Debreceni Egyetem
Informatikai Kar
Információ Technológia Tanszék

Valós idejű rendőr-ágens irányító a Robocar World Championshiphez

Témavezető:
dr. Bátfai Norbert
egyetemi adjunktus

Készítette:
Balkus Gergő Máté
programtervező informatikus hallgató

A dolgozat benyújtásához hozzájárulok.

dr. Bátfai Norbert

Debrecen
2016

Tartalomjegyzék

1. fejezet

Bevezetés

1.1. Robotautók

A korábbi években igen sok fejlődés tapasztalható az autógyártásban, elsősorban az autonóm autók kapcsán. Egyre több autógyártó és techóriás fejleszt robotautót, mint például a területen úttörő Google (a fejlesztett autó a 1.1. képen látható) vagy a Volkswagen cégcsoport, de a Tesla elektromos autó gyártó termékei is egyre inkább önjáróak. Ezek a trendek folyamatosan gyorsulni látszanak. A 2000-es és 2010-es éveket főként az autókba épített, vezetőt segítő elektronikai eszközök uralták (úgymint a gyalogosfigyelő rendszerek, sávelhagyásra figyelmeztető rendszerek, stb.), a 2020-as éveket várhatóan az autonóm autók széleskörű elterjedése fogja meghatározni. Ezek mellett a tisztán elektromos hajtású, valamint 2015-től a piacon is elérhető hidrogénhajtású autók is egyre szélesebb körben elérhetőek. Világosan látható, hogy az autóipar paradigmaváltás előtt áll. A vezetői élmény jelentősége és a fosszilis energiaforrásokon alapú hajtásláncok folyamatosan visszaszorulnak, helyettük a hosszú távon is fenntartható, környezetbarát, az utazást, mint pihentető, emberbarát eseményt vizionáló megoldások kerülnek a figyelem középpontjába.

A két terület találkozási pontja egyértelmű: egy teljesen ésszerű elképzelés, hogy a már most is fejlett elektronikával szerelt autóknak a város adjon útvonalat. Ugyanis a városnak rendelkezésére állhat minden olyan információ, amellyel az utazás optimálissá, gazdaságosabbá, gyorsabbá tehető. Ilyen információk az útlezárások, felújítások, elterelések, vagy például ritka események, balesetek.

1.1. ábra. A Google által fejlesztett robotautó. Forrás: [**?**].

1.2. Okos városok

2050-re a világ lakosságának 70%-a várhatóan városokban fog élni [**?**]. Ez a nagy mértékű urbanizáció új kihívások elé állítja a városi infrastruktúrát. Hogyan kezelhető egy ekkora méretű népesség? Milyen szolgáltatásokat nyújtson a városi adminisztráció annak érdekében, hogy élhető maradjon a város? Többek között ezekre a kérdésekre ad választ a Smart City kutatási terület, melynek megoldásai az utóbbi években kezdenek egyre inkább a mindennapok részévé válni. Kiemelten fontos problémakör lehet (és részben már most is az) a városi közlekedés. Egyre többen használják a városi infrastruktúrát, így a városi úthálózatot is. Milyen alkalmazást tud nyújtani a városi menedzsment, hogy a lakosok optimális módon tudjanak közlekedni? Erre a kérdésre adhat választ az okos közlekedés menedzsment (Smart traffic management).

Az Okos Város kutatási, fejlesztési terület jelentős szerepet fog játszani az következő évtizedekben. Előrejelzések szerint 2023-ig több mint 170 milliárd dollár beruházás fog megvalósulni világszerte [**?**]. A különböző informatikai megoldások egyre nagyobb számban vannak jelen a hétköznapokban már napjainkban is, mint például a VITAL [**?**], a FIWare [**?**] vagy az iCity [**?**]. Ezen felül egyre több kezdeményezés tesz kísérletet arra, hogy a városi lakosság hétköznapjait okosabbá és biztonságosabbá tegye ([**?**], [**?**], [**?**]). Emellett sok tanulmány foglalkozik a smart city fogalomkörével, illetve hogyan mérhető

egy város „okossága” ([**?**], [**?**], [**?**], [**?**]). A vészhelyzetek kezelése is kiemelt szerepet kap (például [**?**]).

