# Szoftver Labor 4
# Projekt dokumentáció

36 – Bigimot

Konzulens:
Bíró Barna


Csapattagok

| | | |
|---|---|---|
| Boros Dávid | xs5qzg | hopaci@yahoo.com |
| Molnár László | tcb38u | laszlomoln@gmail.com |
| Takács Rajmund | q9u7zy | takraj@gmail.com |
| Rapp Gábor | qecowt | gabor894@gmail.com |


2010.05.12

# 2. Követelmény, projekt, funkcionalitás

## *2.1 Követelmény definíció*

### 2.1.1 A program célja, alapvető feladatai

Az elkészítendő alkalmazás egy játékprogram, amely egy városban játszódik. A rabló autóját irányítva kell eljutnunk a bank épületétől a búvóhelyig. Útközben el kell kerülni a közlekedőket és rendőröket.
A fejlesztőcsapat célja a követelményeket kielégítő, élvezhető játékélményt nyújtó alkalmazás létrehozása.

### 2.1.2 A program futtatásához szükséges környezet

Szükséges 1.6.0_14 –es verziójú Java Runtime Environment telepítése. Ennek minimális rendszerkövetelménye: 166 MHz Pentium processzor, 32 MB RAM memória. A játék további 1 MB körüli helyet igényel a háttértárolón.

### 2.1.3 A felhasználói felület

A programot a felhasználó grafikus felhasználói felületen keresztül az egér és a billentyűzet segítségével vezérelheti.

### 2.1.4 A program fejlesztésével kapcsolatos minőségi követelmények

**Teljesítmény:** A cél az, hogy a játék élvezhetően játszható legyen a minimális rendszer követelmények mellett. A programban törekedünk a felesleges overheadek és memóriafoglalások elkerülésére. A grafikus felületnél törekedni fogunk a folyamatos animációk alkalmazására (DoubleBuffer), és ügyelni fogunk a folyamatos játékmenet biztosítására.

**Újrafelhasználhatóság:** A grafikus felhasználói felületet a program többi részétől teljesen különválasztjuk, így az a későbbiekben egyszerűen és gyorsan cserélhető lesz.

**Rugalmasság:** A Java környezet biztosítja a program rugalmasságát. Minden számítógépen és platformon amely megvalósítja a követelményekben leírtakat a játék futtatható lesz.

**Felhasználhatóság:** A használat különösebb tanítást nem igényel, akár a felhasználói kézikönyv elolvasása nélkül is könnyen játszható.

### 2.1.5 A kibocsátás

A dokumentációkat és leírásokat hétről-hétre továbbítjuk a konzulensnek. A program tesztelését és végső futtatását a HSZK gépein végezzük.

## *2.2 Projekt terv*

### 2.2.1 A fejlesztőcsapat tagjai, feladatkörei

| **A csapattag neve** | **Feladatköre** |
|---|---|
| Boros Dávid | csapatvezető, kódolás, dokumentáció |
| Molnár László | kódolás, dokumentáció |
| Takács Rajmund | dokumentáció, tesztelés |
| Rapp Gábor | dokumentáció, tesztelés |

A beosztás nem végleges, a projekt folyamán változhat.

### 2.2.2 A fejlesztői környezet

Az UML diagramok elkészítéséhez a StarUml programot használjuk. A program könnyen kezelhető és átlátható felületű. Teljes mértékben megfelel a projekt során felmerülő feladatok megvalósítására.
A program fejlesztéséhez Eclipse fejlesztői környezetet használunk főként testreszabhatósága miatt. A környezet használatával kapcsolatos korábbi tapasztalatok is előnyt jelentenek.

### 2.2.3 Fejlesztési ütemterv

A fejlesztési ütemtervben négy fő lépcsőfokot határoztunk meg:

**Modell:** A belső működés leírása (osztályok, hívások …) UML diagramok segítségével.

**Szkeleton:** Célja annak bizonyítása, hogy az objektum és dinamikus modellek a definiált feladat egy modelljét alkotják. Minden definiált objektum szerepel, de csak az interfészükkel.

**Prototípus:** Célja annak demonstrálása, hogy a program elkészült, helyesen működik, valamennyi feladatát teljesíti.

**Grafikus felület:** A program végső állapota, a grafikus felület teljesen elkülönül a belső megvalósítástól.

### 2.2.4 Az ütemterv részletei, a szükséges dokumentációk

- Csapatok regisztrációja
- Követelmény, projekt, funkcionalitás
- Analízis modell kidolgozása 1.
- Analízis modell kidolgozása 2.
- Szkeleton tervezése
- Szkeleton beadása
- Prototípus koncepciója
- Részletes tervek
- Prototípus készítése, tesztelése
- Prototípus beadása
- Grafikus felület specifikálása
- Grafikus változat készítése
- Grafikus változat beadása
- Összefoglalás

### 2.2.5 Határidők

| | |
|---|---|
| Február 10. | A csapat regisztrációja |
| Február 18. | Követelmény, projekt, funkcionalitás |
| Február 25. | Analízis modell kidolgozása 1. |
| Március 4. | Analízis modell kidolgozása 2. |
| Március 11. | Szkeleton tervezése - beadás |
| Március 18. | Szkeleton - beadás |
| Március 25. | Prototípus koncepciója - beadás |
| Április 1. | Részletes tervek – beadás |
| Április 8. | |
| Április 15. | Prototípus - beadás |

| | |
|---|---|
| Április 22. | |
| Április 29. | Grafikus felület specifikációja - beadás |
| Május 6. | Grafikus változat - beadás |
| Május 13. | Összefoglalás - beadás |

### 2.2.6 Kommunikációs modell

A csapat három tagja (Molnár László, Rapp Gábor, Boros Dávid) már korábban is ismerte egymást, meggyőződtek egymás megbízhatóságáról, ismerik a feladatokhoz való hozzáállásukat. Jól együttműködtek korábban is kisebb volumenű projektek, programok tervezésekor, írásakor. Takács Rajmund mind új, ismeretlen negyedik tagként csatlakozott a csapathoz. Hamar bizonyította elszántságát, hozzáértését és megbízhatóságát.

A projekt során kulcsfontosságú szerep jut a kommunikációnak. A fontosabb megbeszélések szinkron kommunikációs formában zajlanak. Ezekre a személyes találkozókat, telefonbeszélgetéseket és a Windows Live Messenger programot használjuk.
A projekt tényleges fejlesztése és a dokumentációk készítése a Project Hosting on Google Code és a TortoiseSVN rendszer segítségével történik. Így könnyen frissíthetők, elérhetők és nyomon követhetők a dokumentációk és azok változtatásai. Valamint biztosított egy Wiki rendszer a Konzultációk, Tervek és a Napló könnyű kezeléséhez. Mindezek mellett biztosított a projekt levlistája amely szintén megkönnyíti a fejlesztés előrehaladását.

## *2.3 Feladatleírás*

Egy kis német város, Verkehrsmeldungen am Uml útjain járművek közlekednek. Minden jármű automatikusan az úton tartja magát, nem ütközik más járművekkel és a közlekedési szabályoknak engedelmeskedik (pl. piros lámpánál, stop táblánál egy időre megáll).
Elágazásoknál a járművek véletlenszerűen választanak a lehetőségek között. A járművek folyamatosan lépnek ki és be a városhatárnál. A járművek különböző sebességgel közlekedhetnek, de nem előzik meg egymást valamint az ütközéselkerülésből adódóan hátulról nem ütköznek, pusztán lelassulnak az előttük lévő jármű sebességére.
A városban utak vannak, az út jellemzője a foglaltság. A járműre jellemző a sebessége, a sérültsége, a típusa (semleges, rendőr, rabló) és az iránya. Az utak néhol elágazhatnak, az elágazásra jellemző a belőle kivezető szabad utak száma. Az elágazás feladata, hogy a rajta lévő járművet egy olyan kivezető útra továbbítsa, ami nem foglalt. Az egyirányú út egy speciális útfajta, melybe szabályszerűen csakis az egyik végén lehet behajtani, és a másik végén kihajtani. Az elágazáshoz csatlakozó utakat STOP tábla vagy jelzőlámpa

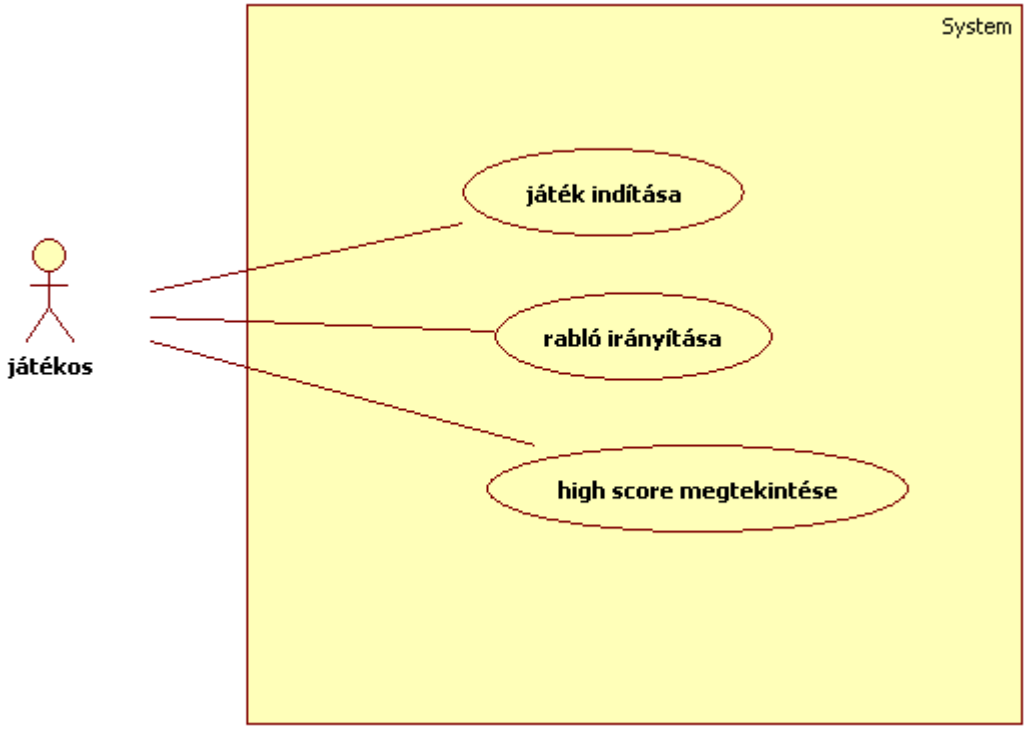
irányíthatja. Ha egy elágazás bejáratánál STOP tábla van, akkor a jármű egészen addig nem hajthat be, amíg az elágazásra olyan is be akar hajtani, akit nem akadályoz tábla. Ha az elágazásnál jelzőlámpa van, akkor amíg az nem zöld, addig az elágazásra nem lehet behajtani a megfelelő útról. A jelzőlámpa jellemzője az állapota. A járművek ismerik a közvetlenül előttük haladó másik járművet. Ha egy jármű elérte a városhatárt, akkor eltűnik, kivéve, ha az rabló. Ebben a városban minden út, ami nem elágazás, az egyirányú, illetve minden elágazásnál van valamilyen forgalomirányítás. A város utcái kétsávosak, a sávok között az üldöző rendőr és a rabló tud váltani, a többiek nem.
A bankrablók speciálisan tuningolt autója kivétel a szabály alól, ők szabadon közlekedhetnek a városban, minden szabályt áthágva. Miközben megpróbálnak eljutni a banktól a rejtekhelyig, ügyelni kell, hogy ne ütközzenek más autóval és az üldöző rendőr se érje el őket. A városban folyamatosan vannak rendőrök, akik, amíg rablót nem észlelnek, ugyanúgy viselkednek, mint a többi jármű. A rendőr jellemzője az állapota (normál vagy üldöző), a rabló jellemzője a zsákmány.
Egy rendőr akkor észlel rablót, ha áthalad azon az utcán, melyen ő is halad. Ha egy rendőr észleli a rablót, akkor addig üldözi, amíg el nem kapja, vagy a rabló el nem érte a rejtekhelyet. A rabló a banktól indul, egy véletlenszerű zsákmánnyal, és a játékos által meghatározott sebességgel. A játékos a rablót irányítja, és az a feladata, hogy úgy jusson el a rejtekhelyig, hogy addig ne kapja el rendőr. Ha egy rendőr elkapta a rablót, vagy a rabló kijutott a városból akkor a játéknak vége, pont úgy, mintha elkapta volna egy rendőr. Az üldöző rendőr megpróbál ütközni a rablóval, akinek minden egyes ütközéskor nő a sérültsége és a sebessége, a sebességkülönbséggel arányos mértékben. Ha a rabló sérültsége elérte a 100%-ot, akkor a rendőr elkapta. A rendőr csak akkor nem lassít le az előtte lévő jármű sebességére, ha az előtte lévő jármű a rabló. A rablóra nem vonatkoznak az elágazásbeli szabályok, és a forgalommal szemben, illetve tolatva is tud haladni.
Ha a rabló ütközik egy civil járművel, akkor a civil jármű felrobban, a rabló sebessége csökken az ütközési sebességgel arányos mértékben, és hasonlóképp nő a sérültsége is. A rabló eldöntheti, hogy merre szeretne egy elágazásban továbbmenni. Ha a rabló korlátnak megy neki, akkor a sebességével arányos sérülést szerez. A rendőr viszont mindig arra tér a szabad kijáratok közül amerre a rablót leghamarabb eléri. Ha a rabló elér a rejtekhelyéhez, akkor a zsákmány értéke hozzáadódik a játékos pontszámához, és új forduló kezdődik.
A város véletlenszerűen keletkezik, ha a rabló sikeresen célba ért, akkor az új forduló egy újabb véletlen városban kezdődik, de ez nem feltétlen jelent nehezebb útvonalakat, ugyanakkor az új városban némileg több jármű fog közlekedni. Ha a rablót elkapták, de több zsákmányt juttatott célba, mint a 10 legtöbbet zsákmányoló rabló, akkor a neve felkerül egy listára, viszont a lista utolsó helyén álló rablót leveszik róla. A játékos menüből is megnézheti ezt a listát, illetve ugyaninnen indíthat új játékot is. A játéknak bármikor véget lehet vetni, ilyenkor automatikusan a menü jelenik meg.

## 2.4 Szótár

| | |
|---|---|
| Város | A játékterület |
| Út | Csak ezen tudnak közlekedni a járművek |
| Jármű | A rabló, a rendőrök, és a civilek összefoglaló neve<br>Rabló: A játékos avatárja |
| Rendőr | A játékost üldözi, ő az ellenfél |
| Civilek, autó | A városban közlekedő autók |
| Ki és belépés a városhatárnál | Civilek, és rendőrök elhagyhatják a játékterületet, ekkor megsemmisülnek, illetve jöhetnek (keletkezhetnek) újak |
| Elágazás | Speciális úttípus, több út találkozása |
| Forgalomirányítás | Minden elágazásnál található, a civilek haladását szabályozza |
| Jelzőlámpa | Forgalomirányítás egy fajtája azt mutatja szabad e az út |
| Stop tábla | Forgalomirányítás egy fajtája |
| Ütközés | Két jármű érintkezése |
| Sérültség | A rabló jellemzője, ha eléri a 100%ot a játékos veszít |
| Tolatás | A kocsi hátuljának az irányába történő haladás |
| Felrobbanás | A megsemmisülés egy fajtája |
| Új forduló | A játék új térképpel újraindul, de a játékos pontszáma megmarad, a civilek száma nő |
| Lista | A legjobb teljesítményt elérő 10 játékos listája |
| Menü | A játék indítása, a lista megtekintése, és a kilépés közül választhat a játékos |

## 2.5 Essential use-case-ek

### 2.5.1 Use-case diagram




### 2.5.2 Use-case leírások

| | |
|---|---|
| **Use-case neve** | Játék indítása |
| **Rövid leírás** | A játékos új játékot kezd, véletlen pálya generálódik |
| **Aktorok** | Játékos |
| **Forgatókönyv** | A játékos futtatja a programot, majd a menüpontok közül kiválasztja a startot. |

| | |
|---|---|
| **Use-case neve** | Rabló irányítása |
| **Rövid leírás** | A játék alatt a rablót a játékos vezérli |
| **Aktorok** | Játékos |
| **Forgatókönyv** | A játékos a kurzor billentyűk lenyomásával irányítja a rablót a célállomás felé, amíg oda nem ér,v agy veszít. |

| | |
|---|---|
| **Use-case neve** | High score megtekintése |
| **Rövid leírás** | A 10 legjobb eredmény megtekintése, illetve ha egy eredményt megdöntött a játékos a nevének megadása |
| **Aktorok** | Játékos |
| **Forgatókönyv** | A játék végeztével ha a játékos rekordot döntött megadja a nevét, ezután (akkor is ha nem döntött rekordot) a képernyőn kilistázásra kerül a tíz legjobb eredmény. A menüből is megtekinthető a megfelelő menüpont kiválasztásával. |

## 2.6 Napló

| Kezdet | Időtartam | Résztvevők | Leírás |
|---|---|---|---|
| 2010.02.01. 21:00 | 0,5 óra | Rapp<br>Takács<br>Boros<br>Molnár | Kapcsolatfelvétel<br>Csapatnév kitalálása<br>Csapat regisztrálása<br>Csapatvezető választása<br>Megegyezés a használt eszközökben |
| 2010.02.15. 18:50 | 0,5 óra | Takács<br>Molnár<br>Boros | Döntés:<br>Takács elkészíti a feladatleírást<br>Boros elkészíti a Követelmény definíciót és a Projekt tervet<br>Molnár elkészíti a Szótárt és az Essential Use-case-eket |
| 2010.02.15. 20:50 | 3 óra | Takács | Elkészíti a Feladatleírást |
| 2010.02.17. 20:00 | 1 óra | Takács | Javítja, kiegészíti a Feladatleírást |
| 2010.02.17. 20:30 | 3 óra | Molnár | Elkészíti a Szótárt és az Essential Use-case-eket |
| 2010.02.17. 20:50 | 3,5 óra | Boros | Elkészíti a Követelmény definíciót és a Projekt tervet |
| 2010.02.18. 01:00 | 1 óra | Rapp | Kiegészíti, javítja és formázza a dokumentumot |

# 3. Analízis modell kidolgozása

## *3.1 Objektum katalógus*

***Bank***
***Building***
***Car***
***City***
***CityElement***
***Hideout***
***Intersection***
***Police***
***RoadBlock***
***Robber***

### *3.1.1 Bank*

*Innen indul a bankrabló, a megszerzett zsákmánnyal.*

### *3.1.2 Building*

*A városban található egyéb épületek.*

### *3.1.3 Car*

*A városban közlekedő autók. Az ő felelőssége nyilvántartani az életét, sebességét, és eldönteni, merre szeretne továbbmenni.*

### *3.1.4 City*

*Város objektum. Ő tartalmazza a várost alkotó „dolgokat" (járművek, épületek, utak, csomópontok). Ő felelőssége a pálya generálása, és a „léptetés", amivel a világot értesíti az idő múlásáról.*

### *3.1.5 CityElement*

*Városban lévő „dolgok" gyűjtőhelye. Ebből fűzünk listát. Azért kell, hogy egy helyen tudjuk tárolni az objektumainkat, és egyszerre tudjuk őket értesíteni az idő múlásáról.*

### *3.1.7 Hideout*

*Bankrabló rejtekhelye. Ha a bankrabló beér, megnyeri a kört.*

### *3.1.9 Intersection*

*Kereszteződés. Összekapcsolja az utakat, épületeket. Megmondja az autónak, merre felé lehet belőle menni.*

### *3.1.10 Police*

*Rendőr. Ő üldözi a rablót, ha utoléri, elkapja.*

### *3.1.11 RoadBlock*

*A rács, amiből az út felépül.*

### *3.1.12 Robber*

*A rabló. Őt lehet irányítani a városban egyedül.*

## *3.2 Osztályok leírása*

### 3.2.1 Bank

A bank épülete, a kiindulópontot jelenti.

- **Felelősség**

- **Ősosztályok**
  Building

- **Interfészek**
  nincs
- **Attribútumok**
  nincs
- **Metódusok**
  nincs

### 3.2.2 Building

Ősosztály az épületek számára.

- **Felelősség**

- **Ősosztályok**
  nincs
- **Interfészek**
  nincs
- **Attribútumok**
  nincs
- **Metódusok**
  nincs

### 3.2.3 Car

A városban haladó autók.

- **Felelősség**
  A haladás mértékének meghatározása a sebessége alapján. A haladási irányának eldöntése.
- **Ősosztályok**
  Vehicle
- **Interfészek**
  nincs
- **Attribútumok**
  - **int speed** : a jármű sebességét jelenti, mennyi útegységet tesz meg időegység alatt
- **Metódusok**
  - **int getSpeed()** : visszaadja a speed attribútum értékét
  - **void setSpeed(s : int)** : beállítja a speed attribútum értékét

### 3.2.4 City

A várost reprezentáló objektum.

- **Felelősség**
  Nyilvántartani a várost felépítő utakat, épületeket és a városban haladó autókat. A forgalmat szabályozni.
- **Ősosztályok**
  nincs
- **Interfészek**
  nincs
- **Attribútumok**
  - **LinkedList<CityElement> cityElements** : a várost alkotó utak, épületek tárolása
  - **LinkedList<Vehicle> vehicles :** a városban haladó autók tárolása
- **Metódusok**
  - **void Step()** a városban mozgó objektumok mozgatásért felel

### 3.2.5 CityElement

A várost felépítő objektumok ősosztálya.

- **Felelősség**
  Informálja az autókat a közlekedési viszonyokról, tárolja az autók pozícióját. Előzéseknél meghívj az autók megfelelő függvényeit.

- **Ősosztályok**
  nincs
- **Interfészek**
  nincs
- **Attribútumok**
  - **Vehicle& vehicle** : a rajta lévő autóra mutató referencia
- **Metódusok**
  - **void Step()** : a mozgatásért felelős függvény (lámpák változása)

### 3.2.6 Hideout

A cél pozíció, a rabló menedékhelye.

- **Felelősség**
  nincs
- **Ősosztályok**
  Building
- **Interfészek**
  nincs
- **Attribútumok**
  nincs
- **Metódusok**
  nincs

### 3.2.7 Intersection

A kereszteződést reprezentáló osztály.

- **Felelősség**
  Megadni a lehetséges kimeneti utak irányát az autónak.
- **Ősosztályok**
  RoadBlock
- **Interfészek**
  nincs
- **Attribútumok**
  - **CityElement& up** : a felfele irányban található útra mutató referencia
  - **CityElement& down** : a lefele irányban található útra mutató referencia

- **Metódusok**
  - **void get()**

### 3.2.8 Police

A városban haladó rendőrautók.

- **Felelősség**
  Ha rabló próbál mellette elhaladni elkapja.
- **Ősosztályok**
  Car
- **Interfészek**
  nincs
- **Attribútumok**
  nincs
- **Metódusok**
  - **void isPassed(Robber r)** : Ha a mellette elhaladni akaró autó egy rabló akkor elkapja.

### 3.2.9 RoadBlock

Az útelemeket reprezentáló osztály.

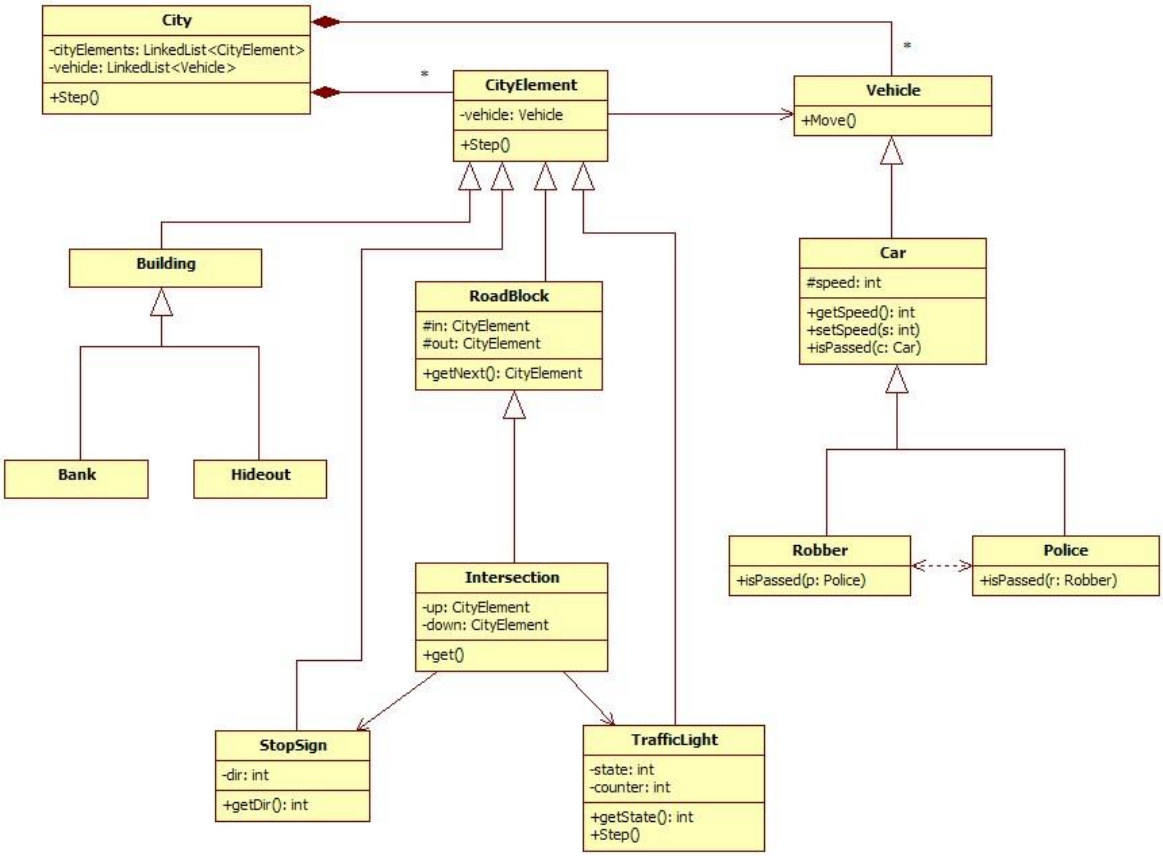
- **Felelősség**
  Figyelni a rajta haladó autókat.
- **Ősosztályok**
  CityElement
- **Interfészek**
- **Attribútumok**
  - **CityElement& in** : az előző útelemre mutató referencia
  - **CityElement& out** : az előző útelemre mutató referencia
- **Metódusok**
  - **CityElement getNext()** : megadja a következő útelemet

### 3.2.10 Robber

A rabló osztály a játékos irányítja.

