

Search and Rescue in Urban Catastrophes

Busca e Salvamento em Catástrofes Urbanas

Graduation Project

(Projecto de Fim de Curso)

João Pedro Bugalho Certo
Nuno Miguel Ferreira Cordeiro



Universidade do Porto

Faculdade de Engenharia

FEUP

Faculdade de Engenharia da Universidade do Porto
Departamento de Engenharia Electrotécnica e de Computadores
Rua Roberto Frias, s/n, 4200-465 Porto, Portugal

December 2005

(Dezembro de 2005)

Search and Rescue in Urban Catastrophes

Busca e Salvamento em Catástrofes Urbanas

Graduation Project

(Projecto de Fim de Curso)

João Pedro Bugalho Certo
Nuno Miguel Ferreira Cordeiro

Students of Engenharia Electrotécnica e de Computadores in Faculdade de Engenharia da
Universidade do Porto

Graduation project of the 5th year of the Licenciatura em Eng.^a Electrotécnica e de Computadores from the
Faculdade de Engenharia da Universidade do Porto, supervised by Luís Paulo Reis¹ (main supervisor),
Nuno Lau² and Francisco Reinaldo³.

Faculdade de Engenharia da Universidade do Porto
Departamento de Engenharia Electrotécnica e de Computadores
Rua Roberto Frias, s/n, 4200-465 Porto, Portugal

December 2005

(Dezembro de 2005)

¹ Luís Paulo Reis – Professor at the University of Porto (FEUP)/LIACC (NIAD&R)

² Nuno Lau – Professor at the University of Aveiro / IEETA

³ Francisco Reinaldo – PhD Student – University of Porto (FEUP)/LIACC (NIAD&R)

Abstract

RoboCup Rescue Simulation is an international joint project [1] that promotes research on distributed artificial intelligence and intelligent robotics. The project was started in 1999 to solve rescue problems, by integrating disaster information, prediction, planning, and training, for rescue actions. Built upon the success of RoboCup Soccer project, it aims to offer a comprehensive urban disaster simulator, forums of technical discussions and competitive evaluation for researchers and practitioners [2]. Through the use of an extensive, and ever evolving, urban disaster simulator, heterogeneous team agents try to minimize damage to both people and property. Burning buildings, civilians trapped under debris, and blocked roads, are just some of the challenges simulation rescue teams (RTs) must overcome, coordinating as many as up to forty agents of six different types [3]. Every year, a RoboCup international competition is organized, where, in a competitive but constructive environment, researchers from all over the world can test their agents against other RTs. By comparing approaches and exchanging ideas, progress is made at an amazing rate, in great part due to the open source nature of the project. After each competition, the source code for every team is released, so that work may be done on top of the best ideas and implementations. Following on that concept, team FC Portugal entered the Rescue project, determined to contribute to the community. By creating tools and building agents capable of competing against the veteran, accomplished, teams, we aim to expand FC Portugal's presence into an entirely new domain. Focusing on these goals, a tool to comprehensively analyze team performance and strategy was developed and, on the agent level, work was erected on top of the code developed by SOS, a reputed RT from Iran. The constant evolutions in the simulator package imply that a lot of effort is required simply to adapt rescue agents to new environment rules. This project involves every challenge in the creation of a new team, from its inception to its current form.

Resumo

O *RoboCup Rescue Simulation* é um projecto internacional [1] que visa promover a pesquisa em inteligência artificial distribuída e robótica inteligente. O projecto foi iniciado em 1999 para resolver problemas de resgate em desastres, integrando informação de catástrofes, planeamento, e treino, para acções de salvamento. Partindo do sucesso do projecto *RoboCup Soccer* – futebol robótico, tem como objectivo oferecer um simulador urbano detalhado, fóruns de discussão técnica, e avaliação competitiva para investigadores e entusiastas [2]. Através do uso de um simulador de desastres urbanos extenso, e em constante evolução, equipas de agentes heterogéneos tentam minimizar danos em pessoas e propriedade. Edifícios em chamas, civis presos sob escombros, e estradas bloqueadas, são alguns dos desafios que as equipas de rescue (RTs - *Simulation Rescue Teams*) têm de ultrapassar, coordenando até quarenta agentes de seis tipos diferentes [3]. Todos os anos, uma competição internacional RoboCup é organizada, onde, num ambiente competitivo mas construtivo, investigadores de todo o mundo podem testar os seus agentes contra outras RTs. Comparando teorias e trocando ideias, o progresso dá-se a um ritmo fantástico, em grande parte devido à natureza *open source* (código aberto) do projecto. Depois de cada competição, o código fonte de todas as equipas é libertado, de modo a que o trabalho possa prosseguir sobre as melhores ideias e implementações. Seguindo este conceito, a equipa FC Portugal aderiu ao projecto Rescue, determinada a contribuir para a comunidade. Criando ferramentas e construindo agentes capazes de competir contra as equipas estabelecidas, tencionamos expandir a presença do FC Portugal para um domínio novo. Com estes objectivos em mente, foi desenvolvida uma ferramenta para analisar, detalhadamente, a performance e estratégia de equipas e, a nível de agentes, iniciou-se o trabalho sobre o código desenvolvido pelos SOS, uma respeitada RT do Irão. As constantes evoluções do simulador implicam que um grande esforço é necessário para a simples adaptação de agentes rescue às novas regras da competição. Este projecto inclui todos os desafios encontrados na criação de uma nova equipa, desde os seus primórdios até à presente forma.

Acknowledgments

The hardest part in giving credits, and trying to thank everyone for their help and contribution, is the certainty that someone will be forgotten. To that person we apologize in advance.

We wish to thank our main supervisor Luís Paulo Reis, for his great help, knowledge and experience. We have learned a lot from him and our lives have been largely affected by the opportunity to work with him.

Thanks to Francisco Reinaldo for all his assistance, his intelligence, and his always helpful smile and good mood. Above all, we would like to thank him for his friendship. Thank you, “Gordinho”.

Thanks to Nuno Lau for all his help, especially in Japan. He is, undoubtedly, an expert programmer. We learned much from you, but we believe there’d be a lot more to teach, given the time.

We would like to thank everyone we met in RoboCup 2005, Osaka, for being very nice, extremely polite, and always eager to help and exchange knowledge. You made our participation very pleasant. A particular thanks to Arash Rashid, from team SOS, for being such a great and helpful guy.

We would like to thank some of our teachers for being very understanding in these last few months, rescheduling exams and presentations when the need arose. Thank you André Restivo, Luís Paulo and Eurico Carrapatoso. We would be in a lot of trouble, if not for your good will.

