220:Bokor) | RETURN
         31342381:Pozicio.setAlapelem (381299:FoldiBogyo) | CALL
         31342381:Pozicio.setAlapelem (381299:FoldiBogyo) | RETURN
         8918249:Poziciok.addPozicio (31342381:Pozicio) | CALL
         8918249:Poziciok.addPozicio (31342381:Pozicio) | RETURN
      4121220:Bokor.Bokor () | RETURN
      4121220:Bokor.setBogyokMaxSzama (20:Integer) | CALL
      4121220:Bokor.setBogyokMaxSzama (20:Integer) | RETURN
      24464082:Kukacok.Kukacok (2:Integer) | CALL
      Hány kukacot vegyek fel?
2
         16671929:Kukac.Kukac (0:Point, 1077280768:Double) | CALL
         Hány szelvényt vegyek fel?
3
            25615188:Pozicio.Pozicio () | CALL
            25615188:Pozicio.Pozicio () | RETURN
            11107083:Szelveny.Alapelem (25615188:Pozicio, 0:Dimension) | CALL
            11107083:Szelveny.Alapelem (25615188:Pozicio, 0:Dimension) | RETURN
            11107083:Szelveny.Szelveny (25615188:Pozicio, 0:Dimension, 16671929:Kukac, 0:Integer,
1237:Boolean) | CALL
            11107083:Szelveny.Szelveny (25615188:Pozicio, 0:Dimension, 16671929:Kukac, 0:Integer,
1237:Boolean) | RETURN
            25615188:Pozicio.setAlapelem (11107083:Szelveny) | CALL
            25615188:Pozicio.setAlapelem (11107083:Szelveny) | RETURN
            8918249:Poziciok.addPozicio (25615188:Pozicio) | CALL
            8918249:Poziciok.addPozicio (25615188:Pozicio) | RETURN
            13446204:Pozicio.Pozicio () | CALL
            13446204:Pozicio.Pozicio () | RETURN
            1267757:Szelveny.Alapelem (13446204:Pozicio, 0:Dimension) | CALL
            1267757:Szelveny.Alapelem (13446204:Pozicio, 0:Dimension) | RETURN
            1267757:Szelveny.Szelveny (13446204:Pozicio, 0:Dimension, 16671929:Kukac, 0:Integer,
1237:Boolean) | CALL
            1267757:Szelveny.Szelveny (13446204:Pozicio, 0:Dimension, 16671929:Kukac, 0:Integer,
1237:Boolean) | RETURN
            13446204:Pozicio.setAlapelem (1267757:Szelveny) | CALL
            13446204:Pozicio.setAlapelem (1267757:Szelveny) | RETURN
            8918249:Poziciok.addPozicio (13446204:Pozicio) | CALL
            8918249:Poziciok.addPozicio (13446204:Pozicio) | RETURN
            138093:Pozicio.Pozicio () | CALL
            138093:Pozicio.Pozicio () | RETURN
            5678233:Szelveny.Alapelem (138093:Pozicio, 0:Dimension) | CALL
            5678233:Szelveny.Alapelem (138093:Pozicio, 0:Dimension) | RETURN
            5678233:Szelveny.Szelveny (138093:Pozicio, 0:Dimension, 16671929:Kukac, 0:Integer,
1237:Boolean) | CALL
            5678233:Szelveny.Szelveny (138093:Pozicio, 0:Dimension, 16671929:Kukac, 0:Integer,
1237:Boolean) | RETURN
            138093:Pozicio.setAlapelem (5678233:Szelveny) | CALL
            138093:Pozicio.setAlapelem (5678233:Szelveny) | RETURN
            8918249:Poziciok.addPozicio (138093:Pozicio) | CALL
            8918249:Poziciok.addPozicio (138093:Pozicio) | RETURN
         16671929:Kukac.Kukac (0:Point, 1077280768:Double) | RETURN
         5268497:Kukac.Kukac (0:Point, 1077280768:Double) | CALL
         Hány szelvényt vegyek fel?
3
            31038029:Pozicio.Pozicio () | CALL
            31038029:Pozicio.Pozicio () | RETURN
            18961126:Szelveny.Alapelem (31038029:Pozicio, 0:Dimension) | CALL
            18961126:Szelveny.Alapelem (31038029:Pozicio, 0:Dimension) | RETURN
            18961126:Szelveny.Szelveny (31038029:Pozicio, 0:Dimension, 5268497:Kukac, 0:Integer,
1237:Boolean) | CALL
            18961126:Szelveny.Szelveny (31038029:Pozicio, 0:Dimension, 5268497:Kukac, 0:Integer,
1237:Boolean) | RETURN
            31038029:Pozicio.setAlapelem (18961126:Szelveny) | CALL
            31038029:Pozicio.setAlapelem (18961126:Szelveny) | RETURN
            8918249:Poziciok.addPozicio (31038029:Pozicio) | CALL
            8918249:Poziciok.addPozicio (31038029:Pozicio) | RETURN
            13220408:Pozicio.Pozicio () | CALL
```

```
            13220408:Pozicio.Pozicio () | RETURN
            31505976:Szelveny.Alapelem (13220408:Pozicio, 0:Dimension) | CALL
            31505976:Szelveny.Alapelem (13220408:Pozicio, 0:Dimension) | RETURN
            31505976:Szelveny.Szelveny (13220408:Pozicio, 0:Dimension, 5268497:Kukac, 0:Integer,
1237:Boolean) | CALL
            31505976:Szelveny.Szelveny (13220408:Pozicio, 0:Dimension, 5268497:Kukac, 0:Integer,
1237:Boolean) | RETURN
            13220408:Pozicio.setAlapelem (31505976:Szelveny) | CALL
            13220408:Pozicio.setAlapelem (31505976:Szelveny) | RETURN
            8918249:Poziciok.addPozicio (13220408:Pozicio) | CALL
            8918249:Poziciok.addPozicio (13220408:Pozicio) | RETURN
            32004544:Pozicio.Pozicio () | CALL
            32004544:Pozicio.Pozicio () | RETURN
            22837328:Szelveny.Alapelem (32004544:Pozicio, 0:Dimension) | CALL
            22837328:Szelveny.Alapelem (32004544:Pozicio, 0:Dimension) | RETURN
            22837328:Szelveny.Szelveny (32004544:Pozicio, 0:Dimension, 5268497:Kukac, 0:Integer,
1237:Boolean) | CALL
            22837328:Szelveny.Szelveny (32004544:Pozicio, 0:Dimension, 5268497:Kukac, 0:Integer,
1237:Boolean) | RETURN
            32004544:Pozicio.setAlapelem (22837328:Szelveny) | CALL
            32004544:Pozicio.setAlapelem (22837328:Szelveny) | RETURN
            8918249:Poziciok.addPozicio (32004544:Pozicio) | CALL
            8918249:Poziciok.addPozicio (32004544:Pozicio) | RETURN
         5268497:Kukac.Kukac (0:Point, 1077280768:Double) | RETURN
      24464082:Kukacok.Kukacok (2:Integer) | RETURN
   30518483:Motor.Motor (2:Integer, 20:Integer, 7320:Dimension) | RETURN
[3.3.2. Motor step]
[i=0 iteráció]
   30518483:Motor.run () | CALL
      24464082:Kukacok.leptet () | CALL
         16671929:Kukac.leptet () | CALL
            5678233:Szelveny.leptet () | CALL
            5678233:Szelveny.leptet () | RETURN
         16671929:Kukac.leptet () | RETURN
         5268497:Kukac.leptet () | CALL
            22837328:Szelveny.leptet () | CALL
            22837328:Szelveny.leptet () | RETURN
         5268497:Kukac.leptet () | RETURN
      24464082:Kukacok.leptet () | RETURN
      4121220:Bokor.leptet () | CALL
      4121220:Bokor.leptet () | RETURN
   30518483:Motor.run () | RETURN
Végrehajtsam a következő ciklust a motorban?
true
[i=1 iteráció]
   30518483:Motor.run () | CALL
      24464082:Kukacok.leptet () | CALL
         16671929:Kukac.leptet () | CALL
            5678233:Szelveny.leptet () | CALL
            5678233:Szelveny.leptet () | RETURN
         16671929:Kukac.leptet () | RETURN
         5268497:Kukac.leptet () | CALL
            22837328:Szelveny.leptet () | CALL
            22837328:Szelveny.leptet () | RETURN
         5268497:Kukac.leptet () | RETURN
      24464082:Kukacok.leptet () | RETURN
      4121220:Bokor.leptet () | CALL
      4121220:Bokor.leptet () | RETURN
   30518483:Motor.run () | RETURN
Végrehajtsam a következő ciklust a motorban?
true
[i=2 iteráció]
   30518483:Motor.run () | CALL
      24464082:Kukacok.leptet () | CALL
         16671929:Kukac.leptet () | CALL
            5678233:Szelveny.leptet () | CALL
            5678233:Szelveny.leptet () | RETURN
         16671929:Kukac.leptet () | RETURN
         5268497:Kukac.leptet () | CALL
            22837328:Szelveny.leptet () | CALL
            22837328:Szelveny.leptet () | RETURN
         5268497:Kukac.leptet () | RETURN
      24464082:Kukacok.leptet () | RETURN
```

```
      4121220:Bokor.leptet () | CALL
      4121220:Bokor.leptet () | RETURN
   30518483:Motor.run () | RETURN
Végrehajtsam a következő ciklust a motorban?
false
[3.3.3. Ütközés és 3.3.6. Kőbogyóharapás]
   24464082:Kukacok.getKukac (0:Integer) | CALL
   24464082:Kukacok.getKukac (0:Integer) | RETURN
[Kőbogyó létrehozása]
   31379709:Pozicio.Pozicio () | CALL
   31379709:Pozicio.Pozicio () | RETURN
   23764290:KoBogyo.Alapelem (31379709:Pozicio, 1012:Dimension) | CALL
   23764290:KoBogyo.Alapelem (31379709:Pozicio, 1012:Dimension) | RETURN
   23764290:KoBogyo.Bogyo (31379709:Pozicio, 1012:Dimension, 4121220:Bokor) | CALL
   23764290:KoBogyo.Bogyo (31379709:Pozicio, 1012:Dimension, 4121220:Bokor) | RETURN
   23764290:KoBogyo.KoBogyo (31379709:Pozicio, 1012:Dimension, 4121220:Bokor) | CALL
   23764290:KoBogyo.KoBogyo (31379709:Pozicio, 1012:Dimension, 4121220:Bokor) | RETURN
[Kőbogyó megharapása]
   16671929:Kukac.getSzelveny (0:Integer) | CALL
   16671929:Kukac.getSzelveny (0:Integer) | RETURN
   23764290:KoBogyo.megharapva (5678233:Szelveny) | CALL
      21171036:KoBogyoVisszaharapas.KoBogyoVisszaharapas (23764290:KoBogyo) | CALL
      21171036:KoBogyoVisszaharapas.KoBogyoVisszaharapas (23764290:KoBogyo) | RETURN
      5678233:Szelveny.visszaharap (21171036:KoBogyoVisszaharapas) | CALL
         21171036:KoBogyoVisszaharapas.SzelvenyHarap (5678233:Szelveny) | CALL
            5678233:Szelveny.getKukaca () | CALL
            5678233:Szelveny.getKukaca () | RETURN
            16671929:Kukac.megno () | CALL
            16671929:Kukac.megno () | RETURN
            16671929:Kukac.getSzelveny (0:Integer) | CALL
            16671929:Kukac.getSzelveny (0:Integer) | RETURN
            5678233:Szelveny.setKoBogyoHatas (1231:Boolean) | CALL
            5678233:Szelveny.setKoBogyoHatas (1231:Boolean) | RETURN
            23764290:KoBogyo.getBokra () | CALL
            23764290:KoBogyo.getBokra () | RETURN
            4121220:Bokor.torolBogyo () | CALL
            4121220:Bokor.torolBogyo () | RETURN
         21171036:KoBogyoVisszaharapas.SzelvenyHarap (5678233:Szelveny) | RETURN
      5678233:Szelveny.visszaharap (21171036:KoBogyoVisszaharapas) | RETURN
   23764290:KoBogyo.megharapva (5678233:Szelveny) | RETURN
[3.3.4. Földibogyóharapás]
[Földibogyó létrehozása]
   10410279:Pozicio.Pozicio () | CALL
   10410279:Pozicio.Pozicio () | RETURN
   12755250:FoldiBogyo.Alapelem (10410279:Pozicio, 1012:Dimension) | CALL
   12755250:FoldiBogyo.Alapelem (10410279:Pozicio, 1012:Dimension) | RETURN
   12755250:FoldiBogyo.Bogyo (10410279:Pozicio, 1012:Dimension, 4121220:Bokor) | CALL
   12755250:FoldiBogyo.Bogyo (10410279:Pozicio, 1012:Dimension, 4121220:Bokor) | RETURN
   12755250:FoldiBogyo.FoldiBogyo (10410279:Pozicio, 1012:Dimension, 4121220:Bokor) | CALL
   12755250:FoldiBogyo.FoldiBogyo (10410279:Pozicio, 1012:Dimension, 4121220:Bokor) | RETURN
[Földibogyó megharapása]
   16671929:Kukac.getSzelveny (0:Integer) | CALL
   16671929:Kukac.getSzelveny (0:Integer) | RETURN
   12755250:FoldiBogyo.megharapva (5678233:Szelveny) | CALL
      16416372:FoldiBogyoVisszaharapas.FoldiBogyoVisszaharapas (12755250:FoldiBogyo) | CALL
      16416372:FoldiBogyoVisszaharapas.FoldiBogyoVisszaharapas (12755250:FoldiBogyo) | RETURN
      5678233:Szelveny.visszaharap (16416372:FoldiBogyoVisszaharapas) | CALL
         16416372:FoldiBogyoVisszaharapas.SzelvenyHarap (5678233:Szelveny) | CALL
            5678233:Szelveny.getKukaca () | CALL
            5678233:Szelveny.getKukaca () | RETURN
            16671929:Kukac.megno () | CALL
            16671929:Kukac.megno () | RETURN
            12755250:FoldiBogyo.getBokra () | CALL
            12755250:FoldiBogyo.getBokra () | RETURN
            4121220:Bokor.torolBogyo () | CALL
            4121220:Bokor.torolBogyo () | RETURN
         16416372:FoldiBogyoVisszaharapas.SzelvenyHarap (5678233:Szelveny) | RETURN
      5678233:Szelveny.visszaharap (16416372:FoldiBogyoVisszaharapas) | RETURN
   12755250:FoldiBogyo.megharapva (5678233:Szelveny) | RETURN
[3.3.5. Fűrészbogyóharapás]
[Fűrészbogyó létrehozása]
   25425741:Pozicio.Pozicio () | CALL
   25425741:Pozicio.Pozicio () | RETURN
```

```
   14746332:FureszBogyo.Alapelem (25425741:Pozicio, 1012:Dimension) | CALL
   14746332:FureszBogyo.Alapelem (25425741:Pozicio, 1012:Dimension) | RETURN
   14746332:FureszBogyo.Bogyo (25425741:Pozicio, 1012:Dimension, 4121220:Bokor) | CALL
   14746332:FureszBogyo.Bogyo (25425741:Pozicio, 1012:Dimension, 4121220:Bokor) | RETURN
   14746332:FureszBogyo.FureszBogyo (25425741:Pozicio, 1012:Dimension, 4121220:Bokor) | CALL
   14746332:FureszBogyo.FureszBogyo (25425741:Pozicio, 1012:Dimension, 4121220:Bokor) | RETURN
[Fűrészbogyó megharapása]
   16671929:Kukac.getSzelveny (0:Integer) | CALL
   16671929:Kukac.getSzelveny (0:Integer) | RETURN
   14746332:FureszBogyo.megharapva (5678233:Szelveny) | CALL
      1259414:FureszBogyoVisszaharapas.FureszBogyoVisszaharapas (14746332:FureszBogyo) | CALL
      1259414:FureszBogyoVisszaharapas.FureszBogyoVisszaharapas (14746332:FureszBogyo) | RETURN
      5678233:Szelveny.visszaharap (1259414:FureszBogyoVisszaharapas) | CALL
         1259414:FureszBogyoVisszaharapas.SzelvenyHarap (5678233:Szelveny) | CALL
            5678233:Szelveny.getKukaca () | CALL
            5678233:Szelveny.getKukaca () | RETURN
            16671929:Kukac.megno () | CALL
            16671929:Kukac.megno () | RETURN
         Fűrészhatás élettartama?
4
            5678233:Szelveny.setFureszHatas (4:Integer) | CALL
            5678233:Szelveny.setFureszHatas (4:Integer) | RETURN
            14746332:FureszBogyo.getBokra () | CALL
            14746332:FureszBogyo.getBokra () | RETURN
            4121220:Bokor.torolBogyo () | CALL
            4121220:Bokor.torolBogyo () | RETURN
         1259414:FureszBogyoVisszaharapas.SzelvenyHarap (5678233:Szelveny) | RETURN
      5678233:Szelveny.visszaharap (1259414:FureszBogyoVisszaharapas) | RETURN
   14746332:FureszBogyo.megharapva (5678233:Szelveny) | RETURN
[3.3.7. Szelvényharapás]
[Szelvény létrehozása]
   27687351:Pozicio.Pozicio () | CALL
   27687351:Pozicio.Pozicio () | RETURN
   8451275:Szelveny.Alapelem (27687351:Pozicio, 1012:Dimension) | CALL
   8451275:Szelveny.Alapelem (27687351:Pozicio, 1012:Dimension) | RETURN
   8451275:Szelveny.Szelveny (27687351:Pozicio, 1012:Dimension, 16671929:Kukac, 0:Integer,
1237:Boolean) | CALL
   8451275:Szelveny.Szelveny (27687351:Pozicio, 1012:Dimension, 16671929:Kukac, 0:Integer,
1237:Boolean) | RETURN
[Szelvény megharapása]
   16671929:Kukac.getSzelveny (0:Integer) | CALL
   16671929:Kukac.getSzelveny (0:Integer) | RETURN
   8451275:Szelveny.megharapva (5678233:Szelveny) | CALL
      3374351:SzelvenyVisszaharapas.SzelvenyVisszaharapas (8451275:Szelveny) | CALL
      3374351:SzelvenyVisszaharapas.SzelvenyVisszaharapas (8451275:Szelveny) | RETURN
      5678233:Szelveny.visszaharap (3374351:SzelvenyVisszaharapas) | CALL
         3374351:SzelvenyVisszaharapas.SzelvenyHarap (8451275:Szelveny) | CALL
            8451275:Szelveny.getFureszHatas () | CALL
            8451275:Szelveny.getFureszHatas () | RETURN
            5678233:Szelveny.getFureszHatas () | CALL
            5678233:Szelveny.getFureszHatas () | RETURN
         Meghal a másik kígyó?
true
            8451275:Szelveny.getKukaca () | CALL
            8451275:Szelveny.getKukaca () | RETURN
            16671929:Kukac.setEleMeg (1231:Boolean) | CALL
            16671929:Kukac.setEleMeg (1231:Boolean) | RETURN
            5678233:Szelveny.getKukaca () | CALL
            5678233:Szelveny.getKukaca () | RETURN
            16671929:Kukac.rovidul (5678233:Szelveny) | CALL
            16671929:Kukac.rovidul (5678233:Szelveny) | RETURN
         3374351:SzelvenyVisszaharapas.SzelvenyHarap (8451275:Szelveny) | RETURN
      5678233:Szelveny.visszaharap (3374351:SzelvenyVisszaharapas) | RETURN
   8451275:Szelveny.megharapva (5678233:Szelveny) | RETURN
[3.3.8. Téglaharapás]
[Tégla létrehozása]
   5737707:Pozicio.Pozicio () | CALL
   5737707:Pozicio.Pozicio () | RETURN
   31771588:Tegla.Alapelem (5737707:Pozicio, 1012:Dimension) | CALL
   31771588:Tegla.Alapelem (5737707:Pozicio, 1012:Dimension) | RETURN
   31771588:Tegla.Tegla (5737707:Pozicio, 1012:Dimension, 7615385:JatekTerep) | CALL
   31771588:Tegla.Tegla (5737707:Pozicio, 1012:Dimension, 7615385:JatekTerep) | RETURN
[Tégla megharapása]
```