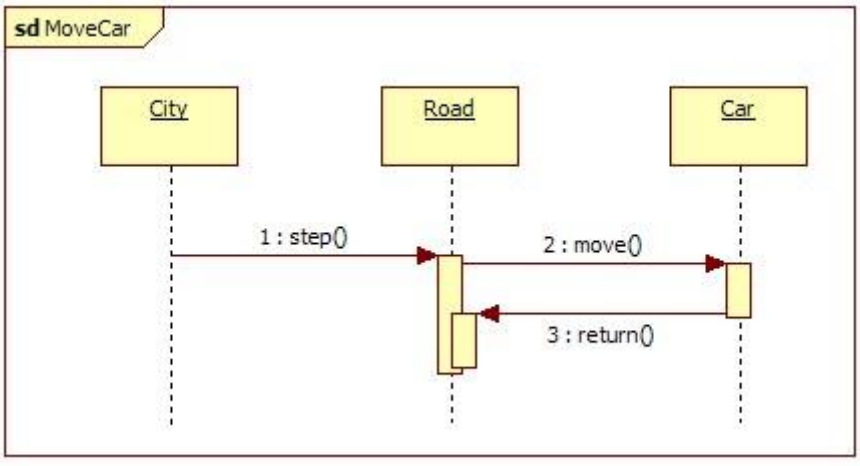
- **Felelősség**
- **Ősosztályok**
  Car
- **Interfészek**
  nincs
- **Attribútumok**
  nincs
- **Metódusok**
  - **isPassed(Police p)** : Ha a mellette elhaladó autó egy rendőr akkor az elkapja

## 3.3 Statikus struktúra diagramok

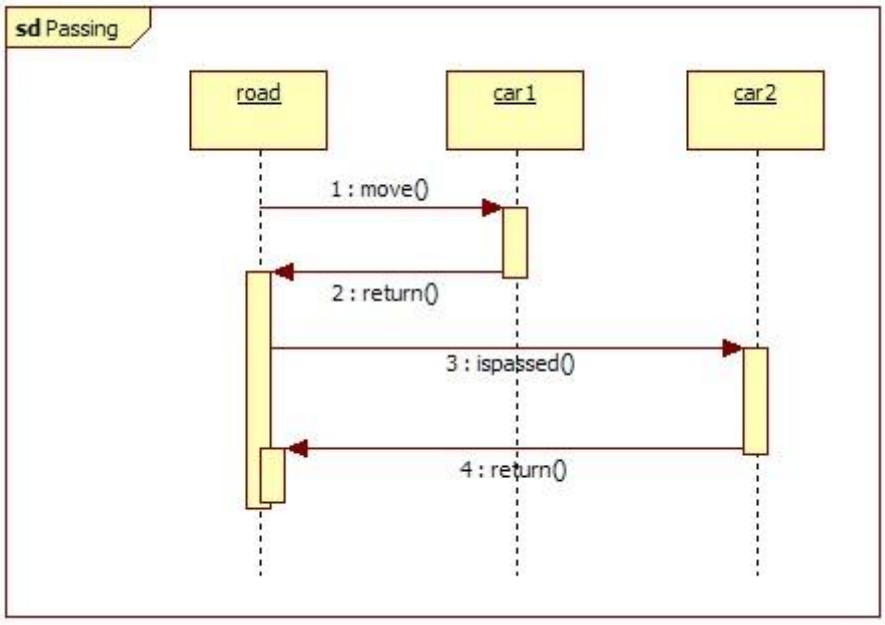



**1. ábra: A játék osztálydiagramja**

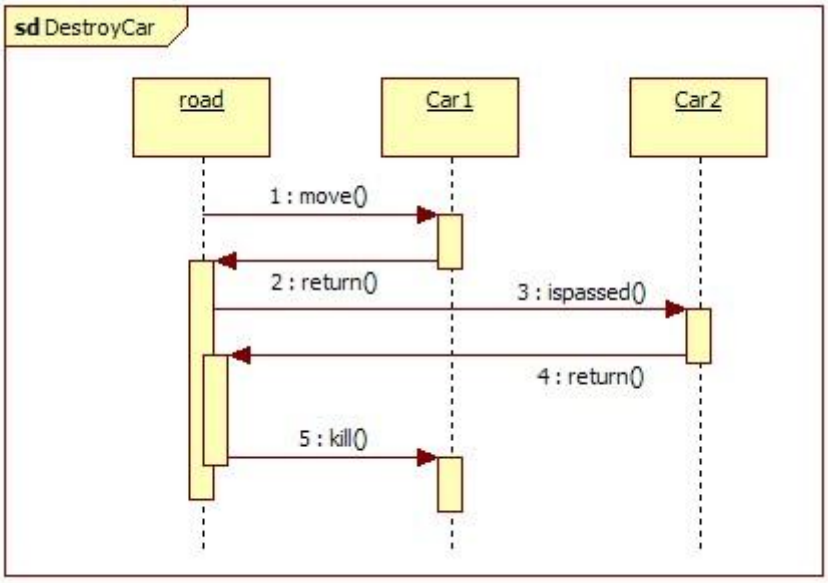
## 3.4 Szekvencia diagramok



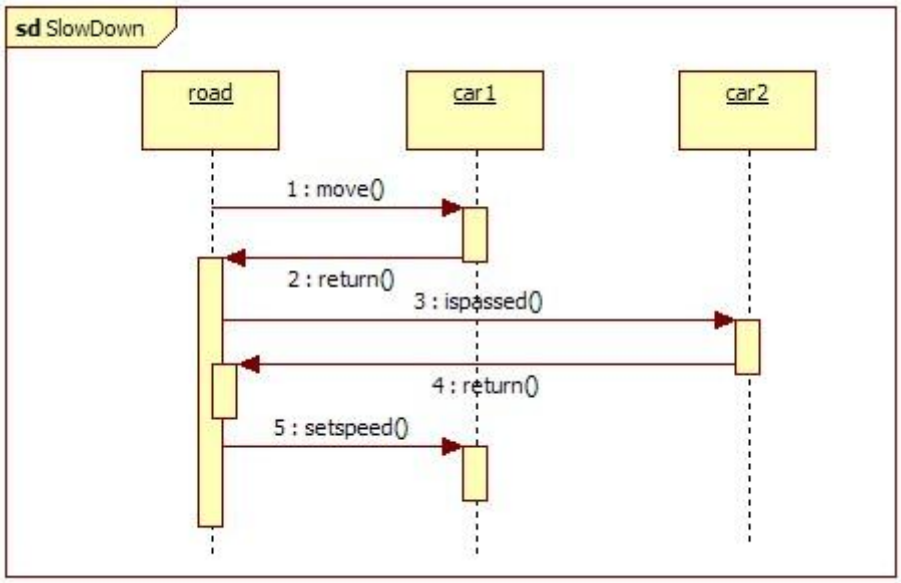

**2. ábra: Szekvencia diagram autó mozgatására**



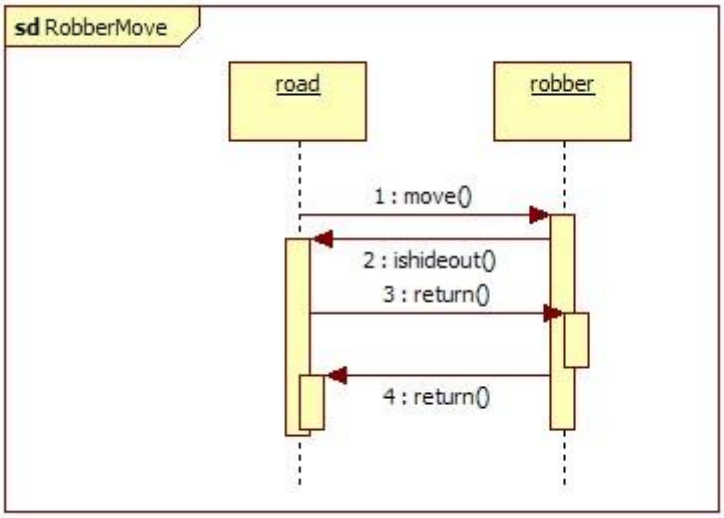

**3. ábra: Szekvencia diagram előzésre**



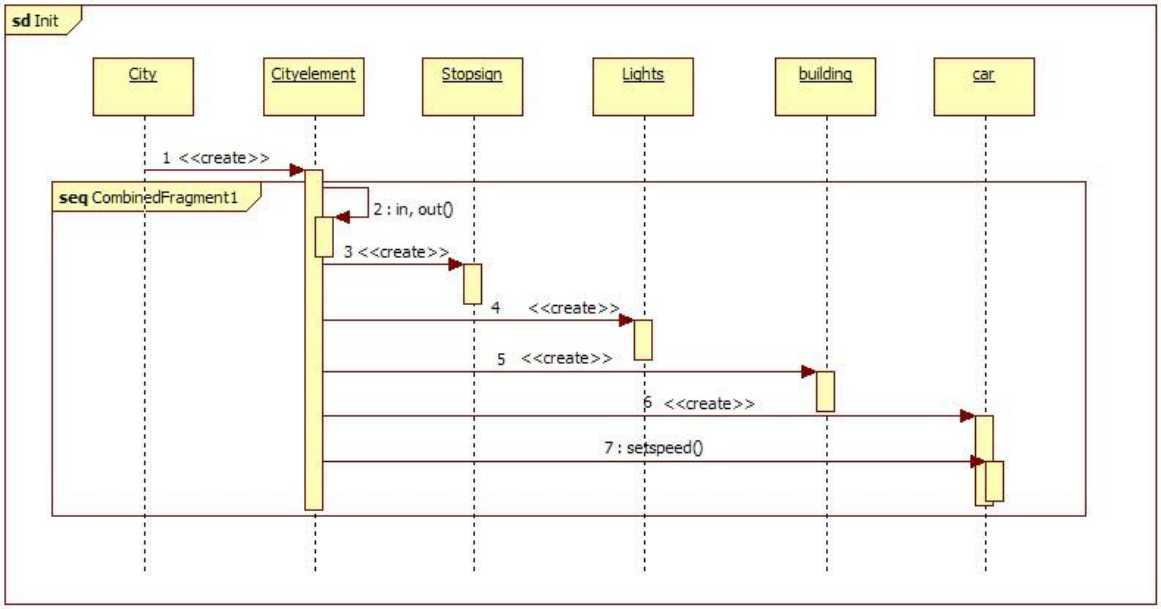

**4. ábra: Szekvencia diagram autó megsemmisítésre (pl. egy rendőr elkapja a rablót)**




**5. ábra: Szekvencia diagram lassításra**

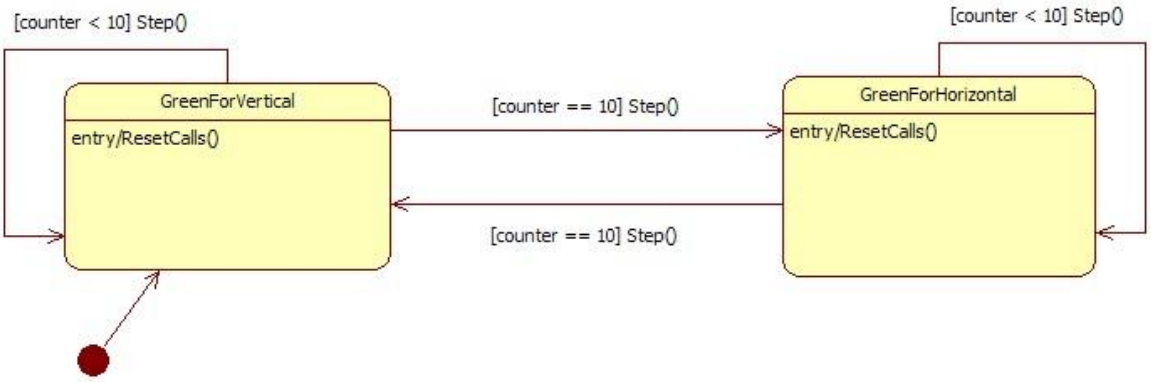



**6. ábra: Szekvencia diagram rabló mozgatására**

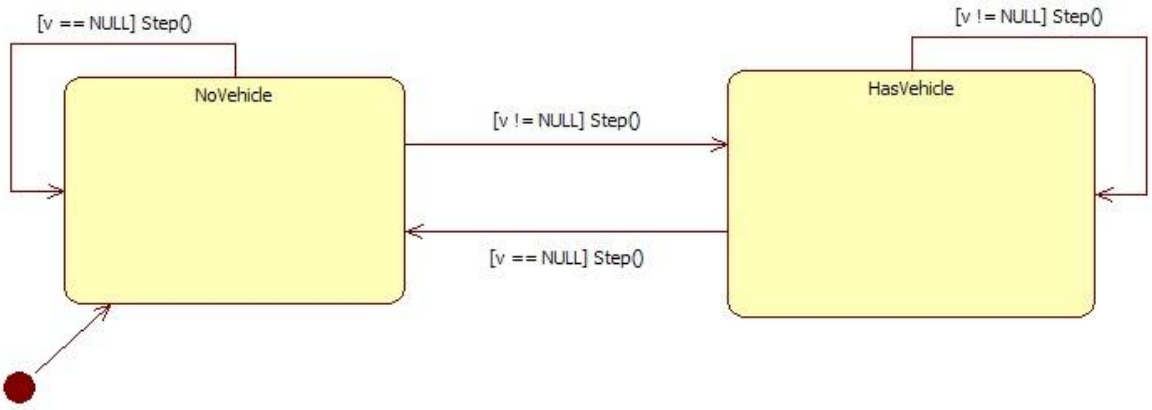



**7. ábra: Szekvencia diagram inicializálásra**

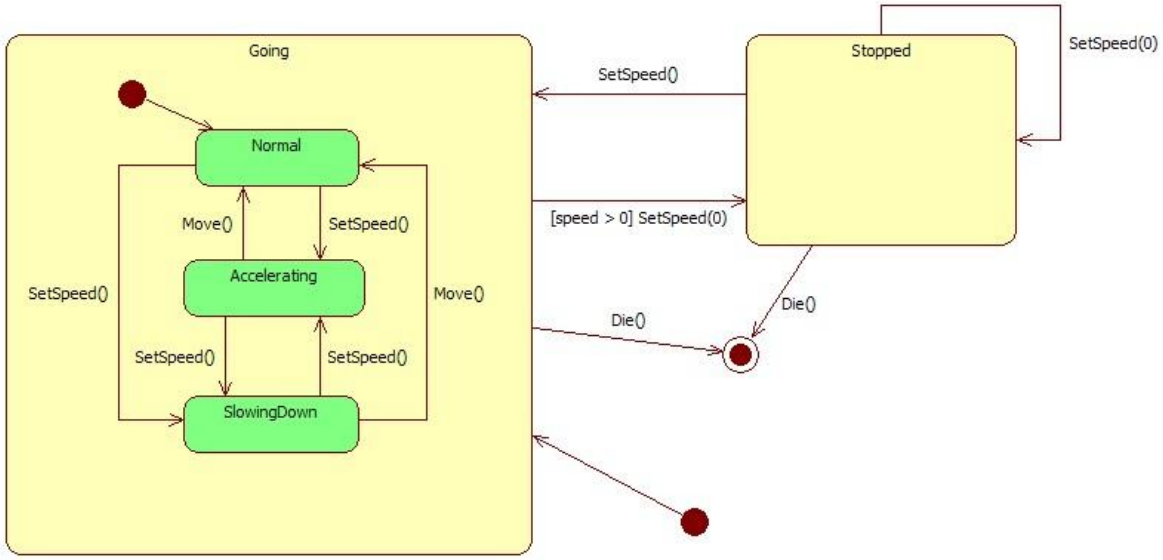
## *3.5 State-chartok*




**8. ábra: A jelzőlámpa állapotdiagramja**




**9. ábra: Az útelem állapotdiagramja**




**10. ábra: Az autó állapotdiagramja**

## *3.6 Napló*

| Kezdet | Időtartam | Résztvevők | Leírás |
|---|---|---|---|
| 2010.02.22. 21:35 | 1 óra | Rapp<br>Takács<br>Boros<br>Molnár | Értekezlet.<br>Döntés: Takács elkészíti a Statikus struktúra diagramokat és a State-chartokat, Boros elkészíti az Osztályok leírását, Rapp elkészíti az Objektum katalógust, Molnár elkészíti a Szekvencia diagramokat. |
| 2010.02.24. 16:00 | 2 óra | Boros | Tevékenység: Egy kezdetleges osztálydiagramot készít. |
| 2010.0224. 20:15 | 2 óra | Rapp | Tevékenység: Objektum katalógust készít. |
| 2010.02.24. 20:00 | 3 óra | Takács | Tevékenység: State chartok, és a végleges osztálydiagram elkészítése. |
| 2010.02.24. 21:00 | 3 óra | Boros | Tevékenység: Elkészíti az osztályok leírását. |
| 2010.02.24. 21:00 | 3 óra | Molnár | Tevékenység: Elkészíti a szekvencia diagramokat. |
| 2010.02.25. 10:00 | 1 óra | Takács | Tevékenység: Formázza, véglegesíti a dokumentumt. |

# 4. Analízis modell kidolgozása 2

## *4.1 Objektum katalógus*

### 4.1.1 Bank

A játék startpozíciója, innen indul a rabló.

### 4.1.2 Building

A városban található épületeket megvalósító objektum.

### 4.1.3 Car

A városban közlekedő autókat megvalósító objektum.

### 4.1.4 City

A város, magában foglalja az őt felépítő elemeket és felelős a mozgatásukért.

### 4.1.5 Hideout

A menedék épülete, a cél pozíciót testesíti meg.

### 4.1.6 ITraffic

A közlekedési szabályok szolgáltatásaiért felelős interface.

### 4.1.7 Police

A rendőrt megtestesítő objektum. Üldözik a rablót.

### 4.1.8 RoadBlock

Az utak építőeleme, nyilvántartja a rajta közlekedő autókat és közlekedési táblákat, lámpákat.

### 4.1.9 Robber

A rabló, őt irányíthatjuk a játék során.

### 4.1.10 TrafficSign

A jelzőlámpa, szabályozza a forgalmat a városban.

### 4.1.11 TrafficTable

A jelzőtáblák, szabályozza a forgalmat a városban.

## *4.2 Osztályok leírása*

### 4.2.1 Bank

- **Felelősség**

  Jelezni a játékosnak a játék végét, ha elérte.

- **Ősosztályok**

  Building

- **Interfészek**

  nincs

- **Attribútumok**

  - **int Role :** megadja az épület szerepét a játékban

- **Metódusok**

  - **int getRole() :** visszaadja az osztály szerepét jelentő belső változó értékét

### 4.2.2 Building

- **Felelősség**

  A városban lévő épületek, felelősségük a szerepkörük nyilvántartása.

- **Ősosztályok**

  nincs

- **Interfészek**

  nincs

- **Attribútumok**
  - **int role:** szerepkör jelölése

- **Metódusok**
  - **int getRole()**: szerepkör lekérdezése

### 4.2.3 Car

- **Felelősség**

  A városban közlekedő autók, felelősségük a szabályok betartása és az ütközések elkerülése, sebességük nyilvántartása.

- **Ősosztályok**

  nincs

- **Interfészek**

  nincs

- **Attribútumok**
  - **int speed :** a kocsi sebessége, hány „steppenként" lép az autó
  - **int timetomove :** várakozási idő lámpák és táblák esetén

- **Metódusok**
  - **int step() :** a mozgást valósítja meg
  - **void move() :** az ütközéselkerülést segítő metódus
  - **void pass(Car) :** az előzést végző függvény
  - **void setSpeed(int) :** beállítja az autó sebességét
  - **int getSpeed() :** visszaadja az autó sebsségét
  - **void destroy() :** törli az autót

### 4.2.4 City

- **Felelősség**
  A várost reprezentáló osztály, felelőssége a városba be és kilépések szabályozása, a városban lévő objektumok nyilvántartása, mozgatása.

- **Ősosztályok**
  nincs

- **Interfészek**
  nincs

- **Attribútumok**
  - **LinkedList<Car> car :** a városban közlekedő autók
  - **LinkedList<RoadBlock> road :** a város útszerkezete
  - **LinkedList<ITraffic> traffic :** a városban lévő közlekedési szabályok
  - **LinkedList<Building> building :** a városban található épületek

- **Metódusok**
  - **void step() :** a város mozgatása

### 4.2.5 Hideout

- **Felelősség**
  Jelezni a játékosnak a játék végét, ha elérte.

- **Ősosztályok**
  Building

- **Interfészek**
  nincs

- **Attribútumok**
  - **int Role :** az épület szerepköre

- **Metódusok**
  - **int getRole() :** visszaadja az épület szerepkörét

### 4.2.6 ITraffic

- **Felelősség**
  A közlekedés szabályozása, interfacet nyújt a közlekedési szabályokat megvalósító osztályok számára.

- **Ősosztályok**
  nincs

- **Interfészek**
  nincs

- **Attribútumok**
  nincs

- **Metódusok**
  - **int getState() :** megadja a jelzés állapotát
  - **void step() :** a váltásokat végző függvény

### 4.2.7 Police

- **Felelősség**
  A rendőrt reprezentáló objektum. Felelőssége ha elkapta a rablót megszakítani a játékot.

- **Ősosztályok**
  Car

- **Interfészek**
  nincs

- **Attribútumok**

- **Metódusok**
  - **void pass(Robber) :** ha egy rabló próbálja megelőzni akkor azt elkapja

### 4.2.8 RoadBlock

- **Felelősség**
  Az utat építi fel, felelőssége a rajta található autók , jelzőlámpák, -táblák és az őt körülvevő RoadBlockok nyilvántartása.

- **Ősosztályok**
  nincs

- **Interfészek**
  nincs

- **Attribútumok**
  - **Car& car :** a rajta található autóra mutató referencia
  - **ITraffic[] traffic :** a rajta található táblák, és lámpák
  - **Building building :** a rajta található épületek
  - **RoadBlock road :** a vele szomszédos útdarabok

- **Metódusok**
  - **void setNeighbour(Neighbour) :** beállítja a szomszédos útdarabokat
  - **void setCar(Car) :** beállítja a rajta lévő autót
  - **void setBuilding(Building) :** beállítja a hozzá tartozó épület
  - **void setTraffic(ITRaffic) :** beállítja a hozzá tartozó lámpákat, táblákat
  - **RoadBlock[] getNeighbour() :** visszaadja a szomszédos útdarabokat
  - **Car getCar() :** visszaadja a rajta lévő autót
  - **Building[] getBuilding() :** visszaadja a hozzá tartozó épület
  - **ITaffic& getTraffic() :** visszaadja a hozzá tartozó lámpákat, táblákat

### 4.2.9 Robber

- **Felelősség**
  A rabló, őt irányíthatjuk a játék során, felelőssége a rendőr detektálása.