From João:

I would like to thank my family for always supporting me through out my graduation. Especially my parents for their tolerance when things wouldn’t go as expected.

A big thanks for my sister, for always being there for me, namely for letting me skip the dish washing tasks, during busier academic days.

I want to acknowledge the friendship of the friends I left in Figueira da Foz, namely Cisco, Lili, Telmo and Marco whom, in spite the distance carry on bothering me.

Thanks to Nuno for the friendship, companionship, the car rides, and all other things that makes him such a good friend.

A special kiss to Rita that in spite of moving to morocco, stayed a close friend.

To all my other, apparently less important (or not), friends.

From Nuno:

I would like to thank my family, parents and brother for being all around great people. We've had our arguments but we always seem to be there for each other when the need arises. A special thank you for my mother, who is getting be an even better person as she ages, and all around easier to deal with. Thank you for always trying to be specially nice and helpful on those times when I look a little beat. As for my father, we always seem to understand each other. Thanks for your time and patience. Thank you for all those little, unimportant things that you know matter.

Also a part of the family is Fátima. Thank you for all these years of loyalty and dedication. I respect you a lot – I really do.

Thank you for my friends – all of them. You know who you are. I want to specially mention Nancy who will always be more than a friend, a sister, and João Certo, my best friend, colleague and coworker. One other great friend is the recently graduated Eng. João Aires, someone I know I can count on, be it for parties or for problems.

A special thanks to a recent friend, who keeps surprising me in ever positive ways. Thank you for our talks and games. You really are a special girl.

Scientific Publications and Certificates

Reinaldo, F., J. Certo, N. Cordeiro, L.P. Reis, R. Camacho, and N. Lau, "Applying Biological Paradigms to Emerge Behaviour in RoboCup Rescue Team," in *Progress in Artificial Intelligence: 12th Portuguese Conference on Artificial Intelligence, EPIA*, Springer-Verlag, LNCS, Vol. 3808, pp. 422 - 434, Covilha, Portugal, 5 - 8 December, 2005

Cordeiro, N., J. Certo, F. Reinaldo, L.P. Reis, R. Camacho, and N. Lau, *Rescue Technical Report I - Simulator System*, 2005, LIACC(NIAD&R) and IEETA

Certo, J., N. Cordeiro, F. Reinaldo, L.P. Reis, R. Camacho, and N. Lau, *Rescue Technical Report II - FCPx, A Tool for Agent Evaluation and Comparison*, 2005, LIACC(NIAD&R) and IEETA

Certo, J., N. Cordeiro, F. Reinaldo, L.P. Reis, and N. Lau, *FCPx: Evaluating Rescue Agents Performance*, in *Robotica 2006*. April, 2006: Guimarães, Portugal. (SUBMITTED)

Participation Certificate for RoboCup 2005, Osaka, Japan.

Contents

1. Introduction	1
1.1 Motivation	3
1.2 Objectives.....	4
1.2.1 Initial Objectives	4
1.2.2 Extended Objectives.....	5
1.3 Report Structure.....	5
2 Simulation System	6
2.1 Initial Considerations	6
2.2 Introduction to the Simulation System	7
2.3 System Structure	9
2.3.1 Kernel.....	10
2.3.2 GIS (<i>Geographical Information System</i>)	10
2.3.3 Current Simulators Modules.....	11
2.3.3.1 Collapse Simulator	11
2.3.3.2 Blockade Simulator	11
2.3.3.3 Traffic Simulator	12
2.3.3.4 Fire Simulator	12
2.3.3.5 Miscellaneous Simulator	13
2.3.4 Simulated World Objects	14
2.3.4.1 World	14
2.3.4.2 Roads and Crossings.....	14
2.3.4.3 Rivers and River Nodes	14
2.3.4.4 Buildings	14
2.3.4.5 Refuge.....	14
2.3.5 Agents.....	15
2.3.5.1 Civilians / Cars.....	15
2.3.5.2 Field Agents.....	16
2.3.5.3 Center Agents	18
2.3.6 Viewer.....	19
2.3.6.1 Morimoto Viewer.....	19

2.3.6.2	Freiburg's 3D viewer.....	20
2.3.7	System Configuration	21
2.4	Comunicação.....	Error! Bookmark not defined.
2.5	Final considerations	24
3	Team Implementation	26
3.1	Initial considerations.....	26
3.2	Code organization	26
3.2.1	World State	26
3.2.2	Communication Strategies	27
3.2.3	The State Machine.....	28
3.2.4	Field Agents.....	30
3.2.4.1	Common blocks	30
3.2.4.2	Ambulance Team.....	32
3.2.4.3	Fire Brigade	33
3.2.4.4	Police Force.....	36
3.2.5	Center agents	39
3.2.5.1	Common Blocks	39
3.2.5.2	Ambulance Center.....	39
3.2.5.3	Fire Station	39
3.2.5.4	Police Office.....	39
3.2.6	Libraries	40
3.2.6.1	The "librescue":.....	40
3.2.6.2	The "Libadk"	40
3.2.6.3	Agent Kind Libraries	40
3.2.7	The initial parameters.....	41
3.2.7.1	Generic	41
3.2.7.2	Map Specific	41
3.2.7.3	Agent Specific	41
3.3	Base code selection and development	41
3.4	Adapting the chosen code to new rules, towards RoboCup Osaka 2005.....	42
3.5	RoboCup Osaka 2005	43
3.6	Final Considerations	44
4	FCPx – A Tool for Agent Evaluation and Comparison	45
4.1	Initial Considerations	45

<i>CONTENTS</i>	<i>XI</i>
4.2 Tool Development.....	46
4.2.1 Design Options	46
4.2.2 Extracting data from log files	47
4.3 Analyzing the First Statistical Results	48
4.4 Team Comparison	57
4.5 Final Considerations	61
5 Conclusion and Future Work	62
Bibliography	64
Appendixes	65
1 Apêndice A: Instalação do Simulador	65
1.1 Processo de Instalação do Simulador	65
1.1.1 Problemas encontrados:.....	65
1.1.1.1 Primeira Instalação.....	65
1.1.1.2 Segunda Instalação:.....	69
1.1.1.3 Terceira Instalação:	69
1.2 Apêndice B: Execução do Simulador	69
1.2.1 Execução do Simulador	69
1.2.2 Visualizador em Tempo Real.....	70
1.2.3 Visualização de Logs.....	72
1.2.4 Execução dos Agentes.....	72
1.2.5 Scripts alterados e criados	73
1.2.6 Ficheiros de Configuração	74
1.3 Appendix C – FCPx spreadsheets parameters	75
Annex A	78