```
   16671929:Kukac.getSzelveny (0:Integer) | CALL
   16671929:Kukac.getSzelveny (0:Integer) | RETURN
   31771588:Tegla.megharapva (5678233:Szelveny) | CALL
      11730319:TeglaVisszaharapas.TeglaVisszaharapas (31771588:Tegla) | CALL
      11730319:TeglaVisszaharapas.TeglaVisszaharapas (31771588:Tegla) | RETURN
      5678233:Szelveny.visszaharap (11730319:TeglaVisszaharapas) | CALL
         11730319:TeglaVisszaharapas.SzelvenyHarap (5678233:Szelveny) | CALL
            5678233:Szelveny.getKukaca () | CALL
            5678233:Szelveny.getKukaca () | RETURN
            16671929:Kukac.setEleMeg (1237:Boolean) | CALL
            16671929:Kukac.setEleMeg (1237:Boolean) | RETURN
         11730319:TeglaVisszaharapas.SzelvenyHarap (5678233:Szelveny) | RETURN
      5678233:Szelveny.visszaharap (11730319:TeglaVisszaharapas) | RETURN
   31771588:Tegla.megharapva (5678233:Szelveny) | RETURN
14978587:Tetragon_Team_Kukacfarm.Tetragon_Team_Kukacfarm () | RETURN
```

6.2. Értékelés

6.2.1. Deák véleménye

A csapatagok az első fázistól kezdve könnyen együttműködtek a korábbi ismeretségek miatt. Csapatfőnöknek Katust választottuk, ugyanis ő szeret legjobban elmélyedni az olyan apró „finomságokban”, melyre mások (mi) nem áldozunk időt, és könnyedén tud ilyen rendszer specifikus tanácsokkal ellátni minket. Ugyanakkor sokszor vissza kellett fognunk kalandvágyát, mert egy ekkora projekt esetén csak felesleges munkát okozott volna az általa tanácsolt menedzsment módszerek halmaza.

Végül közösen rávettük, hogy verziókövetésre és kommunikációra Paulik által gyártott fórumot használjuk, valamint az e-mailjeinket; és NetBeans-ben fejlesszünk. NetBeans-ben fejlesztettem az UML modell diagramokat, lassúságán és megbízhatatlanságán kívül azonban nem volt vele probléma, hatékonyan lehetett dolgozni. A kód szerkesztése is egyszerű volt.

A dokumentációkat mindenki elkészítette időben, verziókövetéssel mindössze egyszer adódott probléma a legelső alkalommal, amikor még nem rögzítettük a fórumhasználat szabályait.

A csapattagok mind kellő intenzitással dolgoztak, minden feladat időben és jól elkészült, így nagyszerű pontszámokat kaptunk rájuk.

Összességében elmondható, hogy az apróbb félreértések kiküszöbölésével, még hatékonyabban tudunk dolgozni, ugyanakkor az a néhány hiba ami akadt nem gátolta nagy mértékben a munkát.

6.2.2. Katus véleménye

A csapat már korán elkezdett szerveződni, tulajdonképpen már előző félévben eldöntöttük, hogy mi lesz a felállás. Jól ismerjük egymást, így azt is tudjuk, hogy mindenki komolyan veszi a saját dolgát, a rá kiosztott feladatot akár plusz tartalékok árán is elvégzi.

Csapatfőnökké választásom úgymond természetesen történt, mindenki egyet értett abban, hogy én koordináljak, és nem éreztem úgy, hogy rossz döntést hoztak a többiek.

Már a projekt elején fontosnak tartottam, hogy bevezessünk valamilyen verziókövető rendszert, így a munkák közös helyre való könnyű feltöltése és menedzselése lényegesen egyszerűen megoldható lett volna. Azonban nem sikerült meggyőznöm csapattársaim, így végül maradtunk a honlapunk és egy ftp szerver mellett.

Utóbbi döntés a párhuzamos munkavégzést is nagymértékben gátolja, hiszen később pl. kód írása esetén mergelhetőek lettek volna a beküldött szöveges fájlok. Eddig már kétszer megtörtént, hogy ugyanazon a fájlon kezdett el dolgozni két ember és később nehézkes volt

összefésülni kézzel a változtatásokat (a dokumentációval és forráskóddal is megtörtént). Látható, hogy hiába a megegyezés, hogy más nem nyúl bele az aktuális felelős ember fájljába, balesetek történnek és sokszor bosszúságot okoznak.

Csapatfőnökként elsősorban a feladatbeosztásért, a munka előrehaladásáért és a zökkenőmentes kommunikációért vagyok felelős. Ha valakinek problémája akad, nekem szól és közösen, mások bevonásával próbáljuk azt megoldani. A feladatok kiosztásakor igyekeztem figyelembe venni azt, hogy ki mennyire elfoglalt.

Eddig minden beadandó megszületését végigkövettem, beadás előtt én hagytam jóvá az adott részt. Ez is biztosította azt, hogy igényes dokumentumokat adjunk le, bár hozzá kell tegyem, egyetlen egyszer sem volt szükséges módosítás, ha az adott leadandóért felelős pár azt mondta, hogy az adott változat már végleges (persze előtte legalább kétszer átdolgozták). Egyedül mindig a külalakon kellett javítani, a dokumentációk végleges formáját mindig én alakítottam ki.

Az utólagos ellenőrzés persze nemcsak az én feladatom volt, mindenki más is bekapcsolódott, sőt ösztönöztem is őket, hogy vitázzunk az adott dokumentum helyességéről.

Az első beadandó egyes részeinek és a szkeleton részek kidolgozásának feladata jutott rám, ezeket főleg Ofellával együtt készítettük.

6.2.3. Ofella véleménye

A csapat tagjai – a korábbi ismeretségek miatt is – hamar összeszoktak és aktív, hatékony csapatmunka alakult ki.

A kezdeti fejlesztőkörnyezeti viták alatt eldőlt, hogy nem bonyolítjuk túl sem a verziókövetést, sem a dokumentációkészítést, hiszen egy ekkora méretű projekt esetén az egyszerűbb megoldások hatékonyabbak.

A csapaton belül Katus irányította a feladatok beosztását, próbált egyenletes eloszlást generálni, ami többé-kevésbé sikerült is. A csapattagok nem vették semmibe a munkát, és kezdettől fogva mindent beleadtak, határidőre mindent elkészítettek.

A NetBeans fejlesztőkörnyezet nem bizonyult a legszerencsésebb választásnak, mivel UML szerkesztője nem volt kellően testre szabható, de ez nem okozott komoly fennakadásokat a projektben. Ugyanakkor a kód- és javadoc-generálást egyszerűen el lehetett vele végezni, és a kódszerkesztésben is hatékony eszköznek bizonyult.

A fórumhasználatnak köszönhetően a naplókészítés és a közös egyeztetés zökkenőmentesen zajlott, a Microsoft Word pedig egyszerűsége miatt ideális szövegszerkesztő.

Összességében elmondható, hogy nem volt fennakadás, a csapat nagy lendülettel kezdte a munkát, és ugyanígy szándékozik folytatni azt.

Az én feladatom volt a követelmény definíció kidolgozása Deákkal, kommunikációs diagramok rajzolása, a generált forráskód kommentezése, illetve a többiek munkájának átnézése az összes szakaszban.

6.2.4. Paulik véleménye

A csapat még a félév kezdete előtt saját szerveződésben alakult meg, így próbáltuk meg garantálni a személyi problémák elkerülését a project során és azt, hogy ne legyenek olyan képességbeli különbségek, amelyek a feladat-elosztás egyensúlyát felborítanák.

A csapatvezetői posztra Katust jelöltük ki, mert ismeri a többi csapattag képességeit, így képes a faladatokat a megfelelő párosokra osztani, elősegítve a hatékony fejlesztést, illetve képes a csapattagokkal a határidőket betartatni.

Döntéseink során törekedtünk a problémák minél egyszerűbb megoldására, de úgy hogy az egyszerűség ne menjen a minőség rovására. Így választottuk közös kommunikációs felületnek, a hamar felállítható php alapú fórumot, illetve fejlesztőkörnyezetnek a NetBeanst. Döntéseink során mindig abból indultunk ki, hogy egy valós, nagy project esetén milyen eszközöket kellene alkalmaznunk és ehhez képest jelenlegi projectünk méretét figyelembe véve, milyen engedményeket/elhanyagolásokat engedhetünk meg magunknak. Így vetettük el a verzió követő rendszer használatát, a verziók követésére néhány egyszerű, könnyen betartható házi-szabályt hoztunk létre. Igyekeztünk a honlap nyújtotta lehetőségeket kihasználni, így a naplózást a megbeszéléseket és a fájlmegosztást is a portálon keresztül oldottuk meg.

A csapat a project első szakaszát az általunk elvártaknak megfelelően teljesítette, az előírt határidőre. Döntéseink helyességét a fejlesztés gördülékenysége, illetve a dokumentációkra kapott pontszámok is igazolták.

6.2.5. Százalékos mérték

Nehéz pontos számokat mondani, a munkából mindenki nagyjából egyenlő mértékben kivette a részét. A feladatok nehézsége is eltérő volt, de könnyebb részek esetén is sok időbe tellett egy-egy dokumentáció előállítása.

Talán Deák és Paulik némileg nagyobb százalékkal járult hozzá az eredményességünkhöz, hiszen ők vállalták el a legtöbbet érő rész, az analízis modell megalkotását. Paulik ráadásul a honlap üzemeltetését is vállalta.

Katus ugyanakkor sok időt töltött a dokumentációk összeállításával, Ofella pedig mindig hasznos tanácsokat nyújtott tulajdonképpen minden rész megszületése és beadása között.

Ezeket (is) figyelembe véve a következő értékeket gondoljuk reálisnak:

Deák László	27%
Katus Kristóf	23%
Ofella Péter	23%
Paulik Tamás	27%

7. fejezet Prototípus koncepciója

7.1. A specifikáció változásának kezelése

1. *A leharapott kígyófarok nem pusztul el, hanem egy új kígyóvá alakul (amely kezdetben NE a harapással őt létrehozó kígyófej irányába mozogjon!)*

- *Megoldás:* A modellben változtatást nem kell eszközölni, a megfelelő függvények törzsében lekezelhető a változás. A Kukac.rovidul(..) fv. törzsében az új Kukac létrehozható, melyhez nem fog felhasználói kontroll tartozni, egyenesen fog haladni, míg falnak nem ütközik.

2. *A kígyóban a kőbogyók vándorlási sebessége eltérhet (nagyobb vagy kisebb is lehet!) a kígyó mozgási sebességétől.*

- *Megoldás:* A Motor objektum elindít egy szálat ami a kőbogyók léptetését fogja felügyelni. Ehhez a Kukac objektumban egy leptetKoBogyo() fv.-t hozunk létre. Szükséges módosítások a modellben:
- leptetKoBogyo() fv. elhelyezése a Kukac osztályban
- A Kőbogyó léptetési sebesség felvétele a Motor konstruktorának paraméterei közé. A paraméter egy int ami a léptetési időközt jelenti millisec-ben
- A Motor osztály egy belső KoBogyoSzal extends Thread osztályt hoz létre a konstruktorában, mely adott időközönként meghívja a kukacok leptetKoBogyo() fv.-ét
- Ehhez módosítani kell a Motor osztály konstruktorának szekvencia diagramját, vagyis az „init”-et

7.2. Prototípus koncepció

7.2.1. Prototípus be-, és kimeneti fájlformátum

A felhasználó a megadott parancsok alapján felépíti a pályát és vezérli a kukacokat. A szimuláció bemenete a proto_input.txt, kimenete a proto_output.txt állomány.

A proto_input.txt formátuma:

A felhasználó az alábbi parancsokat adhatja a programnak:

cPalya(int szelesseg, int magassag)

Létrehoz egy pályát (egy keret, ami téglákat tartalmaz).

cKukac(int ID, int irany, int xpos, int ypos, int fureszbogyohatas, int hossz)

Létrehoz egy kukacot, ID azonosítóval, (xpos,ypos) koordinátákra, int ideig tartó fűrészbogyó-hatással, hossz hosszan.

cFoldi(int xpos, int ypos)

Létrehoz egy földibogyót (xpos,ypos) koordinátára.

cFures(int xpos, int ypos)

Létrehoz egy fűrészbogyót (xpos,ypos) koordinátára.

cKobog(int xpos, int ypos)

Létrehoz egy kőbogyót (xpos,ypos) koordinátára.

cTegla(int xpos, int ypos)

Létrehoz egy téglát (xpos, ypos) koordinátára.

dBogyo(int xpos, int ypos)

Töröl egy bogyót (xpos,ypos) koordinátáról, ha van ott bogyó.

dTegla(int xpos, int ypos)

Töröl egy téglát (xpos,ypos) koordinátáról, ha van ott bogyó.

stepKB

Lépteti a kukacokban lévő kőbogyókat.

stepAl

Lépteti a kukacokat. Lényegében a motor egy lépésének lefuttatása.

bal###(int ID)

ID számú kukacot forgatja balra ### fokkal. Pl.: bal012(1)

job###(int ID)

ID számú kukacot forgatja jobbra ### fokkal. Pl.: job012(1)

A bemenet hibáit a program nem kezeli, hanem hibaüzenet kiírása után futása megáll.

Példa:

```
cKukac(1,0,5,5,0,3)
cKobog(5,4)
stepAl
```

A proto_output.txt formátuma:

A fájl minden egyes lépést tartalmazni fog az alábbi formátumban:

1. Előző lépés óta kiadott parancsok (lásd bementi formátum)
2. Földibogyók pozíciója (fb xpos, ypos)
3. Kőbogyók pozíciója (kb xpos, ypos)
4. Fűrészbogyók pozíciója (fu xpos, ypos)
5. Téglák pozíciója (te xpos, ypos)
6. Kukac azonosítója, fejének pozíciója, fűrészhatás (ku#, xpos, ypos, hatás)
    a. szelvényeik pozíciója, kőbogyóhatás (sz#, xpos, ypos, hatás)

Példa (feltételezve, hogy már korábban létezett egy tégla, egy földibogyó és egy másik kígyó):