- **Ősosztályok**
  Car

- **Interfészek**
  nincs
- **Attribútumok**
- **Metódusok**
  - **void pass(Car) :** az előtte haladó autó megelőzése
  - **void step() :** a mozgása eltér a többi járművétől, hiszen nem vonatkoznak rá szabályok

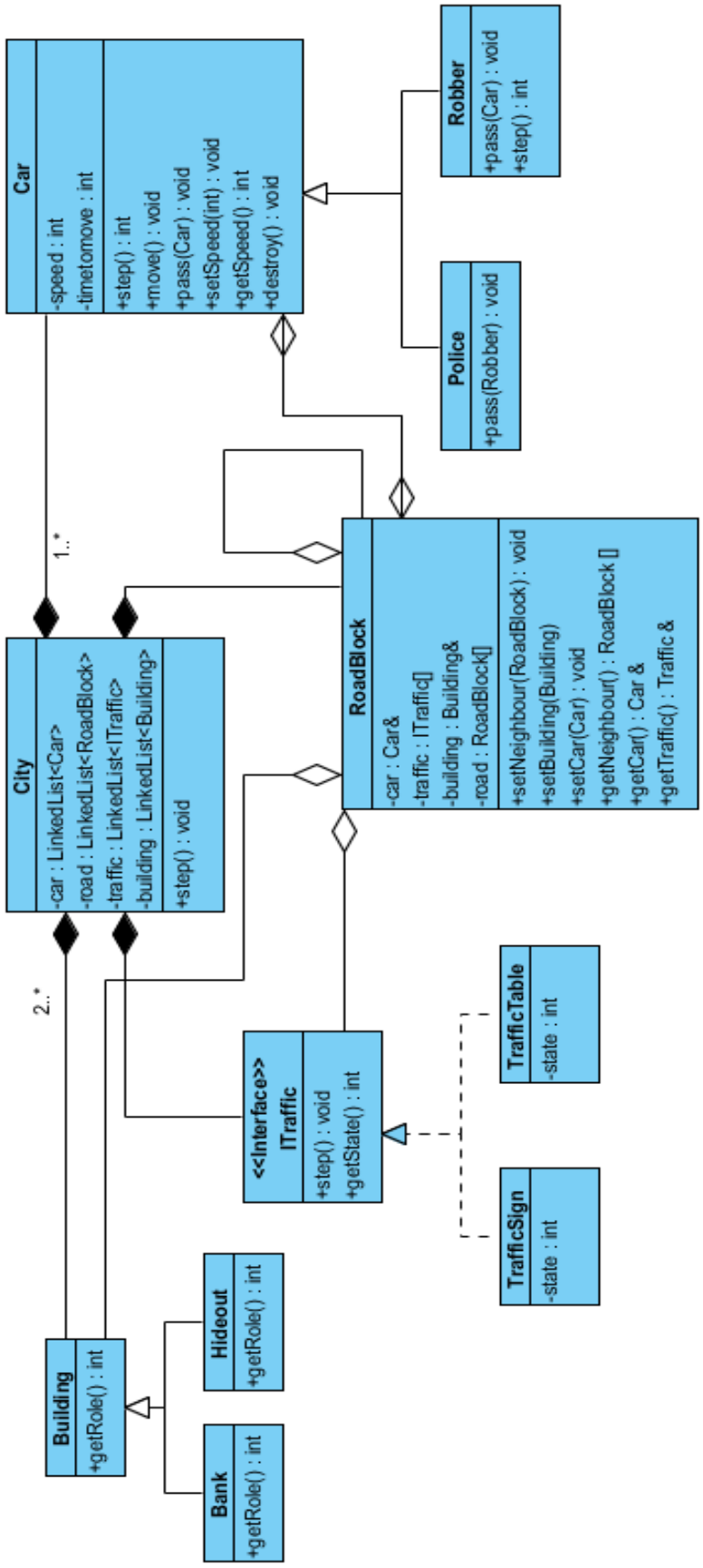
### 4.2.10 TrafficSign

- **Felelősség**
  A jelzőlámpa, felelőssége az állapotának változtatása és annak közlése.
- **Ősosztályok**
  nincs
- **Interfészek**
  ITRaffic
- **Attribútumok**
  - **int state :** aktuális állapota
- **Metódusok**

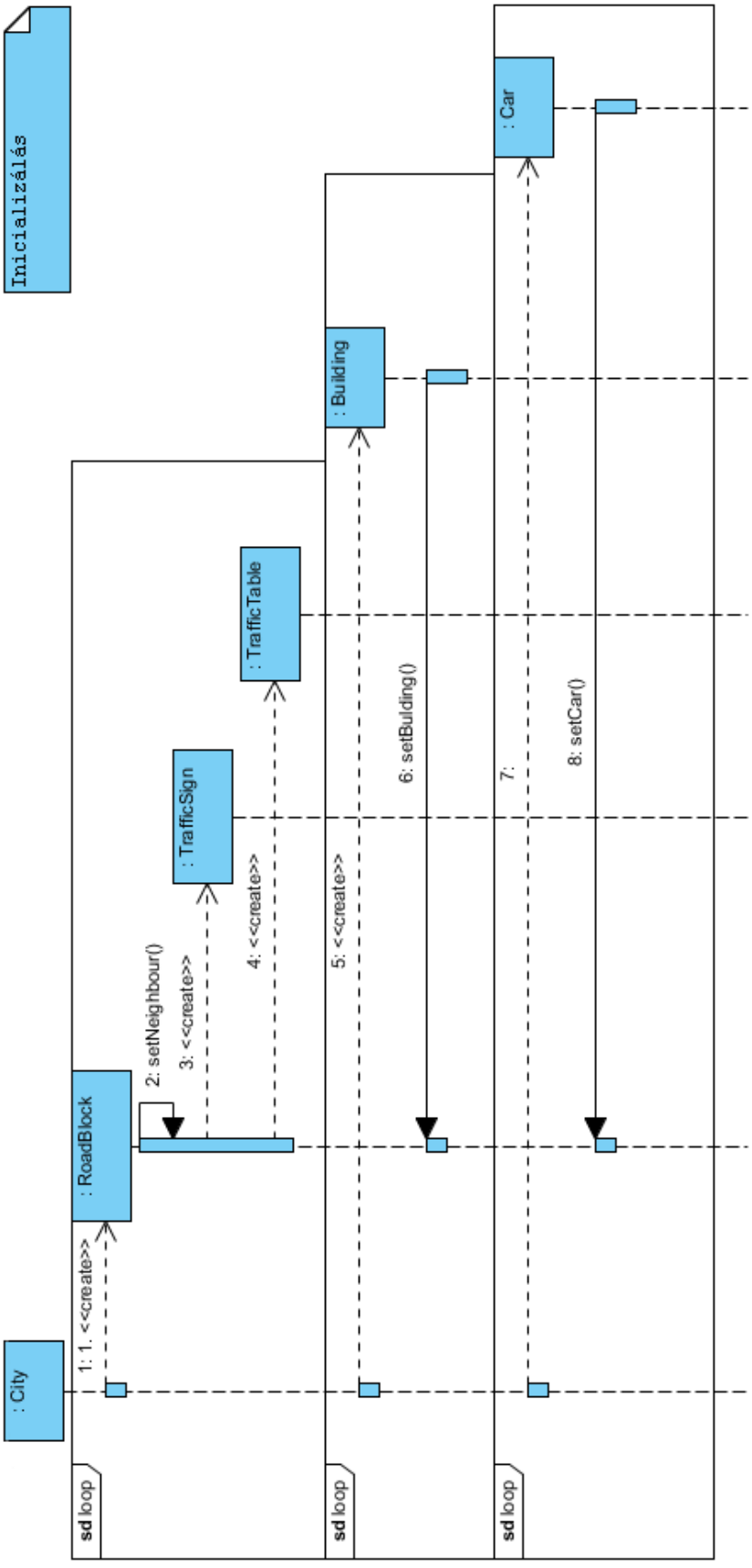
### 4.2.11 TrafficTable

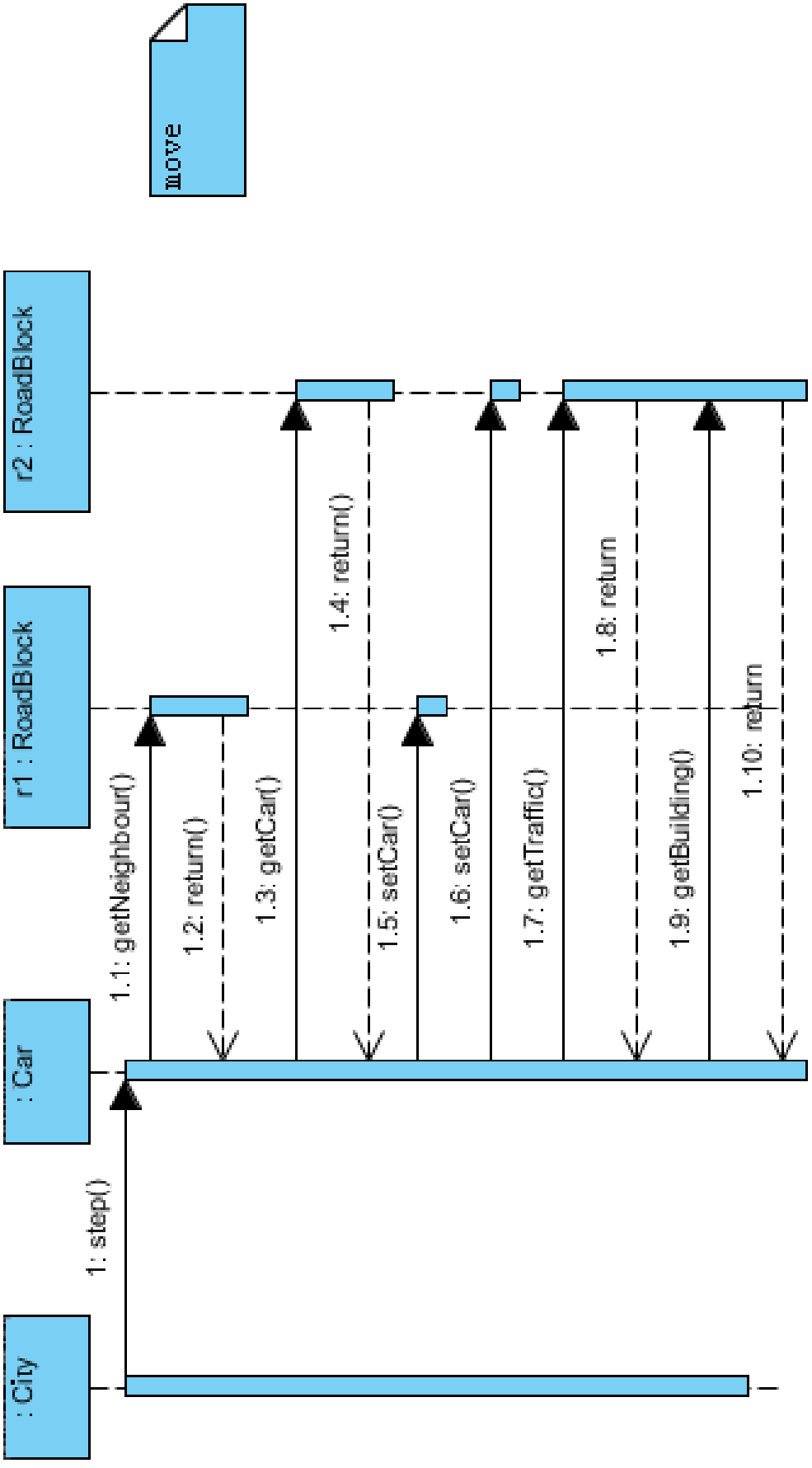
- **Felelősség**
  Tábla, felelőssége a vonatkozó szabály közlése.
- **Ősosztályok**
  nincs
- **Interfészek**
  ITraffic
- **Attribútumok**
  - **int state :** a vonatkozó szabály
- **Metódusok**

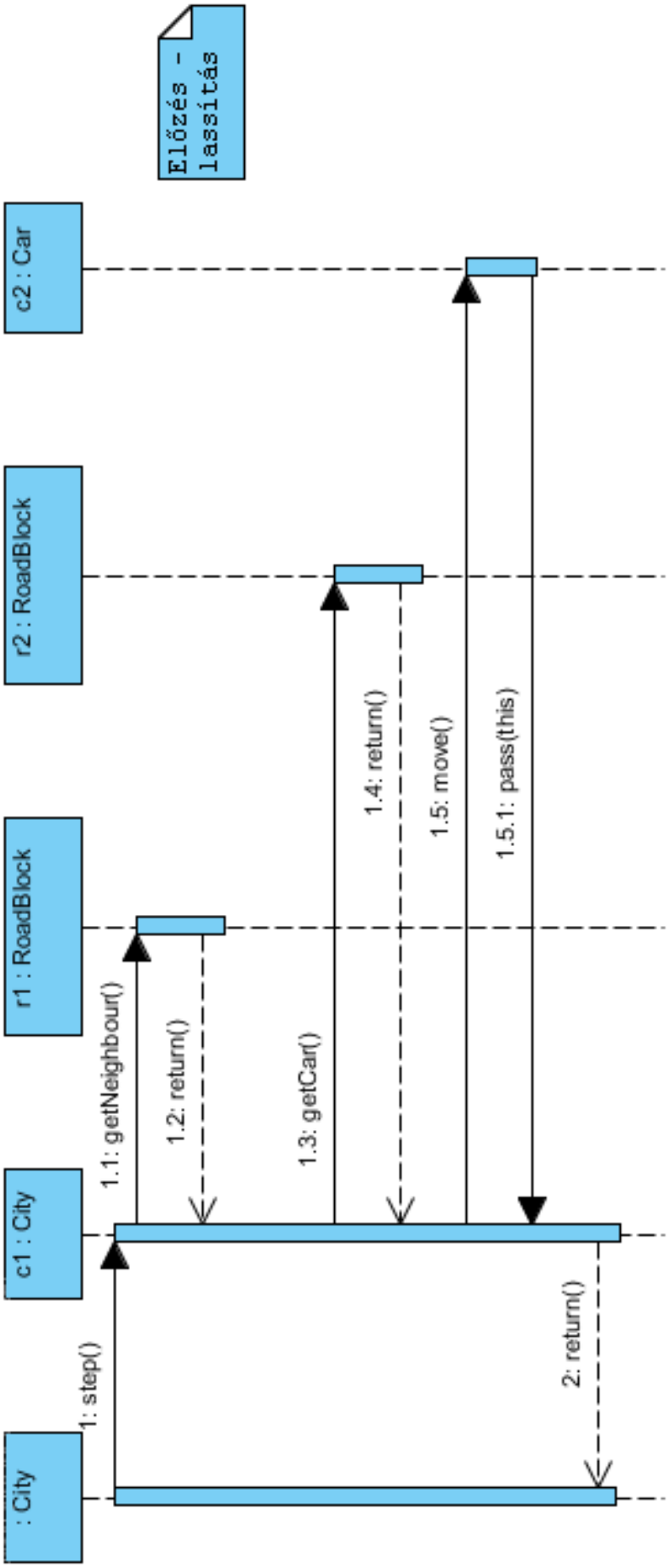
## 4.3 Statikus struktúra diagramok

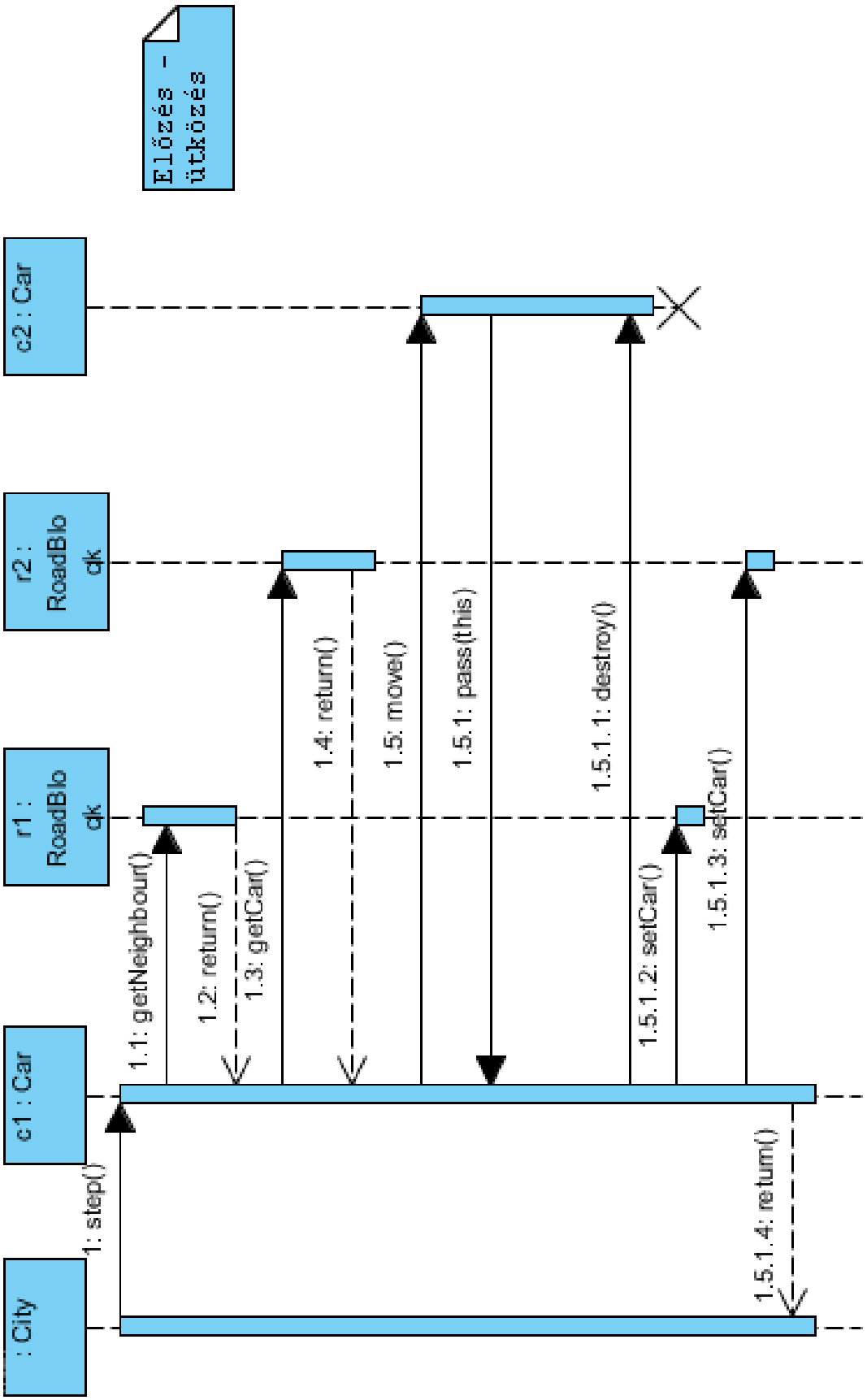


## *4.4 Szekvencia diagramok*

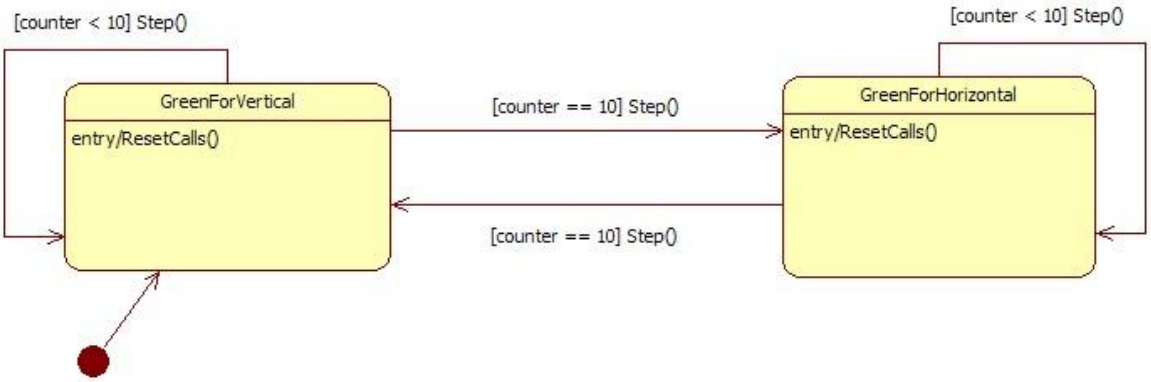



move
r2 : RoadBlock
r1 : RoadBlock
: Car
: City
1: step()
1.1: getNeighbour()
1.2: return()
1.3: getCar()
1.4: return()
1.5: setCar()
1.6: setCar()
1.7: getTraffic()
1.8: return
1.9: getBuilding()
1.10: return

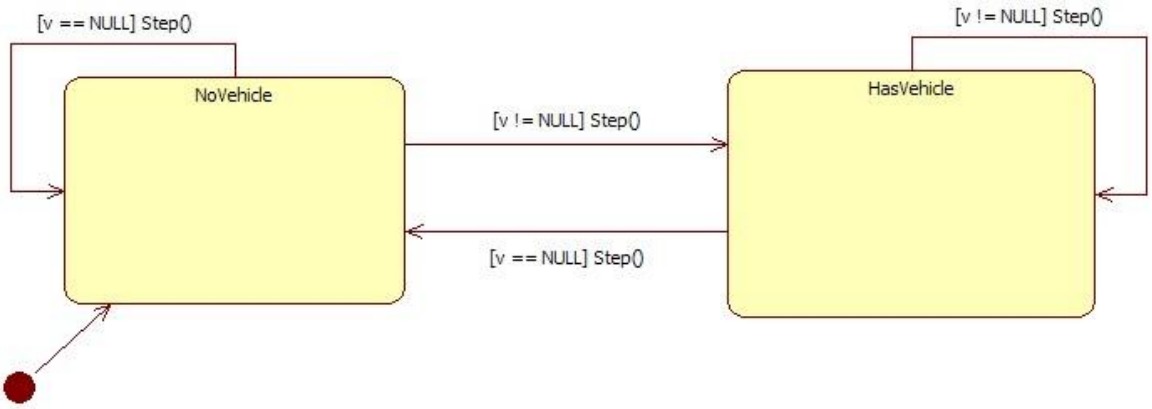

Előzés - lassítás
c2 : Car
r2 : RoadBlock
r1 : RoadBlock
c1 : City
: City
1: step()
1.1: getNeighbour()
1.2: return()
1.3: getCar()
1.4: return()
1.5: move()
1.5.1: pass(this)
2: return()


Előzés - ütközés
c2 : Car
r2 : RoadBlock
r1 : RoadBlock
c1 : Car
: City
1: step()
1.1: getNeighbour()
1.2: return()
1.3: getCar()
1.4: return()
1.5: move()
1.5.1: pass(this)
1.5.1.1: destroy()
1.5.1.2: setCar()
1.5.1.3: setCar()
1.5.1.4: return()

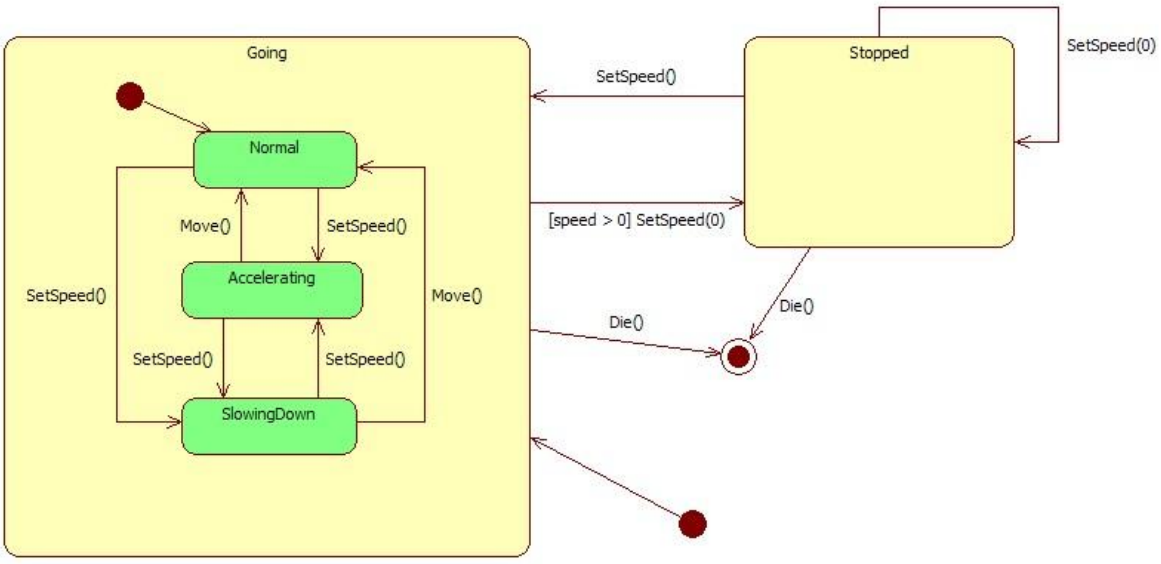
## 4.5 State-chartok




**A jelzőlámpa állapotdiagramja**




**Az útelem állapotdiagramja**




**Az autó állapotdiagramja**

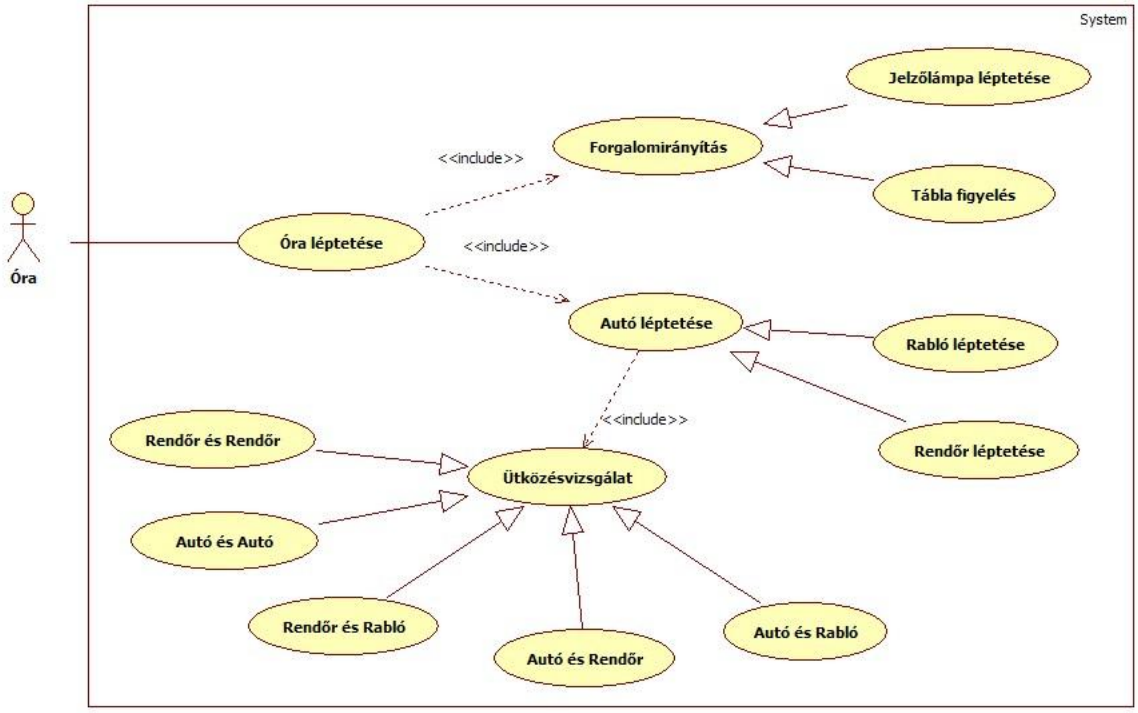
## *4.6 Napló*

| Kezdet | Időtartam | Résztvevők | Leírás |
|---|---|---|---|
| 2010.03.01. 18:00 | 0,5 óra | Molnár<br>Boros | Értekezlet.<br>Döntés: Molnár elkészíti az szekvencia diagramokat és az objektumkatalógust, Boros az osztálydiagrammot, és az osztályok leírását. |
| 2010.03.02. 19:00 | 3 óra | Boros | Elkészíti az osztálydiagrammot |
| 2010.03.03. 08:00 | 1 óra | Molnár | Javítja az osztálydiagrammot |
| 2010.03.03. 09:00 | 2.5 óra | Molnár | Elkészíti a szekvencia diagrammokat |
| 2010.03.03. 20:00 | 2,5 óra | Boros | Elkészíti a az osztályok leírását |
| 2010.03.03. 20:00 | 1.5 óra | Molnár | Elkészíti az objektumkatalógust |
| 2010.03.03. 23:00 | 2 óra | Boros | Átszerkeszti, formázza az elkészült dokumentumot |
| 2010.03.04. 9:00 | 0.5 óra | Rapp | Elkészült dokumentum ellenőrzése, gépelési hibák javítása |