List of Figures

<i>FIGURE 1: SIMULATION SYSTEM FUNCTIONAL OUTLINE [8]</i>	9
<i>FIGURE 2: ROBOCUP RESCUE SIMULATION SYSTEM</i>	10
<i>FIGURE 3: BUILDINGS WITH GROWING COLLAPSE LEVELS (FROM 1 TO 4).</i>	11
<i>FIGURE 4: BLOCKED ROADS.</i>	12
<i>FIGURE 5: AN IMAGE FROM FREIBURG'S 3D VIEWER SHOWS A BUILDING ON FIRE.</i>	13
<i>FIGURE 6: CIVILIANS WITH EVOLVING STATUS.</i>	13
<i>FIGURE 7: THE REFUGE AS DEPICTED IN THE DEFAULT VIEWER (LEFT) AND FREIBURG'S 3D VIEWER (RIGHT).</i>	14
<i>FIGURE 8: CIVILIAN STATUS.</i>	15
<i>FIGURE 9: AMBULANCE AND BURIED CIVILIAN</i>	16
<i>FIGURE 10: FIRE BRIGADES EXTINGUISHING A FIRE</i>	17
<i>FIGURE 11: POLICE AGENTS CLEANING ROADS.</i>	17
<i>FIGURE 12: MORIMOTO VIEWER DISPLAYING A SIMULATION IN THE FOLIGNO MAP.</i>	19
<i>FIGURE 13: MORIMOTO VIEWER DISPLAYING A SIMULATION IN THE</i>	20
<i>FIGURE 14: AGENTS AND THEIR RESPECTIVE STATES AS REPRESENTED IN MORIMOTO VIEWER.</i>	20
<i>FIGURE 15: FREIBURG'S 3D VIEWER (ALSO DISPLAYING ITS 2D VIEW).</i>	21
<i>FIGURE 16: RADIO- COMMUNICATIONS.</i>	23
<i>FIGURE 17: OBJECT CLASS</i>	27
<i>FIGURE 18: AGENTS STATE-MACHINE IMPLEMENTATION</i>	29
<i>FIGURE 19 AMBULANCE TEAM STATES</i>	32
<i>FIGURE 20 FIRE BRIGADE STATES</i>	34
<i>FIGURE 21 POLICE FORCE STATES</i>	37

List of Tables

<i>TABLE 1: PARAMETER RANGES IN ROBOCUPRESCUE 2005 OSAKA</i>	21
<i>TABLE 2: FIRE BRIGADES - BUILDING RELATED DATA</i>	48
<i>TABLE 3 FIRE BRIGADES – CIVILIAN RELATED DATA</i>	49
<i>TABLE 4 FIRE BRIGADES – AGENT RELATED DATA</i>	50
<i>TABLE 5: POLICE FORCES – ROAD RELATED DATA</i>	51
<i>TABLE 6: POLICE FORCES – CIVILIAN RELATED DATA AND AGENT RELATED DATA</i>	52
<i>TABLE 7: AMBULANCE TEAMS – AGENT RELATED DATA</i>	53
<i>TABLE 8 : AMBULANCE TEAMS – CIVILIAN RELATED DATA</i>	53
<i>TABLE 9: CIVILIANS – CIVILIANS DISCOVERED</i>	54
<i>TABLE 10: CIVILIANS – CIVILIANS KILLED AND CIVILIANS RESCUED</i>	55
<i>TABLE 11: SIMULATION SUMMARY</i>	56
<i>TABLE 12: COMPARISON SUMMARY</i>	60

List of Charts

<i>CHART 1: A) FINAL SCORE EVALUATION; B) CIVILIAN CASUALTIES; C) EXPLORATION AND CIVILIAN DISCOVERY; D) NORMALIZED GRAPH OF EXPLORATION AND CIVILIAN DISCOVERY.</i>	<i>55</i>
<i>CHART 2: A) DISCOVERED CIVILIANS BY AGENT TYPE; B) DISCOVERED CIVILIANS (STACKED)</i>	<i>56</i>
<i>CHART 3: SCORE.</i>	<i>57</i>
<i>CHART 4: : A) DISCOVERED CIVILIANS; B) FC PORTUGAL - DISCOVERED CIVILIANS BY AGENT TYPE (STACKED); C) IMPOSSIBLES - DISCOVERED CIVILIANS BY AGENT TYPE (STACKED).....</i>	<i>57</i>
<i>CHART 5: A) CIVILIANS KILLED; B) FC PORTUGAL - CIVILIANS CASUALTIES (STACKED); C) IMPOSSIBLES - CIVILIANS CASUALTIES (STACKED).</i>	<i>58</i>
<i>CHART 6 : A) RESCUE AGENTS BURIED; B) RESCUE AGENTS KILLED.</i>	<i>58</i>
<i>CHART 7: A) BUILDINGS ON FIRE; B) BUILDING AREA DESTROYED.</i>	<i>59</i>
<i>CHART 8: A) ROADS BLOCKED; B) ROADS OBSTRUCTED.....</i>	<i>59</i>
<i>CHART 9: A) CIVILIANS RESCUED; B) CIVILIANS RESCUED ADJUSTED.</i>	<i>60</i>

Chapter 1

1. Introduction

The work described in this graduation project report is related to the study and application of coordination methods on multi-agent systems (MAS), more specifically on the RoboCupRescue domain.

It is focused on three main global areas: a) The RoboCup Rescue Simulation System as a test bed for cooperative models; b) Multi-Agent Systems (MAS) for heterogeneous team coordination and strategies; c) Software Engineering for the development, documentation and deployment of software artifacts.

The first necessary goal is the understanding of the Rescue Simulator as an environment for Multi-Agent Systems (MAS). The concept of Multi-Agent Systems evolved from Distributed Artificial Intelligence (DAI), Distributed Problem Solving (DPS) and Parallel AI (PAI). As for a single intelligent agent, it can be defined as a computational entity, usually called software, that if placed in some environment, perceives it through sensors, and is capable of performing autonomous action, in order to meet its design objectives using its actuators [4]. Being a high-level software abstraction, an agent provides a convenient and powerful way to describe a complex software entity, capable of acting autonomously in order to accomplish tasks. But unlike objects, which are defined in terms of *methods* and *attributes*, an agent is defined in terms of its behavior. The agent concept is a natural evolution of the combined research on artificial intelligence (AI) and network technology. AI is one of the most promising fields in computer science, but also one of the most complex. Since its original conception in the 1950s, its evolution has been fast, but filled with difficult to overcome obstacles.