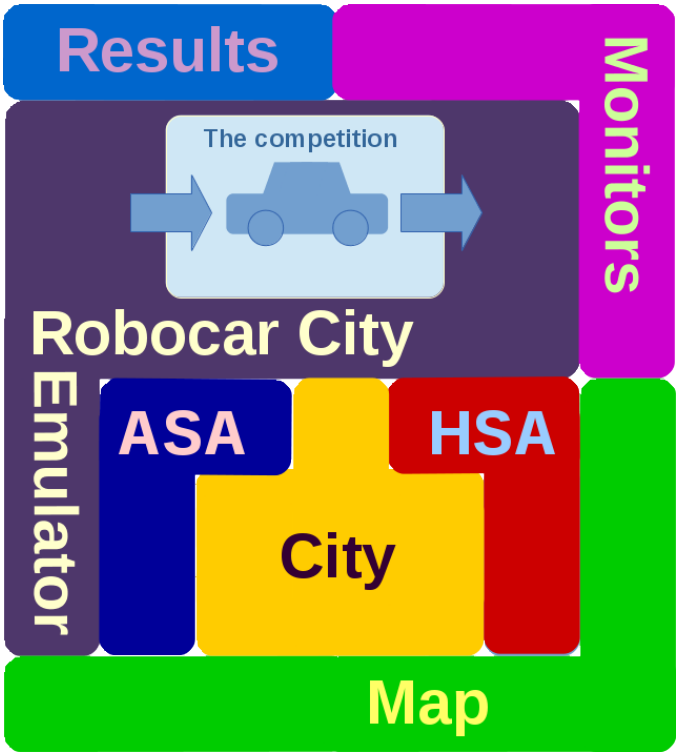
Látható, hogy a Smart City kutatási, fejlesztési terület lassan iparággá növi ki magát, mindenképpen komoly jövő előtt áll.

2. fejezet

A Robocar World Championship (OOCWC)

2.1. Ismerkedés az OOCWC rendszerrel

Az OOCWC rendszer célja, hogy az autonóm autók és az okos városok közötti összefüggéseket vizsgálja, kutatási valamint oktatási platform. A rendszer felépítését a 3.1. ábra szemlélteti.



2.1. ábra. Az OOCWC rendszer tetris terve. Forrás: [**?**].

Ez alapján a rendszer egyes elemei:

– Map – a szimuláció egy adott térképen értelmezett,

– City – a szimuláció működési egysége (City Operating Area),

– The competition – a verseny célja, illetve maga a verseny,

– ASA – automatikus adatgyűjtő rendszer (crowd-sensing),

– HSA – kézi adatgyűjtő rendszer (crowd-sourcing),

– Robocar City Emulator – forgalom emuláció,

– Results – a verseny eredményei, illetve kísérletek eredményei,

– Monitors – megjelenítők, vizualizáció.