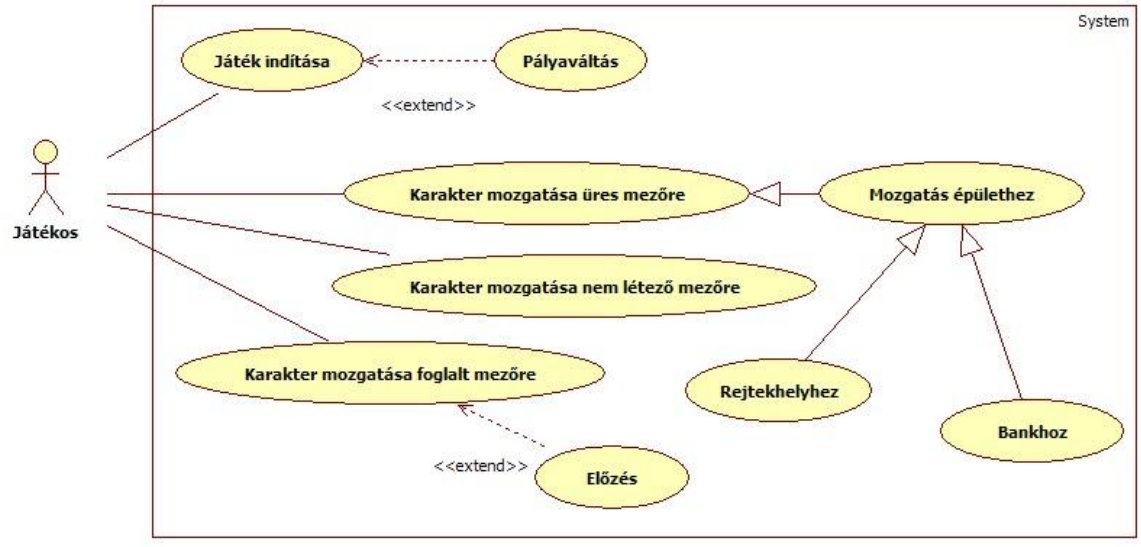
# 5. Szkeleton tervezése

## *5.1 A szkeleton modell valóságos use-case-ei*

### 5.1.1 Use-case diagram




**1. ábra: Az idő múlásával irányított egységek**




**2. ábra: A játékos által vezérelt egységek**

### 5.1.2 Use-case leírások

| **Use-case neve** | Óra léptetése |
|---|---|
| **Rövid leírás** | Amit nem a játékos irányít, azt az óra vezérli. |
| **Aktorok** | Óra |
| **Forgatókönyv** | A játék egy meghatározott idő eltelte után léptet egyet a pálya szereplőin. |

| **Use-case neve** | Forgalomirányítás |
|---|---|
| **Rövid leírás** | Az útkereszteződésekben forgalomirányítás van. |
| **Aktorok** | Óra |
| **Forgatókönyv** | Az óra lép egyet, és az útkereszteződésekben található forgalomirányításnak is jelezzük, hogy új kör van. |

| **Use-case neve** | Jelzőlámpa léptetése |
|---|---|
| **Rövid leírás** | A jelzőlámpa időnként vált. |
| **Aktorok** | Óra |
| **Forgatókönyv** | Az óra lép egyet, és az útkereszteződésekben található jelzőlámpáknak is jelezzük, hogy új kör van. Ezek időnként váltanak. |

| **Use-case neve** | Tábla figyelés |
|---|---|
| **Rövid leírás** | A tábla jelez, ha be lehet hajtani a kereszteződésbe. |
| **Aktorok** | Óra |
| **Forgatókönyv** | Az óra lép egyet, és az útkereszteződésekben található táblának is jelezzük, hogy új kör van. A tábla megvizsgálja, hogy szabad-e behajtani a kereszteződésbe. |

| **Use-case neve** | Autó léptetése |
|---|---|
| **Rövid leírás** | Az utakon autók vannak, ezeket mozgatni kell. |
| **Aktorok** | Óra |
| **Forgatókönyv** | Az óra lép egyet, és az autóknak is jelezzük, hogy léphetnek. |

| **Use-case neve** | Rabló léptetése |
|---|---|
| **Rövid leírás** | A rablóra nem vonatkoznak a szabályok. |
| **Aktorok** | Óra |
| **Forgatókönyv** | Az óra lép egyet, és a rablónak is jelezzük, hogy léphetnek. A rablónak nem kell foglalkoznia a szabályokkal. |

| **Use-case neve** | Rendőr léptetése |
|---|---|
| **Rövid leírás** | A rendőr a rablót keresi, vele ütközni szeretne. |
| **Aktorok** | Óra |
| **Forgatókönyv** | Az óra lép egyet, és a rendőröknek is jelezzük ezt. Ha a rendőr előtt rabló halad, akkor ütköznek. |

| **Use-case neve** | Ütközésvizsgálat |
|---|---|
| **Rövid leírás** | Ha két autó egymás előtt halad, akkor valamit tenni kell. |
| **Aktorok** | Óra |
| **Forgatókönyv** | Az óra lép egyet, és ezt tudatjuk az autókkal is. Ők megvizsgálják, hogy ahova lépni szeretnének oda léphetnek-e. Ha nem, akkor különbözőképp reagálnak. |

| **Use-case neve** | Rendőr és {Rendőr, Autó, Rabló} |
|---|---|
| **Rövid leírás** | Amikor rendőr ütközne valakivel. |
| **Aktorok** | Óra |
| **Forgatókönyv** | Az óra lép egyet, és az autók is lépni szeretnének. A rendőr lelassít az előtte haladó rendőr vagy civil sebességére, de a rablóval ütközik. |

| **Use-case neve** | Rabló és {Rendőr, Autó} |
|---|---|
| **Rövid leírás** | Amikor rabló ütközne valakivel. |
| **Aktorok** | Óra |
| **Forgatókönyv** | Az óra lép egyet, és az autók is lépni szeretnének. A rabló megelőzi az előtte haladó járművet. |

| **Use-case neve** | Autó és {Autó, Rendőr, Rabló} |
|---|---|
| **Rövid leírás** | Amikor két civil ütközne. |
| **Aktorok** | Óra |
| **Forgatókönyv** | Az óra lép egyet, és az autók is lépni szeretnének. Az autó minden esetben lelassít az előtte haladó sebességére. |

| **Use-case neve** | Játék indítása |
|---|---|
| **Rövid leírás** | A játékos új játékot kezd. |
| **Aktorok** | Játékos |
| **Forgatókönyv** | A játékos futtatja a programot, majd a menüpontok közül kiválasztja a startot. |

| **Use-case neve** | Pályaváltás |
|---|---|
| **Rövid leírás** | A játékos pályát választ. |
| **Aktorok** | Játékos |
| **Forgatókönyv** | A játékos elindítja a játékot, majd pályát választ. |

| **Use-case neve** | Karakter mozgatása üres mezőre |
|---|---|
| **Rövid leírás** | A rabló szabad útdarabkára lép. |
| **Aktorok** | Játékos |
| **Forgatókönyv** | A játékos olyan útdarabkára irányítja a rablót, amin nem áll más autó. |

| **Use-case neve** | Mozgatás épülethez |
|---|---|
| **Rövid leírás** | A rabló épülethez ér. |
| **Aktorok** | Játékos |
| **Forgatókönyv** | A játékos olyan útdarabkára irányítja a rablót, ahol épület is található. |

| **Use-case neve** | Rejtekhelyhez |
|---|---|
| **Rövid leírás** | A rabló győz. |
| **Aktorok** | Játékos |
| **Forgatókönyv** | A játékos a rejtekhelyet tartalmazó útdarabkára irányítja a rablót. A pályának vége, a rabló győzött. |

| **Use-case neve** | Bankhoz |
|---|---|
| **Rövid leírás** | A rabló kirabolja a bankot. |
| **Aktorok** | Játékos |
| **Forgatókönyv** | A játékos a bankot tartalmazó útdarabkára irányítja a játékost. A rabló kirabolja a bankot. |

| **Use-case neve** | Karakter mozgatása nem létező mezőre |
|---|---|
| **Rövid leírás** | A rabló ki akar menni a városhatárról. |
| **Aktorok** | Játékos |
| **Forgatókönyv** | A játékos olyan mezőre akar lépni, amely nem létezik. A rabló nem mehet ki. |

| **Use-case neve** | Karakter mozgatása foglalt mezőre |
|---|---|
| **Rövid leírás** | A rabló olyan mezőre szeretne lépni, ahol áll egy autó. |
| **Aktorok** | Játékos |
| **Forgatókönyv** | A játékos olyan mezőre akar lépni, ahol valamilyen autó áll. Rabló esetén ilyenkor előzés történik. |

| **Use-case neve** | Előzés |
|---|---|
| **Rövid leírás** | A rabló megelőzi az előtte haladót. |
| **Aktorok** | Játékos |
| **Forgatókönyv** | A játékos olyan mezőre lépne ahol autó áll, ezért megelőzi azt. |

## *5.2 Architektúra.*

A szkeleton verzió arra hivatott, hogy ellenőrizni lehessen vele a korábban definiált szekvencia diagramok, és use-case modellek működését. Az egyes forgatókönyveket előre elkészített tesztpályákon lehet kipróbálni, melyet a tesztelő felhasználó tud kiválasztani a programból. A program ezen verziója egyszálú, az egyes utasítások, illetve függvényhívások egymás után hívódnak meg.

A teszteléshez létrehoztunk egy osztályt, mely a felhasználóval való kommunikációt biztosítja számunkra. Az egymással analóg eseteket egyazon osztályba soroltuk, a könnyebb áttekinthetőség végett. Az alább található leírásokban olvasható, hogy az egyes pályák mely eseteket mutatják be, illetve mi található rajtuk.

A csapat minden pályán csak a minimálisan szükséges objektumpéldányosításokat végezte el, hiszen most nem a játszhatóságot vagyunk hivatottak bemutatni, hanem az egyes objektumok közötti kommunikációt.

1. tesztpálya:
   - Bemutatja egy jelzőlámpás, vagy táblás útkereszteződés működését.
   - A pályán található egy 4 ágú útkereszteződés, melynek minden pontjához csatlakozik egy útdarabka. A felhasználó döntése szerint helyezkednek el az autók rajtuk, illetve szintén az ő döntése, hogy milyen típusú forgalomirányítás van a kereszteződésben.
2. tesztpálya:
   - Bemutatja, hogy mi történik, mikor a rabló eléri a rejtekhelyet vagy a bankot.
   - A pályán található két útdarabka, ebből az egyikhez csatlakozik a rejtekhely vagy a bank a felhasználó választása szerint.
3. tesztpálya:
   - Bemutat egy egyszerű léptetést, előzést, lassítást, vagy ütközést a lehelyezett autók függvényében.
   - A pályán 3 útdarabka található, ezek egymás után pakolva. Az első két útdarabkán a felhasználó által megadott autók állnak, útdarabkánként maximum 1 darab. A harmadik útdarab üres, nem áll rajta autó.

## *5.3 A szkeleton kezelői felületének terve, dialógusok*

A szkeleton, mint program célja, hogy bemutassa a belső működést, illetve a felvázolt modell működőképességét. Ebben a programban minden objektum szerepel, de azoknak jobbára csak az interfésze definiált. Minden metódushíváskor kiíródik a képernyőre az őt hívó objektum típusa, azonosítója, illetve a saját (és osztályának) neve, paraméterlistája. Ezután a metódus meghívja a működéshez szükséges további metódusokat. Minden olyan pálya indítása előtt, ahol többféle szituáció állhat elő, a program bekéri a felhasználótól a szituációt jellemző adatokat a konzolon. Ha az adott pálya lejátszása közben dönteni kell, akkor a program a felhasználótól kéri azt be egy egyszerű, eldöntendő kérdés formájában. A bekérések alatt a lejátszás szünetel, hiszen csak a kérdésre adott válasz alapján folytatódhat a program. Ha a felhasználó válasza értelmezhetetlen vagy szabálytalan (pl. 2 rablót akar betenni egy szituációba), akkor egy hibaüzenet jelenik meg, majd a program megismétli a kérdést.

A szkeletonnak alkalmasnak kell lennie a szekvencia diagramok ellenőrzésére. Ezt biztosítja a karakteres felület áttekinthetősége, illetve egyszerűsége. A különféle formátumok alant olvashatóak.

A kiírás a következő formátumban történik **metódushívás** esetén:

| **city.car[0]** | **: Robber** | **##CALL##** | **city.road[0].setCar(city.car[0])** |
|---|---|---|---|
| hívó azonosítója | hívó típusa | ez egy hívás | a hívott, és paraméterei |

A kiírás a következő formátumban történik **példányosítás** esetén:

| **city** | **: City** | **##CREATE##** | **city.car[0]** | **: Robber** |
|---|---|---|---|---|
| hívó azonosítója | hívó típusa | példányosítás | a példány neve | és típusa |

A kiírás a következő formátumban történik metódusból **visszatérés** esetén:

| **city.road[1].getBuilding()** | **##RETURN##** | **city.building[1]** | **: Hideout** |
|---|---|---|---|
| metódus azonosítója | visszatérés | visszatérési érték | és típusa |

A kiírás a következő formátumban történik **kérdés** feltevése esetén:

| **##QUESTION##** | **Do you want to step?** | **(Y/N)** | **Y** |
|---|---|---|---|
| ez egy kérdés | kérdés szövege | válaszlehetőségek | tesztelő válasza |

A kiírás a következő formátumban történik **hiba** közlése esetén:

| **##ERROR##** | **You can have only one robber.** |
|---|---|
| ez egy hiba | üzenet |

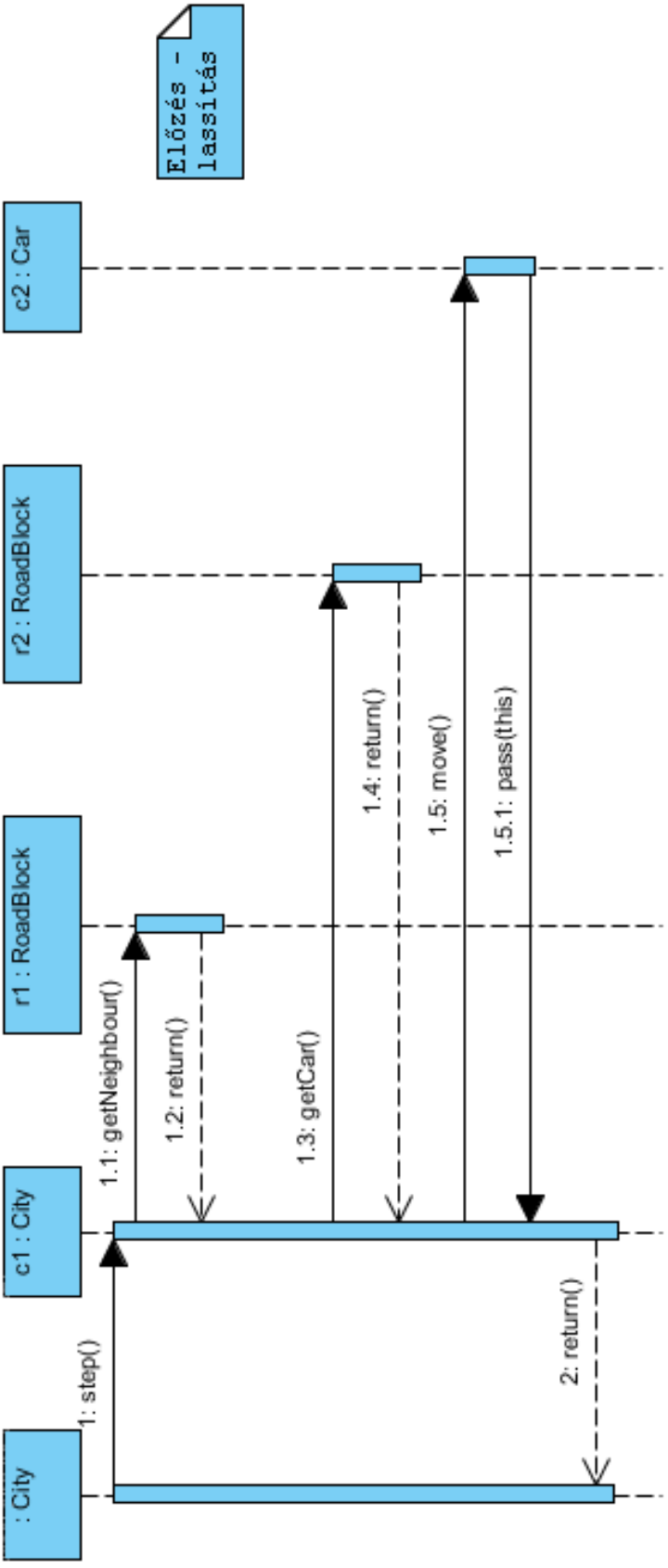
A következő oldalon egy példa log olvasható.

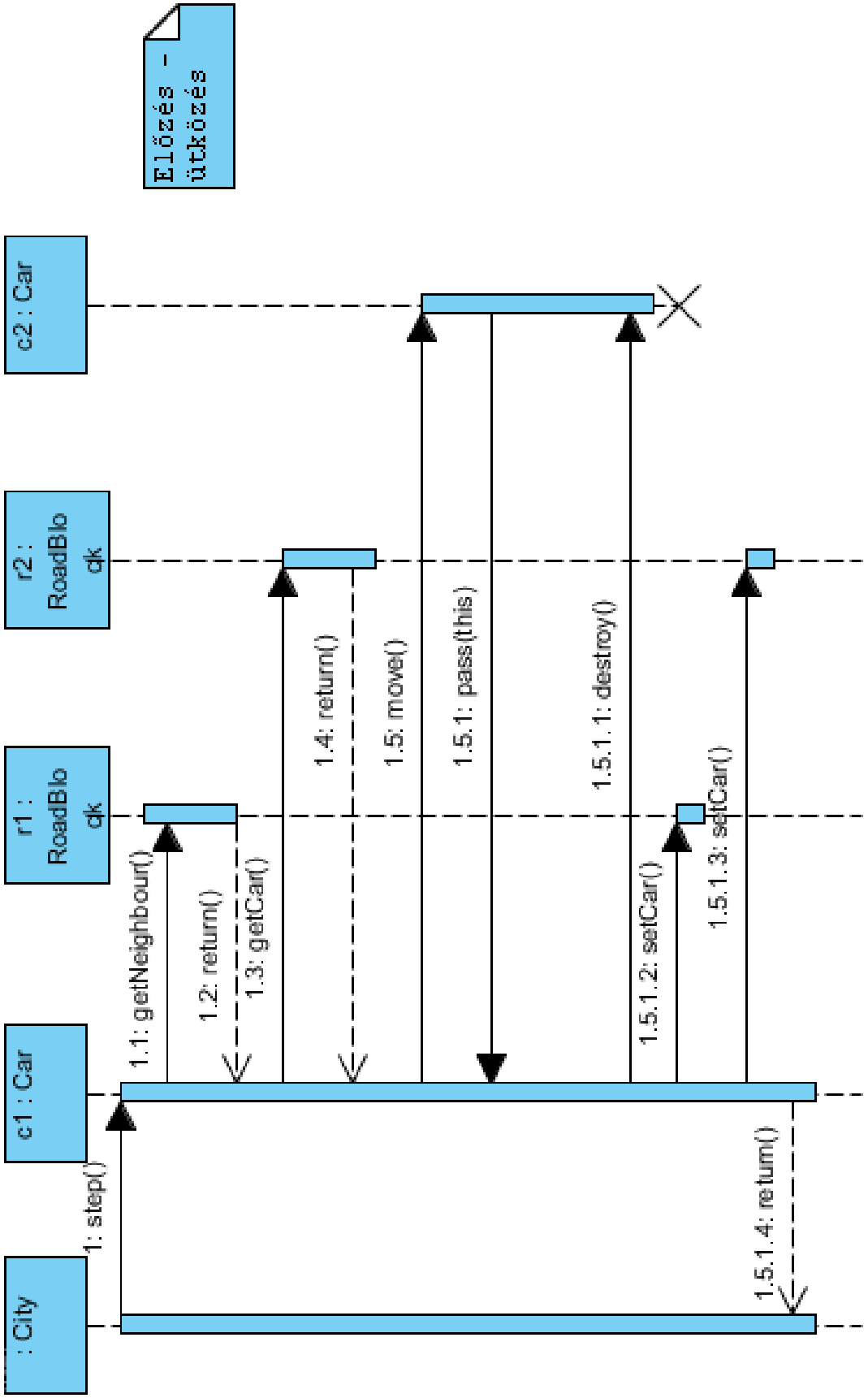
**Példa:**

```
##QUESTION## Do you want to step? (Y/N) Y

city: City ##CALL## car[0].step()
city.car[0] : Robber ##CALL## city.road[0].getNeighbour()
city.road[0].getNeighbour() ##RETURN## city.road[1] : RoadBlock
city.car[0] : Robber ##CALL## city.road[1].getCar()
city.road[1].getCar() ##RETURN## NULL
city.car[0] : Robber ##CALL## city.road[0].setCar(NULL)
city.car[0] : Robber ##CALL## city.road[1].setCar(city.car[0])
city.car[0] : Robber ##CALL## city.road[1].getTraffic()
city.road[1].getTraffic() ##RETURN## NULL
city.car[0] : Robber ##CALL## city.road[1].getBuilding()
city.road[1].getBuilding() ##RETURN## city.building[1] : Hideout
```

## *5.4 Szekvencia diagramok a belső működésre*




Előzés - ütközés
c2 : Car
r2 : RoadBlock
r1 : RoadBlock
c1 : Car
: City
1: step()
1.1: getNeighbour()
1.2: return()
1.3: getCar()
1.4: return()
1.5: move()
1.5.1: pass(this)
1.5.1.1: destroy()
1.5.1.2: setCar()
1.5.1.3: setCar()
1.5.1.4: return()

## 5.5 Napló

| Kezdet | Időtartam | Résztvevők | Leírás |
|---|---|---|---|
| 2010.03.08. 18:00 | 0.5 óra | Boros<br>Takács<br>Rapp | Döntés: Takács elkészíti a játékos use-caset, az architektúrát, és a felület tervet.<br>Döntés: Boros elkészíti a belső szekvenciadiagrammokat.<br>Döntés: Rapp elkészíti az óra use-caset. |
| 2010.03.09 20:30 | 3 óra | Takács | Elkészíti az architekrúrát, felület tervet és dialógu-sokat |
| 2010.03.10. 20:30 | 1.5 óra | Rapp | Elkészíti a use-case diag-ramokat |
| 2010.03.10. 22:30 | 1.5 óra | Takács | Kiegészíti, javítja a use-case diagramokat |
| 2010.03.11. 00:00 | 1.5 óra | Boros | Elkészíti a szekvencia diagramokat, formázza a dokumentumot |

# 6. Szkeleton beadás

## 6.1 Fordítási és futtatási útmutató

### 6.1.1 Fájllista

| Fájl neve | Méret | Keletkezés ideje | Tartalom |
|---|---|---|---|
| Bank.java | 238 | 2010.03.16. | A bankot megvalósító osztály. |
| Building.java | 434 | 2010.03.16. | Az épületeket összefogó osztály. |
| Car.java | 2474 | 2010.03.16. | Az autókat összefogó osztály. |
| City.java | 1360 | 2010.03.16. | A várost reprezentáló osztály. |
| Game.java | 110 | 2010.03.16. | Az elemeket összefogó osztály. |
| Hideout.java | 246 | 2010.03.16. | A menedéket megvalósító osztály. |
| ITraffic.java | 133 | 2010.03.16. | A jelzőrendszert összefogó osztály. |
| Police.java | 2178 | 2010.03.16. | A rendőrt megvalósító osztály. |
| RoadBlock.java | 3268 | 2010.03.16. | A útelemet megvalósító osztály. |
| Robber.java | 2051 | 2010.03.16. | A rablót megvalósító osztály. |
| Skeleton.java | 569 | 2010.03.16. | A tesztelést segítő osztály. |
| TrafficSign.java | 382 | 2010.03.16. | A jelzőlámpát megvalósító osztály. |
| TrafficTable.java | 382 | 2010.03.16. | A jelzőtáblát megvalósító osztály. |
| Compile.bat | 52 | 2010.03.16. | A szkeleton fordításához szükséges batch fájl. |
| Run.bat | 11 | 2010.03.16. | A szkeleton indításához szükséges batch fájl. |

### 6.1.2 Fordítás

Az src.zip fájl ki kell csomagolni a C:\ meghajtóra, ezután a compile.bat futtatásával elkészülnek a .class fájlok. Amennyiben a javac alkalmazás nem a C:\Program Files\Java\jdk1.6.0_17\bin elérési úton található, úgy paranccsorból a javac fájl elérési útját begépelve a C:\elérési út\javac c:\src*.java paranccsal fordíthatóak le a fájlok.