```
**************************
cKukac(1,0,5,5,0,3)
cKobog(5,4)
stepAl
fb 25, 25
kb 5, 4
te 0, 0
ku0, 30, 30, 0
      sz1, 35, 35, 0
      sz2, 40, 40, 1
ku1, 5, 5, 3
      sz1, 5, 10, 0
      sz2, 5, 15, 1
      sz3, 5, 20, 1
**************************
```

7.2.2. Be- és kimeneti hibák észlelése, kezelése

Bármilyen bemeneti hiba esetén (hiányzó file, szintaktikai hiba, stb.) esetén hibára vonatkozó üzenettel leáll a program futása.

7.3. A tesztelés menete

A program különböző funkcióinak tesztelésére létrehozunk tesztforgatókönyveket. Ezeket futtatjuk, a produkált és elvárt kimenetek soronkénti összehasonlításával kiértékeljük, először emberi erővel, majd miután meggyőződtünk a kimenetek helyességéről, a továbbiakban pedig egy szövegfájl-összehasonlító program értékeli ki az eredményt. Az összehasonlító program helyes működését a kézileg ellenőrzött kimenet párokkal ellenőrizzük.

A teszteket többször kell elvégezni, hiszen a programnak ugyanolyan bemenetekre ugyanúgy kell reagálnia. A tesztelést több környezeten és különböző gépeken is elvégezzük. A Java nyelv biztosítja a platformfüggetlenséget.

7.4. Tesztelést támogató segéd- és fordító programok specifikálása

A modellt egy minimális változtatással felkészítjük arra, hogy képes legyen a parancsokra reagálni.

Az új függvények:

Bokor
- bool addBogyo(Bogyo b)
- void torolBogyot(Pozicio p)

Poziciok
- Pozicio csinaljPoz(int x, int y)

JatekTerep
- bool addTeglat(Tegla t)
- torolTegla(Pozicio p)

Kukacok
- addKukac(Kukac k)
- forgatKukac(int ID, int szog)
- leptetKobogyo()

Továbbá az Alapelem, Bokor, Kukac, Kukacok, Játékterep osztályokat elláttuk egy kiir(string file) függvénnyel, ami a kimeneti állományba való kiíráshoz szükséges.

A szöveges állományból történő vezérlést a Motor.run()-ban valósítjuk meg, hiszen a végleges verzióban is ez végzi majd a léptetést.

A tesztelést támogató segédprogram egy két szöveges állomány karakterenkénti összehasonlítását végző program lesz, majd eltérés esetén az eltérés helyét is megadja.

7.4.1. Tesztforgatókönyvek

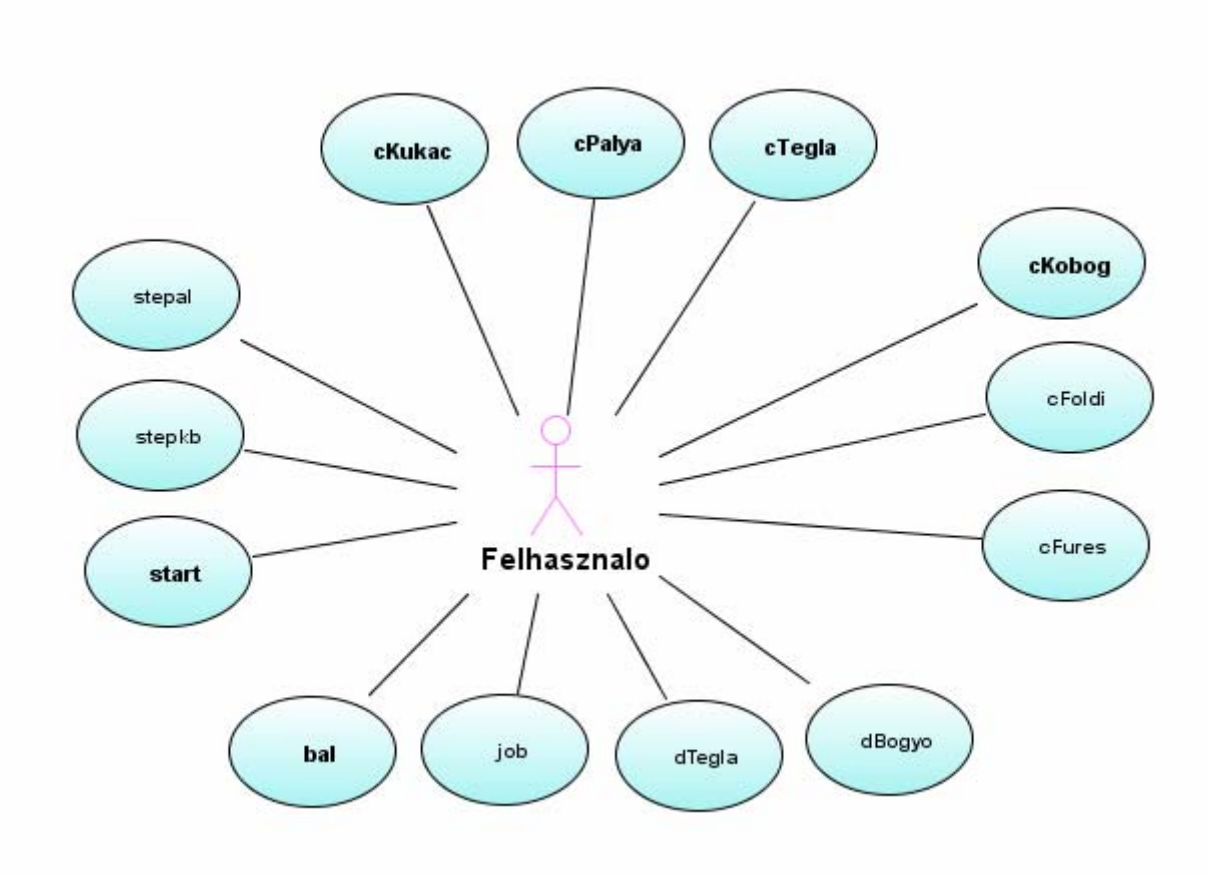
Inicializáció – explicit nem szükséges, minden kimenet tartalmazza az init utáni állapotot.

Kígyó létrehozása
- Szabad pozícióra
- Téglával ütközetve

Bogyóval ütköztetve
Szelvénnyel ütköztetve
Tégla létrehozása
Szabad pozícióra
Téglával ütközetve
Bogyóval ütköztetve
Szelvénnyel ütköztetve
Bogyók létrehozása
Kőbogyó létrehozása szabad pozícióra
Mezeibogyó létrehozása szabad pozícióra
Fűrészbogyó létrehozása szabad pozícióra
Kőbogyó létrehozása Téglával ütköztetve
Mezeibogyó létrehozása Téglával ütköztetve
Fűrészbogyó létrehozása Téglával ütköztetve
Kőbogyó létrehozása másik Bogyóval ütköztetve
Mezeibogyó létrehozása másik Bogyóval ütköztetve
Fűrészbogyó létrehozása másik Bogyóval ütköztetve
Kőbogyó létrehozása Szelvénnyel ütköztetve
Mezeibogyó létrehozása Szelvénnyel ütköztetve
Fűrészbogyó létrehozása Szelvénnyel ütköztetve
Bogyó törlése
Kőbogyó törlése
Fűrészbogyó törlése
Mezeibogyó törlése
Tégla törlés
Ütközések
Ütközés Téglával
Ütközés Bogyóval
Kőbogyóval
Mezeibogyóval
Fűrészbogyóval
Ütközés saját Szelvénnyel
Fűrészbogyóhatás nélkül a cél szelvényen Kőbogyóhatás nélkül
Fűrészbogyóhatás nélkül a cél szelvényen Kőbogyóhatással
Fűrészbogyóhatással a cél szelvényen Kőbogyóhatás nélkül
Fűrészbogyóhatással a cél szelvényen Kőbogyóhatással
Ütközés idegen Szelvénnyel
Fűrészbogyóhatás nélkül a cél szelvényen Kőbogyóhatás nélkül
Fűrészbogyóhatás nélkül a cél szelvényen Kőbogyóhatással
Fűrészbogyóhatással a cél szelvényen Kőbogyóhatás nélkül
Fűrészbogyóhatással a cél szelvényen Kőbogyóhatással
Léptetések
Léptetések különböző irányokba
Léptetés fordulás nélkül
Léptetés 235 fokos balra fordulás után
Léptetés 45 fokos balra fordulás után
Léptetés 45 fokos jobbra fordulás után

Léptetés 235 fokos jobbra fordulás után
Kőbogyó „léptetése” a Kukacban a farok felé.
1 Kőbogyó léptetése üres Kukacban
1 Kőbogyó léptetése Kukacban ami 1 Kőbogyót tartalmaz a farkában
Több (2) Kőbogyó léptetése üres Kukacban
Kőbogyókkal teli Kukac üközése Kőbogyóval

7.5. Összes részletes use-case



8. fejezet Részletes tervek

8.1. A tesztelést támogató programok tervei

A tesztelést egyetlen programmal ellenőrizzük. A program a játék aktuális kimeneti állományait hasonlítja össze az előre definiált kimeneti állományokkal, illetve futtatja a protot a forráskódjában megadott állományokra.

Terv: a program egyetlen main-től elkülönülő függvényt fog tartalmazni, amely paraméterként megkapja a bemeneti állomány és az elvárt kimeneti állomány nevét, ezt parancssori paraméterként a protonak átadva futtatja azt, végül a kimeneti állományt összehasonlítja az elvárttal. Eltérés esetén az eltérés sorának számát is megadja. Az egyes fájlokra helyes kimenet esetén „OK” feliratot jelez. A main függvényben ez a metódus kerül meghívásra minden egyes tesztesetben.

8.2. A tesztek részletes tervei, leírásuk a teszt nyelvén

Inicializáció – explicit nem szükséges, minden kimenet tartalmazza az init utáni állapotot.

Kígyó létrehozása

Szabad pozícióra

`cKukac(0,0,0,0,0,3)`	`**********`
	`cKukac(0,0,0,0,0,3)`
	`ku0, 0, 0, 0`
	`sz1, 0, 1, 0`
	`sz2, 0, 2, 0`
	`sz3, 0, 3, 0`
	`**********`

Téglával ütközetve

`cTegla(0,0)`
`cKukac(0,0,0,0,0,3)`
	`**********`
	`cTegla(0,0)`
	`cKukac(0,0,0,0,0,3)`
	`te 0,0`
	`**********`

Bogyóval ütköztetve

`cKobog(0,0)`
`cKukac(0,0,0,0,0,3)`
	`**********`
	`cKobog(0,0)`
	`cKukac(0,0,0,0,0,3)`
	`kb 0,0`
	`**********`

Szelvénnyel ütköztetve

cKukac(0,0,0,0,0,3) cKukac(1,20,0,0,0,3)	********** cKukac(0,0,0,0,0,3) cKukac(1,20,0,0,0,3) ku0, 0, 0, 0 sz1, 0, 1, 0 sz2, 0, 2, 0 sz3, 0, 3, 0 **********

Tégla létrehozása

Szabad pozícióra

cTegla(0,0)	********** cTegla(0,0) te 0,0 **********

Téglával ütközetve

cTegla(0,0) cTegla(0,0)	********** cTegla(0,0) cTegla(0,0) te 0,0 **********

Bogyóval ütköztetve

cKobog(0,0) cTegla(0,0)	********** cKobog(0,0) cTegla(0,0) kb 0,0 **********

Szelvénnyel ütköztetve

cKukac(0,0,0,0,0,3) cTegla(0,0)	********** cKukac(0,0,0,0,0,3) cTegla(0,0) ku0, 0, 0, 0 sz1, 0, 1, 0 sz2, 0, 2, 0 sz3, 0, 3, 0 **********

Bogyók létrehozása

Kőbogyó létrehozása szabad pozícióra

cKobog(0,0)	********** cKobog(0,0) kb 0,0 **********

Földibogyó létrehozása szabad pozícióra

cFoldi(0,0)	********** cFoldi(0,0) fb 0,0 **********

Fűrészbogyó létrehozása szabad pozícióra

cFures(0,0)	********** cFures(0,0) fu 0,0 **********

Kőbogyó létrehozása Téglával ütköztetve

cTegla(0,0) cKobog(0,0)	********** cTegla(0,0) cKobog(0,0) te 0,0 **********

Földibogyó létrehozása Téglával ütköztetve

cTegla(0,0) cFoldi(0,0)	********** cTegla(0,0) cFoldi(0,0) te 0,0 **********

Fűrészbogyó létrehozása Téglával ütköztetve

cTegla(0,0) cFures(0,0)	********** cTegla(0,0) cFures(0,0) te 0,0 **********

Kőbogyó létrehozása másik Bogyóval ütköztetve

cFoldi(0,0) cKobog(0,0)	********** cFoldi(0,0) cKobog(0,0) fb 0,0 **********

Földibogyó létrehozása másik Bogyóval ütköztetve

cKobog(0,0) cFoldi(0,0)	********** cKobog(0,0) cFoldi(0,0) kb 0,0 **********

Fűrészbogyó létrehozása másik Bogyóval ütköztetve

cFoldi(0,0) cFures(0,0)	********** cFoldi(0,0) cFures(0,0) fb 0,0 **********

Kőbogyó létrehozása Szelvénnyel ütköztetve

cKukac(0,0,0,0,0,3) cKobog(0,0)	********** cKukac(0,0,0,0,0,3) cKobog(0,0) ku0, 0, 0, 0 sz1, 0, 1, 0 sz2, 0, 2, 0 sz3, 0, 3, 0 **********

Földibogyó létrehozása Szelvénnyel ütköztetve

cKukac(0,0,0,0,0,3) cFoldi(0,0)	********** cKukac(0,0,0,0,0,3) cFoldi(0,0) ku0, 0, 0, 0 sz1, 0, 1, 0 sz2, 0, 2, 0 sz3, 0, 3, 0 **********

Fűrészbogyó létrehozása Szelvénnyel ütköztetve

cKukac(0,0,0,0,0,3) cFures(0,0)	********** cKukac(0,0,0,0,0,3) cFures(0,0) ku0, 0, 0, 0 sz1, 0, 1, 0 sz2, 0, 2, 0 sz3, 0, 3, 0 **********

Bogyó törlése

Kőbogyó törlése

cKobog(0,0) stepal dBogyo(0,0)	********** cKobog(0,0) stepal kb 0,0 ********** ********** dBogyo(0,0) **********

Fűrészbogyó törlése

cFures(0,0) stepal dBogyo(0,0)	********** cFures(0,0) stepal fu 0,0 ********** ********** dBogyo(0,0) **********

Földibogyó törlése

cFoldi(0,0) stepal dBogyo(0,0)	********** cFoldi(0,0) stepal fb 0,0 ********** ********** dBogyo(0,0) **********

Tégla törlés

cTegla(0,0) stepal dTegla(0,0)	********** cTegla(0,0) stepal te 0,0 ********** ********** dTegla(0,0) **********

Ütközések

Ütközés Téglával

cKukac(0,0,0,2,0,3) cTegla(0,0) stepal stepal	********** cKukac(0,0,0,2,0,3) cTegla(0,0) stepal te 0,0 ku0, 0, 1, 0 sz1, 0, 2, 0 sz2, 0, 3, 0 sz3, 0, 4, 0 ********** ********** stepal te 0,0 **********

Ütközés Bogyóval

Kőbogyóval

cKukac(0,0,0,2,0,3) cKobog(0,0) stepal stepal	********** cKukac(0,0,0,2,0,3) cKobog(0,0) stepal kb 0,0 ku0, 0, 1, 0 sz1, 0, 2, 0 sz2, 0, 3, 0 sz3, 0, 4, 0 ********** ********** stepal ku0, 0, 0, 1 sz1, 0, 1, 0 sz2, 0, 2, 0 sz3, 0, 3, 0 sz4, 0, 4, 0 **********

Földibogyóval

cKukac(0,0,0,2,0,3) cFoldi(0,0) stepal stepal	********** cKukac(0,0,0,2,0,3) cFoldi(0,0) stepal fb 0,0 ku0, 0, 1, 0 sz1, 0, 2, 0 sz2, 0, 3, 0 sz3, 0, 4, 0 ********** ********** stepal ku0, 0, 0, 0 sz1, 0, 1, 0 sz2, 0, 2, 0 sz3, 0, 3, 0 sz4, 0, 4, 0 **********

Fűrészbogyóval

cKukac(0,0,0,2,0,3) cFures(0,0) stepal stepal	********** cKukac(0,0,0,2,0,3) cFures(0,0) stepal fu 0,0 ku0, 0, 1, 0 sz1, 0, 2, 0 sz2, 0, 3, 0 sz3, 0, 4, 0 ********** ********** stepal ku0, 0, 0, 2 sz1, 0, 1, 0 sz2, 0, 2, 0 sz3, 0, 3, 0 sz4, 0, 4, 0 **********

Ütközés saját Szelvénnyel

Fűrészbogyóhatás nélkül a cél szelvényen Kőbogyóhatás nélkül

cKukac(0,0,0,2,0,6) stepal job090 stepal job090 stepal job090 stepal	********** cKukac(0,0,0,2,0,6) stepal ku0, 0, 1, 0 sz1, 0, 2, 0 sz2, 0, 3, 0 sz3, 0, 4, 0 sz4, 0, 5, 0 sz5, 0, 6, 0 sz6, 0, 7, 0 ********** ********** job090 stepal ku0, 1, 1, 0 sz1, 0, 1, 0 sz2, 0, 2, 0 sz3, 0, 3, 0 sz4, 0, 4, 0 sz5, 0, 5, 0 sz6, 0, 6, 0 ********** ********** job090 stepal ku0, 1, 2, 0 sz1, 1, 1, 0 sz2, 0, 1, 0 sz3, 0, 2, 0 sz4, 0, 3, 0 sz5, 0, 4, 0 sz6, 0, 5, 0 ********** ********** job090 stepal **********

Fűrészbogyóhatás nélkül a cél szelvényen Kőbogyóhatással

cKobog(1,1) cKukac(0,0,0,2,0,7) stepal job090 stepal job090 stepKB stepKB stepKB stepal job090 stepal	********** cKobog(1,1) cKukac(0,0,0,2,0,7) kb 1,1 stepal ku0, 0, 1, 0 sz1, 0, 2, 0 sz2, 0, 3, 0 sz3, 0, 4, 0 sz4, 0, 5, 0 sz5, 0, 6, 0 sz6, 0, 7, 0 sz7, 0, 8, 0 ********** ********** job090 stepal ku0, 1, 1, 1 sz1, 0, 1, 0 sz2, 0, 2, 0 sz3, 0, 3, 0 sz4, 0, 4, 0 sz5, 0, 5, 0 sz6, 0, 6, 0 sz7, 0, 7, 0 sz8, 0, 8, 0 ********** ********** job090 stepKB stepKB stepKB stepal ku0, 1, 2, 0 sz1, 1, 1, 0 sz2, 0, 1, 0 sz3, 0, 2, 1 sz4, 0, 3, 0 sz5, 0, 4, 0 sz6, 0, 5, 0 sz7, 0, 6, 0 sz8, 0, 7, 0 ********** ********** job090 stepal **********

Fűrészbogyóhatással a cél szelvényen Kőbogyóhatás nélkül

cFures(0,1) cKukac(0,0,0,2,0,6) stepal job090 stepal	********** cFures(0,1) cKukac(0,0,0,2,0,6) stepal ku0, 0, 1, 2

job090 stepal job090 stepal	sz1, 0, 2, 0 sz2, 0, 3, 0 sz3, 0, 4, 0 sz4, 0, 5, 0 sz5, 0, 6, 0 sz6, 0, 7, 0 sz7, 0, 8, 0 ********** ********** job090 stepal ku0, 1, 1, 2 sz1, 0, 1, 0 sz2, 0, 2, 0 sz3, 0, 3, 0 sz4, 0, 4, 0 sz5, 0, 5, 0 sz6, 0, 6, 0 sz7, 0, 7, 0 sz8, 0, 8, 0 ********** ********** job090 stepal ku0, 1, 2, 2 sz1, 1, 1, 0 sz2, 0, 1, 0 sz3, 0, 2, 0 sz4, 0, 3, 0 sz5, 0, 4, 0 sz6, 0, 5, 0 sz7, 0, 6, 0 sz8, 0, 7, 0 ********** ********** job090 stepal ku0, 0, 2, 2 sz1, 1, 1, 0 sz2, 0, 1, 0 sz3, 0, 2, 0 ku1, 0, 6, 0 sz1, 0, 5, 0 sz2, 0, 4, 0 sz3, 0, 3, 0 **********

Fűrészbogyóhatással a cél szelvényen Kőbogyóhatással

cKobog(1,1) cFures(0,1) cKukac(0,0,0,2,0,6) stepal job090 stepal job090 stepKB stepKB stepKB stepal job090 stepal	********** cKobog(1,1) cFures(0,1) cKukac(0,0,0,2,0,6) kb 1,1 stepal ku0, 0, 1, 2 sz1, 0, 2, 0 sz2, 0, 3, 0 sz3, 0, 4, 0 sz4, 0, 5, 0 sz5, 0, 6, 0 sz6, 0, 7, 0 sz7, 0, 8, 0 ********** ********** job090 stepal ku0, 1, 1, 3 sz1, 0, 1, 0 sz2, 0, 2, 0 sz3, 0, 3, 0 sz4, 0, 4, 0 sz5, 0, 5, 0 sz6, 0, 6, 0 sz7, 0, 7, 0 sz8, 0, 8, 0 ********** ********** job090 stepKB stepKB stepKB stepal ku0, 1, 2, 2 sz1, 1, 1, 0 sz2, 0, 1, 0 sz3, 0, 2, 1 sz4, 0, 3, 0 sz5, 0, 4, 0 sz6, 0, 5, 0 sz7, 0, 6, 0 sz8, 0, 7, 0 ********** ********** job090 stepal **********

Ütközés idegen Szelvénnyel

Fűrészbogyóhatás nélkül a cél szelvényen Kőbogyóhatás nélkül

cKukac(0,270,-1,0,0,3) cKukac(1,0,0,2,0,3) stepal stepal	********** cKukac(0,270,-1,0,0,3) cKukac(1,0,0,2,0,3) stepal ku0, 0, 0, 0

	sz1, -1, 0, 0 sz2, -2, 0, 0 sz3, -3, 0, 0 ku1, 0, 1, 0 sz1, 0, 2, 0 sz2, 0, 3, 0 sz3, 0, 4, 0 ********** ********** stepal ku0, 1, 0, 0 sz1, 0, 0, 0 sz2, 1, 0, 0 sz3, 2, 0, 0 **********

Fűrészbogyóhatás nélkül a cél szelvényen Kőbogyóhatással

cKukac(0,270,-1,0,0,2) cKukac(1,0,0,2,0,3) cKobog(0,0) stepal stepKB stepal	********** cKukac(0,270,-1,0,0,2) cKukac(1,0,0,2,0,3) cKobog(0,0) stepal ku0, 0, 0, 1 sz1, -1, 0, 0 sz2, -2, 0, 0 sz3, -3, 0, 0 ku1, 0, 1, 0 sz1, 0, 2, 0 sz2, 0, 3, 0 sz3, 0, 4, 0 ********** ********** stepKB stepal ku0, 1, 0, 0 sz1, 0, 0, 1 sz2, 1, 0, 0 sz3, 2, 0, 0 **********

Fűrészbogyóhatással a cél szelvényen Kőbogyóhatás nélkül

```
cKukac(0,270,-1,0,0,3)
cKukac(1,0,0,2,0,2)
cFures(0,1)
stepal
stepal
```

```
**********************
cKukac(0,270,-1,0,0,3)
cKukac(1,0,0,2,0,2)
cFures(0,1)
stepal
ku0, 0, 0, 0
      sz1, -1, 0, 0
      sz2, -2, 0, 0
      sz3, -3, 0, 0
ku1, 0, 1, 2
      sz1, 0, 2, 0
      sz2, 0, 3, 0
      sz3, 0, 4, 0
**********************
**********************
stepal
ku0, 1, 0, 0
ku1, 0, 0, 2
      sz1, 0, 1, 0
      sz2, 0, 2, 0
      sz3, 0, 3, 0
ku2, -2, 0, 0
      sz1, -1, 0
**********************
```

Fűrészbogyóhatással a cél szelvényen Kőbogyóhatással

```
cKukac(0,270,-1,0,0,2)
cKukac(1,0,0,2,0,2)
cFures(0,1)
cKobog(0,0)
stepal
stepKB
stepal
```

```
**********************
cKukac(0,270,-1,0,0,2)
cKukac(1,0,0,2,0,2)
cFures(0,1)
cKobog(0,0)
stepal
ku0, 0, 0, 1
      sz1, -1, 0, 0
      sz2, -2, 0, 0
      sz3, -3, 0, 0
ku1, 0, 1, 2
      sz1, 0, 2, 0
      sz2, 0, 3, 0
      sz3, 0, 4, 0
**********************
**********************
stepKB
stepal
ku0, 1, 0, 0
      sz1, 0, 0, 1
      sz2, -1, 0, 0
      sz3, -2, 0, 0
**********************
```

Léptetések

Léptetések különböző irányokba

Léptetés fordulás nélkül

```
cKukac(0,270,-1,0,0,3)
stepal
```

```
**********************
cKukac(0,270,-1,0,0,3)
stepal
ku0, 0, 0, 0
```

	sz1, -1, 0, 0 sz2, -2, 0, 0 sz3, -3, 0, 0 **********

Léptetés 235 fokos balra fordulás után

cKukac(0,270,-1,0,0,3) bal235 stepal	********** cKukac(0,270,-1,0,0,3) bal235 stepal ku0, -2, 1, 0 sz1, -1, 0, 0 sz2, -2, 0, 0 sz3, -3, 0, 0 **********

Léptetés 45 fokos balra fordulás után

cKukac(0,270,-1,0,0,3) bal045 stepal	********** cKukac(0,270,-1,0,0,3) bal045 stepal ku0, 0, 1,0 sz1, -1, 0, 0 sz2, -2, 0, 0 sz3, -3, 0, 0 **********

Léptetés 45 fokos jobbra fordulás után

cKukac(0,270,-1,0,0,3) job045 stepal	********** cKukac(0,270,-1,0,0,3) job045 stepal ku0, 0, -1,0 sz1, -1, 0, 0 sz2, -2, 0, 0 sz3, -3, 0, 0 **********

Léptetés 235 fokos jobbra fordulás után

```
cKukac(0,270,-1,0,0,3)
job235
stepal
```

```
**********************
cKukac(0,270,-1,0,0,3)
job235
stepal
ku0, -2, -1,0
       sz1, -1, 0, 0
       sz2, -2, 0, 0
       sz3, -3, 0, 0
**********************
```

Kőbogyó „léptetése” a Kukacban a farok felé.

1 Kőbogyó léptetése üres Kukacban

```
cKukac(0,270,-1,0,0,2)
cKobog(0,0)
stepal
stepKB
stepal
```

```
**********************
cKukac(0,270,-1,0,0,2)
cKobog(0,0)
stepal
ku0, 0, 0, 1
       sz1, -1, 0, 0
       sz2, -2, 0, 0
       sz3, -3, 0, 0
**********************
**********************
stepKB
stepal
ku0, 1, 0, 0
       sz1, 0, 0, 1
       sz2, -1, 0, 0
       sz3, -2, 0, 0
**********************
```

1 Kőbogyó léptetése Kukacban ami 1 Kőbogyót tartalmaz a farkában

```
cKukac(0,270,-1,0,0,1)
cKobog(0,0)
stepal
stepKB
stepKB
stepKB
cKobog(1,0)
stepal
stepKB
stepKB
stepal
```

```
**********************
cKukac(0,270,-1,0,0,1)
cKobog(0,0)
stepal
ku0, 0, 0, 1
       sz1, -1, 0, 0
       sz2, -2, 0, 0
**********************
**********************
stepKB
stepKB
stepKB
cKobog(1,0)
stepal
ku0, 1, 0,1
       sz1, 0, 0, 0
       sz2, -1, 0, 0
       sz3, -2, 0, 1
**********************
**********************
stepKB
stepKB
stepal
ku0, 2, 0, 0
       sz1, 1, 0, 0
       sz2, 0, 0, 1
       sz3, -2, 0, 1
**********************
```

Több (2) Kőbogyó léptetése üres Kukacban

```
cKukac(0,270,-1,0,0,1)
cKobog(0,0)
cKobog(1,0)
stepal
stepal
stepKB
stepKB
stepal
```