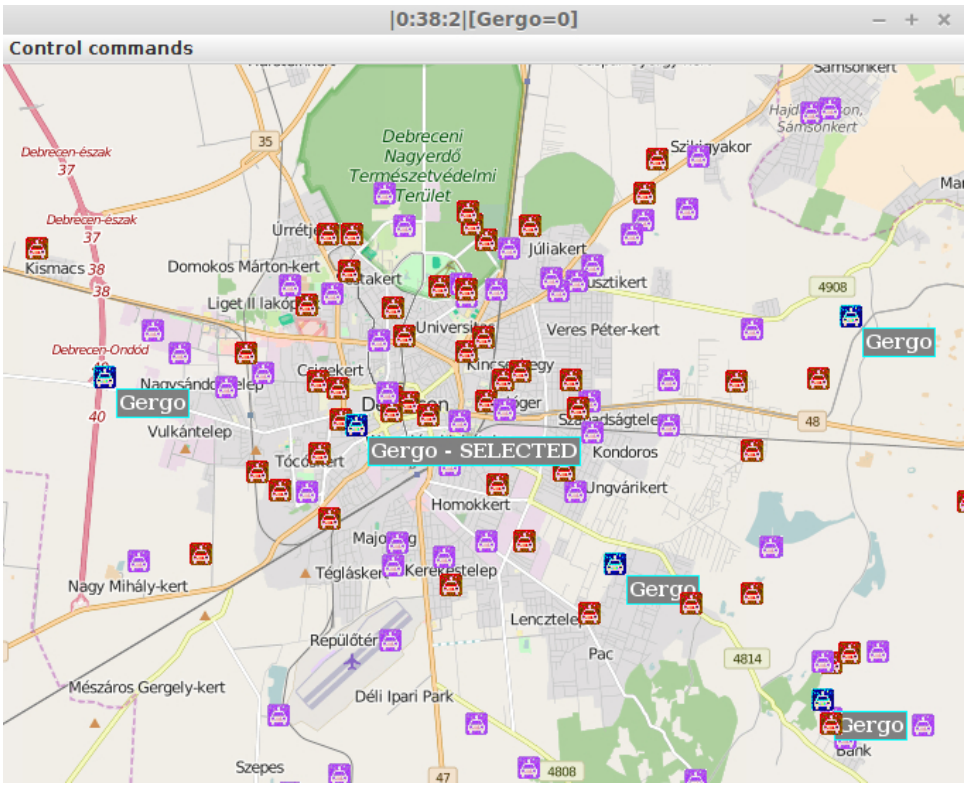
A rendszer többcélú. Egyrészt egy kutatási platformot kínál forgalomelemzésre, szimulációkra. A szimuláció az OpenStreetMap [**?**] térképein fut.

A rendszer másik célja, hogy egy verseny segítségével találjuk meg a lehető legjobb forgalomirányító algoritmust. A rendszer készítői már több sikeres versenyt is lebonyolítottak a Debreceni Egyetem Informatikai Karán [**?**]. Nem ritka, hogy egy-egy kutatási platform versenyezteti a felhasználókat. Erre talán az egyik legjobb példa a mesterséges intelligencia területén a RoboCup [**?**].

Kiemelendő, hogy a rendszer magas fokú alkalmazkodó képességgel rendelkezik. Más és más jellegű útvonaltervezés szükséges kormányzati közlekedéshez, vészhelyzetek esetében, stb. Az oktatási és kutatási oldal támogatására került kifejlesztésre a „Police Edition” (egy-egy pillanatkép ebből a változatból a 2.2. ábrán látható). A cél, hogy rendőr ágensekkel, melyeket a kutató által implementált irányító algoritmus vezérel, minél több gengszter ágenst kapjunk el. Egy ilyen implementációra láthatunk példát a [**?**] absztraktban.

Az OOCWC rendszerben a szimulációt a gyűjtött adatok alapján inicializáljuk. A rendszer jelenleg háromféle forgalmi egységet különböztet meg, routine cars, smart cars és guided cars. A szimuláció kezdőállapota a routine cars és smart cars elhelyezése a térképen a gyűjtött adatok alapján.

A rendszer „Police Edition” változatából készíthetnek egy saját fork-ot a hallgatók és az érdeklődő kutatók. A játék célja, hogy a rendőr ágensekkel minél több gengszter ágenst kapjunk el. A a bemutatott projekt (4. fejezet) is egy ilyen forknak tekinthető.



2.2. ábra. Pillanatkép a rendszer „Police Edition” változatának a projekt szerinti módosításával (lásd: 4. fejezet). A térkép az OSM egy részlete, Debrecen, Hajdú-Bihar Megye, Magyarország. A megjelenítést a JXMapViewer2 [**?**] biztosítja. A térképen a routine car rózsaszínnel, a gengszter ágensek pirossal (smart car), a rendőrágensek (guided car) kékkel jelölve. Az éppen kiálasztott rendőrágens melyet éppen irányítunk pedig el van látva a „SELECTED” felirattal. Forrás: [**?**]