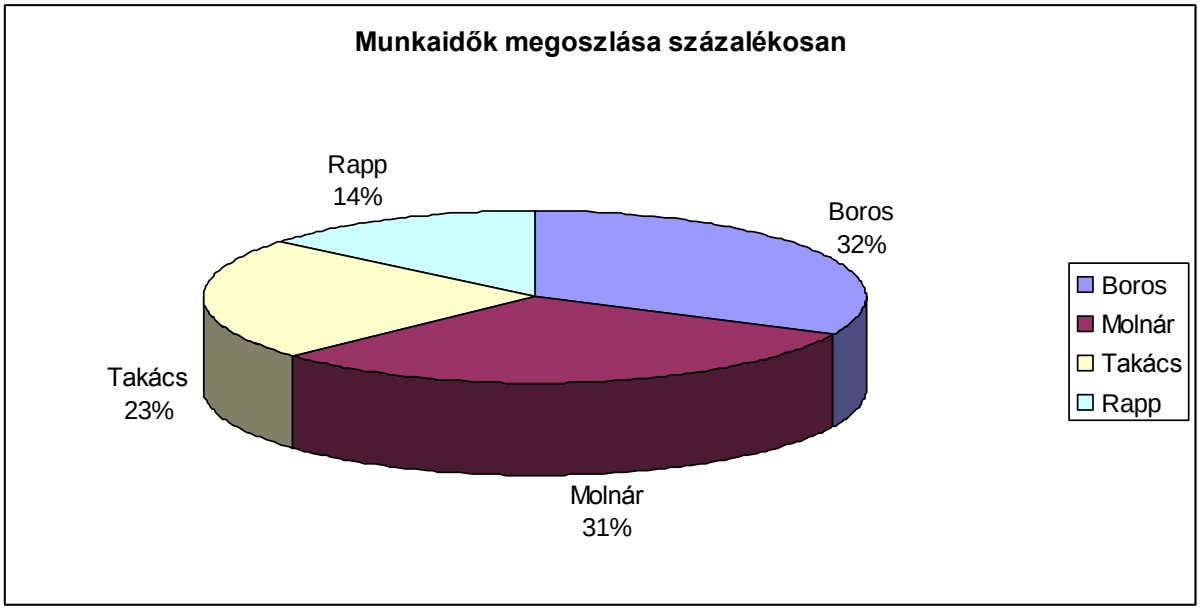
### 6.1.3 Futtatás

A futtatás a Run.bat fájllal végezhető el, hiba esetén a C:\src\java game parancs begépelésével

## 6.2 Értékelés

A csoport munkája során különös fontosságúnak bizonyult a kommunikáció, mivel a csoportmunka ezt megköveteli. A csapattagok személyes találkozókat ritkán tudnak szervezni,

interneten tartják a kapcsolatot, saját levelezőlistával és Windows Messenger programmal. Ötletek gyűjtésére használjuk a levelező listát, és szinkron kommunikációra a Windows Messengert. Ez a módszer megfelelőnek bizonyult, és hatékonyságával is meg vagyunk elégedve.
A csapatban a tagok tudásuk szerint megpróbálják legtöbbet adni a projekt előrehaladásáért, ám eddigi feladatokban nem mindig tudtuk felosztani a részfeladatokat megfelelő arányban a tagok között.




| Tag neve | Munka órában | Munka százalékban | Aláírás |
|---|---|---|---|
| Boros Dávid | 21 | 30 | |
| Molnár László | 20 | 30 | |
| Takács Rajmund | 15 | 25 | |
| Rapp Gábor | 9 | 15 | |

## *6.3 Napló*

| Kezdet | Időtartam | Résztvevők | Leírás |
|---|---|---|---|
| 2010.03.13. 18:00 | 0,5 óra | Takács<br>Rapp<br>Molnár<br>Boros | Értekezlet. Döntés: Molnár elkészíti a szkeletont és a futtatással kapcsolatos dokumentációt, Rapp elkészíti az értékelést, Boros elkészíti a fájllistát |
| 2010.03.16. 19:30 | 0,5 óra | Rapp | Tevékenység: elkészíti az értékelést. |
| 2010.03.17. 17:30 | 0,5 óra | Boros | Tevékenység: elkészíti az fájllistát. |
| 2010.03.16. 19:30 | 6,5 óra | Molnár | Tevékenység: elkészíti szkeletont és a futtatással kapcsolatos dokumentációt |

# 7. Prototípus koncepciója

## *7.1 Prototípus interface-definíciója*

### 7.1.1 Az interfész általános leírása

A prototípus egy karakteres felületű program, melynek elsődleges célja a játék működésének demonstrálása. A szkeleton verzióval ellentétben, itt már nem a belső függvényhívásokra vagyunk kíváncsiak, hanem sokkal inkább a teljes játék logikájának helyes működésére. A könnyű tesztelhetőség végett a program képes bármikor kimenteni az aktuális állapotát egy fájlba, illetve onnan visszaolvasni azt. Lehetőséget biztosítunk arra is, hogy automatizált módon egy előre megírt szkript szerint kapjon utasításokat a programunk, ugyanakkor természetesen ezeket a standard inputon is olvasni képes. A tesztelés szempontjából lényeges, hogy a véletlenszerűen történő események is determinisztikus módon történjenek meg. Ezért a programban lehetőséget biztosítunk az események determinisztikus és véletlenszerű lefutásának futás idejű váltására.

Alapvető koncepció, hogy a tesztek bármikor könnyedén felismerhetőek, illetve reprodukálhatóak legyenek. A csapat is eszerint állítja elő a prototípust.

### 7.1.2 Bemeneti nyelv

A legtöbb itt felsorolt parancs a program végleges verziójában nem kerül felhasználásra, mert csak tesztelési célokat szolgál. Ettől függetlenül lehetőséget fogunk biztosítani arra, hogy a program grafikus változatában is használhatóak legyenek. A prototípust a felhasználó a következő parancsokkal vezérelheti:

***map <mapfile.txt>***

**Leírás:** Betölt egy pályát a <mapfile.txt> fájlból, és felépíti.

**Opciók:**

- **<mapfile.txt>** egy szöveges fájl, melyben a játék egy pályája található meg. A pályaleíró fájl egyszerű szomszédossági mátrixot tárol, ahol a főátlóban kódoljuk, hogy mi van az adott útdarabkán. (pl. jelzőlámpa, stop tábla, bank, stb…)

***run <scriptfile.txt> [echo]***

**Leírás:** Betölt egy utasítássorozatot a <scriptfile.txt> fájlból, majd végrehajtja azt. Opcionálisan kérhető az [echo] paraméterrel, hogy végrehajtáskor az éppen soron következő utasítást a szabványos outputra is kiírja a program.

**Opciók:**

- **<scriptfile.txt>** egy szöveges fájl, melyben a program által értelmezhető utasítások találhatóak, sorban, egymás után.
- **[echo]** Opcionális bináris értékkészletű {0, 1} paraméter, melynek alapértéke 0. Értékei a következőképp értelmezendők:
  - **1** = Az éppen soron következő utasítás megjelenik a szabványos kimeneten is.
  - **0** = Az utasítások „néma” módban hajtódnak végre. Ilyenkor csak az adott utasítás hatására jelenhetnek meg tájékoztató üzenetek, vagy egyéb kimenetek. Maguk az utasítások nem kerülnek kiírásra.

***load <statefile.txt>***

**Leírás:** Betölt egy játékállapotot a <statefile.txt> fájlból. Az aktuális állapot törlődik a memóriából, helyére a beolvasott kerül.
**Opciók:**

- **<statefile.txt>** egy szöveges fájl, melyben a játék egy adott pillanatbeli állapota található meg, egy könnyedén értelmezhető formátumban.

***save <statefile.txt> [logfile.txt]***

**Leírás:** Menti a játék aktuális állapotát a <statefile.txt> fájlba. Ez a fájl értelmezhető a load parancs számára is. Opcionálisan képes elmenteni a [logfile.txt] fájlba a képernyőn megjelenített információkat is, beleértve a felhasználói inputot, és a program konzolra írt outputját.
**Opciók:**

- **<statefile.txt>** egy szöveges fájl, melyben a játék egy adott pillanatbeli állapota található meg, egy könnyedén értelmezhető formátumban.
- **[logfile.txt]** egy szöveges fájl, melyben könnyen értelmezhető módon vannak tárolva a program képernyőjén megjelenített információk. Opcionális paraméter, melyet, ha megadunk, akkor a megadott fájlba ment, egyébként semmit sem csinál.

***random <mode>***

**Leírás:** Beállítja az események (döntések) viselkedését. Lehet választani véletlenszerű és determinisztikus között.
**Opciók:**

- **<mode>** egy bináris {0, 1} paraméter, mely értékei az alábbiakat jelentik:
  - **1** = a döntések véletlenszerűek.
  - **0** = a döntések determinisztikusak.

***exit***

**Leírás:** Kilép a programból.
**Opciók:** -

***step [count]***

**Leírás:** [count] számú lépést vált ki. Ha nincs paraméter, akkor csak egyet lép.
**Opciók:**

- **[count]** egy opcionális pozitív egész paraméter, amelynek megfelelő számú lépés hajtódik végre a programban.

***win***

**Leírás:** Hatása olyan, mintha a bankrabló elért volna a rejtekhelyig.
**Opciók:** -

***loose***

**Leírás:** Hatása olyan, mintha a rendőr elkapta volna a rablót.
**Opciók:** -

***bunny***

**Leírás:** Hatása olyan, mintha a rabló elütött volna egy húsvéti nyulat.
**Opciók:** -

***rob***

**Leírás:** Hatása olyan, mintha a rabló kirabolta volna a bankot.
**Opciók:** -

***money <amount>***

**Leírás:** Hatására a rabló pénze <amount> lesz.
**Opciók:**

- **<amount>** Pozitív egész szám, vagy 0. A pénzmennyiséget reprezentálja.

***health <hp>***

**Leírás:** Hatására a rablónak <hp> életpontja lesz.
**Opciók:**

- **<hp>** Pozitív egész szám, vagy 0. A rabló életpontját reprezentálja.

***speed <v>***

**Leírás:** Hatására a rabló autója <v> sebességgel fog menni.
**Opciók:**

- **<v>** Pozitív egész szám, vagy 0. A rabló sebességét reprezentálja.

***move <dir>***

**Leírás:** Hatása olyan, mintha a <dir> -nek megfelelő gombot nyomtuk volna le az iránybeállító billentyűzeten.
**Opciók:**

- **<dir>** Egy négyelemű értékkészletből {up, down, left, right} előálló paraméter, melynek értelmezése az alábbi:
  - **up** = Hatása olyan, mintha a „fel” gombot nyomtuk volna meg.
  - **down** = Hatása olyan, mintha a „le” gombot nyomtuk volna meg.
  - **left** = Hatása olyan, mintha a „bal” gombot nyomtuk volna meg.
  - **right** = Hatása olyan, mintha a „jobb” gombot nyomtuk volna meg.

***say <message>***

**Leírás:** Hatására a <message> üzenet fog megjelenni a képernyőn.
**Opciók:**

- **<message>** Tetszőleges hosszúságú szöveg.

### 7.1.3 Kimeneti nyelv

A program két fájlt készít tesztelés során. Egyet, amiben a képernyőn megjelenő eseményeket „logolja”, és egyet, amibe elmenti a programban szereplő összes objektum állapotát.
Ezt a mentést a program a következő utasításra készíti el:

```
save state.txt log.txt
```

Ahol a `save` kulcsszó indítja a mentést (fájlba írást), első paraméter a program állapotainak létrehozandó fájl neve, második pedig a képernyőn történt események mentési helye. A második opcionális, az első kötelező.

Példa a log fájl tartalmára:

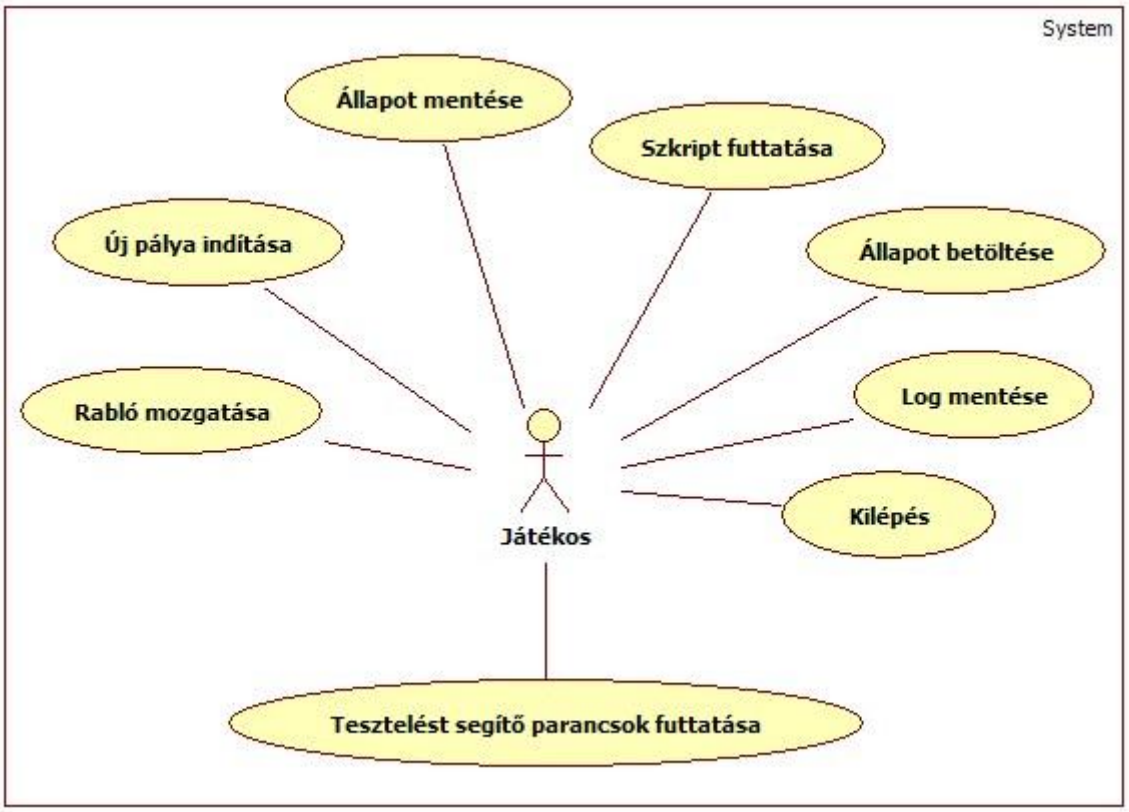
```
INPUT: save state.txt log.txt
OUTPUT: Files saved: state.txt, log.txt
```

Példa az objektumok állapotának mentésére szolgáló fájl tartalmára:

```
OBJECT     car1  TYPE  Car
Attr1      value1
Attr2      value2
```

Ahol `OBJECT` után lévő string a példány neve, `TYPE` után pedig van a típus, amiből visszatöltésnél példányosítanunk kell. Az attribútumok ezek alatt vannak felsorolásszerűen, egymástól újsorral elválasztva, attribútum neve és attribútum értéke pedig tabulátorral elválasztva.

## 7.2 Összes részletes use-case




**1. ábra: A végleges program részletes use-case diagramja**

| **Use-case neve** | Tesztelést segítő parancsok futtatása |
|---|---|
| **Rövid leírás** | A játékos képes futtatni a tesztelési célú parancsokat is. A végleges játékprogramban ezeket hívják „cheat”-eknek. |
| **Aktorok** | Játékos |
| **Forgatókönyv** | A játékos begépeli a specifikált parancsok valamelyikét. |

| **Use-case neve** | Rabló mozgatása |
|---|---|
| **Rövid leírás** | A játékos képes a rabló autóját mozgatni. |
| **Aktorok** | Játékos |
| **Forgatókönyv** | A játékos betölt egy pályát, vagy egy állapotot, majd azon irányítja a rablót. |

| **Use-case neve** | Új pálya indítása |
|---|---|
| **Rövid leírás** | A játékos képes teljesen új, fájlból betöltött pályán kezdeni a játékot. |
| **Aktorok** | Játékos |
| **Forgatókönyv** | A játékos a program indítása után betölt egy pályaleíró fájlt, mire a játék elindul. A játékos irányíthatja a rablót. |

| **Use-case neve** | Állapot mentése |
|---|---|
| **Rövid leírás** | A játékos kimentheti a program aktuális állapotát. |
| **Aktorok** | Játékos |
| **Forgatókönyv** | A játékos a megfelelő parancs kiadásával kimenti a program aktuális állapotát. |

| **Use-case neve** | Szkript futtatása |
|---|---|
| **Rövid leírás** | A játékos képes szkripteket futtatni, a tesztelés automatizálása végett. |
| **Aktorok** | Játékos |
| **Forgatókönyv** | A játékos betölt egy szkriptfájlt a program indítása után, melyet a program végrehajt. |

| **Use-case neve** | Állapot betöltése |
|---|---|
| **Rövid leírás** | A játékos betölthet egy korábban kimentett programállapotot fájlból. |
| **Aktorok** | Játékos |
| **Forgatókönyv** | A játékos a megfelelő parancs kiadásával betölt egy korábban elmentett programállapotot, majd innen folytatja a szimulációt. |

| **Use-case neve** | Log mentése |
|---|---|
| **Rövid leírás** | A játékos diagnosztikai célból kimentheti a program által küldött és fogadott üzeneteket egy fájlba. |
| **Aktorok** | Játékos |
| **Forgatókönyv** | A játékos a megfelelő parancs kiadásával kimenti a (prototípus esetén képernyőn megjelenő) információkat. |

| **Use-case neve** | Kilépés |
|---|---|
| **Rövid leírás** | A játékos befejezheti a program futtatását. |
| **Aktorok** | Játékos |
| **Forgatókönyv** | A játékos a megfelelő parancs kiadásával megszakítja a program futását, és kilép. |

## *7.3 Tesztelési terv*

| **Teszt-eset neve** | Inicializálás |
|---|---|
| **Rövid leírás** | Pálya betöltésének ellenőrzése |
| **Teszt célja** | Ellenőrizni, hogy a pálya betöltésével kapcsolatos folyamatok megfelelően zajlanak le, és a kapcsolódó objektumok attribútumai megfelelő értéket vesznek-e fel. Tesztelni kívánt osztályok: **City, Building, Bank, Hideout.**<br>Forgatókönyv: Játék indítása, pálya betöltése, kilépés |

| **Teszt-eset neve** | Rabló irányítása |
|---|---|
| **Rövid leírás** | Rabló irányításának, játék megnyerése. |
| **Teszt célja** | Rabló irányíthatóságának tesztelése, sebesség, irány módosítása, rabló elvezetése a rejtekhelyig.<br>Tesztelni kívánt osztályok: **City, Robber, Car, Building, Hideout, Roadblock.**<br>Forgatókönyv: Játék indítása, pálya betöltése, rabló létrehozása, rabló irányításának sorozata, rejtekhely elérése, játék megnyerése, kilépés |

| **Teszt-eset neve** | Rendőr elkapja a rablót |
|---|---|
| **Rövid leírás** | Rendőr a rabló mögé kerül, és pár lépés után elkapja. |
| **Teszt célja** | Rendőr elkapási képességének tesztelése. A rendőr a rabló mögé kerül, és lépések meghatározott sorozata után elkapja.<br>Tesztelni kívánt osztályok: **City, Police, Car, Robber, Roadblock.**<br>Forgatókönyv: Játék indítása, pálya betöltése, rabló, rendőr létrehozása, rabló irányításának sorozata, rendőr elkapja, játék vége. |

| **Teszt-eset neve** | Ütközés elkerülés |
|---|---|
| **Rövid leírás** | Szemlélteti, mi történik, ha két autó utoléri egymást**,** illetve ha rabló ér utol autót. |
| **Teszt célja** | Tesztelni, az autók valóban nem ütköznek-e egymásnak, azaz az ütközés elkerülés megfelelően működik-e.<br>Tesztelni kívánt osztályok: **City, Car, Robber, RoadBlock, ITraffic, TrafficSign, TrafficTable.**<br>Forgatókönyv: játék indítása, pálya betöltése, autók elhelyezése, autók mozgatása, hogy egyik hátulról gyorsabban menjen az előtte lévőnél. Következő lépésben pedig úgy mozgatni, hogy a rabló tudjon előzni**.** |

| **Teszt-eset neve** | Nyuszi elütés |
|---|---|
| **Rövid leírás** | Rablót üldözi a rendőr, rabló áthajt a nyuszin, a rendőr egy ideig nem tudja elkapni.. |
| **Teszt célja** | Rabló immunisságát ellenőrizni Nyuszi elütés esetén.<br>Tesztelni kívánt osztályok: **City, Car, Robber, Police, Bunny, Roadblock, Traffictable, TrafficSign**<br>Forgatókönyv: játék indítása, pálya betöltése, rendőr, rabló, nyuszi elhelyezése, rablónak utasítássorozat megadása, áthajt a nyuszin, rabló mozgatása, játék vége. |

## 7.4 Tesztelést támogató segéd- és fordítóprogramok specifikálása

A tesztelés során a program kimentett állapotát a várható teszteredményekkel Microsoft Excel programmal hasonlítjuk össze.

A program képes szövegfájlokat beolvasni, például ha az adatokat újsor és tabulátor jel választja el. Beépített függvényei megkönnyítik a tesztelést, fejlett grafikus felülete pedig könnyen átláthatóvá teszi az adatokat, segít a hibákat azonosítani, például feltételes formázással kiemelhetjük a nem várt eredménynek megfelelő adatokat, összesíthetjük a hibás sorokat, esetleg diagramot készíthetünk a hibák előfordulásának okairól.

## 7.5 Napló

| Kezdet | Időtartam | Résztvevők | Leírás |
|---|---|---|---|
| 2010.03.22 21:00 | 0,5 óra | Boros<br>Molnár<br>Takács<br>Rapp | Értekezlet:<br>Döntés:<br>Rapp elkészíti a Tesztelési tervet, a Tesztelést segítő program specifikálását, és a kimeneti nyelv meghatározását.<br>Takács elkészíti a Prototípus interfész definícióját a Kimeneti nyelv kivételével, valamint a részletes Use-Case-eket. |
| 2010.03.23 17:00 | 1,5 óra | Rapp | Elkészíti a Kimeneti nyelvet, és a Tesztelést támogató programok specifikációját. |
| 2010.03.24 19:00 | 3,5 óra | Rapp | Elkészíti a Tesztelési tervet. |
| 2010.03.24 19:00 | 3 óra | Takács | Elkészíti a Prototípus interfész definícióját a Kimeneti nyelv kivételével. |
| 2010.03.24 22:00 | 2 óra | Takács | Elkészíti a részletes Use-Case-eket. |
| 2010.03.25 10:00 | 0,5 óra | Takács | A dokumentum ellenőrzése, helyesírási hibák javítása, formázás. |

# 8. Részletes tervek

## *8.1 Osztályok és metódusok tervei.*

### 8.1.1 Bank

- **Felelősség**

  A játékos avatárjának kiindulópontja.

- **Ősosztályok**

  Building

- **Interfészek**

  nincs

- **Attribútumok**

- **Metódusok**

  - GetType():visszatér egy az osztályra mutató referenciával

### 8.1.2 Building

- **Felelősség**

  A városban lévő épületek, felelősségük a szerepkörük nyilvántartása.

- **Ősosztályok**

  nincs

- **Interfészek**

  nincs

- **Attribútumok**

- **Metódusok**
  - GetType():visszatér egy az osztályra mutató referenciával

### 8.1.3 Bunny

- **Felelősség**

  Ha a rabló átmegy rajta egy ideig immortal lesz.

- **Ősosztályok**

  nincs

- **Interfészek**

  nincs

- **Attribútumok**
  - **int speed :** a kocsi sebessége, hány „steppenként” lép az autó
  - **int timetomove :** várakozási idő lámpák és táblák esetén

- **Metódusok**
  - **void pass(Car) :** az előzést végző függvény

### 8.1.4 Car

- **Felelősség**

  A városban közlekedő autók, felelősségük a szabályok betartása és az ütközések elkerülése, sebességük nyilvántartása.

- **Ősosztályok**

  nincs

- **Interfészek**

  nincs

- **Attribútumok**
  - **int speed :** a kocsi sebessége, hány „steppenként” lép az autó
  - **int timetomove :** várakozási idő lámpák és táblák esetén

- **Metódusok**
  - **int  step() :** a mozgást valósítja meg
  - **void move() :** az ütközéselkerülést segítő metódus
  - **void pass(Car) :** az előzést végző függvény
  - **void setSpeed(int) :** beállítja az autó sebességét
  - **int getSpeed() :** visszaadja az autó sebsségét

- **void destroy() :** törli az autót

### 8.1.5 City

- **Felelősség**
  A várost reprezentáló osztály, felelőssége az inicializálás irányítása a léptetés vezérlése.
- **Ősosztályok**
  nincs
- **Interfészek**
  nincs
- **Attribútumok**
  - **LinkedList<Car> car :** a városban közlekedő autók
  - **LinkedList<RoadBlock> road :** a város útszerkezete
  - **LinkedList<ITraffic> traffic :** a városban lévő közlekedési szabályok
  - **LinkedList<Building> building :** a városban található épületek
- **Metódusok**
  - **void step() :** a város mozgatása

### 8.1.6 Hideout

- **Felelősség**
  Jelezni a játékosnak a játék végét, ha elérte.
- **Ősosztályok**
  Building
- **Interfészek**
  nincs
- **Attribútumok**

nincs

- **Metódusok**
  - **int getType() :** visszaadja az épület szerepkörét

### 8.1.7 ITraffic

- **Felelősség**
  A közlekedés szabályozása, interfacet nyújt a közlekedési szabályokat megvalósító osztályok számára.
- **Ősosztályok**
  nincs
- **Interfészek**
  nincs
- **Attribútumok**
  nincs
- **Metódusok**
  - **int getState() :** megadja a jelzés állapotát
  - **void step() :** a váltásokat végző függvény
  - Itraffic getTraffic(): visszatér egy az objektumra mutató referenciával

### 8.1.8 Police

- **Felelősség**
  A rendőrt reprezentáló objektum. Felelőssége ha elkapta a rablót megszakítani a játékot.
- **Ősosztályok**
  Car
- **Interfészek**
  nincs
- **Attribútumok**
- **Metódusok**
  - **void pass(**Car**) :** ha egy rabló próbálja megelőzni akkor azt elkapja, egyébként Carként viselkedik

### 8.1.9 RoadBlock

- **Felelősség**
  Az utat építi fel, felelőssége a rajta található autók , jelzőlámpák, -táblák és az őt körülvevő RoadBlockok nyilvántartása.
- **Ősosztályok**
  nincs

- **Interfészek**
  nincs

- **Attribútumok**
  - **Car& car :** a rajta található autóra mutató referencia
  - **ITraffic[] traffic :** a rajta található táblák, és lámpák
  - **Building building :** a rajta található épületek
  - **RoadBlock road :** a vele szomszédos útdarabok

- **Metódusok**
  - **void setNeighbour(Neighbour) :** beállítja a szomszédos útdarabokat
  - **void setCar(Car) :** beállítja a rajta lévő autót
  - **void setBuilding(Building) :** beállítja a hozzá tartozó épület
  - **void setTraffic(ITRaffic) :** beállítja a hozzá tartozó lámpákat, táblákat
  - **RoadBlock[] getNeighbour() :** visszaadja a szomszédos útdarabokat
  - **Car getCar() :** visszaadja a rajta lévő autót
  - **Building[] getBuilding() :** visszaadja a hozzá tartozó épület
  - **ITaffic& getTraffic() :** visszaadja a hozzá tartozó lámpákat, táblákat

### 8.1.10 Robber

- **Felelősség**
  A rabló, őt irányíthatjuk a játék során, felelőssége a rendőr detektálása.