In 1956 John McCarthy, the inventor of the Lisp programming language, used the term artificial intelligence (AI) for the first time. Much of the original focus of artificial intelligence research draws from an experimental approach to psychology, and emphasizes what may be called linguistic intelligence. Historically, there are two broad styles of AI research - the "neats" and "scruffies". The first one, also called classical, or symbolic, involves symbolic manipulation of abstract concepts and was greatly studied during the 60s and 70s in detriment of the later. Knowledge-based programs were created and, in 1972,

the Prolog language was introduced by Alain Colmerauer and Robert Kowalski. "Scruffy", or connectionist, approaches, of which artificial neural networks are the main example, was later pursued, during the 80s, when limitations to the "neat" approach of the time became apparent. During this decade, research in the area was very heavily funded by the Defense Advanced Research Projects Agency (DARPA) in the United States and by the fifth generation computer systems project in Japan. With large investments in a rapidly evolving field, great scientific advances were achieved. In the late 80s, due to the failure of work funded at the time to produce immediate results, there were major cutbacks in funding by government agencies, leading to a general downturn in activity in this field known as AI winter. During the 90s, goals were revised, and research largely moved to related areas such as machine learning, robotics, and computer vision. Still, vital achievements marked this decade, such as the Garry Kasparov defeat against Deep Blue, a chess playing computer, in 1997. In this same year, the first official RoboCup soccer match took place in Nagoya, Japan, featuring table-top matches with 40 teams of interacting robots, and over 5000 spectators[5, 6].

RoboCup was created as an international research and education initiative, aiming to foster artificial intelligence and robotics research, by providing a standard problem, where a wide range of technologies can be examined and integrated. With the objective of dynamise the evolution of AI, in particular MAS, the project was launched by Dr. Hiroaki Kitano, an AI researcher that became president and founder of the RoboCup Federation. It is currently divided in three major categories: soccer, rescue, and junior; each with its different leagues. Due to its prominence, soccer was the main motivator behind RoboCup. Being an extremely popular sport across most of the globe, it is able to attract people from different countries, cultures and religions into the same competition. Furthermore, it presents interesting scientific challenges, chiefly because it is a team game, mingling individual efforts with collective strategy. On the other hand, Junior was created to give younger students a chance to participate, promoting their interest in the field with several entertaining challenges. Last, but certainly not least, came Rescue.

The incredible success of the RoboCupSoccer international research and education initiative led the RoboCup Federation to create the RoboCupRescue project. With the intention of using the scientific knowledge developed, and apply it in a socially significant domain, two new RoboCup Leagues were created on this category: RoboCupRescue Robot League and RoboCupRescue Simulation League. A new league will be started in 2006 to bridge the gap between these two.

Search and rescue of victims in large-scale disasters are serious and very difficult tasks presenting several challenges from a scientific point of view. Unprepared cities can suffer tremendous consequences in a natural catastrophe as was reported in Kobe's earthquake [2] or, more recently, the south Asian tsunami. Every city needs an emergency plan, to reduce the loss of human life in a natural disaster. In recent years, staggering technological breakthroughs brought some science fiction dreams closer to us. The innovations in robot-

ics and artificial intelligence have opened doors, and allowed for a complete new use of rescue agents and emergency plans [2].

The RoboCupRescue Simulation League consists of a simulated city in which heterogeneous, intelligent, agents, acting in a dynamic environment, coordinate efforts to save people and property. Heterogeneous agents in a multi-agent system share a common goal, but have different abilities and specializations, adding further complexity and strategic options. These systems can manifest self-organization and complex behaviors even when the individual strategies of all their agents are simple. Furthermore, the simulation environment behaves in a dynamic way, depending mostly on internal variables describing its current state, and on the functions describing its evolution. It is mostly outside the control of any single agent and the ability to exert any significant change requires coordinated action towards defined goals.

The team-programmed agents are of six different types: Fire Brigades, Police Forces, Ambulance Teams and the respective centre agents. Fire Brigades are responsible for extinguishing fires; Police Forces open up blocked routes and Ambulance Teams unbury civilians trapped under debris. In order to obtain a good score, all these agents work together to explore the city, extinguish fires and unbury civilians, communicating through the centre agents, responsible for coordination and strategy.

1.1 Motivation

FC Portugal Rescue team is the result of a cooperation project between the Universities of Porto (LIACC/NIAD&R Lab) and Aveiro (IEETA Lab) in Portugal. The team was launched in January 2004, but only in January 2005 acquired the financial support to start as planned (FCT/POSI/EIA/63240/2004). This project adds to an established relationship between these two Portuguese Universities in the RoboCup competition, particularly on the simulation league and associated competitions: coach competition, simulation league presentation competition and simulation 3D competition [7]. The FC Portugal Rescue project aims to adapt its coordination methodologies, developed for the FC Portugal simulated soccer team, to the search and rescue scenario. It is also our goal is to introduce new strategies into this complex competition.

The field of Artificial Intelligence is a relatively new and a rapidly evolving one. Although the competition is stiff, the applications are numerous and promising. Despite being an area filled with long-term objectives and only a few short-term rewards, research conducted in FC Portugal Rescue project may be partially applied to several very useful social areas. A few examples of possible applications are: fire combat, mine clearance, land exploration, public transport coordination, satellite control, and cleanup of radioactive and toxic contamination, amongst other problems that imply team coordination [7]. Other fu-

ture applications profiting from the research in this area include self driving vehicles and robotic networks, in both industrial, domestic, and battlefield applications.

Future goals for FC Portugal include:

- Definition of new strategies and coordination mechanisms for teams of heterogeneous agents, performing complex tasks in dynamic domains;
- Creation of an agent architecture suitable for implementing agents with coordination, communication and learning capabilities;
- Application of learning methodologies for Rescue agents;
- Creation of a successful FC Portugal Rescue Team capable of achieving top places in RoboCupRescue international competitions.

1.2 Objectives

1.2.1 Initial Objectives

The ‘FC Portugal: Coordination of Heterogeneous Teams in Search and Rescue Scenarios’ main project objective was the creation of a new team capable of participating in the RoboCup Rescue 2005 Osaka competition. This project is intimately related to the FC Portugal’s rescue FCT Project and was the basis for the creation of FC Portugal’s rescue team.

This core objective can be divided into several smaller objectives:

- Study and documentation of the Rescue Simulator package;
- Familiarization with the competition, by watching and analyzing simulation logs from previous contests;
- Analysis of several source codes in order to choose a base code to develop our rescue team;
- Adaptation of the code to the new 2005 competition rules;
- Development of strategies for agent coordination;
- Participation in the RoboCup Rescue 2005 competition in Osaka, Japan;
- Prepare the team for future developments by new members.