3. fejezet

Felhasznált technlológiák

3.1. Java

A Java általános célú, objektumorientált programozási nyelv, amelyet a Sun Microsystems fejlesztett az 1990-es évek elejétől kezdve egészen 2009-ig, amikor a céget felvásárolta az Oracle. 2011-ben a Java 1.7-es verzióját, 2014-ben pedig az 1.8-as verzióját már az új tulajdonos gondozásában adták ki.

```
1 public class HelloVilag {
2     public static void main(String[] args) {
3         System.out.println("Hello Vilag!");
4     }
5 }
```

Forráskód 3.1. A klaszzikus „Hello World!” Javában

A nyelv 2016-ban is sikeres a szerver oldalon a servlet, a JSP és Enterprise JavaBeans, JDBC technológiákkal, integrációs lehetőségeivel, nyelvi eszközeivel, jvm nyelveivel és a nyílt forráskódú közösség tudására is építve.

A Java alkalmazásokat jellemzően bájtkód formátumra alakítják, de közvetlenül natív (gépi) kód is készíthető Java forráskódból. A bájtkód futtatása a Java virtuális géppel történik, ami vagy interpretálja a bájtkódot, vagy natív gépi kódot készít belőle, és azt futtatja az adott operációs rendszeren. Létezik közvetlenül Java bájtkódot futtató hardver is, az úgynevezett Java processzor.

A Java nyelv a szintaxisát főleg a C és a C++ (3.2. fejezet) nyelvektől örökölte, viszont sokkal egyszerűbb objektummodellel rendelkezik, mint a C++. Példaként szolgál az egyik fő különbsége a C++ nyelvtől: nincs az osztályok között többszörös öröklődés (ezt a célt az interfészek valósítják meg).

A JavaScript szintaxisa és neve hasonló ugyan a Java-éhoz, de a két nyelv nem áll olyan szoros rokonságban, mint azt ezekből a hasonlóságokból gondolhatnánk.

Négy fontos szempontot tartottak szem előtt, amikor a Javát kifejlesztették:

- objektumorientáltság;
- függetlenség az operációs rendszertől, amelyen fut (platformfüggetlenség);
- olyan kódokat és könyvtárakat tartalmazzon, amelyek elősegítik a hálózati programozást;
- távoli gépeken is képes legyen biztonságosan futni.

3.1.1. Története

Bár a nyelv neve kezdetben Oak (tölgyfa) volt, (James Gosling, a nyelv atyja nevezte így az irodája előtt növő tölgyfáról), később kiderült, hogy ilyen elnevezésű nyelv már létezik, ezért végül Java néven vált ismertté. A Java nyelvet kávézás közben találták ki, innen ered a kávéscsésze ikon [**?**].

3.1.2. Objektumorientáltság

A nyelv első tulajdonsága, az objektumorientáltság („OO”), a programozási stílusra és a nyelv struktúrájára utal. Az OO fontos szempontja, hogy a szoftvert „dolgok” (objektumok) alapján csoportosítja, nem az elvégzett feladatok a fő szempont. Ennek alapja, hogy az előbbi sokkal kevesebbet változik, mint az utóbbi, így az objektumok (az adatokat tartalmazó entitások) jobb alapot biztosítanak egy szoftverrendszer megtervezéséhez. A cél az volt, hogy nagy fejlesztési projekteket könnyebben lehessen kezelni, így csökken az elhibázott projektek száma.

3.1.3. Platformfüggetlenség

Ez a tulajdonság azt jelenti, hogy a Java-ban írt programok a legtöbb hardveren ugyanúgy futnak. Ezt úgy érik el, hogy a Java fordítóprogram a forráskódot csak egy úgynevezett Java bájtkódra fordítja le. Léteznek továbbá szabványos könyvtárcsomagok, amelyek, - közvetítve a kód és a gép között, - egységes funkcionalitásként teszik elérhetővé az illető hardver sajátságosságait (grafika, szálak és hálózat).