- **Ősosztályok**
  Car

- **Interfészek**
  nincs

- **Attribútumok**

- **Metódusok**
  - **void pass(Car) :** az előtte haladó autó megelőzése
  - **void step() :** a mozgása eltér a többi járművétől, hiszen nem vonatkoznak rá szabályok

### 8.1.11 TrafficSign

- **Felelősség**
  A jelzőlámpa, felelőssége az állapotának változtatása és annak közlése.

- **Ősosztályok**
  nincs

- **Interfészek**
  ITRaffic

- **Attribútumok**
- **Metódusok**

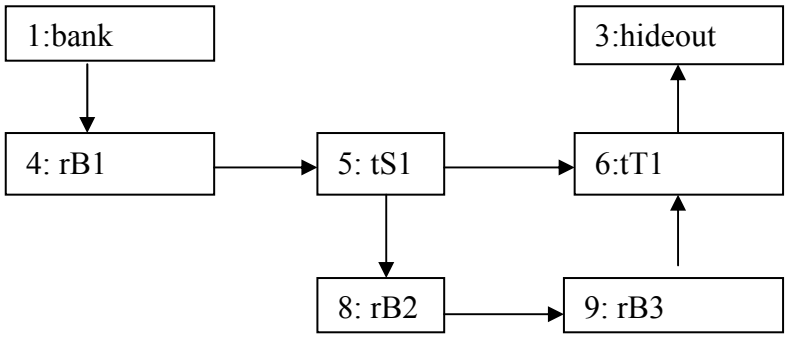
### 8.1.12 TrafficTable

- **Felelősség**
  Tábla, felelőssége a vonatkozó szabály közlése.
- **Ősosztályok**
  nincs
- **Interfészek**
  ITraffic
- **Attribútumok**

## 8.2 A tesztek részletes tervei, leírásuk a teszt nyelvén

Pályák leírásánál a következő mintát kell alkalmazni:
A térképet egy irányított szomszédossági mátrix formájában kell a programnak beadni egy szöveges fájlban.

Példa a térkép megadására:




Ahol a várost, mint egy 3*3 mezős rácsot képzelhetjük el. Ahol nincsen semmi, pl a kettes helyen, ott ki kell hagyni a számozást. A nyilak jelentik az irányított utakat az adott objektumok között.

Jelmagyarázat:
bank = Bank (1)
hideout = Hideout (2)
rB = RoadBlock, (3)
tS =TrafficSign, (4)
tT = TrafficTable (5)
Zárójelben az osztályok neve mögötti szám jelenti, hogy az irányított szomszédossági mátrixban hányas számmal hivatkozunk az objektum típusára, a főátlóban.

A konkrét térképet reprezentáló mátrix:

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | | | 1 | | | | | |
| 2 | | 0 | | | | | | | |
| 3 | | | 2 | | | -1 | | | |
| 4 | -1 | | | 3 | 1 | | | | |
| 5 | | | | -1 | 4 | 1 | | 1 | |
| 6 | | | 1 | | -1 | 5 | | | -1 |
| 7 | | | | | | | 0 | | |
| 8 | | | | | -1 | | | 3 | 1 |
| 9 | | | | | | 1 | | -1 | 3 |

Ez alapján a mapfile.txt tartalma, amivel a tesztelést fogjuk végezni:

```
1    0    0    1    0    0    0    0    0
0    0    0    0    0    0    0    0    0
0    0    2    0    0    -1   0    0    0
-1   0    0    3    1    0    0    0    0
0    0    0    -1   4    1    0    1    0
0    0    1    0    -1   5    0    0    -1
0    0    0    0    0    0    0    0    0
0    0    0    0    -1   0    0    3    1
0    0    0    0    0    1    0    -1   3
```

### 8.2.1 Teszteset - Inicializálás

- **Leírás**
  A teszteset célja annak ellenőrzése, a játék inicializálásakor betöltendő térképen szereplő objektumok megfelelően példányosodnak-e.
- **Ellenőrzött funkcionalitás, várható hibahelyek**
  Pályához tartozó objektumok létrehozásának ellenőrzése. Esetleges hibalehetőség, ha valamely objektum nem jön létre, vagy attribútumai nem megfelelő értékeket vesznek fel.
- **Bemenet**
  ```
  map mapfile.txt
  save state.txt
  ```
- **Elvárt kimenet**
  A state.txt tartalma a következőképp kell, hogy alakuljon:

  ```
  OBJECT    city   TYPE    City
  car       NULL
  road[0]   rB_bank
  road[1]   rB_hideout
  road[2]   rB1
  road[3]   rB_tS1
  road[4]   rB_tT1
  road[5]   rB2
  road[6]   rB3
  traffic[0]       tS1
  traffic[1]       tT1
  building[0]      bank
  building[1]      hideout
  OBJECT    rB_bank   TYPE RoadBlock
  car       NULL
  traffic   NULL
  building  bank
  road      rB1
  OBJECT    rB_hideout     TYPE RoadBlock
  car       NULL
  traffic   NULL
  building  hideout
  road      NULL
  OBJECT    rB1    TYPE    RoadBlock
  car       NULL
  ```

```
traffic   NULL
building  NULL
road      rB_tS1
OBJECT    rB_tS1 TYPE    RoadBlock
car       NULL
traffic   tS1
building  NULL
road      [rB2,rB_tT1]
OBJECT    rB_tT1 TYPE    RoadBlock
car       NULL
traffic   tT1
road      rB_hideout
OBJECT    rB2    TYPE    RoadBlock
car       NULL
traffic   NULL
road      rB3
OBJECT    rB3    TYPE    RoadBlock
car       NULL
traffic   NULL
road      rB_tT1
OBJECT    tS1    TYPE    TrafficSign
state     0
OBJECT    tT1    TYPE    TrafficTable
state     0
OBJECT    bank      TYPE Bank
OBJECT    hideout   TYPE Hideout
```

### 8.2.2 Teszteset – Rabló irányítása

- **Leírás**
  Rabló irányíthatóságának tesztelése, sebesség, irány módosítása, rabló elvezetése a rejtekhelyig.
- **Ellenőrzött funkcionalitás, várható hibahelyek**
  Rabló reagálása az utasításokra, forgalomirányítás tesztelése.
- **Bemenet**

```
map mapfile.txt
load statefile.txt
move up
step 2
move left
step
save state.txt
```

  Ahol statefile.txt a következőket tartalmazza:

```
OBJECT    car[0]    TYPE Robber
speed     10
timetomove     5
OBJECT    road[2]   TYPE RoadBlock
car  car[0]
```

- **Elvárt kimenet**

```
OBJECT    city   TYPE    City
car[0]    car[0]
road[0]   rB_bank
road[1]   rB_hideout
road[2]   rB1
road[3]   rB_tS1
road[4]   rB_tT1
road[5]   rB2
road[6]   rB3
traffic[0]       tS1
traffic[1]       tT1
building[0]      bank
building[1]      hideout
OBJECT    rB_bank   TYPE RoadBlock
car       NULL
traffic   NULL
building  bank
road      rB1
OBJECT    rB_hideout    TYPE RoadBlock
car       car[0]
traffic   NULL
building  hideout
road      NULL
OBJECT    rB1    TYPE    RoadBlock
car       NULL
traffic   NULL
building  NULL
road      rB_tS1
OBJECT    rB_tS1 TYPE    RoadBlock
car       NULL
traffic   tS1
building  NULL
road      [rB2,rB_tT1]
OBJECT    rB_tT1 TYPE    RoadBlock
car       NULL
traffic   tT1
road      rB_hideout
OBJECT    rB2    TYPE    RoadBlock
car       NULL
traffic   NULL
road      rB3
OBJECT    rB3    TYPE    RoadBlock
car       NULL
traffic   NULL
road      rB_tT1
OBJECT    tS1    TYPE    TrafficSign
state     0
OBJECT    tT1    TYPE    TrafficTable
state     0
OBJECT    bank      TYPE Bank
OBJECT    hideout   TYPE Hideout
```

```
OBJECT    car[0] TYPE    Robber
speed     30
timetomove       0-255   int
```

### 8.2.3 Teszteset – Rendőr elkapja a rablót

- **Leírás**
  Rendőr a rabló mögül indul, és 2 lépés után elkapja.
- **Ellenőrzött funkcionalitás, várható hibahelyek**
  Rendőr el tudja-e kapni a rablót, léptetések megfelelően mennek-e.
- **Bemenet**

```
map mapfile.txt
load statefile.txt
step 4
```

  Ahol statefile.txt a következőket tartalmazza:

```
OBJECT    car[0]    TYPE Robber
speed     10
timetomove     5
health    5
OBJECT    car[1]    TYPE Police
speed     30
timetomove     5
OBJECT    road[3]   TYPE RoadBlock
car  car[0]
OBJECT    road[2]   TYPE RoadBlock
car  car[1]
```

- **Elvárt kimenet**

```
OBJECT    city   TYPE    City
car[0]    car[0]
car[1]    car[1]
road[0]   rB_bank
road[1]   rB_hideout
road[2]   rB1
road[3]   rB_tS1
road[4]   rB_tT1
road[5]   rB2
road[6]   rB3
traffic[0]       tS1
traffic[1]       tT1
building[0]      bank
building[1]      hideout
OBJECT    rB_bank   TYPE RoadBlock
car       NULL
traffic   NULL
building  bank
road      rB1
OBJECT    rB_hideout     TYPE RoadBlock
```

```
car       NULL
traffic   NULL
building  hideout
road      NULL
OBJECT    rB1    TYPE    RoadBlock
car       NULL
traffic   NULL
building  NULL
road      rB_tS1
OBJECT    rB_tS1 TYPE    RoadBlock
car       car[1]
traffic   tS1
building  NULL
road      [rB2,rB_tT1]
OBJECT    rB_tT1 TYPE    RoadBlock
car       car[0]
traffic   tT1
road      rB_hideout
OBJECT    rB2    TYPE    RoadBlock
car       NULL
traffic   NULL
road      rB3
OBJECT    rB3    TYPE    RoadBlock
car       NULL
traffic   NULL
road      rB_tT1
OBJECT    tS1    TYPE    TrafficSign
state     0
OBJECT    tT1    TYPE    TrafficTable
state     0
OBJECT    bank      TYPE Bank
OBJECT    hideout   TYPE Hideout
OBJECT car[0] TYPE Robber
speed 30
timetomove 0-255 int
health    0
```

### 8.2.4 Teszteset – Ütközéselkerülés autókkal

- **Leírás**
  Azt az esetet modellezi, amikor két autó halad ugyan azon az útvonalon, és a hátsó gyorsabban halad, beéri a másikat. Ekkor a gyorsabbnak le kell lassítania az előtte lévő sebességére.
- **Ellenőrzött funkcionalitás, várható hibahelyek**
  Ütközéselkerülés, sebességmódosulás.
- **Bemenet**
- `map mapfile.txt`
- `load statefile.txt`
  `step 1`
  Ahol statefile.txt a következőket tartalmazza:
  `OBJECT    car[0]    TYPE Car`

```
speed     10
timetomove     5
OBJECT    car[1]    TYPE Car
speed     30
timetomove     5
OBJECT    road[3]   TYPE RoadBlock
car  car[0]
OBJECT    road[2]   TYPE RoadBlock
car  car[1]
```

- **Elvárt kimenet**

```
OBJECT    city   TYPE    City
car[0]    car[0]
car[1]    car[1]
road[0]   rB_bank
road[1]   rB_hideout
road[2]   rB1
road[3]   rB_tS1
road[4]   rB_tT1
road[5]   rB2
road[6]   rB3
traffic[0]       tS1
traffic[1]       tT1
building[0]      bank
building[1]      hideout
OBJECT    rB_bank   TYPE RoadBlock
car       NULL
traffic   NULL
building  bank
road      rB1
OBJECT    rB_hideout     TYPE RoadBlock
car       NULL
traffic   NULL
building  hideout
road      NULL
OBJECT    rB1    TYPE    RoadBlock
car       NULL
traffic   NULL
building  NULL
road      rB_tS1
OBJECT    rB_tS1 TYPE    RoadBlock
car       car[1]
traffic   tS1
building  NULL
road      [rB2,rB_tT1]
OBJECT    rB_tT1 TYPE    RoadBlock
car       car[0]
traffic   tT1
road      rB_hideout
```

```
OBJECT    rB2    TYPE    RoadBlock
car       NULL
traffic   NULL
road      rB3
OBJECT    rB3    TYPE    RoadBlock
car       NULL
traffic   NULL
road      rB_tT1
OBJECT    tS1    TYPE    TrafficSign
state     0
OBJECT    tT1    TYPE    TrafficTable
state     0
OBJECT    bank      TYPE Bank
OBJECT    hideout   TYPE Hideout
OBJECT    car[0] TYPE    Car
speed     10
OBJECT    car[1] TYPE    Car
speed     10
```

### 8.2.5 Teszteset – Ütközéselkerülés Rablóval

- **Leírás**
  Rabló ér be egy autót, meg kell előzze.
- **Ellenőrzött funkcionalitás, várható hibahelyek**
  Ütközéselkerülés, sebességmódosulás.
- **Bemenet**
  ```
  map mapfile.txt
  load statefile.txt
  step 1
  ```
  Ahol statefile.txt a következőket tartalmazza:
  ```
  OBJECT    car[0]    TYPE Car
  speed     10
  timetomove     5
  OBJECT    car[1]    TYPE Robber
  speed     30
  timetomove     5
  OBJECT    road[3]   TYPE RoadBlock
  car  car[0]
  OBJECT    road[2]   TYPE RoadBlock
  car  car[1]
  ```

- **Elvárt kimenet**

  ```
  OBJECT    city   TYPE    City
  car[0]    car[0]
  car[1]    car[1]
  road[0]   rB_bank
  road[1]   rB_hideout
  road[2]   rB1
  road[3]   rB_tS1
  ```

```
road[4]   rB_tT1
road[5]   rB2
road[6]   rB3
traffic[0]       tS1
traffic[1]       tT1
building[0]      bank
building[1]      hideout
OBJECT    rB_bank   TYPE RoadBlock
car       NULL
traffic   NULL
building  bank
road      rB1
OBJECT    rB_hideout     TYPE RoadBlock
car       NULL
traffic   NULL
building  hideout
road      NULL
OBJECT    rB1    TYPE    RoadBlock
car       NULL
traffic   NULL
building  NULL
road      rB_tS1
OBJECT    rB_tS1 TYPE    RoadBlock
car       car[0]
traffic   tS1
building  NULL
road      [rB2,rB_tT1]
OBJECT    rB_tT1 TYPE    RoadBlock
car       car[1]
traffic   tT1
road      rB_hideout
OBJECT    rB2    TYPE    RoadBlock
car       NULL
traffic   NULL
road      rB3
OBJECT    rB3    TYPE    RoadBlock
car       NULL
traffic   NULL
road      rB_tT1
OBJECT    tS1    TYPE    TrafficSign
state     0
OBJECT    tT1    TYPE    TrafficTable
state     0
OBJECT    bank      TYPE Bank
OBJECT    hideout   TYPE Hideout
OBJECT    car[0] TYPE    Car
speed     10
OBJECT    car[1] TYPE    Car
speed     30
```

### 8.2.6 Teszteset – Nyuszielütés

- **Leírás**
  Rablót üldözi a rendőr, rabló áthajt a nyuszin, a rendőr egy ideig nem tudja elkapni.
- **Ellenőrzött funkcionalitás, várható hibahelyek**
  Nyuszi elütése valóban immunisságot ad-e a rablónak.
- **Bemenet**

```
map mapfile.txt
load statefile.txt
step 4
```

  Ahol statefile.txt a következőket tartalmazza:

```
OBJECT    car[0]    TYPE Robber
speed     10
timetomove     5
health    5
OBJECT    car[1]    TYPE Police
speed     20
timetomove     5
OBJECT    car[2]    TYPE Bunny
speed     0
timetomove     100
OBJECT    road[3]   TYPE RoadBlock
car  car[0]
OBJECT    road[2]   TYPE RoadBlock
car  car[1]
OBJECt    road[4]   TYPE RoadBlock
car  car[2]
```

- **Elvárt kimenet**

```
OBJECT    city   TYPE    City
car[0]    car[0]
car[1]    car[1]
car[2]    car[2]
road[0]   rB_bank
road[1]   rB_hideout
road[2]   rB1
road[3]   rB_tS1
road[4]   rB_tT1
road[5]   rB2
road[6]   rB3
traffic[0]       tS1
traffic[1]       tT1
building[0]      bank
building[1]      hideout
OBJECT    rB_bank   TYPE RoadBlock
car       NULL
traffic   NULL
building  bank
road      rB1
```

```
OBJECT    rB_hideout     TYPE RoadBlock
car       NULL
traffic   NULL
building  hideout
road      NULL
OBJECT    rB1    TYPE    RoadBlock
car       NULL
traffic   NULL
building  NULL
road      rB_tS1
OBJECT    rB_tS1 TYPE    RoadBlock
car       car[1]
traffic   tS1
building  NULL
road      [rB2,rB_tT1]
OBJECT    rB_tT1 TYPE    RoadBlock
car       car[0]
traffic   tT1
road      rB_hideout
OBJECT    rB2    TYPE    RoadBlock
car       NULL
traffic   NULL
road      rB3
OBJECT    rB3    TYPE    RoadBlock
car       NULL
traffic   NULL
road      rB_tT1
OBJECT    tS1    TYPE    TrafficSign
state     0
OBJECT    tT1    TYPE    TrafficTable
state     0
OBJECT    bank      TYPE Bank
OBJECT    hideout   TYPE Hideout
OBJECT car[0] TYPE Robber
speed 30
timetomove 0-255 int
health    5
immortality      10
```

## 8.3 A tesztelést támogató programok tervei

A teszteredmények összehasonlításának megkönnyítése érdekében készítettünk egy egyszerű fájl összehasonlító programot Java nyelven. A konzolos program parancssori paramétereiben megkapja az összehasonlítandó két fájl nevét, a relatív elérési útvonalukkal együtt. A működése egyszerűen csak annyi, hogy a két fájlt szinkronban beolvassa, és soronként összehasonlítja. Ha eltérés van, akkor azt jelzi a felhasználónak, kiemelve a két fájlban található sort. A program futása végén jelzi, hogy végeredményben egyezik, vagy sem a két fájl. Ha hiba történt a fájlok beolvasása során, akkor ezt hibaüzenettel jelzi.

A megvalósításhoz készítettünk egy különálló osztályt (InputFile), ami a bemeneti fájlt reprezentálja. A főprogramban ezt kétszer példányosítva, majd metódusai segítségével, a fájlokat összehasonlítva értékeljük ki az eredményeket.

A programhoz készült egy JavaDOC is, melyben tételesen megtalálhatóak a program függvényei is, rövid leírásokkal együtt.

Ha hibás bemeneti argumentumokkal futtatjuk a szoftvert, akkor az kiírja a helyes használathoz szükséges szintaxist:

```
Error: Missing or too many argument(s).
Syntax: compare <file1> <file2>
Example: compare out.txt sample.txt
```

Ha helyes paraméterekkel indítjuk, de a két bemeneti fájl közül valamelyik nem található, akkor a következő kimenetet kaphatjuk:

```
Reading files...
Error: File not found.
```

Ha a fájlok léteznek, és az argumentumok is helyesen lettek megadva, illetve a két fájl egyezik, akkor a kimenet a következő: (test1.txt, és test2.txt az összehasonlított fájlok)

```
Reading files...
Comparing file test1.txt to file test2.txt ...
Full match.
```

Ha eltérés van a fájlok között, akkor azokat tételesen felsorolja az alábbiak szerint: (test1.txt, és test2.txt az összehasonlított fájlok, g11, g1, er, err a tartalmazott szövegek, line 1, line 5 értelemszerűen a sorszámok, ahol ezek találhatóak)

```
Reading files...
Comparing file test1.txt to file test2.txt ...

Row mismatch at line 1 in file test1.txt compared to file
test2.txt
test1.txt: g11
test2.txt: g1


Row mismatch at line 5 in file test1.txt compared to file
test2.txt
```

```
test1.txt: er
test2.txt: err

Files are different.
```

## 8.4 Változás a specifikációban

A húsvéti nyuszi időnként feltűnik az utakon. Ha a rabló elüti a nyuszit, akkor rövid időre immunitást élvez, vagyis a rendőrök ekkor nem tudják elkapni, bárhogyan próbálkoznak is.

A húsvéti nyulat a Bunny osztály fogja reprezentálni, aki járműként mozoghat a városban. Így a Bunny a Car ősosztályból származik, rendelkezik a feltűnés időtartamát meghatározó int típusú változóval. Továbbá a rabló osztály implementálunk egy pass(Bunny) : void metódussal ami az immunitását jelző belső változó értékét állítja be, amely minden stepben csökken.

## 8.5 Napló

| Kezdet | Időtartam | Résztvevők | Leírás |
|---|---|---|---|
| 2010.03.29.18:00 | 0,5 óra | Rapp<br>Takács<br>Molnár<br>Boros | Megbeszélés |
| 2010.03.30.18:00 | 2 óra | Rapp | 8.2 Tesztek részletes terveinek kidolgozása |
| 2010.03.31.17:00 | 4 óra | Rapp | 8.2.* Tesztesetek részletes kidolgozása |
| 2010.03.31.19:00 | 2 óra | Takács | 8.3.* A tesztelést támogató programok tervei rész elkészítése |
| 2010.03.31.20:00 | 2 óra | Molnár | 8.1 Osztályleírások elkészítése |
| 2010.03.30.23:00 | 2 óra | Boros | 8.3 Specifikáció-változás kidolgozása, dokumentum javítása, formázása |

# 10. Prototípus beadása

## *10.1 Fordítási és futtatási útmutató*

### 10.1.1 Fájllista

| Fájl neve | Méret | Keletkezés ideje | Tartalom |
|---|---|---|---|
| Bank.java | 464 | 2010.04.18 | A bankot megvalósító osztály. |
| Building.java | 376 | 2010.04.18 | Az épületeket összefogó osztály. |
| Bunny.java | 1.5 | 2010.04.18 | A nyuszit megvalósító osztály. |
| Car.java | 2.6 | 2010.04.18 | Az autókat összefogó osztály. |
| City.java | 4.4 | 2010.04.18 | A várost reprezentáló osztály. |
| Game.java | 2.7 | 2010.04.18 | Az elemeket összefogó osztály. |
| Hideout.java | 468 | 2010.04.18 | A menedéket megvalósító osztály. |
| ITraffic.java | 240 | 2010.04.18 | A jelzőrendszert összefogó osztály. |
| Police.java | 2.1 | 2010.04.18 | A rendőrt megvalósító osztály. |
| RoadBlock.java | 2.0 | 2010.04.18 | A útelemet megvalósító osztály. |
| Robber.java | 2.1 | 2010.04.18 | A rablót megvalósító osztály. |
| Skeleton.java | 569 | 2010.04.18 | A tesztelést segítő osztály. |
| TrafficSign.java | 596 | 2010.04.18 | A jelzőlámpát megvalósító osztály. |
| TrafficTable.java | 100 | 2010.04.18 | A jelzőtáblát megvalósító osztály. |
| Compile.bat | 52 | 2010.04.18 | A szkeleton fordításához szükséges batch fájl. |
| Run.bat | 11 | 2010.04.18 | A szkeleton indításához szükséges batch fájl. |
| Mapfile.txt | 100 | 2010.04.18 | Tesztelési fájl |
| Statefile.txt | 22 | 2010.04.18 | Tesztelési fájl |

### 10.1.2 Fordítás

Az src.zip fájl ki kell csomagolni a D:\ meghajtóra, ezután a compile.bat futtatásával elkészülnek a .class fájlok. Amennyiben a javac alkalmazás nem a D:\Program Files\Java\jdk1.6.0_17\bin elérési úton található, úgy paranccsorból a javac fájl elérési útját begépelve a D:\elérési út\javac c:\src*.java paranccsal fordíthatóak le a fájlok.

### 10.1.3 Futtatás

A futtatás a Run.bat fájllal végezhető el, hiba esetén a D:\src\java game parancs begépelésével

## *10.2 Tesztek jegyzőkönyvei*

### 10.2.1 Initteszt

| | |
|---|---|
| **Tesztelő neve** | Molnár |
| **Teszt időpontja** | 2010.04.18 18:00 |

### 10.2.2 Ütközéstesztek

| | |
|---|---|
| **Tesztelő neve** | Molnár |
| **Teszt időpontja** | 2010.04.18 18:00 |

### 10.2.3 Előzéstesztek

| | |
|---|---|
| **Tesztelő neve** | Molnár |
| **Teszt időpontja** | 2010.04.18 18:00 |

| | |
|---|---|
| **Tesztelő neve** | Molnár |
| **Teszt időpontja** | 2010.04.18 18:00 |
| **Teszt eredménye** | Hibás, a rabló nem előz |
| **Lehetséges hibaok** | Hibás pass(car c) metódus |
| **Változtatások** | A metódus kijavítása |

## *Változtatások a tesztelésben*

A bemenet 4 paraméterű, a térkép, a futtatási fájl, a kimenet elérési helye és a lépések száma.
Pl:
Mapfile.txt
Statefile.txt
Save.txt
10

A térkép szomszédossági mátrix, első értéke a térkép mérete. Az egyes objektumokat kezdőbetűik jelölik.