1.2.2 Extended Objectives

After the successful participation in RoboCup 2005 the Project was extended⁴. It quickly became apparent that some future objectives of the FC Portugal Rescue team required a tool, capable of retrieving detailed information from a simulation, and saving it an organized, accessible manner, enabling future analysis. This would allow analysis and improvement of different team strategies, and the development of learning modules, feeding from the collected data.

The extended project objective became the creation of this tool, which could later be released to the rescue community.

1.3 Report Structure

The rest of this report is organized as follows. The next Chapter presents an overview of the Simulation System, including its Internal Structure and Communication System. In Chapter 3 we discuss the agent implementation, including the base code selection and development. Chapter 4 introduces FCPx, a tool developed for the analysis and comparison of rescue teams, presenting some results. Finally, in Chapter 5, we conclude this paper, present some ideas, and suggest future developments.

⁴ with an FCT scholarship.

Chapter 2

2 Simulation System

2.1 Initial Considerations

The Rescue simulator is a simplified model of a city - only data relevant to the disaster situation is reproduced, while most detail is neglected. There are two main reasons for these simplifications. Firstly, due to computational limitations, it is impossible to simulate reality to a high level of detail. Furthermore, the real world is full of distracting and obscuring detail, and science generally progresses by focusing on artificially simple models of reality. As an example, the existence of air friction delayed a correct interpretation of gravitation for several centuries. Without this factor a feather would take as much time as a brick to hit the floor, if dropped from the same height - simplifying the comprehension of these physics. In 1970 Marvin Minsky and Seymour Papert, of the MIT AI Laboratory, proposed that AI research should likewise focus on developing programs capable of intelligent behaviour in artificially simple situations. This allows both the researcher and the intelligent agents to focus on the important factors without being distracted or confused by minor details.

The simulator package uses a modular approach, allowing different parts to be updated independently. Every year new features are combined with the existing ones, improving the simulation and adding complexity to the environment. The most recent large change was in the fire simulator, which was completely overhauled, requiring some changes in the agents' strategy.

The infrastructure competition was created to encourage the enhancement and addition of simulator modules. Some of the improvements presented in this contest are then added to the main package, enriching the existing city model.

2.2 Introduction to the Simulation System

The action takes place in a simulated city, where a natural disaster (earthquake) has just taken place. This city is dynamically modeled by the following equations,

$$e(t) = \mathbf{f}(\mathbf{x}(t), u(t), t)$$

$$\mathbf{x}(t + \Delta t) = \mathbf{g}(\mathbf{x}(t), e(t))$$

in which:

$e(t)$ represents the effects that create change in the city, calculated by \mathbf{f} .

t represents the current time instant.

\mathbf{f} is the function describing how $\mathbf{x}(t)$ e $u(t)$ affect the simulated world, changing its status.

$\mathbf{x}(t)$ is the status variable. It represents the disaster situation in instant t . Every variable such as the strength of fire or the speed of cars is saved in the form of a vector. The size of this vector proportionally increases with the size of the simulated area.

$u(t)$ is the input vector in instant t , representing external effects like water sprayed by fire brigades and debris removed by Police Forces.

Δt is the time step used to forward the simulation discretely.

Finally, \mathbf{g} is the function that describes the values of statuses $\mathbf{x}(t)$ at the instant immediately after t , i.e. instant $t + \Delta t$ [8].

At $t=0$, $\mathbf{x}(t)$ represents the initial situation.

From $\mathbf{x}(0)$ we can obtain the following values:

Sint: total HP of all agents at start,

Bint: total undamaged area at start,

At any time step we can obtain:

P: number of living agents,

S: remaining total HP of all agents,

B: total undamaged area of buildings.

The simulation score V is calculated using the following equation:

$$V = \left(P + \frac{S}{S_{int}} \right) * \sqrt{\frac{B}{B_{int}}}$$

Evaluation rule: given any simulation, the higher the V value, the better the rescue operation.[9]

Note that at the beginning of the simulation ($t=0$):

$$V = (P+1)$$

As the simulation proceeds, more buildings are damaged and people hurt, causing the score to drop till its final value at $t=300$.

As such the initial value of V is the maximum possible score for a given simulation.

2.3 System Structure

A schematic representation of the simulation system can be seen on *Figure 1*.

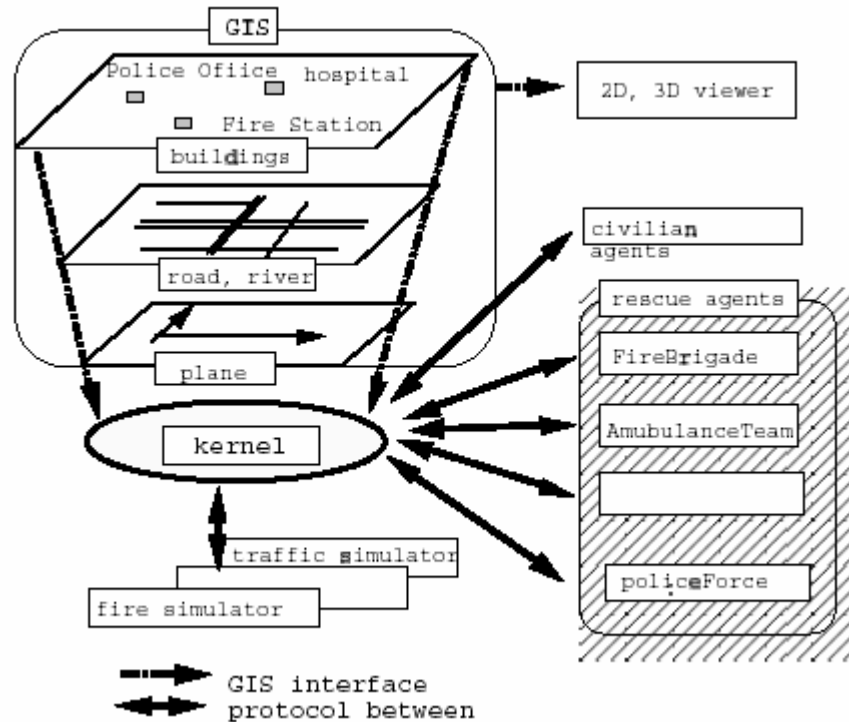


Figure 1: Simulation System functional outline [8]

This structure allows a relatively autonomous development of the different simulator modules, since once the communication protocol is defined, the modules are mostly independent.

The communication between modules takes place by message exchange. The organization depicted on Figure 1 is a flexible one and, due to recent evolutions, it may evolve into something slightly different since, for example, the new fire simulator is now able to communicate directly with the collapse simulator – although the collapse simulator module is not yet ready for this. A description of the different modules and their representation on the system is given below.