Vannak olyan Java fordítóprogramok, amelyek a forráskódot natív gépi kódra fordítják le, - ilyen például a GCJ, - ezzel valamelyest felgyorsítva annak futtatását. Cserébe a lefordított program elveszíti hordozhatóságát.

Egyes cégek, mint például a Microsoft, mégis platformfüggő sajátságokat adtak a nyelvhez, amire a Sun keményen reagált: beperelte a Microsoftot (az amerikai bíróság 20 millió dollár kártérítésre és a sajátos tulajdonságok visszavonására kötelezte a céget). Válaszként a Microsoft kihagyta a Java rendszert a jövőbeli termékekből és Windows-változatokból. Ez azt jelenti, hogy az Internet Explorer webböngésző alapváltozataiból hiányzik a Java. Így abban az olyan weboldalak, amelyek Java-t használnak, nem fognak helyesen megjelenni. A Windows-felhasználók e problémáját megoldva a Sun és más cégek ingyenesen letölthetővé tették a JVM rendszert azon Windows-változatok számára, amelyekből a virtuális gép hiányzik.

A hordozhatóság megvalósítása technikailag nagyon bonyolult. E közben a Java esetében is sok vita volt. Az „írd meg egyszer, futtasd bárhol" szlogenből „írd meg egyszer, keress hibát mindenhol" lett. 2016-ra a hordozhatóság nem okoz tovább problémát, mivel maga a Java is nyílit szabványokra épül, pl. openGL v. Open POSIX vagy az RFC-k, a Java minden jelentősebb platformon elérhető (Linux, Unix, Windows, más rendszerek pl. AS/400).

3.1.4. Swing

3.1.5. JXMapViewer

A JXMapViewer [**?**] egy nyílt forráskódú Java könyvtár, ami egy Swing (3.1.4. fejezet) JPanelt [**?**] szolgáltat, melynek feladata a térkép betöltése, és mutatása.

3.2. C++

A C++ egy általános célú, magas szintű programozási nyelv. Támogatja a procedurális, az objektumorientált és a generikus programozást is. Napjainkban szinte minden operációs rendszer alá létezik C++ fordító. A nyelv a C hatékonyságának megőrzése mellett törekszik a könnyebben megírható, karbantartható és újrahasznosítható kód írására, ez azonban sok kompromisszummal jár. Erre utal, hogy általánosan elterjedt a mid-level minősítése is, bár szigorú értelemben véve egyértelműen magas szintű.

A nyelv tervezésénél fontos szempont volt a C-vel való kompatibilitás, ezt oly mértékben sikerült megvalósítani, hogy minden szintaktikailag helyes C program egyben egy szintaktikailag helyes C++ program is.

```
1 #include <iostream>
2
3 int main()
4 {
5     std::cout << "Hello, vilag!" << std::endl;
6     return 0;
7 }
```

Forráskód 3.2. A klaszzikus „Hello World!” C++-ban

3.2.1. Története

Bjarne Stroustrup kezdte el a C++ programozási nyelv fejlesztését a C programozási nyelv kiterjesztéseként, más nyelvekből véve át megoldásokat, ötleteket. A nyelv első, nem kísérleti körülmények közt való használatára 1983-ban került sor, 1987-ben pedig nyilvánvalóvá vált, hogy a C++ szabványosítása elkerülhetetlen. Ez a folyamat 1991 júniusában kezdődött el, amikor az ISO szabványosítási kezdeményezés részévé vált. A C++ programozási nyelv szabványát 1998-ban hagyták jóvá ISO/IEC 14882:1998 [**?**] (C++98) néven, a 2003-as változat kódjelzése pedig a ISO/IEC 14882:2003 [**?**] (C++03) lett. Az ezt követő 2011-es változatot ISO/IEC 14882:2011 [**?**] (C++11) kódjelzéssel hagyták jóvá. A jelenlegi legfrissebb változata a 2014-es ISO/IEC 14882:2014 [**?**] (C++14) kódjelzésű verzió.