A kimenet objektumlistázás, kezdve a City objektummal, majd mindegy egyes RoadBlock kiírja a rajta található objektumokat és paramétereit.

A térkép és a tesztesetek koncepciója változatlan. A térkép adott 3x3-as elrendezésű az egyes tesztesetek a különböző állapotokban különböznek. A rabló mozgása a megismételhetőség miatt kötött.
A mapfile mindig ugyanaz a statefilek változnak, a step default értéke 10 legyen.

A state fájlok:

Inicializálás:

mapfile.txt
statefile.txt
saveinit.txt
0

Elvárt kimenet:
OBJECT city TYPE City
road[0]
road[1]
road[3]
road[4]
road[5]
road[7]
road[8]
traffic[0]
traffic[1]
OBJECT road[0] TYPE RoadBlock
OBJECT road[1] TYPE RoadBlock
Building hideout
OBJECT road[3] TYPE RoadBlock
OBJECT road[4] TYPE RoadBlock
Traffic sign
OBJECT road[5] TYPE RoadBlock
Traffic table
OBJECT road[7] TYPE RoadBlock
OBJECT road[8] TYPE RoadBlock

Rendőr elkapja a rablót statefile:
2
Robber 3 5
Police 0 1

Elvárt kimenet:
OBJECT city TYPE City
road[0]
road[1]
road[3]
road[4]
road[5]
road[7]
road[8]
traffic[0]
traffic[1]
car[0]
car[1]
OBJECT road[0] TYPE RoadBlock
OBJECT road[1] TYPE RoadBlock
Building hideout
OBJECT road[3] TYPE RoadBlock
OBJECT road[4] TYPE RoadBlock

Traffic sign
OBJECT road[5] TYPE RoadBlock
Traffic table
OBJECT road[7] TYPE RoadBlock
OBJECT road[8] TYPE RoadBlock

Nyuszi elütés statefile:
3
Robber 3 1
Bunny 4 5
Police 8 2

Elvárt kimenet:
OBJECT city TYPE City
road[0]
road[1]
road[3]
road[4]
road[5]
road[7]
road[8]
traffic[0]
traffic[1]
car[0]
car[1]
car[2]
OBJECT road[0] TYPE RoadBlock
Car  robber speed=1
OBJECT road[1] TYPE RoadBlock
Building hideout
OBJECT road[3] TYPE RoadBlock
OBJECT road[4] TYPE RoadBlock
Traffic sign
Car  car speed=5
OBJECT road[5] TYPE RoadBlock
Traffic table
OBJECT road[7] TYPE RoadBlock
OBJECT road[8] TYPE RoadBlock

Ütközéselkerülés statefile:
2
Car 5 5
Car 8 1

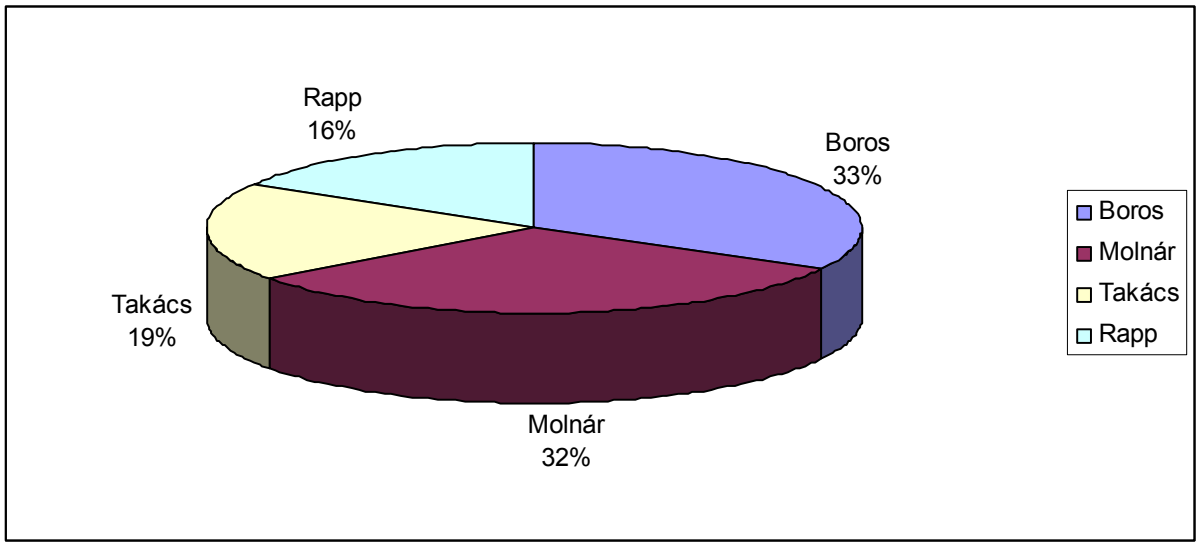
Elvárt kimenet:
OBJECT city TYPE City
road[0]
road[1]
road[3]
road[4]

road[5]
road[7]
road[8]
traffic[0]
traffic[1]
car[0]
car[1]
OBJECT road[0] TYPE RoadBlock
OBJECT road[1] TYPE RoadBlock
Building hideout
OBJECT road[3] TYPE RoadBlock
OBJECT road[4] TYPE RoadBlock
Traffic sign
OBJECT road[5] TYPE RoadBlock
Traffic table
Car  car speed=1
OBJECT road[7] TYPE RoadBlock
OBJECT road[8] TYPE RoadBlock

Ütközéselkerülés rablóval statefile:
2
Robber 3 1
Car 8 2

Elvárt kimenet:
OBJECT city TYPE City
road[0]
road[1]
road[3]
road[4]
road[5]
road[7]
road[8]
traffic[0]
traffic[1]
car[0]
car[1]
OBJECT road[0] TYPE RoadBlock
Car  robber speed=1
OBJECT road[1] TYPE RoadBlock
Building hideout
OBJECT road[3] TYPE RoadBlock
OBJECT road[4] TYPE RoadBlock
Traffic sign
OBJECT road[5] TYPE RoadBlock
Traffic table
OBJECT road[7] TYPE RoadBlock
OBJECT road[8] TYPE RoadBlock

## *10.3 Értékelés*



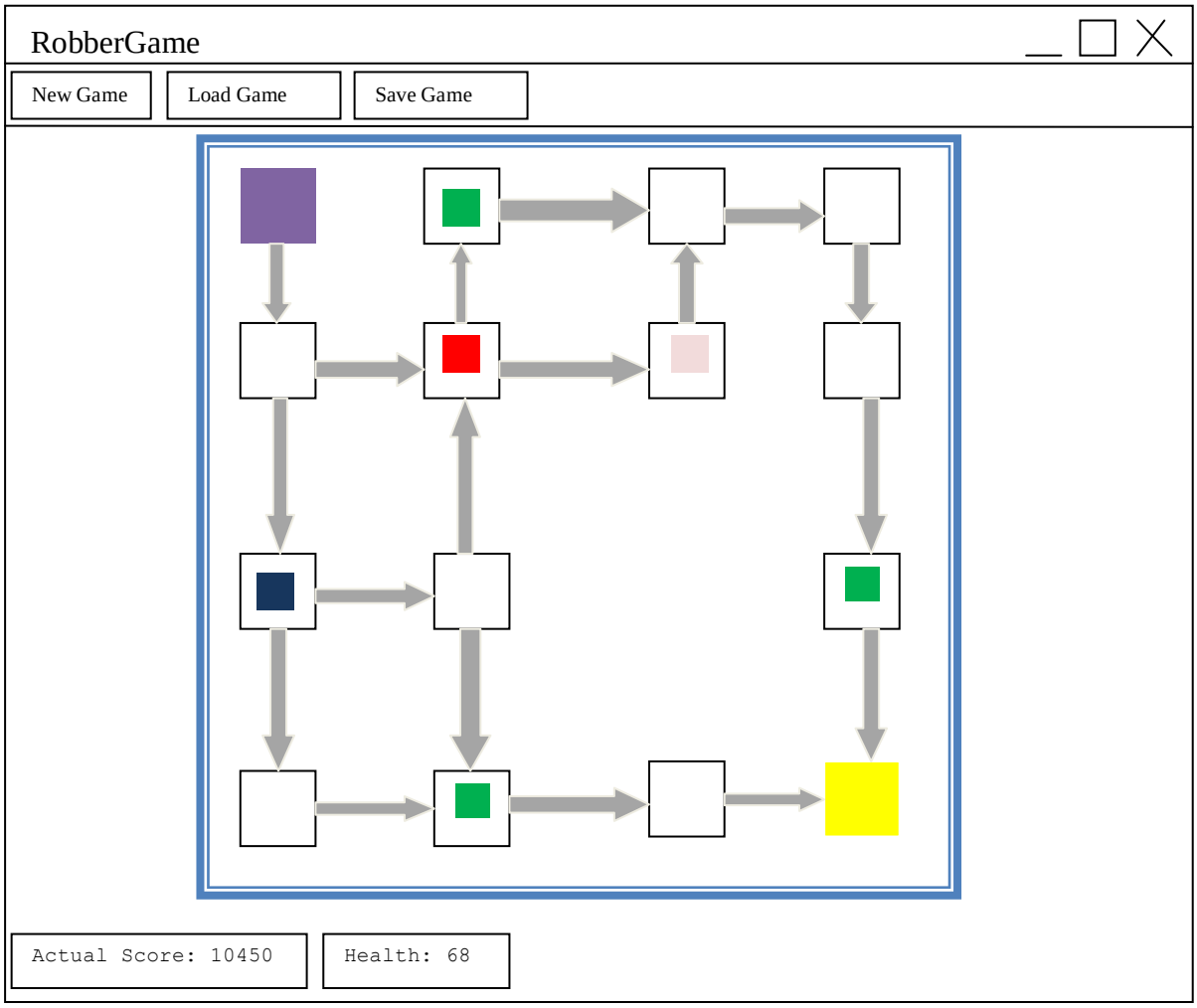

| Tag neve | Munka százalékban | Aláírás |
|---|---|---|
| Boros Dávid | 30 | |
| Molnár László | 30 | |
| Takács Rajmund | 22 | |
| Rapp Gábor | 18 | |

## *10.4 Napló*

| **Kezdet** | **Időtartam** | **Résztvevők** | **Leírás** |
|---|---|---|---|
| 2010.04.05. 18:30 | 0,5 óra | Takács<br>Rapp<br>Molnár<br>Boros | Értekezlet.<br>Döntés: Molnár és Boros elkészíti a prototípust és teszteli. |
| 2010.04.14. 22:00 | 7 óra | Molnár | Kidolgozza a Car és Traffic-al kapcsolatos osztályokat |
| 2010.04.14. 22:00 | 6 óra | Boros | Kidolgozza a City és a Game osztályokat |
| 2010.04.17. 10:00 | 4 óra | Molnár | Javítja és módosítja a mozgást |
| 2010.04.17. 11:00 | 5 óra | Boros | Implementálja a beolvasást |
| 2010.04.18. 16:00 | 5 óra | Boros | Implementálja a tesztelés funkviókat |
| 2010.04.18. 18:00 | 5 óra | Molnár | Teszteli a programot |
| 2010.04.18. 17:00 | 1 óra | Rapp | Módosítja a teszteseteket |
| 2010.04.18. 24:00 | 1 óra | Boros | Megírja a dokumentációt |
| | | | |

# 11. Grafikus felület specifikációja

## *11.1 A grafikus interfész*




**Jelmagyarázat a képernyőképhez:**
A különböző objektumok (épületek, autók, stb..) különböző színnel jelennek meg a játékban:

- rendőr: kék
- rabló: piros
- út: szürke
- nyuszi: rózsaszín
- bank: lila
- rejtekhely: sárga
- civil autók: zöld

## *11.2 A grafikus rendszer architektúrája*

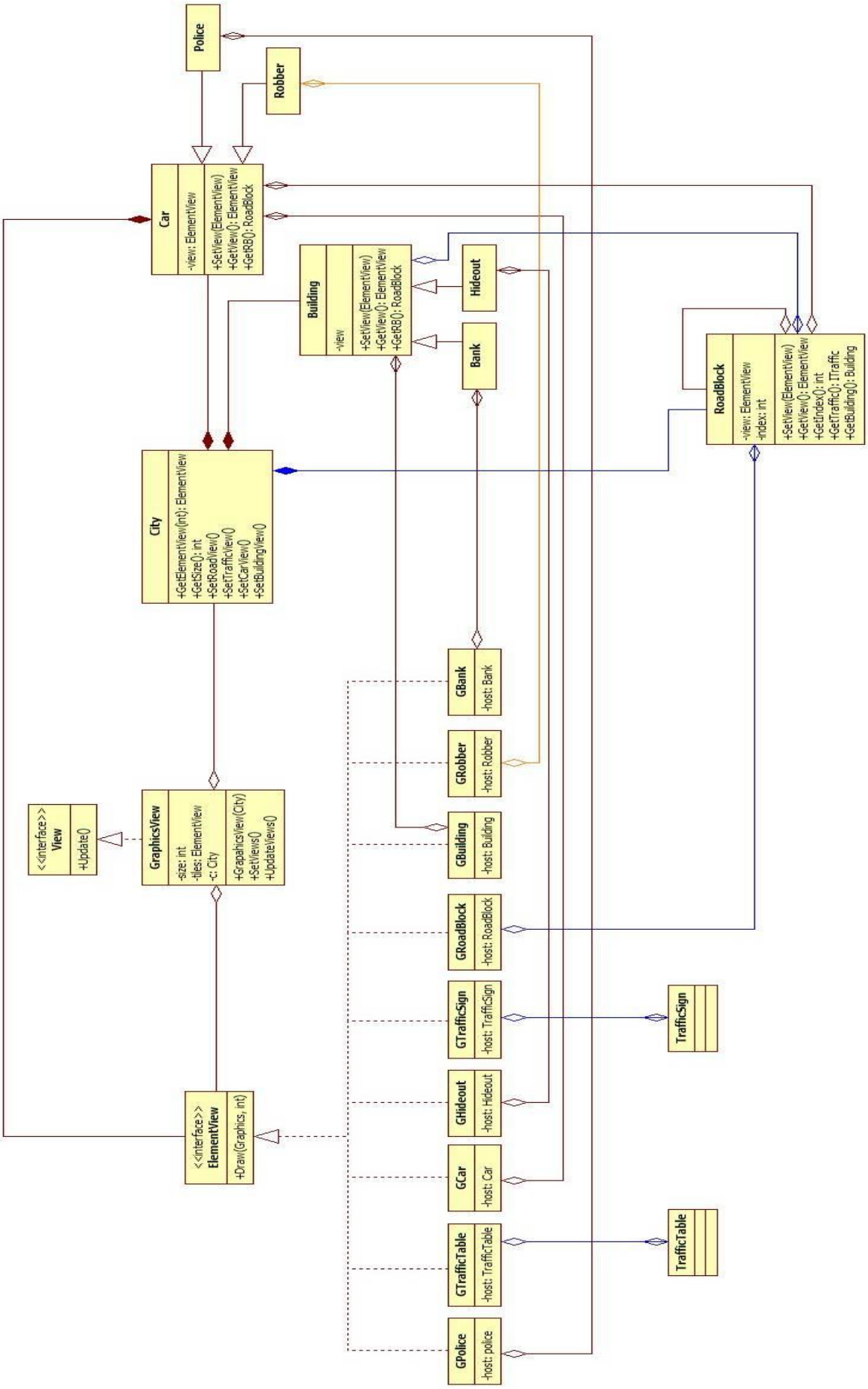
### 11.2.1 A felület működési elve

A program grafikus változatának elkészítésekor a Model-View-Controller tervezési mintát használjuk. Ezzel el tudjuk választani a program játék logikáját a grafikus megjelenítéstől, ezzel eleget téve a követelményeknek, valamint a program egyes részeinek karbantartása is könnyebbé válik. A grafikus felület mindig frissíteni fogja magát az aktuális játékállapothoz, viszont az egyes elemek saját magukat fogják kirajzolni, így a kevert alapelvet részesítettük előnyben.

### 11.2.2 A felület osztály-struktúrája

Sajnos utólag kiderült, hogy a prototípus osztályait a megfelelő grafikus felület implementálásához módosítani kell. A módosítás új tagváltozók és tagfüggvények hozzáadásában merül ki, a meglévőeket nem módosítottuk. Az alábbi osztálydiagramon a grafikus felülettel kiegészült programváz látható, a régi osztályok esetében csak a módosításokat tüntettük fel.

Ahol a vonalak keresztezik egymást, ott a félreértéseket elkerülvén különböző színekkel jelöltük az érintett vonalakat. Fekete-Fehér nyomtatásban sajnos ez nem látszik.


Police
Robber
Car
-view: ElementView
+SetView(ElementView)
+GetView(): ElementView
+GetRB(): RoadBlock
Building
-view
+SetView(ElementView)
+GetView(): ElementView
+GetRB(): RoadBlock
Hideout
Bank
RoadBlock
-view: ElementView
-index: int
+SetView(ElementView)
+GetView(): ElementView
+GetIndex(): int
+GetTraffic(): ITraffic
+GetBuilding(): Building
City
+GetElementView(int): ElementView
+GetSize(): int
+SetRoadView()
+SetTrafficView()
+SetCarView()
+SetBuildingView()
<<interface>>
View
+Update()
GraphicsView
-size: int
-tiles: ElementView
-c: City
+GraphicsView(City)
+SetViews()
+UpdateViews()
<<interface>>
ElementView
+Draw(Graphics, int)
GBank
-host: Bank
GRobber
-host: Robber
GBuilding
-host: Building
GRoadBlock
-host: RoadBlock
GTrafficSign
-host: TrafficSign
TrafficSign
GHideout
-host: Hideout
GCar
-host: Car
GTrafficTable
-host: TrafficTable
TrafficTable
GPolice
-host: police

## *11.3 A grafikus objektumok felsorolása*

### 11.3.1 View

- **Felelősség**

Interfészt nyújt a megjelenítő felületek számára.

- **Ősosztályok**

Nincs.

- **Interfészek**

Nincs.

- **Attribútumok**

Nincs.

- **Metódusok**
  - **+ void Update()**: A felület frissítését végzi el.

### 11.3.2 GraphicsView

- **Felelősség**

A játéklogika grafikus felületre való leképzésért felelős.

- **Ősosztályok**

Nincs.

- **Interfészek**

View

- **Attribútumok**
  - **- size: int**: A pálya méretét tárolja, a grafikus ablak méretét határozzuk meg belőle.
  - **-c: City**: Referenciát tárol a megjelenített városra.
  - **-tiles: ElementView:** Ide kérjük le az aktuálisan kirajzolandó objektum grafikus változatát.

- **Metódusok**
  - **+GraphicsView(city: City)**: Az osztály konstruktora, mely inicializálja a grafikus interfészt is egyben.
  - **+void SetViews()**: Beállítja az egyes objektumok megjelenését. Megadja, hogy ki hogyan rajzolja ki magát.
  - **+void UpdateViews()**: Frissíti az egyes objektumok (esetünkban csak a Car típusúak, mert csak ezek változnak) megjelenését. Pl. ha jött egy új autó akkor ad neki megjelenést is.

### 11.3.3 ElementView

- **Felelősség**

Interfészt nyújt a megjeleníthető objektumok kirajzolásához.

- **Ősosztályok**

Nincs.

- **Interfészek**

Nincs.

- **Attribútumok**

Nincs.

- **Metódusok**
  - **+ void Draw(Graphics, int)**: Kirajzolja az objektum magát az első paraméterben kapott felületre, mely a második paraméterben megadott méretű (utóbbiból számolja ki, hogy pontosan milyen koordinátákba rajzoljon).

### 11.3.4 GPolice, GTrafficTable, GCar, GRobber, GRoadBlock, GHideout, GBank, GBuilding, GTrafficSign

- **Megjegyzés:** A fentieket összevontuk, mert a leírásuk egyezik. Mindössze a Draw() metódus törzse tér el annyiban, hogy más típusú objektumokat hív. A mellékelt szekvencia diagramon ez látható is.

- **Felelősség**

Az egyes objektumok grafikus változatának kirajzolása a képernyőre.

- **Ősosztályok**

Nincs.

- **Interfészek**

ElementView

- **Attribútumok**
  - **- host**: Referenciát tárol arra az objektum példányra, akit kirajzol. Típusát szándékosan nem jelöltük az összevonás miatt, hiszen logikusan Car, Police, Building, Robber, Bank, RoadBlock, stb… referenciát tárol attól függően, hogy éppen melyik változatot nézzük (GCar, GPolice, GBuilding, stb…).

- **Metódusok**
  - **+void Draw(Graphics, int)**: Lásd: ElementView. A rajzolás után előfordulhat, hogy megvizsgáljuk van-e valamilyen tartalmazott objektum (pl. RoadBlock esetében egy az adott útdarabon álló autó, vagy jelzőlámpa, épület, stb…) és azokat is kirajzoljuk a megfelelő Draw() metódusok meghívásával.

### 11.3.5 Módosult: City

- **Új Metódusok**
  - **+ElementView GetElementView(int)**: Visszatér az adott sorszámú RoadBlock ElementView típusú tagváltozójával.
  - **+int GetSize()**: Visszaadja a pálya méretét.
  - **+void SetRoadView()**: A road tömb elemein végiglépkedve beállítja az összes RoadBlock megjelenését a SetView() metódus meghívásával.
  - **+void SetTrafficView()**: A traffic tömb elemein végiglépkedve beállítja az összes ITraffic interfészt implementáló objektum megjelenését a SetView() metódus meghívásával.
  - **+void SetCarView**: A car tömb elemein végiglépkedve beállítja az összes Car megjelenését a SetView() metódus meghívásával.
  - **+void SetBuildingView**: A building tömb elemein végiglépkedve beállítja az összes Building megjelenését a SetView() metódus meghívásával.

### 11.3.6 Módosult: Car

- **Új Tagváltozók**
  - **-ElementView view**: Az aktuális objektumra vonatkozó kirajzolásért felelős objektumot tárolja.

- **Új Metódusok**
  - **+ElementView GetView()**: A view tagváltozót adja vissza.
  - **+void SetView(ElementView):** Beállítja a view tagváltozót.
  - **+RoadBlock GetRB():** Visszaadja, hogy az útdarabot, amelyen az autó épp áll.

### 11.3.7 Módosult: Building

- **Új Tagváltozók**
  - **-ElementView view**: Az aktuális objektumra vonatkozó kirajzolásért felelős objektumot tárolja.

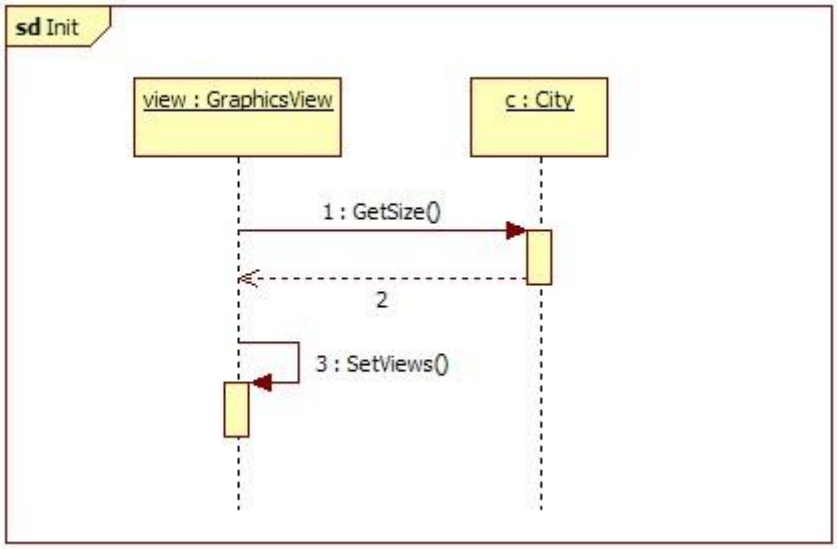
- **Új Metódusok**
  - **+ElementView GetView()**: A view tagváltozót adja vissza.
  - **+void SetView(ElementView):** Beállítja a view tagváltozót.
  - **+RoadBlock GetRB():** Visszaadja, hogy az útdarabot, amelyen az épület áll.

### 11.3.8 Módosult: RoadBlock

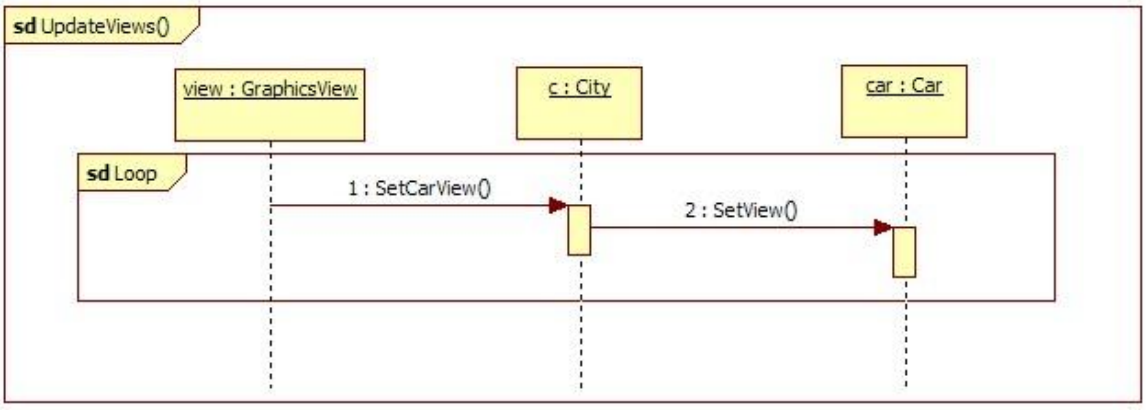
- **Új Tagváltozók**
  - **-ElementView view**: Az aktuális objektumra vonatkozó kirajzolásért felelős objektumot tárolja.
  - **-int index:** Az útdarab sorszámát tárolja.