2.3.1 Kernel

The kernel is the central processing unit of the system, controlling the simulation process and facilitating information exchange between modules. It is responsible for establishing and maintaining communication with the Geographic Information System (GIS), the Simulators (Collapse, Fire, Traffic, etc.), the Viewer, and the Agents; as is depicted on Figure 2.

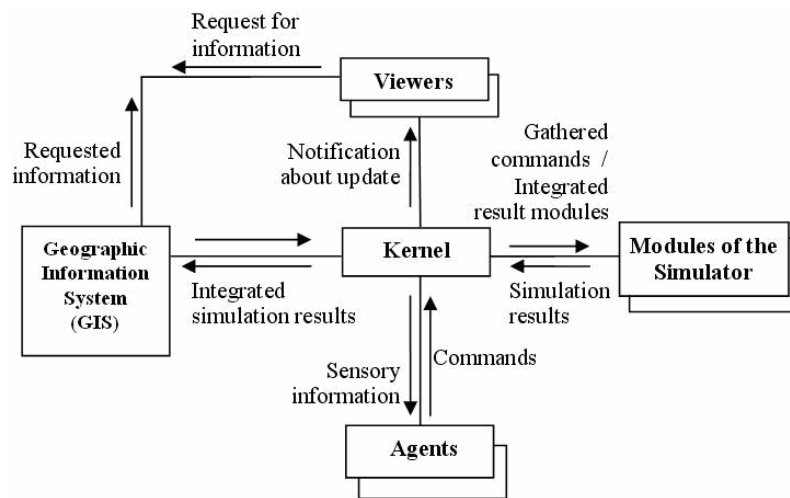


Figure 2: RoboCup Rescue Simulation System

When the program starts, the Kernel receives from the GIS module the initial configuration of the simulated world. At every step of the simulation, the Kernel sends sensory information to all agents and receives their action commands. Information is sent and received from the modules as necessary and, for each data exchange, the command and information validity is verified [10].

2.3.2 GIS (*Geographical Information System*)

The GIS module is responsible for the initial configuration of the simulated world. This is composed by the location and properties of buildings, roads, nodes, refuges, agent centers, civilians, ambulances, fire brigades, police forces and initial fires. It also records the simulation progress into a log file, enabling a detailed offline analysis. Additionally, this module is responsible for feeding data to the viewer.

2.3.3 Current Simulators Modules

As the project evolves, simulators are added or improved, deepening the complexity and adding realism to the simulation

2.3.3.1 Collapse Simulator

This module acts on the physical state of buildings after the earthquake. On a large scale disaster like the one RoboCupRescue aims to emulate, around 80% to 90% of households are at least partially collapsed, shortly after the calamity. Currently, this simulator is triggered only once, at the beginning of the simulation. In the images displayed by the default viewer⁵, like the one displayed in Figure 3, the more damaged a building is, the darker it looks – as can be seen by the buildings numbered 1 to 4.

This is one of the modules schedule to be revised shortly, as new features – such as earthquake aftershocks – are expected.

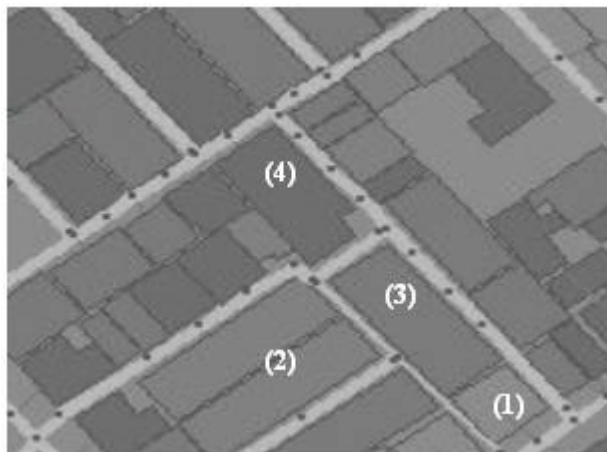


Figure 3: Buildings with growing collapse levels (from 1 to 4).

2.3.3.2 Blockade Simulator

This is the module responsible for defining the state of road obstructions. After the earthquake, a large part of the roads gets blocked, hindering traffic flow. These obstructions may have different causes such as crowds, debris from buildings and traffic accidents. Blocked roads can only be cleared by Police Force agents, this way allowing other agents to freely move through.

⁵ The default viewer, developed by Morimoto can be found at <http://ne.cs.uec.ac.jp/~morimoto/rescue/viewer/index.html>

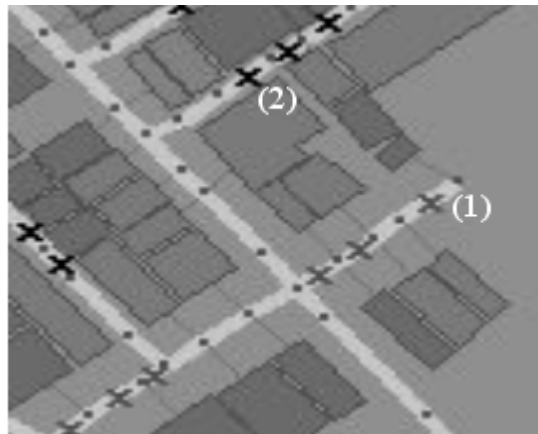


Figure 4: Blocked Roads.

As seen on Figure 4, the default viewer represents road blocks with crosses, which can be either gray or black. A gray cross means that the road is only partially blocked while a black one marks a block which can not be crossed. Although the viewer simplifies this feature it is a fairly complex one; blocks are defined and simulated in millimeters, even though some do not show up in the viewer. These affect the speed at which agents can travel through roads and the number of usable lanes, impacting on traffic jams.

2.3.3.3 Traffic Simulator

Every agent's movement in the world, including civilians, is modelled by this component, which defines the pace allowed on every road section. Width, number of agents present, and the level of "blockness", are some of the factors influencing maximum speed on a street. Usually, a road which is over 50% blocked is not traversable.

2.3.3.4 Fire Simulator

This module simulates the spread of fire in the city. It is currently one of the most evolved components of the simulator package. Right after the earthquake, some buildings ignite and start radiating heat to nearby structures. This component is responsible for the physical simulation of combustion and heat spread. This is done resorting to an intelligent model in which the temperature of a building is, on the one hand increased, either by the its own combustion or by the radiation waves from neighbouring buildings, and on the other hand decreased, due to the evaporation of water pumped by Fire brigade agents. In the simplified combustion model used, the critical factors are ignition temperature and combustible material (buildings), with the supply of oxygen being disregarded. When a building's temperature rises above its material's ignition temperature it bursts into flames, as seen on Figure 5. In contrast, when the temperature drops below its ignition point, its fire is extinguished[11].