3.2.2. Boost library

A Boost a C++ programozási nyelvhez ad támogatást olyan könyvtárakkal, melyek különböző területek különböző problémáit segítenek megoldani. Többek között ilyen te-

3.1. ábra. A Boost logója. Forrás: [**?**].

rületek a lineáris algebra, a pseudorandom számok generálása, a többszálasítás, a képfeldolgozás, a reguláris kiejezések, illetve a unittesztelés.

Az OOCWC (2. fejezet) számára az egyik legfontosabb ilyen könyvtár a BGL (Boost Graph Library) [**?**], melynek segítségével épül fel az OSM [**?**] térkép alapján az OOCWC irányított gráfja.

3.3. Maven

Az Apache Maven (röviden Maven) egy szoftver, amelyet szoftverprojektek menedzselésére és a build folyamat automatizálására lehet használni. Jason van Zyl készítette 2002-ben. Funkcionalitásában hasonlít az Apache Ant eszközhöz. A projektet az Apache Software Foundation hosztolja, ahol korábban a Jakarta Projekt részeként működött.

A Maven bevezeti a POM, azaz a Projekt Objektummodell (angolul: Project Object Model) fogalmát. Egy POM egy buildelendő projektet ír le és annak függőségeit. Az egyes lépéseket céloknak, angolul goal-oknak nevezik. Vannak előre definiált célok a tipikus feladatokra, mint például a kód fordítása és csomagolása, de a felhasználónak lehetősége van saját célokat is definiálni a projektspecifikus lépések végrehajtására.

A Maven hálózatképes, tehát szükség esetén dinamikusan is le tud tölteni komponenseket. Repository névvel illetik a különböző hosztok fájlrendszereinek azon mappáit, ahol a letölthető komponensek találhatók. A Maven nem csak a repository-kból való letöltést támogatja, hanem a készült szoftvercsomag feltöltését is. Ezzel az automatizálható le- és feltöltési mechanizmussal a Maven de facto szabványt próbál teremteni, de elég lassan fogadja el a Java közösség.

A Maven-nek léteznek beépülő moduljai számos népszerű IDE-höz, melyek integrációt nyújtanak a Mavennal való az IDE build mechanizmusához, kódszerkesztő eszközeihez. Ezekkel lehetőség van az IDE-ből lefordítani a projekteket, és beállítani a classpath-t a kód kiegészítéshez ill. fordító hibák színezéséhez stb. Ilyen IDE például az Eclipse, illetve a NetBeans is.

3.4. Git

A Git egy nyílt forráskódú, elosztott verziókezelő szoftver, vagy másképpen egy szoftverforráskód-kezelő rendszer, amely a sebességre helyezi a hangsúlyt. A Gitet eredetileg Linus Torvalds fejlesztette ki a Linux kernel fejlesztéséhez. Minden Git munkamásolat egy teljes értékű repository teljes verziótörténettel és teljes revízió követési lehetőséggel, amely nem függ a hálózat elérésétől vagy központi szervertől.

Számos nagy volumenű projekt használja jelenleg a Gitet verziókezelő rendszerként; a legfontosabbak ezek közül: Linux-rendszermag, GNOME, Samba, X.org, Qt. A Git karbantartásának felügyeletét jelenleg Junio Hamano látja el.

A Git rendszerét a BitKeeper és a Monotone ihlette. Eredetileg alacsony szintű verziókövető rendszernek tervezték, azonban teljes értékű revíziókövető rendszerré szélesedett, amelyet már közvetlenül is lehet használni. Torvalds, aki jártas a nagy, szétosztott fejlesz-

tési projektekben és a fájlrendszerekben, és gyorsan akart működő rendszert fejleszteni, ezekre az elvekre alapozta a Gitet:

– A nemlineáris fejlesztés erős támogatása. A Git támogatja az ágak (branchek) készítését, összefésülésüket, és tartalmaz eszközöket a nem-lineáris fejlesztési történet ábrázolására.

– Elosztott fejlesztés. A Git minden fejlesztő helyi munkaváltozatában rendelkezésre bocsátja a teljes addigi fejlesztési történetet, és a változtatások másolása mindig két repository között történik. Ezeket a változtatásokat, mint külön ágakat importálják és összefésülhetőek, hasonlóan a helyben létrehozott fejlesztési branchokhoz.