- **Új Metódusok**
  - **+ElementView GetView()**: A view tagváltozót adja vissza.
  - **+void SetView(ElementView):** Beállítja a view tagváltozót.
  - **+int GetIndex():** Visszaadja az index tagváltozó értékét.
  - **+ITraffic GetTraffic():** Visszaadja a forgalomirányítót, ha van.
  - **+Building GetBuilding():** Visszaadja az épületet, ha van.

## *11.4 Kapcsolat az alkalmazói rendszerrel*



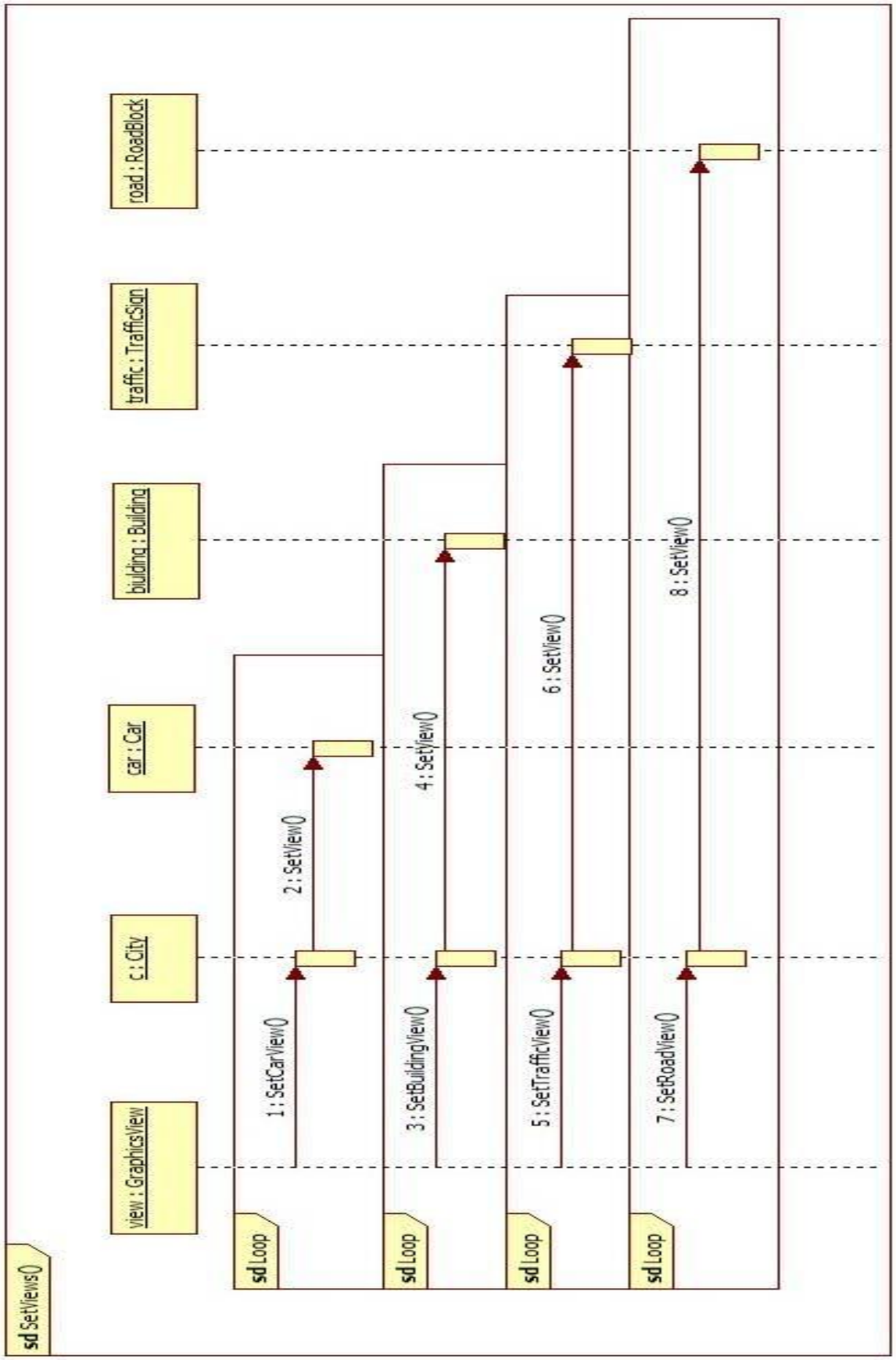

**1. ábra: Grafikus felület inicializálása**

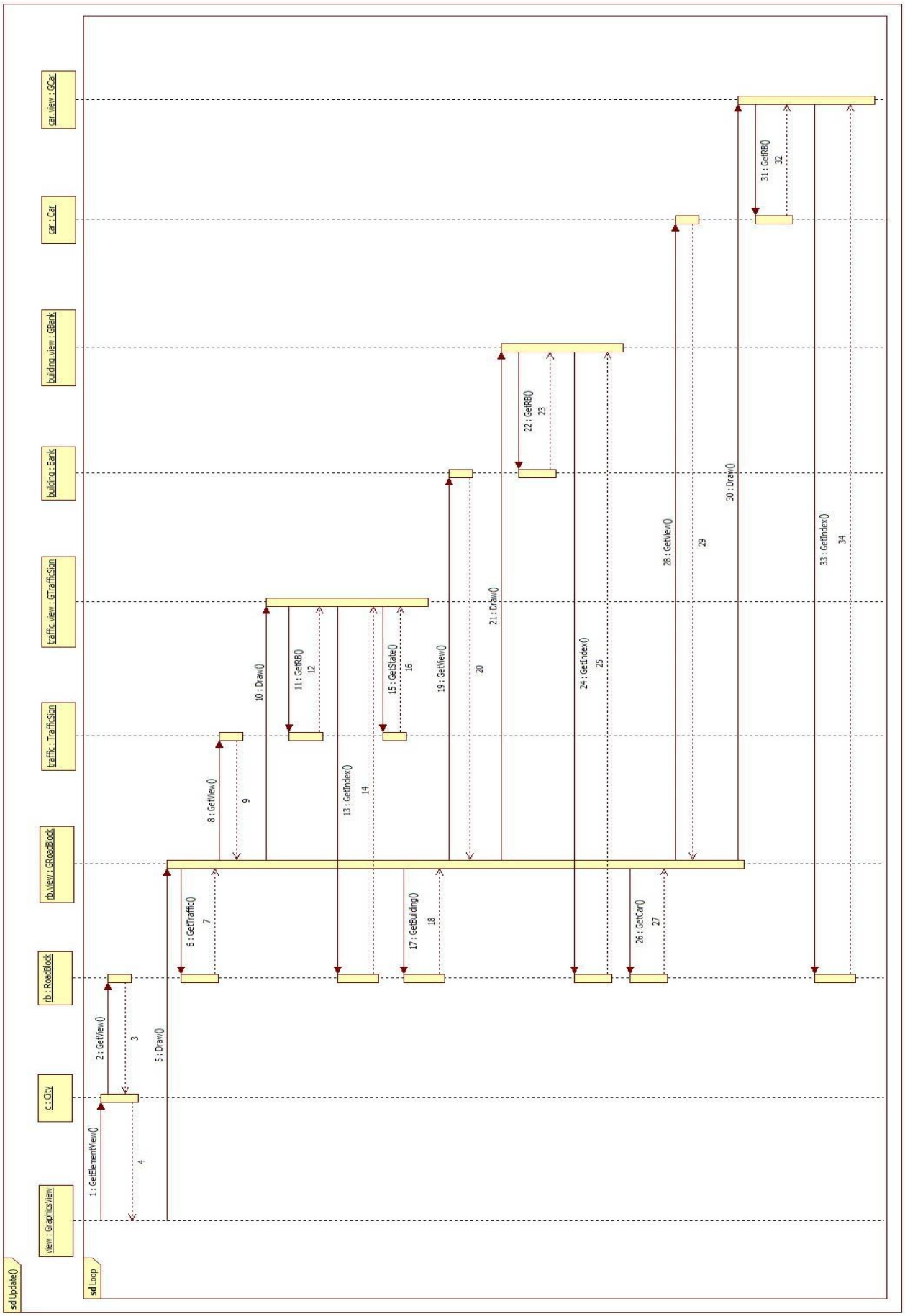



**2. ábra: Grafikus felület nézeteinek frissítése**

A következő két ábra sorrendben:

- Nézetek beállítása.
- Grafikus felület frissítése, újrarajzolása.


sd SetViews()
view : GraphicsView
c : City
car : Car
biulding : Building
traffic : TrafficSign
road : RoadBlock
sd Loop
1 : SetCarView()
2 : SetView()
3 : SetBuildingView()
4 : SetView()
5 : SetTrafficView()
6 : SetView()
7 : SetRoadView()
8 : SetView()


sd Update()
sd Loop
view : GraphicsView
c : City
rb : RoadBlock
rb view : GRoadBlock
traffic : TraffSign
traffic view : GTraffSign
building : Bank
building view : GBank
car : Car
car view : GCar
1 : GetElementView()
2 : GetView()
3
4
5 : Draw()
6 : GetTraffic()
7
8 : GetView()
9
10 : Draw()
11 : GetRB()
12
13 : GetIndex()
14
15 : GetState()
16
17 : GetBuilding()
18
19 : GetView()
20
21 : Draw()
22 : GetRB()
23
24 : GetIndex()
25
26 : GetCar()
27
28 : GetView()
29
30 : Draw()
31 : GetRB()
32
33 : GetIndex()
34

## *11.5 Napló*

| Kezdet | Időtartam | Résztvevők | Leírás |
|---|---|---|---|
| 2010.04.19. 20:00 | 1,5 óra | Rapp<br>Takács | Értekezlet. A grafikus felület elvi felépítésének megbeszélése. |
| 2010.04.21. 20:00 | 6 óra | Rapp | Tevékenység: 11.1 és 11.2 elkészítése, alapvető célok, elvárások, elvek rögzítése. |
| 2010.04.21. 20:00 | 4 óra | Takács | Tevékenység: Az osztálydiagram megtervezése, szükséges módosítási pontok felmérése. |
| 2010.04.22. 8:00 | 4 óra | Takács | Tevékenység: A végleges osztálydiagram elkészítése, objektumok leírása, módosítási pontok dokumentálása, szekvencia diagramok készítése. |
| 2010.04.22 12:00 | 1 óra | Takács | Tevékenység: A dokumentum formázása, helyesírási hibák javítása, ellenőrzés. |

# 13. Grafikus változat beadása

## *13.1 Fordítási és futtatási útmutató*

### 13.1.1 Fájllista

| Fájl neve | Méret | Keletkezés ideje | Tartalom |
|---|---|---|---|
| compile.bat | 317 bájt | 2010-05-06 | Fordító állomány. |
| run.bat | 110 bájt | 2010-05-06 | Futtató állomány. |
| bigmap2.txt | 6724 bájt | 2010-05-06 | Teszt térkép. |
| bigmap.txt | 289 bájt | 2010-05-06 | Teszt térkép. |
| mapfile.txt | 100 bájt | 2010-05-06 | JAVASOLT teszt térkép. |
| saved.txt | 581 bájt | 2010-05-06 | Mentett játékállás. |
| state1.txt | 134 bájt | 2010-05-06 | Játékállás. |
| state2.txt | 194 bájt | 2010-05-06 | Játékállás. |
| state3.txt | 178 bájt | 2010-05-06 | Játékállás. |
| state4.txt | 183 bájt | 2010-05-06 | Játékállás. |
| state5.txt | 287 bájt | 2010-05-06 | Játékállás. |
| Bank.java | 616 bájt | 2010-05-06 | Bank osztály. |
| Building.java | 1051 bájt | 2010-05-06 | Building osztály. |
| bunny.java | 2220 bájt | 2010-05-06 | bunny osztály. |
| Car.java | 5526 bájt | 2010-05-06 | Car osztály. |
| City.java | 9625 bájt | 2010-05-06 | City osztály. |
| ArrowDrawer.java | 1514 bájt | 2010-05-06 | ArrowDrawer osztály. |
| GameForm.java | 9250 bájt | 2010-05-06 | GameForm osztály. |
| GBank.java | 1042 bájt | 2010-05-06 | GBank osztály. |
| GBuilding.java | 1141 bájt | 2010-05-06 | GBuilding osztály. |
| GBunny.java | 934 bájt | 2010-05-06 | GBunny osztály. |
| GCar.java | 895 bájt | 2010-05-06 | GCar osztály. |
| GHideout.java | 952 bájt | 2010-05-06 | GHideout osztály. |
| GMessage.java | 1539 bájt | 2010-05-06 | GMessage osztály. |
| GPolice.java | 898 bájt | 2010-05-06 | GPolice osztály. |
| GRoadBlock.java | 2367 bájt | 2010-05-06 | GRoadBlock osztály. |
| GRobber.java | 874 bájt | 2010-05-06 | GRobber osztály. |
| GTextInput.java | 1712 bájt | 2010-05-06 | GTextInput osztály. |
| GTrafficSign.java | 1042 bájt | 2010-05-06 | GTrafficSign osztály. |
| GTrafficTable.java | 968 bájt | 2010-05-06 | GTrafficTable osztály. |
| Hideout.java | 525 bájt | 2010-05-06 | Hideout osztály. |
| ITraffic.java | 369 bájt | 2010-05-06 | ITraffic osztály. |
| MenuForm.java | 2117 bájt | 2010-05-06 | MenuForm osztály. |
| Police.java | 2444 bájt | 2010-05-06 | Police osztály. |
| Randomizer.java | 651 bájt | 2010-05-06 | Randomizer osztály. |
| RoadBlock.java | 2340 bájt | 2010-05-06 | RoadBlock osztály. |
| Robber.java | 2849 bájt | 2010-05-06 | Robber osztály. |
| TrafficSign.java | 1073 bájt | 2010-05-06 | TrafficSign osztály. |
| TrafficTable.java | 958 bájt | 2010-05-06 | TrafficTable osztály. |
| View.java | 299 bájt | 2010-05-06 | View osztály. |
| DrawingCoords.java | 3534 bájt | 2010-05-06 | DrawingCoords osztály. |

| | | | |
|---|---|---|---|
| ElementView.java | 342 bájt | 2010-05-06 | ElementView osztály. |
| game.java | 9 169 bájt | 2010-05-06 | game osztály. (belépési pont) |

### 13.1.2 Fordítás és telepítés

A HSZK-ban előzetesen informálódtunk a fordító, és a runtime environment elérési útvonalairól, ezért a **compile.bat** és a **run.bat** állományok abban a környezetben hiba mentesen fordíthatók. Amennyiben mégsem, úgy a fent említett két fájlban a megfelelő elérési útvonalakat a saját konfigurációnkhoz kell igazítani.

A továbbiakban feltételezzük, hogy a fordító és futtató fájlok az alábbi helyen találhatóak:

```
D:\Program Files\Java\jdk1.6.0_14\bin\
```

A szoftver forráskódját a **compile.bat** állománnyal fordíthatjuk le, ezzel elkészítve a futtatható kódokat tartalmazó fájlokat.

### 13.1.3 Futtatás

A programot az előző bekezdésekben említetthez hasonlóan, a **run.bat** állománnyal indíthatjuk el. Javasoljuk tesztelésre a **mapfile.txt** betöltését, ami a protó teszten is használt pálya.

## *13.2 Értékelés*

Az eredeti elképzelés szerint két ember foglalkozott volna a kód elkészítésével, és kettő a dokumentációval. A munka során nehéz volt az együttműködés, a csapat szinte teljesen szétesett, és szinte senki sem azt csinálta, amit kellett volna. Szinte minden beadásnál előfordult, hogy a specifikációnak homlokegyenest ellentmondott a kód. Probléma volt, hogy ritkán kerültek áttanulmányozásra a korábban beadott dokumentumok, így meglehetősen nagy inkonzisztencia lépett fel a projekt végére. Sajnos a most beadott programkód se teljesen konzisztens a korábbi dokumentumokkal, mivel a prototípus abban a formában, ahogy beadtuk szinte teljesen használhatatlan volt. A kód 60%-át újra kellett írni, így idő hiányában inkább a futtathatóságra törekedtünk, mintsem a helyes modellezésre vagy dokumentálásra. A program jelenlegi verziója talán így közelebb áll így a tervezetthez. A csapatban a kommunikáció szinte csak heti rendszerességű volt, a különböző tagok elérhetetlensége miatt. A grafikus verzió elkészítését, és a kód javítását egyetlen ember végezte el, illetve a csapat egy másik tagja ugyan dolgozott volna a terv szerint a kód kommentelésén, illetve dokumentálásán, de sajnos nem túl nagy sikerrel, így azt is a kódoló embernek kellett befejeznie. Idő hiányában viszont ez hagy némi kívánni valót maga után.

| Tag neve | Munka százalékban | Aláírás |
|---|---|---|
| Rapp | 19% | |
| Takács | 38% | |
| Boros | 22% | |
| Molnár | 22% | |

## *13.3 Napló*

| Kezdet | Időtartam | Résztvevők | Leírás |
|---|---|---|---|
| 2010.04.26. 18:00 | 0,5 óra | Rapp<br>Takács | Értekezlet. |
| 2010.05.01. 14:00 | 8 óra | Takács | Tevékenység: A prototípus hibáinak átnézése, kód javítása. |
| 2010.05.02. 14:00 | 2 óra | Takács | Tevékenység: A grafikus megjelenítő felszín elkészítése. |
| 2010.05.03. 16:00 | 8 óra | Takács | Tevékenység: A prototípus és a grafikus felszín összehangolása, megfelelő kódrészletek újraírása, eseménykezelők bekötése. |
| 2010.05.04. 8:00 | 14,5 óra | Takács | Tevékenység: A prototípus kód megfelelő részeinek teljes újraírása, a játéklogika működésre bírása a specifikációnak megfelelően. |
| 2010.05.05. 20:00 | 4,5 óra | Rapp | Tevékenység: Dokumentáció elkészítése. |
| 2010.05.06. 9:00 | 2 óra | Takács | Tevékenység: A dokumentáció javítása. |

# 14. Összefoglalás

## *14.1 Statisztika*

### 14.1.1 A projektre fordított összes munkaidő személyenként

| Személy | Szkeketon | Prototípus | Grafikus | Összesen |
|---|---|---|---|---|
| Rapp | 5,5 óra | 12 óra | 12,5 óra | 30 óra |
| Takács | 12,5 óra | 7,5 óra | 45,5 óra | 65,5 óra |
| Boros | 18 óra | 17 óra | 0 óra | 35 óra |
| Molnár | 17,5 óra | 20 óra | 0 óra | 37,5 óra |

### 14.1.2 Forrássorok száma fázisonként

| Szkeleton | Prototípus | Grafikus |
|---|---|---|
| 719 sor | 1049 sor + 166 sor | 3257 sor |

## *14.2 Értékelés*

### 14.2.1 Takács értékelése

A projekt során voltak nehézségeink az együttműködéssel, de végül mégis sikerült befejeznünk. Talán nem lett minden olyan, mint ahogy azt magunktól elvártuk volna, de a program működik, többé-kevésbé játszható, és a végleges verzió már nem rúgja fel a specifikációt sem annyira, mint a köztes bemutatódarabok. Induláskor arra törekedtünk, hogy a lehető legmagasabb pontszámokat kapjuk. Ez abban is látszott, hogy már hetekkel a szorgalmi időszak kezdete előtt elkezdtük tervezgetni, hogy ilyen-olyan extra feature-öket teszünk a játékba. A fejlesztést támogató eszközöket is idejekorán kiválasztottuk. Minden tagot megkérdeztem, hogy milyen jegyet szeretne kapni, mikor jelentkeztem a csapathoz, és természetesen mindenki masszívan 5-öst akart a végére 25-25-25-25%-os részvétellel. A projekt az első hetekben még jól haladt, tartottunk konzultációt, megbeszéltük kinek mi a dolga az adott hétre. Úgy tűnt ez egy jó csapat lesz, sok hozzáértő emberrel. Az első hónap után elfogyott a lelkesedés, és egyre romlott a leadott dokumentációk minősége. A projekt ezt eléggé megsínylette, egyáltalán nem lett olyan, amit bármelyikünk is büszkén bemutatna a barátainak. Én sajnálom, hogy így alakult, általában nem szoktam selejtes munkát kiadni a kezem alól, de most sikerült.

*Mit tanultak a projektből konkrétan és általában?*

A projektből megtanultam, hogyan kell irányítani egy csapatot, annak ellenére is, hogy nem én voltam a kijelölt csapatkapitány. A legtöbb adminisztrációs feladatot nekem kellett végezni, és nekem kellett a csapatot rábírni is a munkára, illetve én osztottam azt ki is egy idő után. Továbbá megtanultam, hogy egy szoftver fejlesztését elvállalva milyen követelményeket támasszak fel magammal szemben. Megtanultam azt az aranyszabályt is, hogy amit nem tudok adott időre elkészíteni, azt inkább el se vállaljam (ne írjam bele a specifikációba).

Sokkal rosszabb, ha az ügyfél kér valamit, majd vár, vár, vár, és még mindig csak vár. Egy idő után sajnos dühös lesz. ☺

*Mi volt a legnehezebb és a legkönnyebb?*

A legnehezebb feladat talán a modell megtervezése volt, lévén ilyet még sosem csináltunk ezelőtt, főleg ilyen rövid idő alatt. Olyat kellett (volna) tervezni, amit végig tudunk használni, és nem kell rajta hatszázszor módosítani, hogy egyáltalán életképes legyen.

*Összhangban állt-e az idő és a pontszám az elvégzendő feladatokkal? Ha nem, akkor hol okozott ez nehézséget?*

A pontszámok szerintem korrektek voltak, bár annak ellenére, hogy nem túl nagy súllyal számít bele a jegybe a grafikus verzió, mégis sok munka volt vele. De ez valószínűleg csak a majdnem használhatatlan prototípusunknak volt köszönhető. Az idő viszont kevés volt, lévén hétről hétre kellett dolgozni, a sok egyéb teendőnk mellett.

*Milyen változtatási javaslatuk van?*

A tárgyon úgy kellene változtatni, hogy ne kettő kreditért küzdjön az ember. Nem lenne nehéz a teljesítése, ha csak ezzel kellene foglalkozni a félév során, és nem kellene hat másik tárgyból beugrókra, kis-ZH-ra készülni, illetve házi feladatokat beadni. Még aki egyenesben van az is megszenved vele emiatt. Tudom, hogy 3-4 fő van egy csapatban, és máris 6-8 kreditre értékelődik a tárgy, de az egyes tagok nem részesülnek így megfelelő honoráriumban. Sőt, így ha valaki úgy dönt egy csapatban, hogy ő inkább megbukik, és cserbenhagyja a többieket, az sem olyan nagy tragédia neki, hiszen csak két kredit. Ezen ugyanakkor változtatni nemigen lehet, hiszen csak laboros tárgyra nem járhat több pont.

*Milyen feladatot ajánlanának a projektre?*

Nem tudom volt-e már, de jó ötletnek találnám egy egyszerű szerepjáték készítését projektre. Akár karakteres, akár felülnézetes, nyilakkal irányíthatót érdekes lenne megvalósítani. Olyanra gondolok, mint a régi „Legend of Zelda” legelső része. Persze a tárgy kedvéért bőségesen egyszerűsítve. ☺

### 14.2.2 Rapp értékelése

*Mit tanultak a projektből konkrétan és általában?*

A projekt rávilágított számomra arra, milyen összetett munka szükséges egy ilyen kis program megvalósításához is, illetve annak koordinálása mennyi egyeztetést igényel.

*Mi volt a legnehezebb és a legkönnyebb?*

Legnehezebb a programozási rész és a dokumentációk szinkronban tartása volt, legkönnyebb pedig ezen elbukni. Nehéz a munkaórákat arányban tartani. A kódolási rész több munka, és ha a dokumentálásba is bevonjuk a programozókat, nagyon elcsúsznak a munkaórák, ha meg nem, lehet, az elképzelt megoldás nem fog tetszeni a programozónak, és hiába dolgoztunk például a prototípus specifikáción.

*Összhangban állt-e az idő és a pontszám az elvégzendő feladatokkal? Ha nem, akkor hol okozott ez nehézséget?*

A tárgy elég sok munkát kíván meg a hallgatóktól, két kreditért talán túl sok ráfordítást igényel. Mivel a munkából levonni a színvonal rovására menne, a laborfoglalkozások pedig egységesen 2 kreditesek, megoldást nehéz lenne találni a problémára, de könnyen látható, hogy néhány 4-5 kredites tárgy kevesebb ráfordítást igényel. Más nevet kéne adni a tárgynak, olyant, amiben nem szerepel a "labor" szó.

*Milyen változtatási javaslatuk van?*

Jó lenne, ha pontosabb útmutatást kapnánk a konzulenstől, például a csapaton belüli kommunikációról. A konzultáción csak a beadandó követelményei lettek ismertetve, meg persze az ehhez szükséges ismeretek, de a munkafolyamat nem. Nem triviális megszervezni 4 ember munkáját, még ha év elején annak is tűnt.

*Milyen feladatot ajánlanának a projektre?*

-

### 14.2.3 Boros értékelése

*Mit tanultak a projektből konkrétan és általában?*

Bevezetett a projekttervezés és megvalósítás lépéseibe, betekintést adott, milyen lehet egy csapat tagjaként dolgozni.

*Mi volt a legnehezebb és a legkönnyebb?*

A legnehezebb megtalálni a közös hangot a csapattársakkal, a legkönnyebb a kódolás rész (skeleton, protó)

*Összhangban állt-e az idő és a pontszám az elvégzendő feladatokkal? Ha nem, akkor hol okozott ez nehézséget?*

Összhangban állt.

*Milyen változtatási javaslatuk van?*

-

*Milyen feladatot ajánlanának a projektre?*

-