Figure 5: An image from Freiburg's 3D viewer shows a building on fire.

2.3.3.5 Miscellaneous Simulator

The agent's status is modelled by this simulator. When an agent is inside a burning building or trapped under debris its health is affected and starts decreasing. This is the module responsible for controlling the agent's properties in these situations. As a simple example, a large value for the agent's property buriedness describes its state as trapped under debris. When Ambulances use their rescue ability, this value is progressively reduced until the agent is free.

In Figure 6, the status of the depicted civilians has changed. In the default viewer, civilians are represented by green circles that get darker as their health degrades. The civilian in the bottom changed from healthy to deceased in just a few cycles, because the buildings he was trapped in burned to the ground.

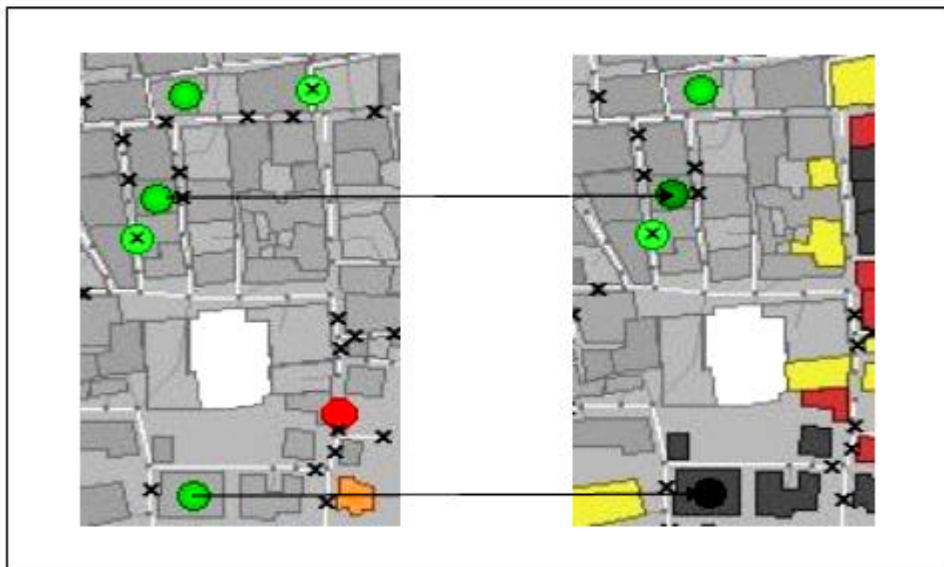


Figure 6: Civilians with evolving status.

2.3.4 Simulated World Objects

2.3.4.1 World

The parameters about the simulated world are stored in this object. These parameters include location, time, and wind (strength and direction).

2.3.4.2 Roads and Crossings

The road network is represented by a graph, in which edges are roads and nodes are crossings.

2.3.4.3 Rivers and River Nodes

Like with roads, the river network is also represented by a graph. Although possible, there are no current implementations of these objects.

2.3.4.4 Buildings

Buildings have parameters like position, number of floors, primary construction material, fieriness, brokenness and total area, amongst many others.

2.3.4.5 Refuge

The refuge (see Figure 7) is a special kind of building where civilians take shelter and fire brigades can fill their water tanks. It has mostly the same properties as any other building, but it is indestructible.

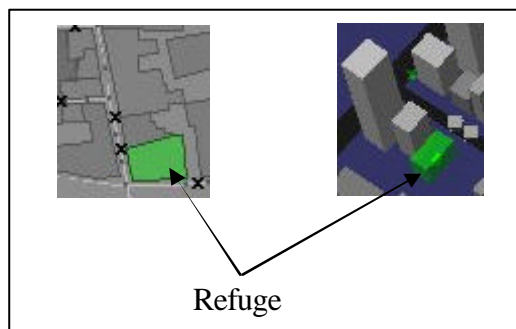


Figure 7: The refuge as depicted in the default viewer (left) and Freiburg's 3D viewer (right).

2.3.5 Agents

Whereas every team is responsible for the creation of their own rescue agents, civilian agents are directly controlled by the simulator. Some new types of agents were recently proposed and their possible implementation is being discussed. The number of each type of agent is defined for every simulation.

2.3.5.1 Civilians / Cars

Civilians

Civilians represent people in the system, but due to current computational limitations, every civilian represents a family or similar aggregate. They possess parameters such as location, health and buriedness, amongst others [8]. Figure 8 shows how the default viewer represents different health values for civilians. Civilians buried under debris can be rescued by ambulances.

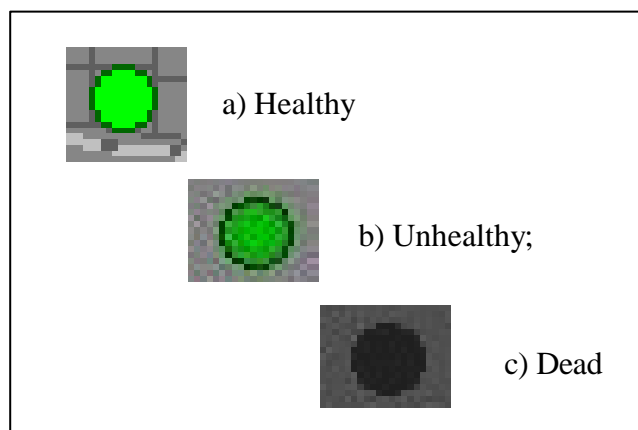


Figure 8: Civilian status.

Cars

The car is a purely conceptual agent, created because civilian agents act like motorized vehicles when moving around the city. From every point of view cars and civilians are only one type of agent.

2.3.5.2 Field Agents

These types of agents are developed by rescue teams. They all have the same properties as a civilian, plus some type specific ones. Field agent types are described bellow. In the default viewer agents are represented by colored circles. Police and Ambulance activity is manifested by a surrounding halo, while fire brigades shoot blue water streams (See Figure 14 in Section 2.3.6).

Ambulance Team agent

This is the agent responsible for rescuing civilians from collapsed buildings. It bears all civilian properties plus the ones specific to its function, namely: unbury civilians, load them to the ambulance and unload them in refuges. Figure 9 shows two ambulances and one of them is rescuing a civilian.