– Nagy objektumok hatékony használata.

– Kriptográfiailag hitelesített történet. A Git-történet olyan módon tárolódik, hogy a szóban forgó revízió neve függ a hozzá vezető fejlesztési történettől. Ha egyszer publikálták, akkor már nem lehetséges észrevétlenül megváltoztatni egy régi revíziót. A struktúra hasonlatos a hash-fához, de hozzáadott adatokkal az egyes csomópontokon és leveleken.

– Eszközkészlet-szerű megoldás. A Git maga C-ben írt programok halmaza, számos shell-szkripttel megtűzdelve, amelyek összefűzik ezeket a C-ben írt programokat. Annak ellenére, hogy a szkriptek többségét újraírták C-ben a Microsoft Windowsra való portolás eredményeként, a struktúra megmaradt, és továbbra is könnyű az egyes komponenseket láncba rendezni az igényeknek megfelelően.

3.5. OpentStreetMap (OSM)

Az OpenStreetMap (OSM) csoportmunkán alapuló térképfejlesztés, melynek célja egy szabadon szerkeszthető és felhasználható térkép készítése az egész világról.

A térképek hordozható GPS eszközökből, légifotókból, szabadon használható, nemzeti kormányzati nyilvántartásokból és egyéb szabad forrásokból származó adatok, vagy egyszerű helyismeret alapján készültek. A térinformatikai adatbázis teljes tartalma Open Database License licenc alatt érhető el letöltésre [**?**] (a konkrétan megrajzolt térképrészletek általában változatos szabad licencek alatt attól függően, hogy azokat ki rajzolta az adatbázisból).

Az OpenStreetMapot olyan oldalak ihlették, mint a Wikipédia – a térkép-megjelenítésnél elérhető egy jellegzetes „Szerkesztés” fül, és a változtatások teljes története is rendelke-

zésre áll. A regisztrált felhasználók feltölthetnek GPS nyomvonalakat, és szerkeszthetik a vektoros adatokat adott szerkesztőeszközök használatával.

A `www.openstreetmap.org` [**?**] címen a regisztrációt követően bárki elkezdheti a szerkesztést. A GPS készülékkel vagy PDA-val személyesen gyűjtött adatokat GPX formátumban lehet a megadott oldalra feltölteni, majd a rögzített pontok alapján a terep-tárgyakat megrajzolni és beazonosítani. A projektben résztvevők térképező hétvégéket szoktak szervezni egy-egy kiemelt terület felmérésére. A projektnek már 2006-ban több magyar tagja volt, pedig akkor a szerkesztők száma még csak kb. 1000 volt.

4. fejezet

Az alkalmazás (Cop Controller)

Ennek a projektnek a célja az, hogy az OOCWC kvalifikációs részét valós-idejű irányítás válthassa ki, ezáltal érdekesebbé, és kézügyességtől (is) függővé válna a gengszterek elkapása.

4.1. Az alkalmazás funckiói

4.2. Az adat áramlása

5. fejezet

Jövőbeli munka

6. fejezet

Összefoglalás

Irodalomjegyzék

[] R. Besenczi, T. Katona, and M. Szilágyi. A Fork Implementation of the Police Edition of the OOCWC System. In *Cognitive Infocommunications (CogInfoCom), 2015 6th IEEE Conference on*, pages 163–164, Oct 2015.

[] Boost Graph Library. Boost graph library, 2001. URL `http://www.boost.org/doc/libs/1_60_0/libs/graph/doc/index.html`.

[] Boost logo. Boost logo, 2005. URL `http://boost.cvs.sourceforge.net/viewvc/boost/boost/boost.png?revision=1.2&view==markup`.

[] N. Bátfai. robocar-emulator, 2015. URL `https://github.com/nbatfai/robocar-emulator`.

[] N. Bátfai, R. Besenczi, A. Mamenyák, and M. Ispány. Traffic Simulation based on the Robocar World Championship Initiative. *Infocommunications Journal*, 7(3): 50–59, 2015.