```
**********************
cKukac(0,270,-1,0,0,1)
cKobog(0,0)
cKobog(1,0)
stepal
kb 1, 0
ku0, 0, 0, 1
       sz1, -1, 0, 0
       sz2, -2, 0, 0
**********************
**********************
stepal
ku0, 1, 0, 1
       sz1, 0, 0, 1
       sz2, -1, 0, 0
       sz3, -2, 0, 0
**********************
**********************
stepKB
stepKB
stepal
ku0, 2, 0, 0
       sz1, 1, 0, 0
       sz2, 0, 0, 1
       sz3, -1, 0, 1
**********************
```

Kőbogyókkal teli Kukac üközése Kőbogyóval

A teszteset bár fontos, nem szorul külön kivizsgálásra, ugyanis egy kőbogyó felvétele automatikusan megnöveli a kukac hosszát, majd lépteti a kőbogyókat hátrafelé, így a kukac fejében biztosan lesz üres hely. Tehát ha az előző esetek mindegyikében hátralépett a fejből a kőbogyó akkor ez a teszteset is hibátlan lesz.

8.3. Objektumok és metódusok tervei (State chartok és Activity diagramok)

Motor Operation Summary	
Kukacok	**getKukacok**() Visszaadja a Kukacokra mutató referenciát.
void	**setKukacok**(Kukacok val) Beállítja a Kukacokra mutató referenciát.
Poziciok	**getPoziciok**() Visszaadja a Poziciokra mutató referenciát.
void	**setPoziciok**(Poziciok val) Beállítja a Poziciokra mutató referenciát.
Bokor	**getBokor**() Visszaadja a Bokorra mutató referenciát.
void	**setBokor**(Bokor val) Beállítja a Bokorra mutató referenciát.
JatekTerep	**getJatekTerep**() Visszaadja a JatekTerepre mutató referenciát.
void	**setJatekTerep**(JatekTerep val) Beállítja a JatekTerepre mutató referenciát.
void	**run**() Beolvassa az aktuális lépésre vonatkozó parancsokat a proto bemeneti állományból és végrehajtja azokat. Beolvasásuk sorrendjében. A stepal parancs: Meghívja a Kukacok.leptet() metódust. Meghívja a Kukacok.getKukacokSzama() fv.t. Minden egyes Kukacot lineárisan elkér a getKukac() fv-nyel, és minden Kukac-tól elékéri annak 0. Szelveny-ét a getSzelveny(0) metódussal. Az így kapott szelvényt paraméterül adja Poziciok.utkozes(Alapelem a) fv-nek, végül meghívja a Bokor.leptet() metódust. Végül meghívja a , JatekTerep,kiir(kimeneti fájlnév), Bokor,kiir(kimeneti fájlnév), Kukacok,kiir(kimeneti fájlnév) metódusokat A stepKB parancs: Meghívja a Kukacok.leptetKoBogyo()-t bal###, job### parancsok: Meghívja a Kukacok.forgatKukac(ID, szog) fv-t, ahol szog minden esetben az óramutató járásával ellentétes irányban véve pozitív, tehát a job### esetén -### az átadott érték. c**** és d**** parancsok: Létrehozza illetve törli a megadott objektumokat.
KoBogyoSzal	**getKBSzal**() Visszaadja a KBszalra mutató referenciát.
void	**setKBSzal**(KoBogyoSzal val) Beállítja a KBszalra mutató referenciát.

Pozicio Operation Summary

Point	**getHely**() Visszadja az eltárolt helyet.
void	**setHely**(Point val) Beállít egy helyet.
Alapelem	**getAlapelem**() Visszaadja az alapelemre mutató referenciát.
void	**setAlapelem**(Alapelem val) Beállítja az alapelemre mutató referenciát.
	Pozicio() Konstruktor, létrehozza a Poziciot egy véletlenszerű szabad helyre.

Poziciok Operation Summary

void	**utkozes**(Alapelem a) Lineárisan végigmegy a Poziciok kollekción, és minden elemre megvizsgálja, hogy fedésben van-e a –val. Ha igen, akkor meghívja annak megharapva(a) függvényét.
void	**addPozicio**(Pozicio p) Eltárol egy Pozicio elemet
void	**deletiPozicio**(Pozicio p) Kitorol egy poziciot
Pozicio	**csinaljPoz**(int x, int y) Létrehoz egy Poziciot (x,y) koordinátára

KobogyoSzal Operation Summary

void	**run**() A kőbogyók léptetéséért felel, steptime (ms) időközönként meghívja a k.leptetKoBogyo() függvényt
int	**getSteptime**() Steptime lekérése
void	**setSteptime**(int val) Steptime beállítása
Kukacok	**getK**() K lekérése
void	**setK**(Kukacok val) K beállítása

Bokor Operation Summary

int	**getBogyokMaxSzama**(int hanybogyo) Lekéri a bogyók maximális számát
void	**setBogyokMaxSzama**(int val) beállítja a bogyók maximális számát
int	**getIdoHatraAKovetkezoBogyoMeghalasaig**() Lekérdezi, hogy hány „leptet” van még hátra a következő bogyó meghalásáig

void	**setIdoHatraAKovetkezoBogyoMeghalasaig**(int val) Beállítja, hogy hány „leptet” van még hátra a következő bogyó meghalásáig
void	**leptet**() Csökkenti, a IdoHatraAKovetkezoBogyoMeghalasaig számlálót és ha nulla, kitörli a bogyókból álló FIFO első elemét a többieket pedig lépteti a FIFOban.
void	**torolBogyo**(Bogyo b) Kitorli b-t a bogyók közülm ha b érvényes bogyó. Az ellenőrzést a bogyok-ban történő lineáris kereséssel végzi.
bool	**addBogyo**(Bogyo b) Hozzáadja b-t a bogyók FIFOhoz. A BogyokMaxSzama-t növeli, ha kell, hogy az új bogyó mindenképp beférjen.
void	**torolBogyot**(Pozicio p) Kitorli a p pozíción álló bogyót, ha létezik. Ehhez lineárisan lekérdezi bogyok minden Bogyo-jának Pozicio-jat

Bogyo Operation Summary

void	**megharapva**(Alapelem a) Absztarkt metódus, helyileg implementálva.
Bokor	**getBokra**() Lekérdezi az adott bogyó bokrát
void	**setBokra**(Bokor val) Beállítja az adott bogyó bokrát
void	**visszaharap**(Visszaharapas visszaharapo) Absztarkt metódus, helyileg implementálva.

Jatekterep Operation Summary

void	**addTegla**(Pozicio p) Felvesz egy téglát adott p Pozicio-ra
void	**keretez**(Dimension teglameret,Point honnan, Point hova) Létrehoz egy Tegla-kból álló téglalapot „honnan” bal felső és „hova” jobb alsó sarokkal. A teglák mérete teglameret.
void	**torolTegla**(Pozicio p) Kitorli a p pozíción álló Tegla-t, ha létezik. Ehhez lineárisan lekérdezi teglak minden Tegla-jának Pozicio-jat
bool	**addTeglat**(Tegla t) Felveszi, t Teglat

Tegla Operation Summary

void	**megharapva**(Alapelem a) Létrehoz egy TeglaVisszaharapas objektumot, és paraméterül saját magát átadja. meghívja a.visszaharap(Visszaharapas visszaharapo) függvényét paraméterként a létrehozott osztályt átadva.
JatekTerep	**getTerepe**() Lekérdezi az adott Tegla terep-ét
void	**setTerepe**(JatekTerep val)

	Beállítja az adott Tegla terep-ét
void	**visszaharap**(Visszaharapas visszaharapo) Meghívja a visszaharapo.TeglaHarap(this) függvényt.

Kukacok Operation Summary

Kukac	**getKukac**(int melyiket) Visszadja a melyiket-ik Kukacot
void	**ellenoriz**() Leellenőrzi, hogy van e halott Kukac, ha igen akkor törli. Ehhez lineárisan végigellenőrzi a kukacok kollekciót. A halott Kukac-ot kitörli a kollekcióból és kitörli a hozzá tartozó Pozicio-kat is.
void	**getKukacokSzama**() Lekéri a kukacok számát
void	**leptet**() Meghívja minden Kukac leptet() függvényét
void	**addKukac**(Kukac k) Felvesz egy Kukacot
void	**forgatKukac**(int ID, int szog) Lineárisan megkeresi az adott IDjű Kukacot(egy Kukac IDje a kollekcióban elfoglalt helye) és módosítja a haladási irányát annak setIranyszog(double val) függvényével.
void	**leptetKoBogyo**() Meghívja minden Kukac leptetKoBogyo() függvényét

Kukac Operation Summary

void	**megno**() A kukac hosszát megnöveli eggyel úgy, hogy a végéhez tesz egy új Szelvenyt. Annak friss változóját boolba állítja, majd lépteti a kőbogyókat, így olyan mintha a kukac a fejénél nőne
void	**rovidul**(Szelveny honnantol) Létrehoz egy új Kukacot, hossz paraméterként nullát átadva, irányként az utolsó szelvényének pozíciójából kivonja utolsó előtti szelvényének pozícióját, ez a vektor adja az új kukac irányát, majd a levágandó szelvényeket hátulról előre odaadja a kezdetben üres kukacnak, annak addSzelveny metódusát használva. Ha a kukac egy hosszú (tehát csak fej), akkor nem jön létre új kukac, a harapás helyén lévő szelvény minden esetben törlődik, kivéve, ha fej.
Szelveny	**getSzelveny**(int melyiket) Lekéri a kollekció melyiket-edik szelvényét.
bool	**getEleMeg**() Lekéri, hogy a kukac él e még.
void	**setEleMeg**(bool val) Beállítja a kukac él-e még változóját.
bool	**elemee**(Szelveny sz)

	Igazat ad vissza, ha a Szelveny a Kukac eleme, hamisat, ha nem. Lineáris keresést használ.
void	**leptet**() A kukac szelvények kollekciójának legutolsó szelvényének pozícióját a kukac haladási irányban a legelső szelvény pozíciója elé rakja. Az új pozíciót a haladási irányból számolja ki, a szelvényt magát a kollekció legelejére teszi, ha volt rajta kőbogyóhatás, akkor azt az utolsó kőbogyóhatással nem rendelkező szelvényre teszi, amelyet hátulról előre lineárisan keres meg. Megnézi, hogy a régi fejen volt-e fűrészbogyóhatás, ha igen akkor azt átmásolja ide. Ha a szelvény „friss” volt, akkor a friss változót falseb a állítja Meghívja a legelső szelvény léptet függvényét.
double	**getIranyszog**() Visszaadja a kukac irányszögét.
void	**setIranyszog**(double val) Beállítja a kukac irányszögét.
void	**leptetKoBogyo**() A kukacon hátulról előre végigmenve minden kőbogyóhatást eggyel hátrébb mozgat, ha a kőbogyós utáni szelvény kőbogyómentes. Hátulról végzi, a kollekción lineárisan végigmenve
void	**addSzelveny(Szelveny sz)** Hozzáad egy szelvényt a kukac legutolsó szelvénye után. A szelvények kollekciót bővíti. Az adott szelvény setKukaca függvényét meghívja saját magával mint paraméterrel.

Szelveny Operation Summary

void	**megharapva**(Alapelem a) Létrehoz egy SzelvenyVisszaharapas objektumot, és paraméterül saját magát átadja. meghívja a.visszaharap(Visszaharapas visszaharapo) függvényét paraméterként a létrehozott osztályt átadva.
Kukac	**getKukaca**() Visszaadja a szelvény Kukacát
void	**setKukaca**(Kukac val) Beállítja, hogy a szelvény melyik Kukachoz tartozzon.
bool	**getFureszHatas**() Lekérdezi a fűrészhatást.
void	**setFureszHatas**(int val) Beállítja a fűrészhatást.
bool	**getKoBogyoHatas**() Lekéri a kőbogyóhatást.
void	**setKoBogyoHatas**(bool val) Beállítja a kőbogyóhatást.
void	**leptet**() Csökkenti a fűrészhatást, ha az nagyobb, mint 0.

void	**visszaharap**(Visszaharapas visszaharapo) Meghívja a visszaharapo.SzelvenyHarap(this) függvényt.

Alapelem Operation Summary

Pozicio	**getPozicio**() Visszaadja a Pozicio-ját az alapelemnek.
void	**setPozicio**(Pozicio val) Beállítja a Pozicio-ját az alapelemnek.
Dimension	**getMeret**() Visszaadja a méretét az alapelemnek.
void	**setMeret**(Dimension val) Beállítja a méretét az alapelemnek.
void	**megharapva**(Alapelem a) absztrakt metódus az örökölt osztályokban van definiálva, célja a visitor minta megvalósítása
void	**visszaharap**(Visszaharapas visszaharapo) absztrakt metódus az örökölt osztályokban van definiálva, célja a visitor minta megvalósítása

TeglaVisszaharapasOperation Summary

Tegla	**getT**() Visszadja a megharapott téglát.
void	**setT**(Tegla val) Beállítja a megharapott téglát.
void	**SzelvenyHarap**(Szelveny sz) Lekéri az adott szelvény Kukac-át és annak él-e mégjét False-ra állítja.
void	**TeglaHarap**(Tegla t) jelenlegi specifikációban ez a függvény nem hívódhat meg
void	**FoldiBogyoHarap**(FoldiBogyo fb) jelenlegi specifikációban ez a függvény nem hívódhat meg
void	**KoBogyoHarap**(KoBogyo kb) jelenlegi specifikációban ez a függvény nem hívódhat meg
void	**FureszBogyoHarap**(FureszBogyo fb) jelenlegi specifikációban ez a függvény nem hívódhat meg

KoBogyoVisszaharapas Operation Summary

KoBogyo	**getKb**() Visszadja a megharapott Kobogyo-t.
void	**setKb**(KoBogyo val) Beállítja a megharapott Kobogyo-t.
void	**SzelvenyHarap**(Szelveny sz) Lekéri az adott Szelveny Kukac-át, és meghívja annak megno() metódusát. Meghívja a Kukac getSzelveny(0) metódusát, és az így kapott szelvény kőbogyóhatását beállítja.

void	**TeglaHarap**(Tegla t) jelenlegi specifikációban ez a függvény nem hívódhat meg
void	**FoldiBogyoHarap**(FoldiBogyo fb) jelenlegi specifikációban ez a függvény nem hívódhat meg
void	**KoBogyoHarap**(KoBogyo kb) jelenlegi specifikációban ez a függvény nem hívódhat meg
void	**FureszBogyoHarap**(FureszBogyo fb) jelenlegi specifikációban ez a függvény nem hívódhat meg

FoldiBogyoVisszaharapas Operation Summary

FoldiBogyo	**getFb**() Visszadja a megharapott FoldiBogyo-t.
void	**setFb**(FoldiBogyo val) Beállítja a megharapott FoldiBogyo -t.
void	**SzelvenyHarap**(Szelveny sz) Lekéri az adott Szelveny Kukac-át, és meghívja annak megno() metódusát.
void	**TeglaHarap**(Tegla t) jelenlegi specifikációban ez a függvény nem hívódhat meg
void	**FoldiBogyoHarap**(FoldiBogyo fb) jelenlegi specifikációban ez a függvény nem hívódhat meg
void	**KoBogyoHarap**(KoBogyo kb) jelenlegi specifikációban ez a függvény nem hívódhat meg
void	**FureszBogyoHarap**(FureszBogyo fb) jelenlegi specifikációban ez a függvény nem hívódhat meg

FureszBogyoVisszaharapas Operation Summary

FureszBogyo	**getFb**() Visszadja a megharapott FureszBogyo-t.
void	**setFb**(FureszBogyo val) Beállítja a megharapott FureszBogyo -t.
void	**SzelvenyHarap**(Szelveny sz) Lekéri az adott Szelveny Kukac-át, és meghívja annak megno() metódusát.. Meghívja a Kukac getSzelveny(0) metódusát, és az így kapott szelvény fűrészbogyóhatását megnöveli 15-tel.
void	**TeglaHarap**(Tegla t) jelenlegi specifikációban ez a függvény nem hívódhat meg
void	**FoldiBogyoHarap**(FoldiBogyo fb) jelenlegi specifikációban ez a függvény nem hívódhat meg
void	**KoBogyoHarap**(KoBogyo kb) jelenlegi specifikációban ez a függvény nem hívódhat meg
void	**FureszBogyoHarap**(FureszBogyo fb) jelenlegi specifikációban ez a függvény nem hívódhat meg

SzelvenyVisszaharapas Operation Summary

Szelveny	**getSz**() Visszadja a megharapott Szelveny-t.
void	**setSz**(Szelveny val) Beállítja a megharapott Szelveny-t.
void	**SzelvenyHarap**(Szelveny sz) Megnézi, hogy this.sz szelvény friss-e. Ha igen: nem tesz semmit Ha nem: Lekéri az kapott Szelveny Kukac-át. Meghívja a Kukac getSzelveny(0) metódusát, és az így kapott szelvény fűrészbogyóhatását lekéri. Ha nincs neki, akkor Kukac él-e mégjét False-ra állítja. Ha van, akkor lekéri a this.sz kőbogyóhatását. Ha this.sz rendelkezik kőbogyóhatással, akkor sz Kukacának él-e mégjét False-ra állítja. ha this.sz nem rendelkezik kőbogyóhatással, akkor lekéri this.sz Kukac-át, és meghívja annak rovidul(this.sz) függvényét.
void	**TeglaHarap**(Tegla t) jelenlegi specifikációban ez a függvény nem hívódhat meg
void	**FoldiBogyoHarap**(FoldiBogyo fb) jelenlegi specifikációban ez a függvény nem hívódhat meg
void	**KoBogyoHarap**(KoBogyo kb) jelenlegi specifikációban ez a függvény nem hívódhat meg
void	**FureszBogyoHarap**(FureszBogyo fb) jelenlegi specifikációban ez a függvény nem hívódhat meg

FureszBogyo Operation Summary

void	**megharapva**(Alapelem a) Létrehoz egy FureszBogyoVisszaharapas objektumot, és paraméterül saját magát átadja. meghívja a.visszaharap(Visszaharapas visszaharapo) függvényét paraméterként a létrehozott osztályt átadva.
void	**visszaharap**(Visszaharapas visszaharapo) Meghívja a visszaharapo.FureszBogyoHarap(this) függvényt.

FoldiBogyo Operation Summary

void	**megharapva**(Alapelem a) Létrehoz egy FoldiBogyoVisszaharapas objektumot, és paraméterül saját magát átadja. meghívja a.visszaharap(Visszaharapas visszaharapo) függvényét paraméterként a létrehozott osztályt átadva.
void	**visszaharap**(Visszaharapas visszaharapo) Meghívja a visszaharapo.FoldiBogyoHarap(this) függvényt.

KoBogyo Operation Summary

void	**megharapva**(Alapelem a) Létrehoz egy KoBogyoVisszaharapas objektumot, és paraméterül saját magát átadja.

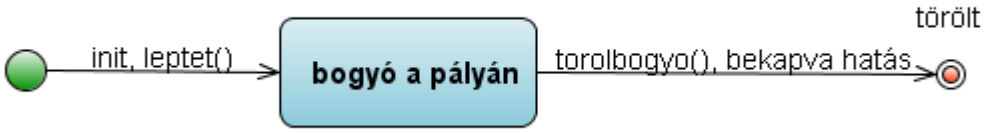
	meghívja a.visszaharap(Visszaharapas visszaharapo) függvényét paraméterként a létrehozott osztályt átadva.
void	**visszaharap**(Visszaharapas visszaharapo) Meghívja a visszaharapo.KoBogyoHarap(this) függvényt.

A protó kimenti állományba való íráshoz szükséges kiir(string file) fv a Bokor, JatekTerep, Kukacok esetében csak a kollekcióik elemein lineárisan végigmenve, meghívják azok kiik(string metódusait.

A kiir(string file) fv a Kukac, Tégla és az egyes bogyók a kimeneti állományba a specifikált módon történő kiírására szolgál.

8.3.1. State chartok

Bogyó



Kukac

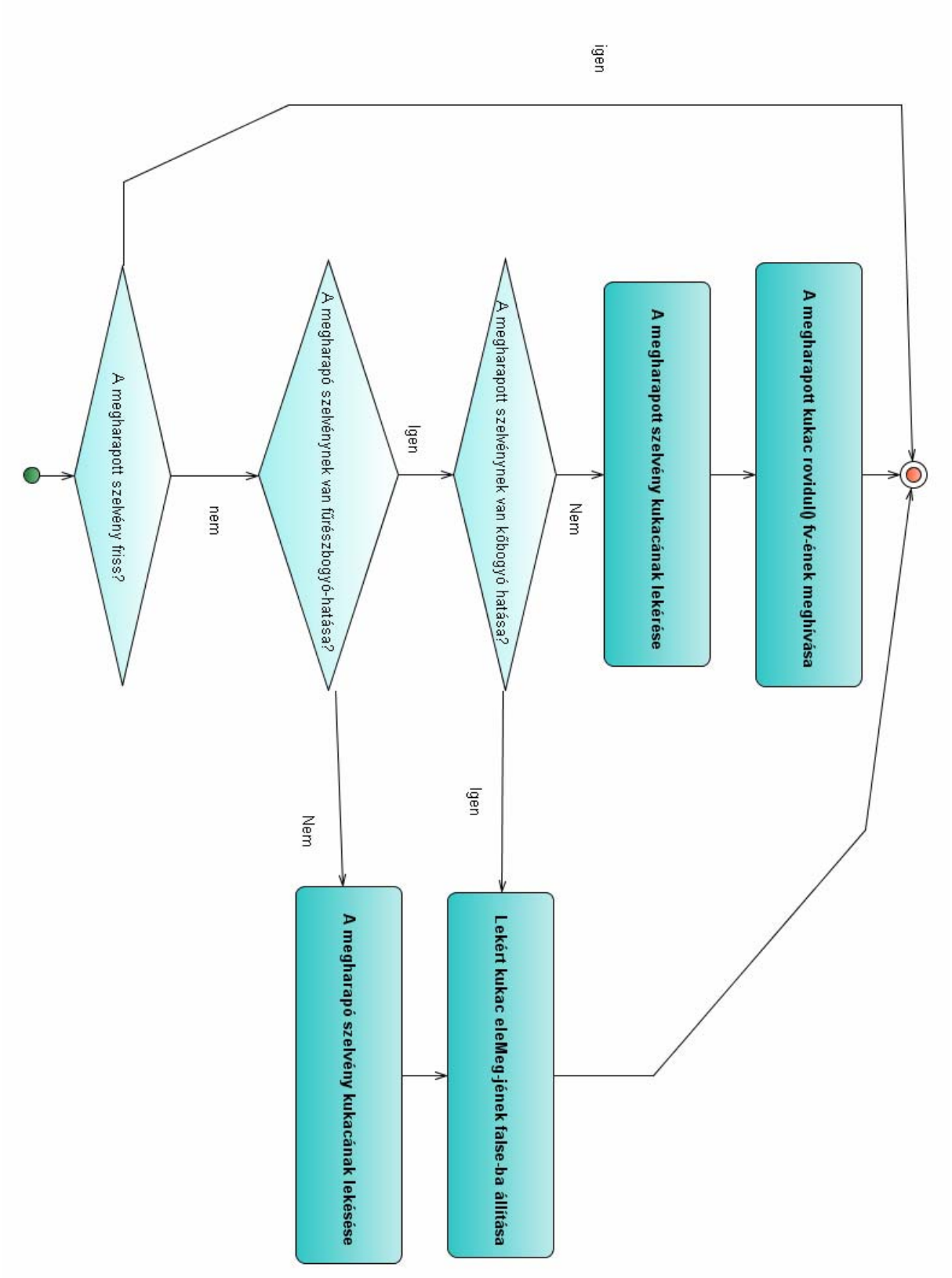
Szelvény

szelveny
friss=1
friss=0
a kukac lép, vagy a szelvényt a konstruktor hozta létre
fureszhatas=0
a szelveny megszunik fej lenni, vagy a számláló lejár
fureszhatas!=0
a kukac fűrészbogyót eszik a fűrészhatás ideje megnő
a kukac fűrészbogyót eszik
kobogyohatas=0
a kőbogyót lelépetélik a szelvényről
kobogyohatas=1
lepetéssel kőbogyót kap, vagy a szelvény fej és kőbogyót evett
vesztí hatás, levágva hatás
indít, bekap

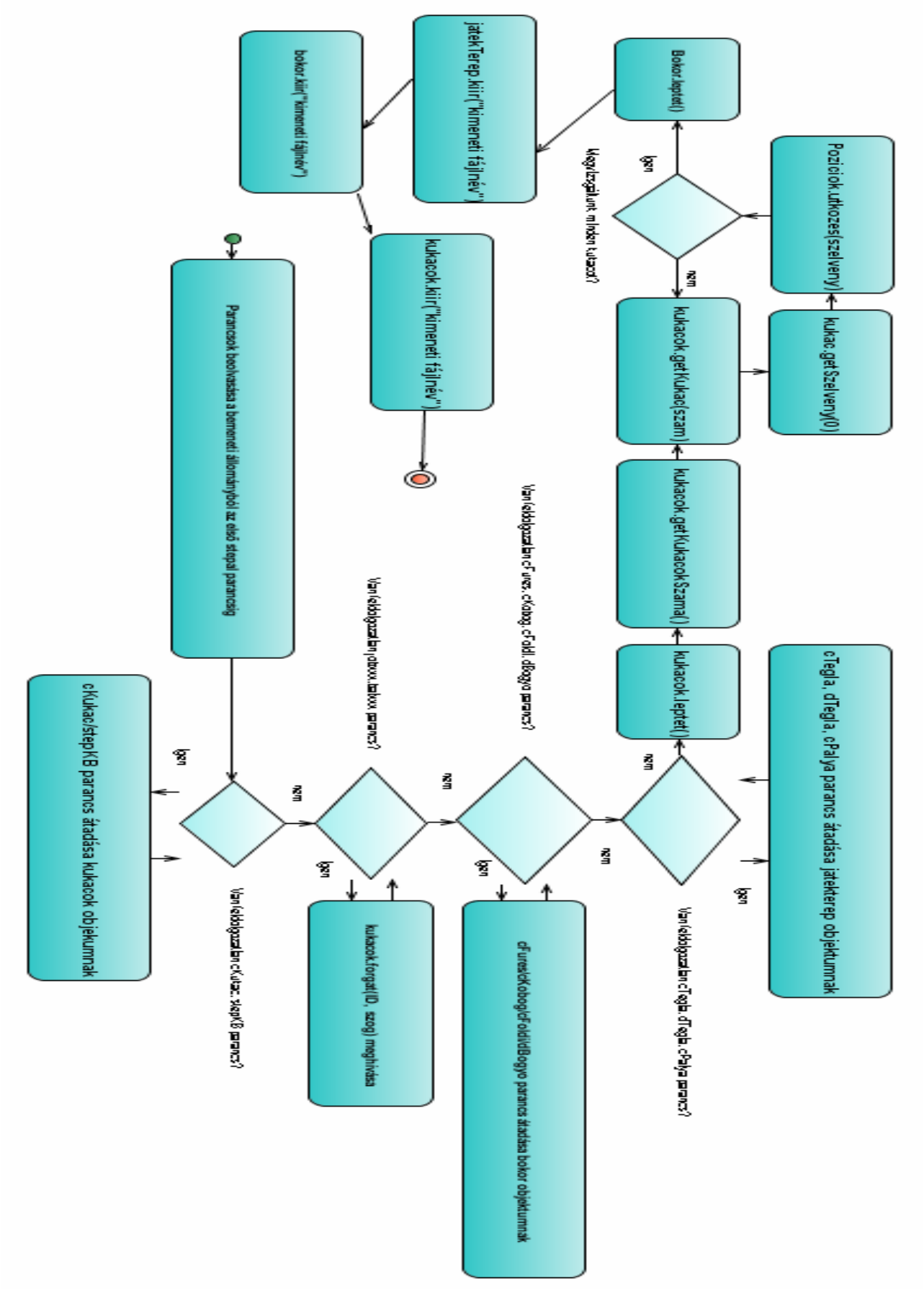
8.3.2. Activity diagramok

kobogyovisszaharapas_szelvenyharap

SzelvenyVisszaharapas_szelvenyHarap



motor_run()


Pozíciók.utkozes(szelveny)
kukac.getSzelveny(0)
kukacok.get(kukac(szam)
kukacok.get(kukacokSzama()
kukacok.leptet()
cTegla, dTegla, cPalya parancs átadása játéktérep objektumnak
\Van-e többszörös cTegla, dTegla, cPalya parancs?
cFureszek.oboogl(cFoldi,dBogyo parancs átadása bokor objektumnak
\Van-e többszörös cFureszek.oboogl(cFoldi,dBogyo parancs?
kukacok.forgat(ID, szogij meghívása
\Van-e többszörös cKukac.stepKB parancs?
cKukac/stepKB parancs átadása kukacok objektumnak
\Van-e többszörös cKukac.stepKB parancs?
Parancsok beolvasása a bemeneti állományból az első stepsil parancsig
kukacok.kiir("kimeneti fájlnév")
bokor.kiir("kimeneti fájlnév")
játékTerep.kiir("kimeneti fájlnév")
Bokor.leptet()
Kiegyenlítjük a különbséget?
igen
nem

9. fejezet Prototípus készítése, tesztelése

- Ezen a héten nem kellett anyagot beadni.

10. fejezet Prototípus beadása

10.1. Útmutatás a prototípushoz

10.1.1. Fájllista

Fájl neve	*Fájl mérete*	*Keletkezési idő*	*Leírás*
compile.bat	52 bájt	2008. április 15., 22:39:56	A forrásállományok lefordítását végző batch-fájl.
run.bat	6 306 bájt	2008. április 15., 22:39:56	Az összes előre megírt tesztesetet futtató batch-fájl.
test.bat	5 998 bájt	2008. április 15., 22:39:56	Az összes előre megírt tesztesetet az elvárttal összehasonlító, az eredményt képernyőre író batch-fájl.
test_to_file.bat	8 491 bájt	2008. április 15., 22:39:56	Az összes előre megírt tesztesetet az elvárttal összehasonlító, az eredményt fájlokba író batch-fájl.
Tetragon_Proto\javadoc	849 143 bájt	2008. április 15., 22:44:52	Ez a könyvtár tartalmazza a Javadoc speciális kommentekből és a forrásfájlokból generált fejlesztői referenciát.
Tetragon_Proto\tetragonteamkukacfarm	140 631 bájt	2008. április 15., 22:39:57	Ez a könyvtár tartalmazza a forrásfájlokat, a továbbiakban nem írjuk ki a teljes elérési útvonalat.
%\Alapelem.java	4 934 bájt	2008. április 15., 22:39:57	Az alapelem az objektumrendszer egy alapvető, absztrakt eleme, amelyből minden olyan osztály leszármazik, amely részt vesz a játéktér kialakításában. Minden elemnek lehet helye mérete és azonosítója.
%\Bogyo.java	3 444 bájt	2008. április 15., 22:39:57	A bogyó egy absztrakt osztály, belőle származnak le az egyes bogyófajták. A bogyó osztály az alapelemből származik le.
%\Bokor.java	9 216 bájt	2008. április 15., 22:39:57	A bokor az az osztály, amely egy listát tartalmaz a pályán jelenlévő bogyókról. Tárolja, kezeli és menedzseli a pályán lévő bogyókat. Felel a bogyók létrehozásáért és törléséért. Egy példány létezik belőle.
%\FoldiBogyo.java	2 896 bájt	2008. április 15., 22:39:57	A bogyó osztály leszármazottja, a játékspecifikációban leírt mezei bogyót valósítja meg.
%\FoldiBogyoVisszaharapas.java	6 004 bájt	2008. április 15., 22:39:57	Ha a megharapott alapelem földibogyó, akkor az földibogyóvisszaharapás objektumot hoz létre, és átadja azt a kölcsönhatásban résztvevő félnek, a látogató modell szerint. A visszaharapás interface-t valósítja meg.
%\FureszBogyo.java	2 695 bájt	2008. április 15., 22:39:57	A bogyó osztály leszármazottja, a játékspecifikációban leírt fűrészbogyót valósítja meg.
%\FureszBogyoVisszaharapas.java	6 250 bájt	2008. április 15., 22:39:57	Ha a megharapott alapelem fűrészbogyó, akkor az fűrészbogyóvisszaharapás objektumot hoz létre, és átadja azt a kölcsönhatásban résztvevő félnek, a látogató modell szerint. A visszaharapás interface-t valósítja meg.
%\JatekTerep.java	6 026 bájt	2008. április 15., 22:39:57	A játékterep az az osztály, amely egy listát tartalmaz a pályán jelenlévő fal objektumokról. Tárolja, kezeli és menedzseli őket. Felel a falak létrehozásáért. Egy példány létezik belőle.
%\KoBogyo.java	2 649 bájt	2008. április 15., 22:39:57	A bogyó osztály leszármazottja, a játékspecifikációban leírt kőbogyót valósítja meg.
%\KoBogyoSzal.java	1 207 bájt	2008. április 15., 22:39:57	A kőbogyószál a kőbogyók léptetéséért felelős.
%\KoBogyoVisszaharapas.java	5 985 bájt	2008. április 15., 22:39:57	Ha a megharapott alapelem kőbogyó, akkor az kőbogyóvisszaharapás objektumot hoz létre, és átadja azt a

			kölcsönhatásban résztvevő félnek, a látogató modell szerint. A visszaharapás interface-t valósítja meg.
%\Kukac.java	19 929 bájt	2008. április 15., 22:39:57	Egy kukacot reprezentáló osztály, rendelkezik attribútumként egy olyan változóval hogy életben van e, ismeri saját mozgásának irányát, képes saját hosszát megnövelni, megrövidíteni, lépni és tárolja szelvényeit.
%\Kukacok.java	6 595 bájt	2008. április 15., 22:39:57	Egy homogén kollekció Kukacok tárolására és az azokon elvégzendő alapvető műveletek koordinálására.
%\Motor.java	17 855 bájt	2008. április 15., 22:39:57	A játék léptetéséért az objektumok összehangolásával foglalkozó objektum. Az időzítéshez szálat használ, melyet majd a proto fázis után implementálunk.
%\Pozicio.java	4 646 bájt	2008. április 15., 22:39:57	Az osztály Alapelem típusú objektumok térbeli pozíciójának tárolására szolgál.
%\Poziciok.java	4 485 bájt	2008. április 15., 22:39:57	A Pozicio-k tárolására szolgáló osztály.
%\Proto_Tester.java	4 961 bájt	2008. április 15., 22:39:57	A prototípus tesztelésére szolgáló osztály.
%\Szelveny.java	8 112 bájt	2008. április 15., 22:39:57	A Kukac alapvető eleme.
%\SzelvenyVisszaharapas.java	6 760 bájt	2008. április 15., 22:39:57	Ha a megharapott alapelem szelvény, akkor az szelvényvisszaharapás objektumot hoz létre, és átadja azt a kölcsönhatásban résztvevő félnek, a látogató modell szerint. A visszaharapás interface-t valósítja meg.
%\Tegla.java	4 197 bájt	2008. április 15., 22:39:57	Az alapelemből leszármazó osztály. A falat reprezentálja.
%\TeglaVisszaharapas.java	5 751 bájt	2008. április 15., 22:39:57	Ha a megharapott alapelem tégla, akkor az téglavisszaharapás objektumot hoz létre, és átadja azt a kölcsönhatásban résztvevő félnek, a látogató modell szerint. A visszaharapás interface-t valósítja meg.
%\Tetragon_Team_Kukacfarm.java	4 552 bájt	2008. április 15., 22:39:57	A Tetragon_Team_Kukacfarm osztály felelős az inicializáció elindításáért. Ő hozza létre a Motor osztályt, amely elvégzi az inicializációt. Egy példány létezik belőle. Új játékot lehet vele indítani, és meglévőt lehet törölni.
%\Visszaharapas.java	1 482 bájt	2008. április 15., 22:39:57	A visszaharapás interface a visszaharapások kohézív tulajdonságainak összessége. Leírja, hogyan kell kinéznie az őt megvalósító interfacek-nek. A függvények leírása az adott – interface-t megvalósító – osztály leírásában található.
Tetragon_Proto\tesztek	11 692 bájt	2008. április 15., 22:39:56	Ez a könyvtár tartalmazza az előre definiált teszteseteket, és azok elvárt kimeneteit. Ezek a következők:
000-Init_cKukac_Szabad_be.txt	18 bájt	2008. április 15., 22:39:56	Kígyó létrehozása Szabad pozícióra
000-Init_cKukac_Szabad_elvart.txt	127 bájt	2008. április 15., 22:39:56	
001-Init_cKukac_Teglaval_be.txt	33 bájt	2008. április 15., 22:39:56	Téglával ütközetve
001-Init_cKukac_Teglaval_elvart.txt	90 bájt	2008. április 15., 22:39:56	
002-Init_cKukac_Bogyoval_be.txt	33 bájt	2008. április 15., 22:39:56	Bogyóval ütköztetve

002-Init_cKukac_Bogyoval_elvart.txt	90 bájt	2008. április 15., 22:39:56	
003-Init_cKukac_Szelvennyel_be.txt	41 bájt	2008. április 15., 22:39:56	Szelvénnyel ütköztetve
003-Init_cKukac_Szelvennyel_elvart.txt	150 bájt	2008. április 15., 22:39:56	
004-Init_cTegla_Szabad_be.txt	13 bájt	2008. április 15., 22:39:56	Tégla létrehozása Szabad pozícióra
004-Init_cTegla_Szabad_elvart.txt	70 bájt	2008. április 15., 22:39:56	
005-Init_cTegla_Teglaval_be.txt	26 bájt	2008. április 15., 22:39:56	Téglával ütközetve
005-Init_cTegla_Teglaval_elvart.txt	83 bájt	2008. április 15., 22:39:56	
006-Init_cTegla_Bogyoval_be.txt	26 bájt	2008. április 15., 22:39:56	Bogyóval ütköztetve
006-Init_cTegla_Bogyoval_elvart.txt	83 bájt	2008. április 15., 22:39:56	
007-Init_cTegla_Szelvennyel_be.txt	33 bájt	2008. április 15., 22:39:56	Szelvénnyel ütköztetve
007-Init_cTegla_Szelvennyel_elvart.txt	140 bájt	2008. április 15., 22:39:56	
008-Init_cKobog_Szabad_be.txt	11 bájt	2008. április 15., 22:39:56	Bogyók létrehozása Kőbogyó létrehozása szabad pozícióra
008-Init_cKobog_Szabad_elvart.txt	70 bájt	2008. április 15., 22:39:56	
009-Init_cFoldi_Szabad_be.txt	13 bájt	2008. április 15., 22:39:56	Földibogyó létrehozása szabad pozícióra
009-Init_cFoldi_Szabad_elvart.txt	70 bájt	2008. április 15., 22:39:56	
010-Init_cFures_Szabad_be.txt	13 bájt	2008. április 15., 22:39:56	Fűrészbogyó létrehozása szabad pozícióra
010-Init_cFures_Szabad_elvart.txt	70 bájt	2008. április 15., 22:39:56	
011-Init_cKobog_Teglaval_be.txt	26 bájt	2008. április 15., 22:39:56	Kőbogyó létrehozása Téglával ütköztetve
011-	83 bájt	2008. április	

Init_cKobog_Teglaval_elvart.txt		15., 22:39:56	
012-Init_cFoldi_Teglaval_be.txt	26 bájt	2008. április 15., 22:39:56	Földibogyó létrehozása Téglával ütköztetve
012-Init_cFoldi_Teglaval_elvart.txt	83 bájt	2008. április 15., 22:39:56	
013-Init_cFures_Teglaval_be.txt	26 bájt	2008. április 15., 22:39:56	Fűrészbogyó létrehozása Téglával ütköztetve
013-Init_cFures_Teglaval_elvart.txt	83 bájt	2008. április 15., 22:39:56	
014-Init_cKobog_Bogyoval_be.txt	26 bájt	2008. április 15., 22:39:56	Kőbogyó létrehozása másik Bogyóval ütköztetve
014-Init_cKobog_Bogyoval_elvart.txt	83 bájt	2008. április 15., 22:39:56	
015-Init_cFoldi_Bogyoval_be.txt	26 bájt	2008. április 15., 22:39:56	Földibogyó létrehozása másik Bogyóval ütköztetve
015-Init_cFoldi_Bogyoval_elvart.txt	83 bájt	2008. április 15., 22:39:56	
016-Init_cFures_Bogyoval_be.txt	26 bájt	2008. április 15., 22:39:56	Fűrészbogyó létrehozása másik Bogyóval ütköztetve
016-Init_cFures_Bogyoval_elvart.txt	83 bájt	2008. április 15., 22:39:56	
017-Init_cKobog_Szelvennyel_be.txt	33 bájt	2008. április 15., 22:39:56	Kőbogyó létrehozása Szelvénnyel ütköztetve
017-Init_cKobog_Szelvennyel_elvart.txt	40 bájt	2008. április 15., 22:39:56	
018-Init_cFoldi_Szelvennyel_be.txt	33 bájt	2008. április 15., 22:39:56	Földibogyó létrehozása Szelvénnyel ütköztetve
018-Init_cFoldi_Szelvennyel_elvart.txt	40 bájt	2008. április 15., 22:39:56	
019-Init_cFures_Szelvennyel_be.txt	33 bájt	2008. április 15., 22:39:56	Fűrészbogyó létrehozása Szelvénnyel ütköztetve
019-Init_cFures_Szelvennyel_elvart.txt	140 bájt	2008. április 15., 22:39:56	
020-Init_cKobog_dBogyo_be.txt	34 bájt	2008. április 15., 22:39:56	Bogyó törlése Kőbogyó törlése
020-Init_cKobog_dBogyo	139 bájt	2008. április 15., 22:39:56	

_elvart.txt			
021-Init_cFures_dBogyo_be.txt	34 bájt	2008. április 15., 22:39:56	Fűrészbogyó törlése
021-Init_cFures_dBogyo_elvart.txt	139 bájt	2008. április 15., 22:39:56	
022-Init_cFoldi_dBogyo_be.txt	34 bájt	2008. április 15., 22:39:56	Földibogyó törlése
022-Init_cFoldi_dBogyo_elvart.txt	139 bájt	2008. április 15., 22:39:56	
023-Init_cTegla_dTegla_be.txt	34 bájt	2008. április 15., 22:39:56	Tégla törlés
023-Init_cTegla_dTegla_elvart.txt	139 bájt	2008. április 15., 22:39:56	
024-Utkozes_Teglaval_be.txt	49 bájt	2008. április 15., 22:39:56	Ütközések Ütközés Téglával
024-Utkozes_Teglaval_elvart.txt	222 bájt	2008. április 15., 22:39:56	
025-Utkozes_Kobogyoval_elvart.txt	287 bájt	2008. április 15., 22:39:56	Ütközés Bogyóval Kőbogyóval
025-Utkozes_Kobogyoval_be.txt	49 bájt	2008. április 15., 22:39:56	
026-Utkozes_Foldibogyoval_be.txt	49 bájt	2008. április 15., 22:39:56	Földibogyóval
026-Utkozes_Foldibogyoval_elvart.txt	287 bájt	2008. április 15., 22:39:56	
027-Utkozes_Fureszbogyoval_be.txt	49 bájt	2008. április 15., 22:39:56	Fűrészbogyóval
027-Utkozes_Fureszbogyoval_elvart.txt	287 bájt	2008. április 15., 22:39:56	
028-Utkozes-Sajattal_nincsFureszhatas_nincsKobogyo_be.txt	85 bájt	2008. április 15., 22:39:56	Ütközés saját Szelvénnyel Fűrészbogyóhatás nélkül a cél szelvényen Kőbogyóhatás nélkül
028-Utkozes-Sajattal_nincsFureszhatas_nincsKobogyo_elvart.txt	589 bájt	2008. április 15., 22:39:56	
029-Utkozes-Sajattal_nincsFureszhatas_vanKobogyo_be.txt	122 bájt	2008. április 15., 22:39:56	Fűrészbogyóhatás nélkül a cél szelvényen Kőbogyóhatással

029-Utkozes-Sajattal_nincsFureszhatas_vanKobogyo_elvart.txt	710 bájt	2008. április 15., 22:39:56	
030-Utkozes-Sajattal_vanFureszhatas_nincsKobogyo_elvart.txt	750 bájt	2008. április 15., 22:39:56	Fűrészbogyóhatással a cél szelvényen Kőbogyóhatás nélkül
030-Utkozes-Sajattal_vanFureszhatas_nincsKobogyo_be.txt	98 bájt	2008. április 15., 22:39:56	
031-Utkozes-Sajattal_vanFureszhatas_vanKobogyo_be.txt	143 bájt	2008. április 15., 22:39:56	Fűrészbogyóhatással a cél szelvényen Kőbogyóhatással
031-Utkozes-Sajattal_vanFureszhatas_vanKobogyo_elvart.txt	731 bájt	2008. április 15., 22:39:56	
032-Utkozes-Idegennel_nincsFureszhatas_nincsKobogyo_be.txt	56 bájt	2008. április 15., 22:39:56	Ütközés idegen Szelvénnyel Fűrészbogyóhatás nélkül a cél szelvényen Kőbogyóhatás nélkül
032-Utkozes-Idegennel_nincsFureszhatas_nincsKobogyo_elvart.txt	334 bájt	2008. április 15., 22:39:56	
033-Utkozes-Idegennel_nincsFureszhatas_vanKobogyo_be.txt	77 bájt	2008. április 15., 22:39:56	Fűrészbogyóhatás nélkül a cél szelvényen Kőbogyóhatással
033-Utkozes-Idegennel_nincsFureszhatas_vanKobogyo_elvart.txt	355 bájt	2008. április 15., 22:39:56	
034-Utkozes-Idegennel_vanFureszhatas_nincsKobogyo_be.txt	69 bájt	2008. április 15., 22:39:56	Fűrészbogyóhatással a cél szelvényen Kőbogyóhatás nélkül
034-Utkozes-Idegennel_vanFureszhatas_nincsKobogyo_elvart.txt	390 bájt	2008. április 15., 22:39:56	
035-Utkozes-Idegennel_vanFureszhatas_vanKobogyo_be.txt	90 bájt	2008. április 15., 22:39:56	Fűrészbogyóhatással a cél szelvényen Kőbogyóhatással
035-Utkozes-Idegennel_vanFureszhatas_vanKobogyo_elvart.txt	368 bájt	2008. április 15., 22:39:56	
036-Leptetes_Fordulas_nelkul_be.txt	28 bájt	2008. április 15., 22:39:56	Léptetések Léptetések különböző irányokba Léptetés fordulás nélkül

036-Leptetes_Fordulas_nelkul_elvart.txt	138 bájt	2008. április 15., 22:39:56	
037-Leptetes_Fordulas_235_balra_be.txt	39 bájt	2008. április 15., 22:39:56	Léptetés 235 fokos balra fordulás után
037-Leptetes_Fordulas_235_balra_elvart.txt	150 bájt	2008. április 15., 22:39:56	
038-Leptetes_Fordulas_45_balra_be.txt	39 bájt	2008. április 15., 22:39:56	Léptetés 45 fokos balra fordulás után
038-Leptetes_Fordulas_45_balra_elvart.txt	150 bájt	2008. április 15., 22:39:56	
039-Leptetes_Fordulas_45_jobbra_be.txt	39 bájt	2008. április 15., 22:39:56	Léptetés 45 fokos jobbra fordulás után
039-Leptetes_Fordulas_45_jobbra_elvart.txt	149 bájt	2008. április 15., 22:39:56	
040-Leptetes_Fordulas_235_jobbra_be.txt	39 bájt	2008. április 15., 22:39:56	Léptetés 235 fokos jobbra fordulás után
040-Leptetes_Fordulas_235_jobbra_elvart.txt	151 bájt	2008. április 15., 22:39:56	
041-Leptetes_1Kobogyo_Ures_Kukac_be.txt	57 bájt	2008. április 15., 22:39:56	Kőbogyó „léptetése” a Kukacban a farok felé. 1 Kőbogyó léptetése üres Kukacban
041-Leptetes_1Kobogyo_Ures_Kukac_elvart.txt	276 bájt	2008. április 15., 22:39:56	
042-Leptetes_1Kobogyo_NemUres_Kukac_be.txt	110 bájt	2008. április 15., 22:39:56	1 Kőbogyó léptetése Kukacban ami 1 Kőbogyót tartalmaz a farkában
042-Leptetes_1Kobogyo_NemUres_Kukac_elvart.txt	421 bájt	2008. április 15., 22:39:56	
043-Leptetes_2Kobogyo_Ures_Kukac_be.txt	86 bájt	2008. április 15., 22:39:56	Több (2) Kőbogyó léptetése üres Kukacban
043-Leptetes_2Kobogyo_Ures_Kukac_elvart.txt	406 bájt	2008. április 15., 22:39:56	

10.1.2. Tesztelési útmutatás

Fordítás

Amennyiben a java fordító és futtató fájl elérési helye környezeti változó, a fordítás a compile.bat futtatásával történik. Ha nem, akkor kénytelenek vagyunk begépelni a parancssorba a fordító teljes elérési útját. A következő parancsot kell kiadnunk (Tetragon_Proto legyen az aktuális könyvtár; a class-ok package-ben vannak, ezért kell innen kiadni a parancsot):