[] C++1998. ISO/IEC 14882:1998, 1998. URL `http://www.iso.org/iso/catalogue_detail.htm?csnumber=25845`.

[] C++2003. ISO/IEC 14882:2003, 2003. URL `http://www.iso.org/iso/catalogue_detail.htm?csnumber=38110`.

[] C++2011. ISO/IEC 14882:2011, 2011. URL `http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=50372`.

[] C++2014. ISO/IEC 14882:2014, 2014. URL `http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=64029`.

[] R. Carli, M. Dotoli, R. Pellegrino, and L. Ranieri. Measuring and managing the smartness of cities: A framework for classifying performance indicators. In *Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference on*, pages 1288–1293. IEEE, 2013.

[] R. De Santis, A. Fasano, N. Mignolli, and A. Villa. Smart city: fact and fiction, 2014. URL `https://mpra.ub.uni-muenchen.de/54536/`.

[] C. Du and S. Zhu. Research on urban public safety emergency management early warning system based on technologies for the internet of things. *Procedia Engineering*, 45:748–754, 2012.

[] Fi-Ware. Fi-Ware, 2015. URL http://www.fiware.org/.

[] Future City. Future City Glasgow, 2015. URL http://futurecity.glasgow.gov.uk/.

[] Google Self-driving car image. Google Self-driving car image, 2015. URL http://www.newsnish.com/cars/latest-car-news/robocar--google-deploys-nextgen-cars/.

[] iCity. iCity Project, 2015. URL http://www.icityproject.eu/.

[] Java Swing JPanel. Java swing jpanel, 2016. URL https://docs.oracle.com/javase/7/docs/api/javax/swing/JPanel.html.

[] JavaName. Why is it called java?, 1991. URL http://mathbits.com/MathBits/Java/Introduction/BriefHistory.htm.

[] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, and E. Osawa. RoboCup: The Robot World Cup Initiative. In *Proceedings of the First International Conference on Autonomous Agents*, AGENTS '97, pages 340–347. ACM, 1997.

[] MyNeighbourhood. MyNeighbourhood Initiative, 2015. URL http://my--neighbourhood.eu/.

[] Navigant. Smart Cities Navigant Research, 2015. URL http://www.navigantresearch.com/research/smart-cities.

[] P. Neirotti, A. De Marco, A. C. Cagliano, G. Mangano, and F. Scorrano. Current trends in Smart City initiatives: Some stylised facts. *Cities*, 38:25–36, 2014.

[] OOCWC Competitions. Competition Result of Debrecen, 2015. URL http://justine.inf.unideb.hu/2015/Europe/Hungary/Debrecen/.

[] OSM. OpenStreetMap, 2016. URL https://www.openstreetmap.org/about.

[] R. Weait. Openstreetmap data license is odbl, 2012. URL https://blog.openstreetmap.org/2012/09/12/openstreetmap-data--license-is-odbl/.

[] SmartSantander. SmartSantander, 2015. URL http://www.smartsantander.eu/.

[] M. Steiger. jxmapviewer2, 2016. URL https://github.com/msteiger/jxmapviewer2.

[] United Nations. World Urbanization Prospects. The 2007 Revision, 2015. URL http://www.un.org/esa/population/publicationslwup2007/2007/WUP_Highlights_web.pdf.

[] VITAL. VITAL, 2015. URL http://www.vital−iot.eu/.

[] T. Yamauchi, M. Kutami, and T. Konishi-Nagano. Development of Quantitative Evaluation Method regarding Value and Environmental Impact of Cities. *Fujitsu Sci. Tech. J*, 50(2):112–120, 2014.

Függelék

COPCONTROLLER is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.
COPCONTROLLER is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
You should have received a copy of the GNU General Public License along with COPCONTROLLER. If not, see <http://www.gnu.org/licenses/>.
CSTS is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.
CSTS is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
You should have received a copy of the GNU General Public License along with CSTS. If not, see <http://www.gnu.org/licenses/>.

Köszönetnyilvánítás