```
X:\...\Tetragon_Proto>"Y:\...\Java\jdk...\bin\javac.exe" -encoding utf8
tetragonteamkukacfarm\*.java
```

A karakterkódolással is problémák lehetnek, ugyanis a konzolra való kiírások ill. a kommentek ékezetes karaktereket is tartalmaznak és a szöveges forrásfájlok UTF-8-ban lettek kódolva, így ezt is meg kell adni.

Futtatás

A futtatót indítjuk el a fő osztállyal paraméterezve. A program két paramétere: a bemeneti parancsok, illetve a kimeneti fájlnév. Az előre elkészített teszteseteket így könnyen lehet futtatni. Pl.:

```
X:\...\Tetragon_Proto>"Y:\...\Java\jdk...\bin\java.exe"
tetragonteamkukacfarm/Tetragon_Team_Kukacfarm tesztek\028-Utkozes-
Sajattal_nincsFureszhatas_nincsKobogyo_be.txt tesztek\028-Utkozes-
Sajattal_nincsFureszhatas_nincsKobogyo_ki.txt
```

A mellékelt run.bat állomány futtatásával az összes előre létrehozott tesztesetet lefuttathatjuk.

Kimenet ellenőrzése

Az ellenőrzésre létrehozott Proto_Tester osztályt kell futtatni. Paraméterei: a kimeneti fájl neve, az elvárt kimenet fájlneve, és opcionálisan az ellenőrző osztály kimeneti fájlneve. Ha ezt elhagyjuk, a képernyőre írja ki a kimenetet. Pl.:

```
X:\...\Tetragon_Proto>"Y:\...\Java\jdk...\bin\java.exe
tetragonteamkukacfarm/Proto_Tester tesztek\028-Utkozes-
Sajattal_nincsFureszhatas_nincsKobogyo_ki.txt tesztek\028-Utkozes-
Sajattal_nincsFureszhatas_nincsKobogyo_elvart.txt tesztek\028-Utkozes-
Sajattal_nincsFureszhatas_nincsKobogyo_teszteredmeny.txt
```

A mellékelt test.bat illetve test_to_file.bat állományok futtatásával lefut az összes teszteset ellenőrzése a képernyőre, illetve a névnek megfelelő fájlba.

A Proto_Tester osztály a whitespace-ek figyelmen kívül hagyásával leellenőrzi a sorok egyezését, ahol eltér, ott kiírja az eltérést és az eltérő adatokat tartalmazó sorok sorszámát. Az ellenőrzés végén kiír egy üzenetet a teszteset sikerességéről.

10.2. Eredmények összefoglalása

10.2.1. Tesztek eredményei

Minden, a Részletes tervek c. beadandó korábbi dokumentumban szereplő teszteset sikeresen lefutott a szükséges módosítások után. Részletesen kifejtve:

Inicializáció

Kígyó létrehozása

Szabad pozícióra: sikeresen lefutott
Téglával ütközetve: sikeresen lefutott
Bogyóval ütköztetve: sikeresen lefutott
Szelvénnyel ütköztetve: sikeresen lefutott

Tégla létrehozása

Szabad pozícióra: sikeresen lefutott
Téglával ütközetve: sikeresen lefutott
Bogyóval ütköztetve: sikeresen lefutott
Szelvénnyel ütköztetve: sikeresen lefutott

Bogyók létrehozása

Kőbogyó létrehozása szabad pozícióra: sikeresen lefutott
Mezeibogyó létrehozása szabad pozícióra: sikeresen lefutott
Fűrészbogyó létrehozása szabad pozícióra: sikeresen lefutott
Kőbogyó létrehozása Téglával ütköztetve: sikeresen lefutott
Mezeibogyó létrehozása Téglával ütköztetve: sikeresen lefutott
Fűrészbogyó létrehozása Téglával ütköztetve: sikeresen lefutott
Kőbogyó létrehozása másik Bogyóval ütköztetve: sikeresen lefutott
Mezeibogyó létrehozása másik Bogyóval ütköztetve: sikeresen lefutott
Fűrészbogyó létrehozása másik Bogyóval ütköztetve: sikeresen lefutott
Kőbogyó létrehozása Szelvénnyel ütköztetve: sikeresen lefutott
Mezeibogyó létrehozása Szelvénnyel ütköztetve: sikeresen lefutott
Fűrészbogyó létrehozása Szelvénnyel ütköztetve: sikeresen lefutott

Bogyó törlése

Kőbogyó törlése: sikeresen lefutott
Fűrészbogyó törlése: sikeresen lefutott
Mezeibogyó törlése: sikeresen lefutott

Tégla törlés

Ütközések

Ütközés Téglával: sikeresen lefutott
Ütközés Bogyóval: sikeresen lefutott
Kőbogyóval: sikeresen lefutott
Mezeibogyóval: sikeresen lefutott
Fűrészbogyóval: sikeresen lefutott
Ütközés saját Szelvénnyel: sikeresen lefutott
Fűrészbogyóhatás nélkül a cél szelvényen Kőbogyóhatás nélkül: sikeresen lefutott
Fűrészbogyóhatás nélkül a cél szelvényen Kőbogyóhatással: sikeresen lefutott
Fűrészbogyóhatással a cél szelvényen Kőbogyóhatás nélkül: sikeresen lefutott
Fűrészbogyóhatással a cél szelvényen Kőbogyóhatással: sikeresen lefutott
Ütközés idegen Szelvénnyel
Fűrészbogyóhatás nélkül a cél szelvényen Kőbogyóhatás nélkül: sikeresen lefutott
Fűrészbogyóhatás nélkül a cél szelvényen Kőbogyóhatással: sikeresen lefutott
Fűrészbogyóhatással a cél szelvényen Kőbogyóhatás nélkül: sikeresen lefutott
Fűrészbogyóhatással a cél szelvényen Kőbogyóhatással: sikeresen lefutott

Léptetések

Léptetések különböző irányokba
Léptetés fordulás nélkül: sikeresen lefutott
Léptetés 235 fokos balra fordulás után: sikeresen lefutott
Léptetés 45 fokos balra fordulás után: sikeresen lefutott
Léptetés 45 fokos jobbra fordulás után: sikeresen lefutott
Léptetés 235 fokos jobbra fordulás után: sikeresen lefutott
Kőbogyó „léptetése” a Kukacban a farok felé.
1 Kőbogyó léptetése üres Kukacban: sikeresen lefutott
1 Kőbogyó léptetése Kukacban ami 1 Kőbogyót tartalmaz a farkában: sikeresen lefutott
Több (2) Kőbogyó léptetése üres Kukacban: sikeresen lefutott

10.2.2. Változtatások

- A tesztesetekben a Kukacok létrehozásánál a szöghöz hozzá kellett adni 90°-ot (jobbra van a 0°). Emiatt a fordulások eredménye is más lett, mindenhol át kellett írni.
- A tesztesetekben jobxxx és balxxx parancsnál pótolni kellett a kukac azonosítóját a zárójelben.
- Ütközés saját szelvénnyel, kőbogyóval: nem a 3., hanem a 4. szelvény fog ütközni, ezért a kőbogyót még egyszer léptetni kell.
- 030-Utkozes-Sajattal_vanFureszhatas_nincsKobogyo_elvart.txt: ha a kukac fölveszi a fűrészbogyót, akkor is csak 7 szelvénye lesz a fején kívül.
- 2 függvény lett hozzáadva:
- `boolean Pozicio.equals(Object o)`
- `boolean Poziciok.contains(Pozicio p)`
- A protó specifikációja szerint egy helyre csak egy objektumot lehet létrehozni, ehhez érték szerint is össze kell tudni hasonlítani.
- Kőbogyó és fűrészhatás fejben egyszerre hogyan jelenik meg parancsszinten? Megoldás: kő + fűrész = 3.

10.3. Értékelés

10.3.1. Deák véleménye

A kódolásos feladatokat a csapat nem tervező része végezte el, Paulik és én ebben nem vettünk részt, de megterveztük a protot, és segítettük a kódolókat a dokumentáció helyes megértésében, illetve a folyamat közben fellépő hibákat kommenteztük, megoldásokat javasoltunk. Sajnos a megtervezett rendszer egy apró részlete nem került implementálásra, ugyanis rövid ideig úgy tűnt a kódolóknak, hogy az csak felesleges redundancia, de mi tervezők később rávilágítottunk ezen entitás (kukac ID-je) jelentőségére. További észrevétel – sajnos ez is már későn érkezett – hogy két sor plusz kóddal, hatalmas dokumentációs feladat (a kukacok irányszögének értelmezése) alól mentesíthették volna magukat a programírók. A proto tervezése során apróbb kommunikációs hiba lépett fel a feladat kiírója, és a csapat között, melyet végül kompromisszumos döntéssel sikerült megoldani.

A csapat együttműködésében én nem észleltem kommunikációs hibát úgy, mint a korábbi fejlesztési időszakokban, de ez köszönhető annak is, hogy kevésbé aktívan vettem

részt az implementációs feladatban. Ennek ellenére úgy gondolom, hogy ha a megjegyzéseinket hamarabb nyilvánosságra tudjuk hozni, és a kódolók szíveskednek azokat figyelembe venni, akkor hatékonyabb lehetett volna a munka. Összességében pozitívan értékelem ezt a fázist.

10.3.2. Katus véleménye

Ezen fázis végére a csapattagok munkában való részvételének ideje nagyjából teljesen kiegyenlítődött. Összességében immáron elmondható, hogy mindenki egyenlő mértékben vette ki a részét a munkában a projekt kezdete óta.

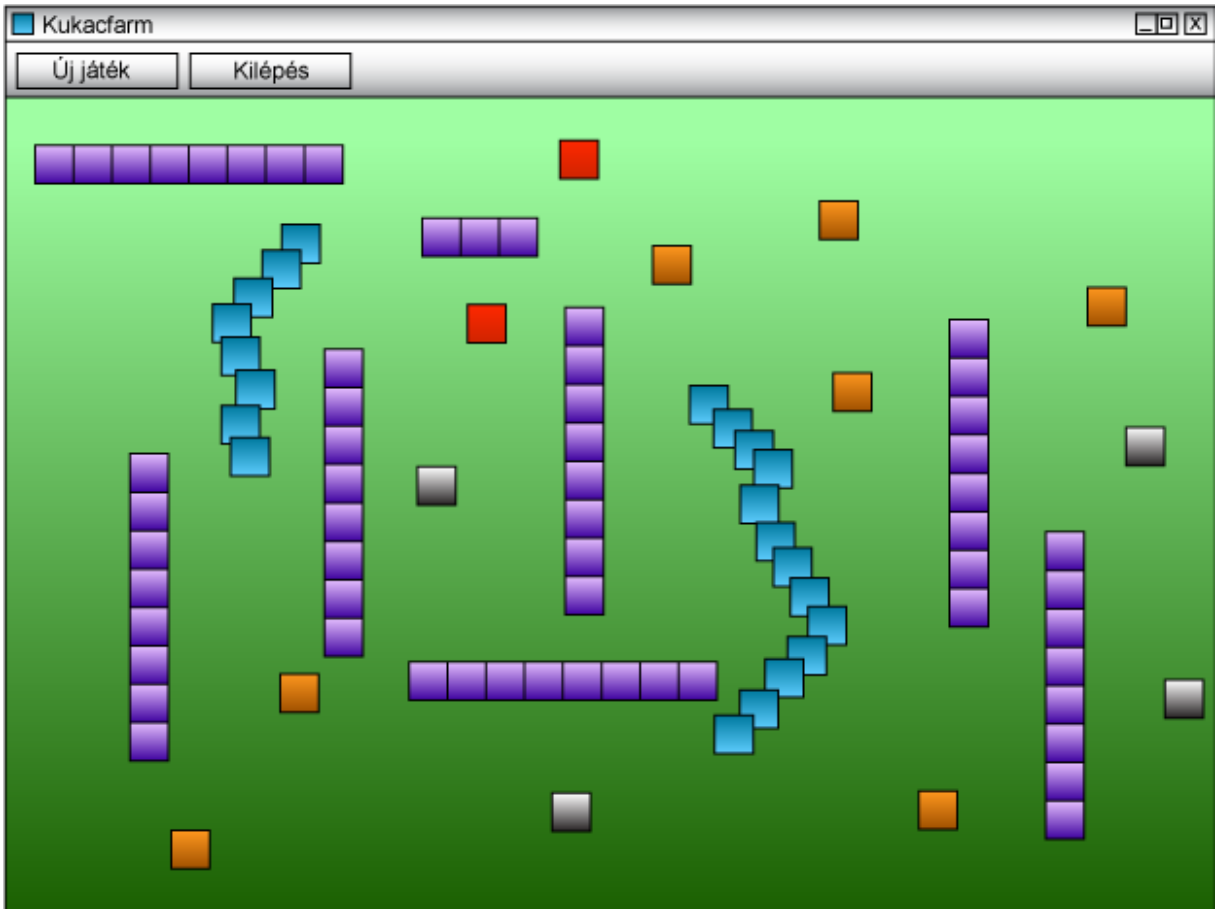
10.3.3. Százalékos mérték

A következő értékeket gondoljuk reálisnak:

Deák László	25%
Katus Kristóf	25%
Ofella Péter	25%
Paulik Tamás	25%

11. fejezet Grafikus felület specifikálása

11.1. A menürendszer, a kezelői felület grafikus képe



Az ábrán a grafikus felület látható. A felületen helyet kapott két gomb, új játék kezdéséhez, ill. a kilépéshez. Ezek az ablak felső sávjában helyezkednek el. Az ablak fő részét a játéktér tölti ki, itt jelenik meg a pálya, rajta az elkerülendő téglák, a bekapható bogyók és maguk a kígyók. Amint látható, minden különböző típusú objektumnak más a színe, alakjuk viszont mindegyiknek egy egyszerű négyzet.

Az új játék két kígyóval indul, amiket billentyűzetről lehet vezérelni, tehát jobbra vagy balra forgatni.

11.2. A felület működésének elve

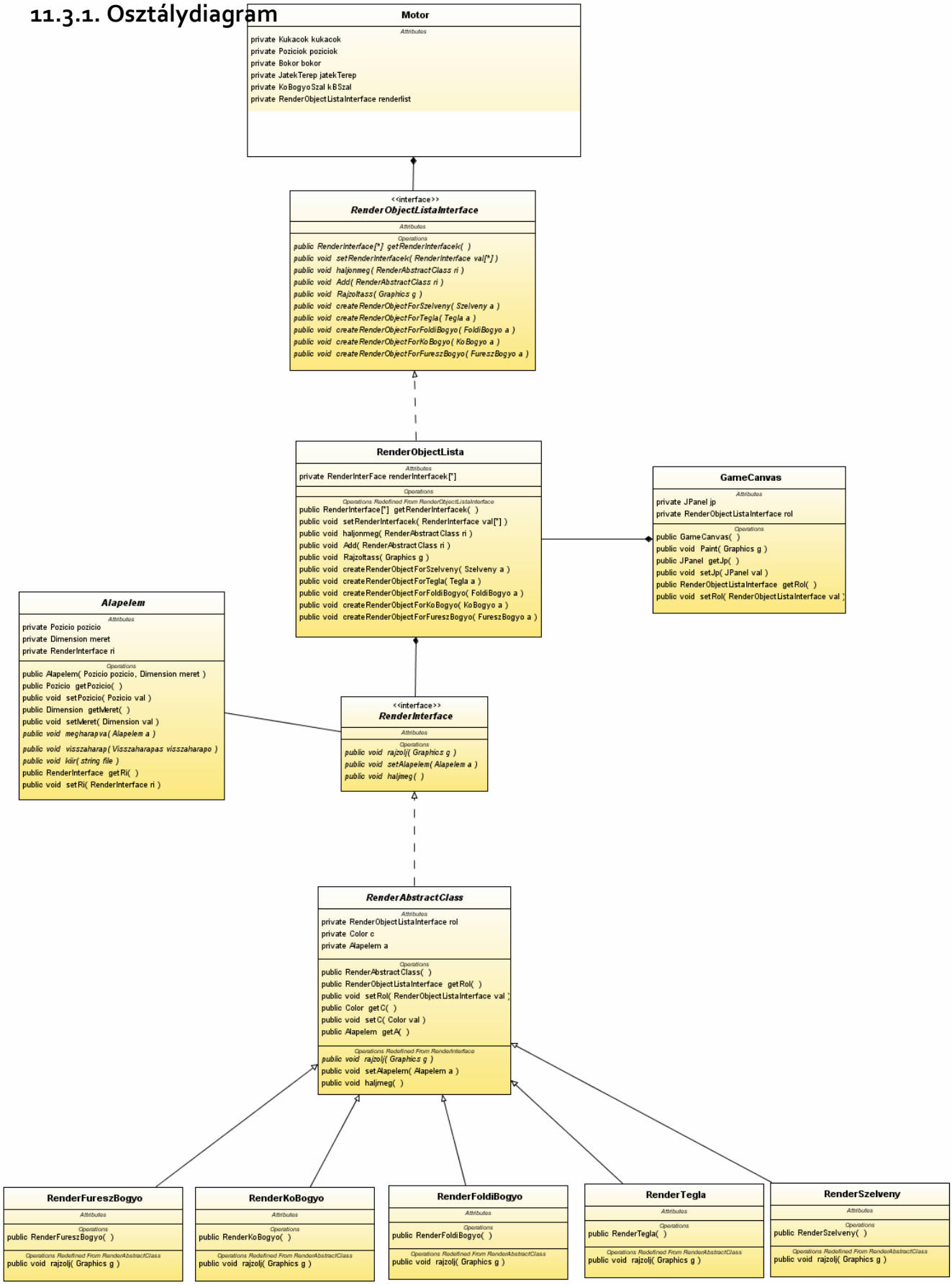
A grafikus oldalon kialakított osztályhierarchiában minden kirajzolandó objektum megvalósít egy RenderInterface interfészt. Ez biztosítja az egységes kezelését minden grafikus objektumnak. Mivel minden modellbeli nem wrapper és nem kollekció osztály – tehát ami kirajzolható – az Alapelem osztályból származik, ezért, hogy a modell és a grafikus rész között kapcsolatot teremtsünk csak ebben az osztályban kell elhelyezni modell részről RenderInterface típusú referenciát az adott grafikus objektumra. A grafikus objektumok pedig tudnak egy referencián keresztül a modellbeli megfelelőjükről. Ez az adatredundancia-elkerüléséhez feltétlenül szükséges, hiszen így nem történik felesleges adatduplikáció, a modelltől a szükséges információk elkérhetők.

Grafikus oldalon a tényleges példányosítást és az aktuális kirajzolást a RenderObjectListaInterface-t megvalósító RenderObjectLista objektum végzi. Ez tulajdonképpen egy egyszerű lista, ami RenderInterface-szel kompatibilis objektumokat tárol. Kirajzoláskor ezen a listán fut végig, meghívva minden egyes tárolt elem rajzolj() metódusát. Mivel minden kirajzolandó modellbeli objektumnak van grafikus megfelelője, ezért valahogy jelezni kell, hogyha modell oldalon az adott objektum megszűnik. Ezt az Alapelem jelzi a haljmeg() híváson keresztül a RenderObjectLista-nak.

A RenderObjectLista-ba új elem felvétele akkor történik, amikor az az elem éppen létrejön a modellben. A modell a grafikus részt a RenderObjectListaInterface-szel kompatibilis objektumon keresztül látja, ezen keresztül szól a megfelelő függvényhívással (createRenderObjectFor…), hogy a megfelelő grafikus objektum jöjjön létre.

11.3. A grafikus rendszer architektúrája

11.3.1. Osztálydiagram



11.3.2. Osztályleírások

RenderObjectListaInterface

A kirajzoló osztályokat tartalmazó osztály interfésze. Ennek segítségével később más implementációjú kirajzoló osztályt is könnyedén lehet beilleszteni a programba. A create… kezdetű függvényekkel hozható létre új rajzoló objektum a megadott alapelemnek, a haljonmeg függvénnyel lehet egy ilyet törölni, a Rajzoltass(Graphics g) függvény pedig a játéktér újrarajzolását eredményezi.

RenderObjectLista

A kirajzoló osztályokat tartalmazó osztály implementációja. Tartalmaz egy tömböt a kirajzoló osztályokról, és a Rajzoltass(Graphics g) függvény meghívásakor meghívja ezek rajzolj(Graphics g) függvényét.

GameCanvas

A kirajzolást egy Jpanelra végezzük, ezt tartalmazza ez az osztály, ami rajzolás szüksége esetén meghívja a RenderObjectLista osztály Rajzoltass(Graphics g) függvényét.

RenderInterface

A rajzoló osztályok interfésze. A rajzolj(Graphics g) függvénnyel rajzolják ki a hozzájuk tartozó alapelemet, a setAlapelem-mel beállítható ez az alapelem, a haljmeg()-gel pedig törli magát a tárolójából.

RenderAbstractClass

A rajzoló osztályok összes közös metódusának implementációját tartalmazza. Mivel mindegyik rajzoló objektum rendelkezik szín és alapelem tulajdonságokkal, ezeket elegendő egy helyen megvalósítani. Az interfészből nem valósítja meg a rajzolj(Graphics g) függvényt, mert a különböző objektumokat esetleg más-más módon kell kirajzolni.

RenderFureszBogyo

A fűrészbogyóhoz tartozó rajzoló osztály.

RenderKoBogyo

A kőbogyóhoz tartozó rajzoló osztály.

RenderFoldiBogyo

A földibogyóhoz tartozó rajzoló osztály.

RenderTegla

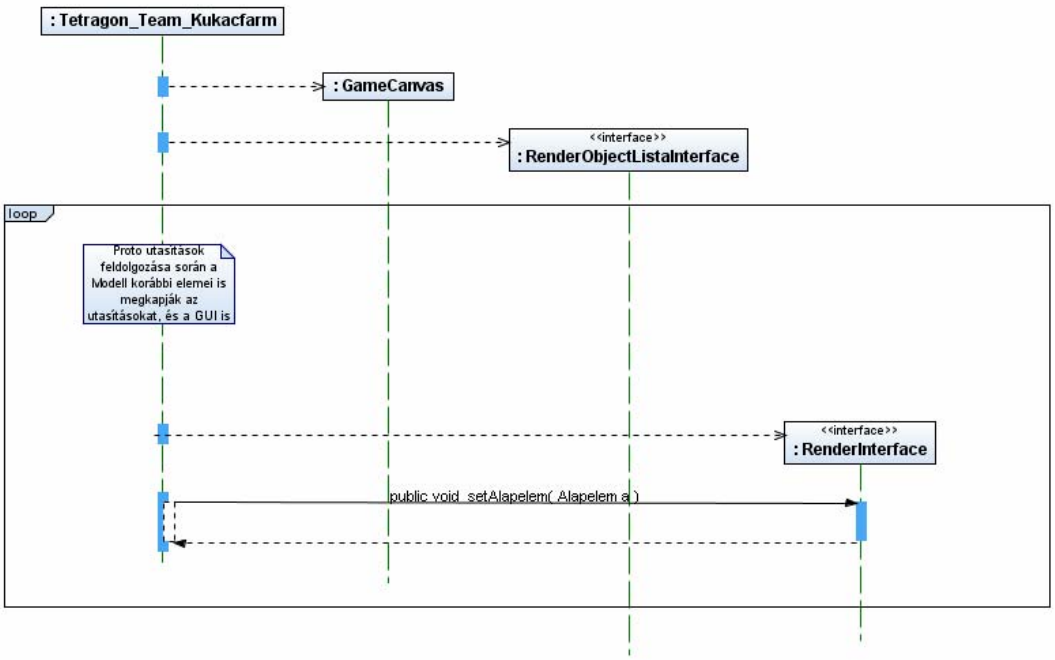
A téglához tartozó rajzoló osztály.

RenderSzelveny

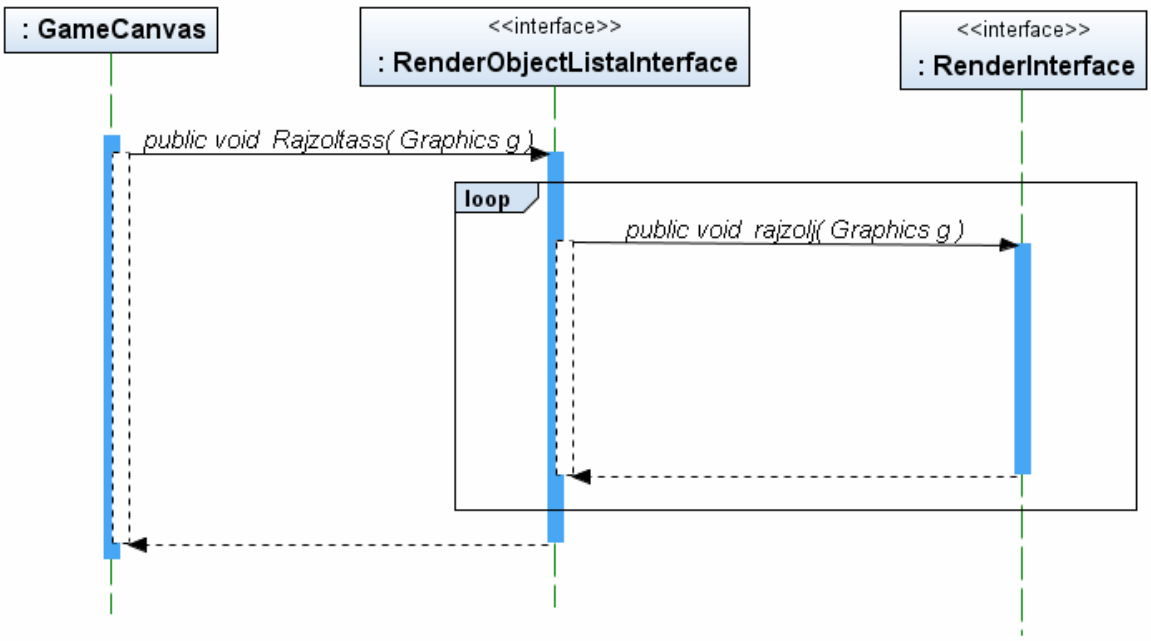
A szelvényhez tartozó rajzoló osztály.

11.4. A grafikus objektumok kapcsolata az alkalmazói rendszerrel

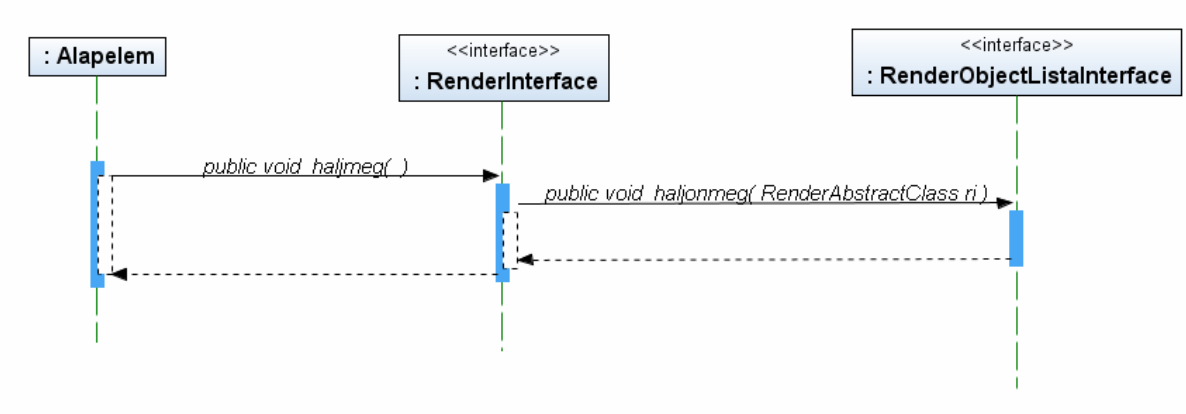
11.4.1. Grafikus felület inicializálása



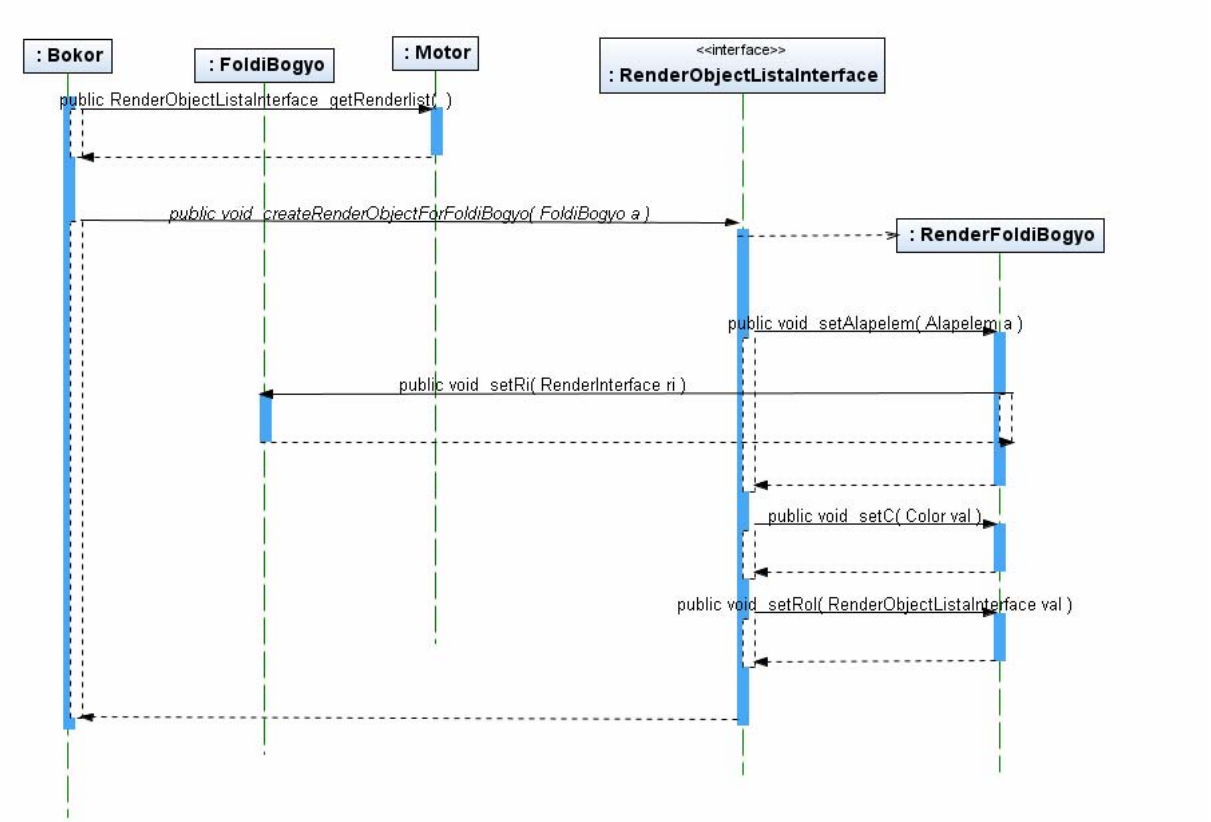
11.4.2. Grafikus objektum kirajzolása



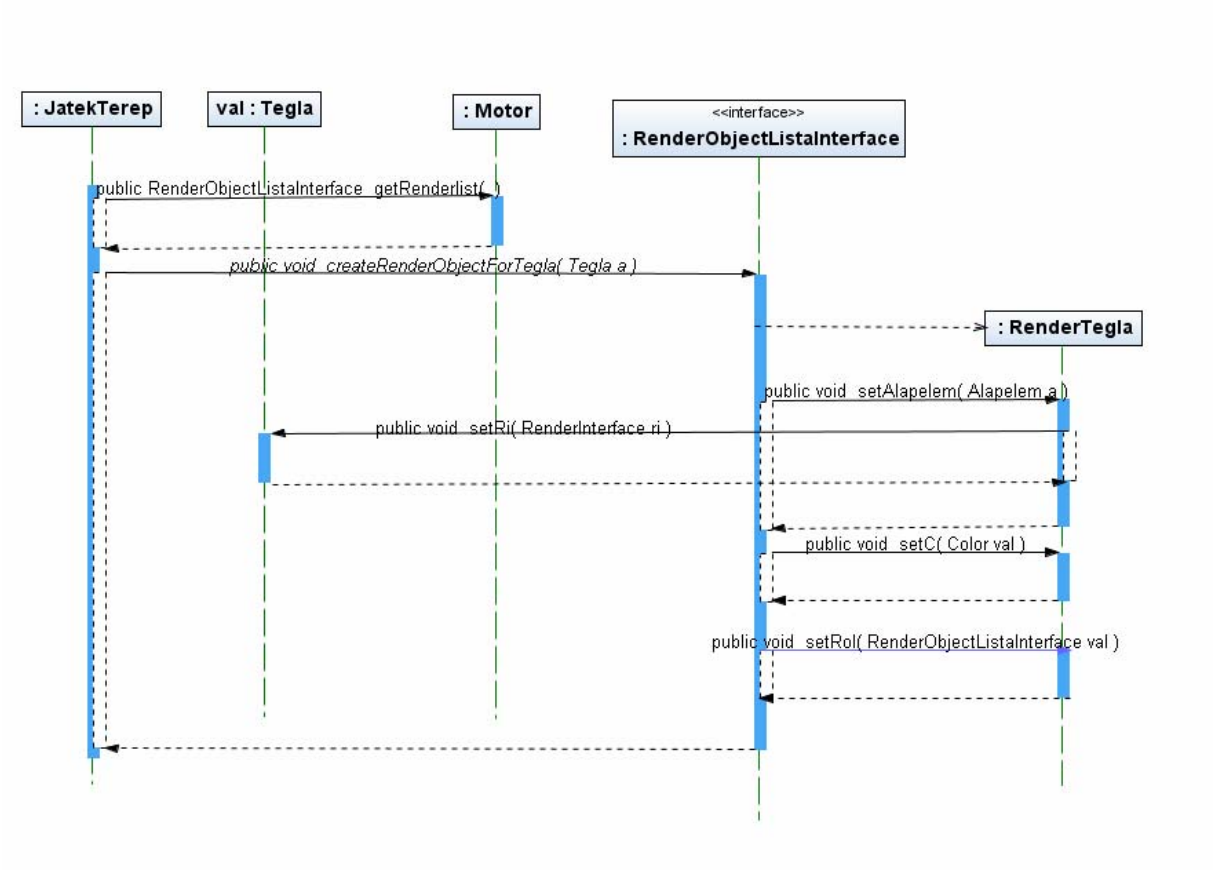
11.4.3. Grafikus objektum megszüntetése



11.4.4. Bogyó grafikus objektum létrehozása



11.4.5. Tégla grafikus objektum létrehozása



11.4.6. Szelvény grafikus objektum létrehozása

12. fejezet Grafikus változat készítése

- Ezen a héten nem kellett anyagot beadni.

13. fejezet Grafikus változat beadása

13.1. Útmutatás a grafikus változathoz

13.1.1. Fájllista

Fájl neve	*Fájl mérete*	*Keletkezési idő*	*Leírás*
compile.bat	155 bájt	2008.05.07., 00:23	A forrásállományok lefordítását végző batch-fájl.
runc.bat	72 bájt	2008.05.07., 00:23	A játékot egy adott pályával futtató batch-fájl (nem csomagolt, a class-ok nincsenek jar-ban).
runj.bat	102 bájt	2008.05.07., 00:23	A játékot egy adott pályával futtató batch-fájl (jar fájlt futtat).
tesztpalya.txt	94 bájt	2008.05.07., 00:23	A játékhoz mellékelt egy pálya.
tesztpalya2.txt	95 bájt	2008.05.07., 00:23	A játékhoz mellékelt másik pálya.
\dist	146980 bájt	2008.05.07., 00:23	Az installálható változatot tartalmazó könyvtár.
\javadoc	1197289 bájt	2008.05.07., 00:23	Ez a könyvtár tartalmazza a Javadoc speciális kommentekből és a forrásfájlokból generált fejlesztői referenciát.
\tetragonteamkukacfarm	178770 bájt	2008.05.07., 00:23	Ez a könyvtár tartalmazza a forrásfájlokat, a továbbiakban nem írjuk ki a teljes elérési útvonalat.
%\Alapelem.java	5111 bájt	2008.05.07., 00:23	Az alapelem az objektumrendszer egy alapvető, absztrakt eleme, amelyből minden olyan osztály leszármazik, amely részt vesz a játéktér kialakításában. Minden elemnek lehet helye mérete és azonosítója.
%\Bogyo.java	3446 bájt	2008.05.07., 00:23	A bogyó egy absztrakt osztály, belőle származnak le az egyes bogyófajták. A bogyó osztály az alapelemből származik le.
%\Bokor.java	11602 bájt	2008.05.07., 00:23	A bokor az az osztály, amely egy listát tartalmaz a pályán jelenlévő bogyókról. Tárolja, kezeli és menedzseli a pályán lévő bogyókat. Felel a bogyók létrehozásáért és törléséért. Egy példány létezik belőle.
%\FoldiBogyo.java	2900 bájt	2008.05.07., 00:23	A bogyó osztály leszármazottja, a játékspecifikációban leírt mezei bogyót valósítja meg.
%\FoldiBogyoVisszaharapas.java	6004 bájt	2008.05.07., 00:23	Ha a megharapott alapelem földibogyó, akkor az földibogyóvisszaharapás objektumot hoz létre, és átadja azt a kölcsönhatásban résztvevő félnek, a látogató modell szerint. A visszaharapás interface-t valósítja meg.
%\FureszBogyo.java	2699 bájt	2008.05.07., 00:23	A bogyó osztály leszármazottja, a játékspecifikációban leírt fűrészbogyót valósítja meg.
%\FureszBogyoVisszaharapas.java	6250 bájt	2008.05.07., 00:23	Ha a megharapott alapelem fűrészbogyó, akkor az fűrészbogyóvisszaharapás objektumot hoz létre, és átadja azt a kölcsönhatásban résztvevő félnek, a látogató modell szerint. A visszaharapás interface-t valósítja meg.
%\GameCanvas.java	7350 bájt	2008.05.07., 00:23	A kirajzolást egy Jpanelra végezzük, ezt tartalmazza ez az osztály, ami rajzolás szüksége esetén meghívja a RenderObjectLista osztály Rajzoltass(Graphics g) függvényét.
%\JatekTerep.java	7221 bájt	2008.05.07., 00:23	A játékterep az az osztály, amely egy listát tartalmaz a pályán jelenlévő fal objektumokról. Tárolja, kezeli és menedzseli őket. Felel a falak létrehozásáért. Egy példány létezik belőle.
%\KoBogyo.java	2653 bájt	2008.05.07., 00:23	A bogyó osztály leszármazottja, a játékspecifikációban leírt

			kőbogyót valósítja meg.
%\KoBogyoSzal.java	1285 bájt	2008.05.07., 00:23	A kőbogyószál a kőbogyók léptetéséért felelős.
%\KoBogyoVisszaharapas.java	5985 bájt	2008.05.07., 00:23	Ha a megharapott alapelem kőbogyó, akkor az kőbogyóvisszaharapás objektumot hoz létre, és átadja azt a kölcsönhatásban résztvevő félnek, a látogató modell szerint. A visszaharapás interface-t valósítja meg.
%\Kukac.java	20870 bájt	2008.05.07., 00:23	Egy kukacot reprezentáló osztály, rendelkezik attribútumként egy olyan változóval hogy életben van e, ismeri saját mozgásának irányát, képes saját hosszát megnövelni, megrövidíteni, lépni és tárolja szelvényeit.
%\Kukacok.java	8606 bájt	2008.05.07., 00:23	Egy homogén kollekció Kukacok tárolására és az azokon elvégzendő alapvető műveletek koordinálására.
%\Motor.java	23505 bájt	2008.05.07., 00:23	A játék léptetéséért az objektumok összehangolásával foglalkozó objektum. Az időzítéshez szálat használ, melyet majd a proto fázis után implementálunk.
%\Pozicio.java	4646 bájt	2008.05.07., 00:23	Az osztály Alapelem típusú objektumok térbeli pozíciójának tárolására szolgál.
%\Poziciok.java	4501 bájt	2008.05.07., 00:23	A Pozicio-k tárolására szolgáló osztály.
%\Proto_Tester.java	4961 bájt	2008.05.07., 00:23	A prototípus tesztelésére szolgáló osztály.
%\RenderAbstractClass.java	3949 bájt	2008.05.07., 00:23	A rajzoló osztályok összes közös metódusának implementációját tartalmazza. Mivel mindegyik rajzoló objektum rendelkezik szín és alapelem tulajdonságokkal, ezeket elegendő egy helyen megvalósítani. Az interfészből nem valósítja meg a rajzolj(Graphics g) függvényt, mert a különböző objektumokat esetleg más-más módon kell kirajzolni.
%\RenderFoldiBogyo.java	888 bájt	2008.05.07., 00:23	A földibogyóhoz tartozó rajzoló osztály.
%\RenderFureszBogyo.java	892 bájt	2008.05.07., 00:23	A fűrészbogyóhoz tartozó rajzoló osztály.
%\RenderInterface.java	1069 bájt	2008.05.07., 00:23	A rajzoló osztályok interfésze. A rajzolj(Graphics g) függvénnyel rajzolják ki a hozzájuk tartozó alapelemet, a setAlapelem-mel beállítható ez az alapelem, a haljmeg()-gel pedig törli magát a tárolójából.
%\RenderKoBogyo.java	847 bájt	2008.05.07., 00:23	A kőbogyóhoz tartozó rajzoló osztály.
%\RenderObjectLista.java	4590 bájt	2008.05.07., 00:23	A kirajzoló osztályokat tartalmazó osztály implementációja. Tartalmaz egy tömböt a kirajzoló osztályokról, és a Rajzoltass(Graphics g) függvény meghívásakor meghívja ezek rajzolj(Graphics g) függvényét.
%\RenderObjectListeInterface.java	2932 bájt	2008.05.07., 00:23	A kirajzoló osztályokat tartalmazó osztály interfésze. Ennek segítségével később más implementációjú kirajzoló osztályt is könnyedén lehet beilleszteni a programba. A create… kezdetű függvényekkel hozható létre új rajzoló objektum a megadott alapelemnek, a haljonmeg függvénnyel lehet egy ilyet törölni, a Rajzoltass(Graphics g) függvény pedig a játéktér újrarajzolását eredményezi.
%\RenderSzelveny.java	1401 bájt	2008.05.07., 00:23	A szelvényhez tartozó rajzoló osztály.
%\RenderTegla.java	877 bájt	2008.05.07., 00:23	A téglához tartozó rajzoló osztály.
%\Szelveny.java	8112 bájt	2008.05.07., 00:23	A Kukac alapvető eleme.
%\SzelvenyVisszaharapas.java	6874 bájt	2008.05.07., 00:23	Ha a megharapott alapelem szelvény, akkor az szelvényvisszaharapás objektumot hoz létre, és átadja azt a kölcsönhatásban résztvevő félnek, a látogató modell szerint. A visszaharapás interface-t valósítja meg.

%\Tegla.java	4201 bájt	2008.05.07., 00:23	Az alapelemből leszármazó osztály. A falat reprezentálja.
%\TeglaVisszaharapas.java	5874 bájt	2008.05.07., 00:23	Ha a megharapott alapelem tégla, akkor az téglavisszaharapás objektumot hoz létre, és átadja azt a kölcsönhatásban résztvevő félnek, a látogató modell szerint. A visszaharapás interface-t valósítja meg.
%\Tetragon_Team_Kukacfarm.java	5155 bájt	2008.05.07., 00:23	A Tetragon_Team_Kukacfarm osztály felelős az inicializáció elindításáért. Ő hozza létre a Motor osztályt, amely elvégzi az inicializációt. Egy példány létezik belőle. Új játékot lehet vele indítani, és meglévőt lehet törölni.
%\Visszaharapas.java	1482 bájt	2008.05.07., 00:23	A visszaharapás interface a visszaharapások kohézív tulajdonságainak összessége. Leírja, hogyan kell kinéznie az őt megvalósító interfacek-nek. A függvények leírása az adott – interface-t megvalósító – osztály leírásában található.

13.1.2. Útmutatás

Fordítás

Amennyiben a java fordító és futtató fájl elérési helye környezeti változó, a fordítás a compile.bat futtatásával történik. Ha nem, akkor kénytelenek vagyunk begépelni a parancssorba a fordító teljes elérési útját. A következő parancsot kell kiadnunk (a tetragonteamkukacfarm könyvtár könyvtára legyen az aktuális könyvtár; a class-ok package-ben vannak, ezért kell innen kiadni a parancsot):

```
X:\...\Tetragon_Proto>"Y:\...\Java\jdk...\bin\javac.exe" -encoding utf8
tetragonteamkukacfarm\*.java
X:\...\Tetragon_Proto>"Y:\...\Java\jdk...\bin\jar.exe" cvf
tetragonteamkukacfarm.jar tetragonteamkukacfarm\*.class tesztpalya.txt
tesztpalya2.txt
```

A karakterkódolással is problémák lehetnek, ugyanis a konzolra való kiírások ill. a kommentek ékezetes karaktereket is tartalmaznak és a szöveges forrásfájlok UTF-8-ban lettek kódolva, így ezt is meg kell adni.

Futtatás

A futtatót indítjuk el a fő osztállyal paraméterezve. A program egy paramétere: a bemeneti parancsokat tartalmazó fájl neve, ami egy pálya felépítéséhez szükséges. Pl.:

```
X:\...\Tetragon_Proto>"Y:\...\Java\jdk...\bin\java.exe"
tetragonteamkukacfarm/Tetragon_Team_Kukacfarm tesztpalya2.txt
```

VAGY

```
X:\...\Tetragon_Proto>"Y:\...\Java\jdk...\bin\java.exe" -cp
tetragonteamkukacfarm.jar tetragonteamkukacfarm/Tetragon_Team_Kukacfarm
tesztpalya2.txt
```

A mellékelt runc.bat vagy runj.bat állomány futtatásával elindíthatjuk a játék nem csomagolt ill. csomagolt változatát.

Telepíthető változat

A telepíthető változatot a \dist könyvtárban mellékeltük. A Tetragon_Project_java.jar-t érdemes felparaméterezni valamelyik mellékelt pályával. Pl.:

```
java -jar Tetragon_Project_java.jar tesztpalya2.txt
```

A két játékos az ’A’ és ’D’ ill. a bal és jobb nyilakkal tudja irányítani a saját kukacát.

13.2. Módosítások

- Se a kukacok, se a bokor, se a játékterep nem ismerte a motort, ezért a konstruktorban hozzá kellett adni.
- Fűrészbogyós szelvényen van egy piros kör.
- Kukacok hozzáadása egy paraméter a motornak, ami létrehoz annyi kukacot, amennyi meg van adva, egy képlet szerint elhelyezve
- Bogyóhozzáadás rosszul volt megírva, a 'bogyokmaxszama'-t mindig növelte, ha hozzáadtunk egy bogyót.
- A motornak esetleg nem ártana ismernie valamilyen repaint()-et, ha netán újra akar rajzoltatni valamit. Ehhez a RenderObjectListaInterface-hez hozzáadtunk egy repaint()-et.
- Az újraindításnál a motornak tudnia kell a GameCanvas-t beállítani, így ezt is hozzáadtuk az interface-hez, csak átmenetileg van eltárolva a motorban.
- A játéktér mérete eltárolódik a motorban, hogy a bogyókat hozzá tudja adni.
- Hogy ne csak 90°-osakat lehessen fordulni, különválasztottuk a szelvények méretét, és a köztük lévő távolságot – sqrt(2)*méret már elegendő távolság, hogy ne legyen ütközés.

13.3. Értékelés

13.3.1. Százalékos mérték

A csapattagok hozzájárulása nem sokat változott, így marad mindenkinek a 25%-os részesedés.

Deák László	25%
Katus Kristóf	25%
Ofella Péter	25%
Paulik Tamás	25%

14. fejezet Projekt összefoglalás

A projekt során nyert tapasztalatokat, észrevételeket a következő kérdések megválaszolásán keresztül próbáljuk megfogalmazni.

14.1. Mit tanultak a projektből konkrétan és általában?

A tárgyat nagyon hasznosnak találtuk, a csapatmunka szépségeit most tapasztaltuk meg először a szoftverfejlesztés területén.

Bár a félév folyamán rendelkezésre álló idő nyilvánvalóan csak egy jelképes példán tette lehetővé a tanult eszközök, módszerek kipróbálását, mégis érdekes bepillantást nyerhettünk a csoportos, tervezett programfejlesztés részleteibe. A megoldandó feladat a tanult eszközökhöz képest egyértelműen kicsi, így nem is szerezhettünk a programfejlesztésből nagyobb gyakorlatot a félév során, a tárgynak azonban ez a tanulmányi lehetőségek korlátai miatt nem is lehetett célja. Viszont így is sok hasznos tapasztalatot szerezhettünk a tanultak gyakorlati alkalmazásából.

A tárgy egyik legfontosabb tapasztalata az volt, ahogy az információ áramlott a csapattagok között, hogy egymás munkáinak megértéséhez a megfelelő információt biztosítsuk, hiszen az elvégzendő feladatok szorosan egymásra épültek. Az egyik legnehezebb feladat a munkák megfelelő szétosztása volt. A kapott feladat sokszor csak a másik csoporttag munkájának befejezése után volt elvégezhető, és igényelte a más által elvégzett feladat megértését és egyértelmű értelmezését is.

14.2. Mi volt a legnehezebb és a legkönnyebb?

Viszonylag nehezebb volt a feladat modelljének alapötleteit kitalálni, a terveket részletesen kidolgozni, valamint a működő prototípust a tesztelésekkel együtt előállítani. Ezekhez képest viszonylag könnyebb volt a követelmények dokumentálása, a grafikus változat tervezése és beadása.

14.3. Összhangban állt-e az idő és a pontszám az elvégzendő feladatokkal?

Az első három héten a rendelkezésre álló idő elegendő volt, a pontszámot az elvégzett feladatnak megfelelőnek éreztük. Az analízis modell és a szkeleton tervezése és elkészítése már jóval több munkát jelentett, a feladatért járó pont azonban itt is megfelelt az elvégzendő munkának.

14.4. Ha nem, akkor hol okozott ez nehézséget?

A prototípus tervezésénél és a részletes terveknél már nem volt olyan nagy ugrás az elvégzendő munka mennyiségében, sőt, itt úgy éreztük, hogy a feladatért járó pontokat már kevesebb munkával szereztük. A prototípus elkészítése és tesztelése nagyon nagy munkát jelentett, az ezért járó pontokat kevésnek találtuk. Véleményünk szerint ez volt a legnagyobb feladat, mely a pontok számában nem tükröződött eléggé.

A grafikus változat tervezése és készítése viszonylag kevés munka volt, melyhez képest aránytalanul sok pont járt a feladatért.

14.5. Milyen változtatási javaslatuk van?

A tárgyból nagy segítség volna, ha az elkészítendő dokumentumok jobban meghatározattak lennének, azaz a csapattagoknak kevésbé kellene a saját fantáziájukra hagyatkozniuk. Felmerült ötletként, hogy a csapatok félév közben kicseréljék egymás között a dokumentált munkákat, de ez elég igazságtalannak tűnik azokkal a csapatokkal szemben, akik rendesen dolgoznak. További probléma lenne az is, hogy aki úgy véli, hogy rossz modellt kap, az ráerőlteti a saját ötleteit a már meglévő elemekre.

14.6. Milyen feladatot ajánlanának a projektre?

14.6.1. Erdőtűz-oltás

Valószínűség-alapú erdőtűz szimuláció kiegészítve az oltást végző helikopterekkel és repülőkkel. A játékos célja az irányított eszközökkel a tűz mielőbbi megfékezése. A játékosnak adott számú eszköz áll rendelkezésre, ezek újratöltése időbe telik. A játékos a berepülés irányát és hosszát adhatja meg. A példa ihletője:

- http://demonstrations.wolfram.com/AProbabilisticForestFireModel/

Többfajta növényzet-típus is bevezethető; a mesterséges erdőirtások is szerepet játszanak a tűz terjedésében.

14.6.2. Egyéb, felsorolás szinten

TIM

- http://en.wikipedia.org/wiki/The_Incredible_Machine

Worms

- http://en.wikipedia.org/wiki/Worms_%28series%29

Pillage the village:

- http://www.xgenstudios.com/play/pillage

Stick Soldiers:

- http://www.andrewrussellstudios.com/wsu/

XiaoXiao Game 9:

- http://www.stickpage.com/xiao9play.shtml

Teenage Mutant Ninja Puppets:

- http://www.fun-motion.com/physics-games/teenage-mutant-ninja-puppets/

Phun:

- http://phun.cs.umu.se/wiki

15. fejezet Napló

Előkészületek

2008. január 31.

Összeáll a végső csapat.

09:40 – 13:20

Katus ötletelést és szavazást ír ki a csapatnév eldöntésére. A szavazás eredményeképp a csapatnév Tetragon lesz.

Katus számba veszi az elérhető fejlesztőeszközöket, megbízza Paulikot a kommunikációs infrastruktúra kiépítésével.

13:20 – 14:20

Deák csapatneveket gyűjt; csapatregisztrációhoz szükséges feladatok összegyűjtése, amíg Katus távol lesz.

2008. február 5.

11:30 – 12:00

Deák csapatregisztrál Katust helyettesítve.

20:00 – 23:40

Katus elkészíti a csapat logó terveit.

2008. február 10.

10:00 – 18:00

Paulik üzembe helyezi a platina512.uw.hu címen az együttműködést segítő portált.

Első hét

2008. február 11.

16:00 – 16:45

Deák meetingel Ofellával: brainstorming a feladat specifikációról (multiplayer rész).

16:45 - 17:00

Deák meetingel Katussal: multiplayer rész elvetése.

Javaslat előzetes munkabeosztásra.

Csapatlogó véleményezése.

20:40 – 22:20

Katus kialakítja a dokumentum stílusokat. Felhasznált források:

- http://en.wikipedia.org/wiki/Samples_of_Serif_typefaces

- http://en.wikipedia.org/wiki/Samples_of_Sans_Serif_typefaces
- http://www.microsoft.com/typography/ClearTypeFonts.mspx
  Dokumentumstílusok véleményezése.

21:00 – 22:00

Paulik megoldja a fórummotor időzóna-beállításának gondjait.

2008. február 12.

15:02 – 17:45

Deák: Visual Studio 2008 telepítése véleményformálás céljából.
Deák: időpont egyeztetés Ofellával egy holnapi meetingre. Döntés: délután négy óra.
Deák: egyéni felkészülés a meetingre. Kidolgozandó vázlatpontok gyűjtése.

2008. 02. 13.

16:00 – 17:00

Deák és Ofella brainstormingolnak. A meeting eredménye: mi kerüljön az első beadandóba (Követelmény definíció, Projekt terv, a feladat leírása, szótár, use-case).

19:00 – 19:50

Deák követelmény specifikációt készít.

2008. február 14.

20:00 – 21:00

Ofella az első beadandó rá eső részén dolgozik (2.3.-on és 2.4.-en kívül majdnem minden).

2008. február 15.

08:00 – 09:37

Deák szótárat készít. Dokumentumok feltöltése.

20:30

Ofella feltölti a feladat ráeső részének első verzióját.

20:40 – 21:20

Deák és Katus az elkészült dokumentumokat futja át, egészíti ki.

21:20 – 23:00

Katus dokumentumot formáz és fogalmaz át; naplót összesít. Az első beadandó dokumentum első verziójának kiadása ellenőrzésre.

2008. február 17.

09:00 – 10:00

Deák dokumentumot fogalmaz át. Az első beadandó dokumentum második verziójának kiadása ellenőrzésre.

13:00 – 15:00

Paulik kidolgozza a kódírás egységes formátumát.

Második hét

2008. február 20.

A csapat leadja az első beadandót, majd megbeszélik, hogy az aktuális feladatot Deák és Paulik készíti el. Deák és Paulik megegyeznek, hogy a modell pénteken (2008. február 22.) kerül kidolgozásra.

2008. február 22.

15:00 – 21 :30

Deák és Paulik kidolgozzák a modell első verzióját, elkészítik a modellhez tartozó szekvencia diagramokat. Átnézik a modellt hiányosságokat keresve és megoldást keresnek a felmerült problémákra. *Döntés:* Az átvizsgálás eredményeképp a meglévő modellt nagyrészt elvetik és megegyeznek, hogy az új osztály-diagramot Paulik készíti el másnapra és annak alapján Deák elkészíti a szekvencia és állapot diagramokat.

23:30 – 02:30

Paulik elkészíti az új modell alapján az osztály-diagramot és eljuttatja Deáknak.

2008. február 22. ›

Katus határozatlan időre nem rendelkezik otthoni internet-eléréssel a kerületi kábellopás miatt. A munkában való részvételét ez jelentősen hátráltatja, mivel mindenkinek kevés ideje van és az adott dologra azonnal reagálnia kellene.

2008. február 23.

10:00 – 14:00

Deák a megkapott osztály-diagram alapján elkészíti a szekvencia és állapot diagramokat.

15:00 – 17:00

Deák és Paulik átvizsgálják a diagramokat. *Döntés:* Deák és Paulik kijavítja a talált hibákat pótolja a hiányosságokat.

2008. február 24.

Deák és Paulik a nap folyamán pár kisebb módosítást végez a diagramokon.

Harmadik hét

2008. február 25.

10:00 – 12:00

A konzultáció időpontjában a csapat bemutatja a modell elvét a csapat konzulensének, majd döntést hoznak arról, hogy meglévő működőképes modelljüket átalakítják a visitor tervezési mintának megfelelően.

16:00 – 18:00

Paulik elkészíti az osztály-diagramot a visitor mintának megfelelően, elkészíti a szekvencia diagramok egy részét, majd átküldi a fájlokat Deáknak, hogy elkészíthesse a hátralévő szekvencia diagramokat és állapot diagramokat.

19:00 – 21:00

Deák elkészíti a hátralévő diagramokat.

2008. február 26.

9:00 – 11:00

Deák elkészíti a dokumentáció első felének nyers változatát, majd átküldi azt Pauliknak. Paulik elkészíti a dokumentáció második felének nyers változatát.

17:00 – 21:00

Paulik a nyers változatok alapján elkészíti a dokumentáció végleges változatát.

Negyedik hét

2008. március 3.

10:52 – 10:59

Deák apróbb módosításokat eszközöl az előző beadandón.

2008. március 4.

19:26 – 20:33

Katus az előző heti beadandó végső változatát formázza át, hogy kultúráltabban nézzen ki. Lényeges tartalmi módosítás nem szükséges.

2008. március 3. ‹

Katus ismét rendelkezik otthoni internet-eléréssel.

2008. március 6.

20:55 - 21:08

Feladatok kiosztása. *Döntés:* Ofella a kommunikációs diagrammokat készíti el, Katus pedig a többit.

21:25 - 21:55

Ofella ismerkedik a NetBeans UML- és kommunikációs diagram-szerkesztőjével.

2008. március 8.

12:20 – 13:13

Katus készít egy egy oldalas leírást arról, hogy mi legyen benne az ötödik beadandóban. Katus a Deák és Paulik által készített UML diagramokkal ismerkedik.

17:00 – 17:15

Katus felhívja Deákot, felvilágosítást kér a modell egyes részeiről.

17:16 – 18:05

Katus a szkeleton use case-eit gondolja át és megrajzolja a diagramot a NetBeans UML szerkesztőjében. Egy könyvtár-átnevezés következtében a NetBeans elszáll és az elkészült diagram elveszik. A diagram rész mentve lett közben, de a NetBeans által magának fenntartott modell-leírás valószínűsíthetően nem, ugyanis a project újbóli megnyitása a következő felugró ablakkal indul: "Information - The diagram is being modified based on the changes in the model", majd a diagram üresen jelenik meg. A fájlok tehát megvannak, de nem lehet őket semmilyen módon felhasználni, az IDE és az UML plugin frissítése sem oldja meg a problémát.

18:41 – 19:15

Katus újra nekilát a use-case elkészítésének, ezúttal új koncepcióval.

19:21 – 20:09

Katus vitára invitálja Paulikot. A vita tárgya az eddigi koncepció és az elkészült use-case. A kommunikáció MSN-en folyik.

20:09 – 21:05

Katus elkészíti az 5.2-es fejezet előzetes változatát.

21:50 – 22:13

Katus befejezi a dokumentáció többi részét az 5.4.-es fejezet kivételével, hiszen az Ofella dolga. Katus átküldi az előzetes dokumentációt Ofellának.

2008. március 9.

19:52 – 20:02

Katus és Ofella az elkészült előzetes dokumentációt nézik át.

20:30 - 22:30

Ofella a NetBeans helyett a Visual Paradigm használatára kényszerül, aminek NetBeans-es pluginját sehogy sem tudja működésre bírni, ezért az önállóan futtatott programmal a nulláról készít el pár kommunikációs diagramot.

Ötödik hét

2008. március 10.

17:02 - 17:40

Ofella befejezi a kommunikációs diagramok elkészítését.

2008. március 11.

18:23 – 20:10

Katus módosítja a teljes dokumentációt a hétfői konzultáción kapott útmutatások alapján.

21:13 – 21:54

Katus kiadja ellenőrzésre az ötödik heti beadandót. Katus és Deák vitáznak az ötödik heti beadandóról MSN-en. Ofella is bekapcsolódik.

2008. március 15.

16:00 – 17:00

Katus az UML modellből generált kódot ellenőrzi, berakja a megfelelő importokat. Átgondolja a kiíró class-t.

17:00 – 18:00

Katus a kiíró class-t implementálja.

18:00 – 18:30

Katus a kiíró class-t teszteli.

18:30 – 19:00

Katus átalakítja úgy a kódot, hogy abc sorrendben a FureszBogyoVisszaharapas-ig minden a helyére kerüljön, egyéb apróbb módosításokat végez.

20:52 – 21:35

Katus elkészül a Tetragon_Team_Kukacfarm-ig minden osztállyal.

21:35 – 23:10

Katus fordításidejű és futásidejű hibákkal küszködik, a hibaforrásokat nehezen találja meg.

23:10 – 0:05

Katus elkészül az inicializációs rész kiíratásával, az egyes dialógusokat berakja a helyükre.

2008. március 16.

00:05 – 00:35

Katus a kiírás módján próbál javítani.

10:00 – 11:40

Katus a látogató mintát megvalósító osztályokat próbálja kiíratásra bírni, de a magyar elnevezésektől többször összezavarodik, és végül feladja.

18:00 – 20:30

Ofella a Katus által elkészített szkeletont kommentezi, az analízis modellnek megfelelően.

22:10 – 22:50

Katus a kommenteket egészíti ki, a saját kiegészítő osztályait is magyarázatokkal látja el.

Hatodik hét

2008. március 18.

08:00 – 10:00

Deák kisegíti Katust, Katus által megbeszélt útmutatások alapján kitölti a megfelelő függvényhívásokkal a látogató mintát megvalósító osztályok függvényeinek belsejét.

16:00 – 16:20

Deák némileg javít a délelőtt írt kódján.

19:45 – 20:45

Ofella az értékelést írja, majd elküldi a többieknek és Katussal konzultál. *Döntés:* az értékelés jó, a többiek is írjanak személyes értékelést.

21:10 – 21:20

Deák és Katus közös megbeszélésük közben rájön, hogy Deák a forráskód egy korábbi verzióját módosította.

21:20 – 22:00

Deák a módosításait másolja át a forráskód újabb verziójába, ügyelve a konzisztenciára.

22:00 – 22:16

Katus és Deák közösen megoldanak néhány fordítási problémát.

22:16 – 22:41

Katus befejezi az ütközések kiíratásának megvalósítását.

21:45 - 22:45

Paulik elkészíti ez értékelést.

22:46 – 22:57

Katus kiszedi azokat az elemeket a kódból, amelyek esetleg megakadályozhatják a kód fordítását korábbi Java verziókon (ilyenek pl.: template, annotation).

22:58 – 23:55

Katus nekilát a dokumentum összeállításának.

23:55 – 0:54
Katus próbálja konzolról lefordítani a forrásfájlokat.

01:26 – 02:01
Katus a naplót szerkeszti.

2008. március 21.

16:00 – 21:00
Paulik és Deák kidolgozza a proto tervét. A bemeneti formátumot Paulik készíti el, míg Deák a kimeneti formátumot dolgozza ki. A szükséges dokumentációkat közös döntés keretében írják meg. Elkészítik a teszt forgatókönyvek nagyobb témaköreit.

23:46 – 00:48
Katus átolvassa a Deák és Paulik által készített dokumentációt. A use-case-n kívül mindent rendben talál, azt viszont lecseréli. Katus átformázza a dokumentumot.

Hetedik hét

2008. március 25.

18:45 – 21:45
Deák és Paulik rájön a kidolgozott proto hibájára, és teljesen újraszerkeszti a már elkészült tervet, majd ezt dokumentálja. Paulik a bemeneti formátumot, míg Deák a kimeneti formátumot készíti el. Paulik a kész modellt a szükséges függvényekkel egészíti ki, meg Deák a use-case-eket rajzolja újra. Együttesen elkészítik a részletes teszt-forgatókönyveket és a tesztelőprogram specifikációját.

22:40 – 23:00
Katus átolvassa az újonnan elkészített dokumentációt, és rendben találja.

2008. március 29.

16:00 – 23:59
Deák és Paulik elkészíti a következő részfeladatokat:

- Objektumok és metódusok tervei. (State chartok és Activity diagramok)
- A tesztek részletes tervei, leírásuk a teszt nyelvén
- A tesztelést támogató programok tervei

2008. március 30.

10:30 – 12:30
Paulik kisebb javításokat végez az ismertetett algoritmusokban, kiküszöböl néhány potenciális hiba lehetőséget, olyan módosítások alkalmazásával, melyek nem indukálnak új teszt-forgatókönyveket, de biztosítják a helyes működést.

15:00 – 16:00

Deák a be és kimeneti teszteseteket ellenőrzi, és hibáit javítja.

19:22 – 19:42

Katus az első kikerült dokumentációt nézi át, formáz, és helyesírási hibákat javít.

Nyolcadik hét

2008. március 31.

21:00 – 23-59

Paulik kiegészíti az algoritmusokat; elkészíti a motor.run() activity diagramját, a szelvény state chartját. A dokumentum hibás struktúrája lehetetlenné tette annak formázását; így a nyers szövegből újraépíti a dokumentációt. Karbantartja a honlapon található dokumentumokat és felhasználóbarátabbá teszi a letöltési rendszert.

2008. április 1.

19:00 – 19:15

Katus átnézi a végső dokumentációt, és kiegészíti azt a naplóval.

Kilencedik hét

2008. április 11.

19:15 – 21:19

Katus és Ofella megbeszélést tartanak. Átnézik a protóhoz kapcsolodó dokumentumokat. A kódot két egyenlő részre osztják, elkezdenek kódolni.

21:30 – 24:00

Ofella a rá bízott részt kódolja.

2008. április 12.

11:00 – 12:00

15:00 – 18:00

20:00 – 22:00

Mivel Ofella elutazott, ezért Katus kódolja le a teljes belső működést.

2008. április 13.

09:40 – 12:30

Katus elkezdi megírni a teszteléshez szükséges kódrészleteket.

14:00 – 17:10

Katus befejezi a teszteléshez szükséges kódrészletek megírását. Katus kipróbálja az első néhány tesztesetet. A program nem száll el, a kimenet is helyes. Azonban ekkor már látszik, hogy az elvárt kimenetek több helyen helytelenül lettek megadva, esetenként hiányosak.

17:40 – 18:35

Katus átadja a kódot Ofellának, elmondja hogy mi történt eddig a kóddal, Ofella mire vigyázzon, és milyen munkamenetet kövessen.

20:18 – 20:30

Katus és Ofella megvitatják a dolgok állását.

20:30 – 01:00

Ofella konzisztenssé teszi a kódot, javítja a bugokat.

Tizedik hét

2008. április 14.

20:40 – 22:41

Ofella már a sztringkomparátort leszámítva mindennel kész van. Így Katussal próbálja fordítani és futtatni az Ural2-n a programot. Paulikkal és Deákkal konzultálnak bizonyos problémákról a parancsnyelvet illetően.

2008. április 15.

09:30 – 11:30

Ofella megírja a kimenet-ellenőrző osztályt és a teszteléshez szükséges batch-fájlokat.

20:30 – 24:00

Ofella apróbb módosításokat végez a kimenet-ellenőrző osztályon, és megírja a dokumentációt. Közben folyamatosan konzultál Katussal.

00:10 – 01:00

Katus a dokumentációt állítja össze.

2008. április 18.

08:00 – 12:00

Katus, Paulik és Deák kidolgozza a grafikus modellt a kukac játékhoz. Egy kérdés marad megoldatlanul.

2008. április 20.

12:00 – 13:00

Paulik megoldást talál az egyetlen fennmaradó problémára.

15:00 – 15:30

Deák megvizsgálja Paulik megoldását, azzal egyet ért, és egy interface-s kiegészítést javasol hozzá.

Tizenegyedik hét

2008. április 21.

11:30 – 11:45

A csapat prezentálja a grafikus rész koncepcióját a megfelelő utólagos kiegészítésekkel a konzulensnek, ő pedig azzal egyetért.

16:00 – 18:00

Katus a NetBeans UML szerkesztőjével próbál sikereket elérni. Végül nagy nehezen módosítja az osztálydiagramot és a szekvencia diagramokat. Katus megrajzolja a grafikus felületet.

2008. április 22.

19:00 – 23:40

Ofella az észlelt hibák és hiányosságok kijavításának kezd neki. Az öröklési hierarchia megfelelő részébe beiktat egy RenderAbstractClass absztrakt osztályt a RenderInterface interfész alá. Ezen kívül a szekvencia diagrammokat is módosítja. Megpróbál a NetBeans-szel report-ot generálni, de az többszöri próbálkozás után nem sikerül neki, sőt az több diagramot teljesen elront, elemek törlődnek [!]. Végül egy egyszerűsített objektum felsorolást és rövid leírást ír meg.

2008. április 27.

12:00 – 24:00

Ofella a grafikus változatot készíti.

Tizenharmadik hét

2008. május 5.

10:00 – 11:00

Deák és Paulik tesztelik az addig elkészült grafikus változatot, és a szükséges módosításokat közlik Ofellával.

13:00 – 15:00

Ofella elvégzi a változtatásokat, és az eredményt feltölti a tárhelyre.

2008. május 6.

22:00 – 00:30

Katus a dokumentációt és a feltöltendőt állítja össze.

Tizennegyedik hét

2008. május 13.

22:50 – 24:00

Katus és Ofella megírják az utolsó beadandót. Katus az összesített dokumentációt állítja össze.


team
Tetragon4