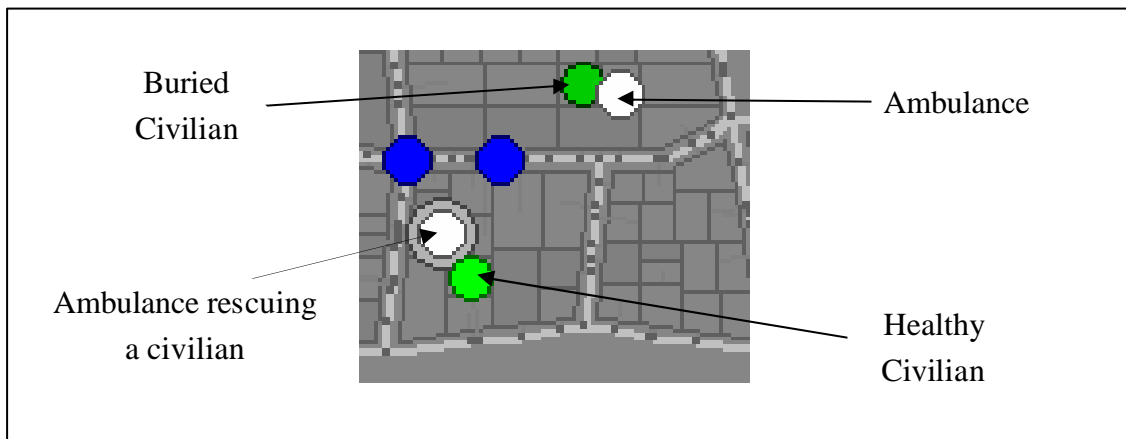


Figure 9: Ambulance and buried civilian

Fire Brigade agent

Fire brigades are responsible for putting out fires all over the city. They share all the civilian properties plus the ones related to its ability to throw water at buildings - such as water quantity. Figure 10 shows two fire brigades. One is inside a burnt out building fighting the flames around him (as denoted by the water stream), and the other one is finding a suitable position to cooperate with his teammate.

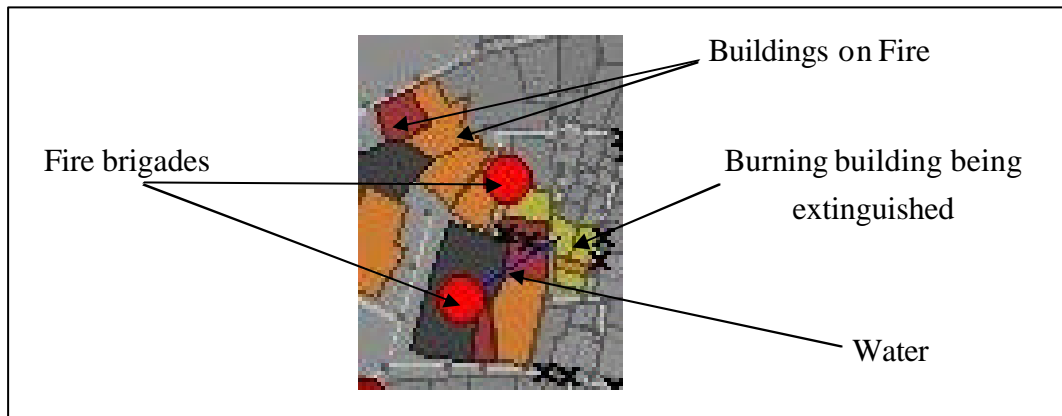


Figure 10: Fire brigades extinguishing a fire

Police Force agent

This agent is responsible for cleaning up road blocks such as debris, traffic accidents and other kind of obstacles. It possesses all civilian properties plus the ability to remove road obstructions. Figure 11 shows two Police Forces clearing roads. One of them is approaching an obstacle while the other is actively removing one.

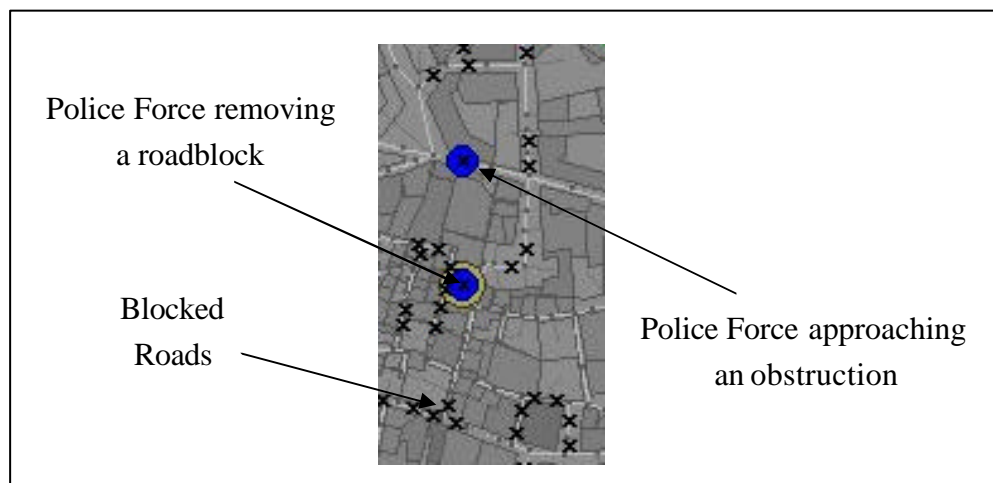


Figure 11: Police agents cleaning roads

2.3.5.3 Center Agents

These agents are represented as ordinary buildings – they appear in a lighter color in the default viewer – possessing all their properties except for the particularity of being indestructible. Although represented as buildings, Centers are, conceptually, very different. They are developed by rescue teams and, as will be seen in section **Error! Reference source not found.** they have, relatively to field agents, enhanced communication skills. This makes them ideal for the coordination and tactical organization of field agents. Each Center can communicate with the field agents of the related type and with the other Center agents

Fire Station

This is the Center agent for the Fire Brigades.

Police Office

This is the Center agent for the Police Forces.

Ambulance Center

This is the Center agent for the Ambulances Teams.

2.3.6 Viewer

A viewer is the graphical interface used to display the actions taking place in the simulated city. Several viewers exist, but the one included in the official package, and used in the competitions, is Morimoto Viewer, developed by Morimoto⁶.

2.3.6.1 Morimoto Viewer

Snapshots of the Morimoto Viewer can be found in Figure 12 and Figure 13. This viewer shows agents as colored circles. Ambulance Teams are white; Firer Brigades red; Police Forces blue; civilians green and any of them gets darker when hurt, turning completely black if they die Figure 14. Buildings also have different colors according to their function or status. While refuges are green and Centre Agent buildings are white, those on fire evolve from yellow, to orange, to red. Flooded and extinguished buildings have different shades of blue, while those burnt down are dark grey (almost black). Roadblocks are marked with crosses and current time and score is displayed on top of the map. The team name can also be displayed on the top bar, but it is optional.

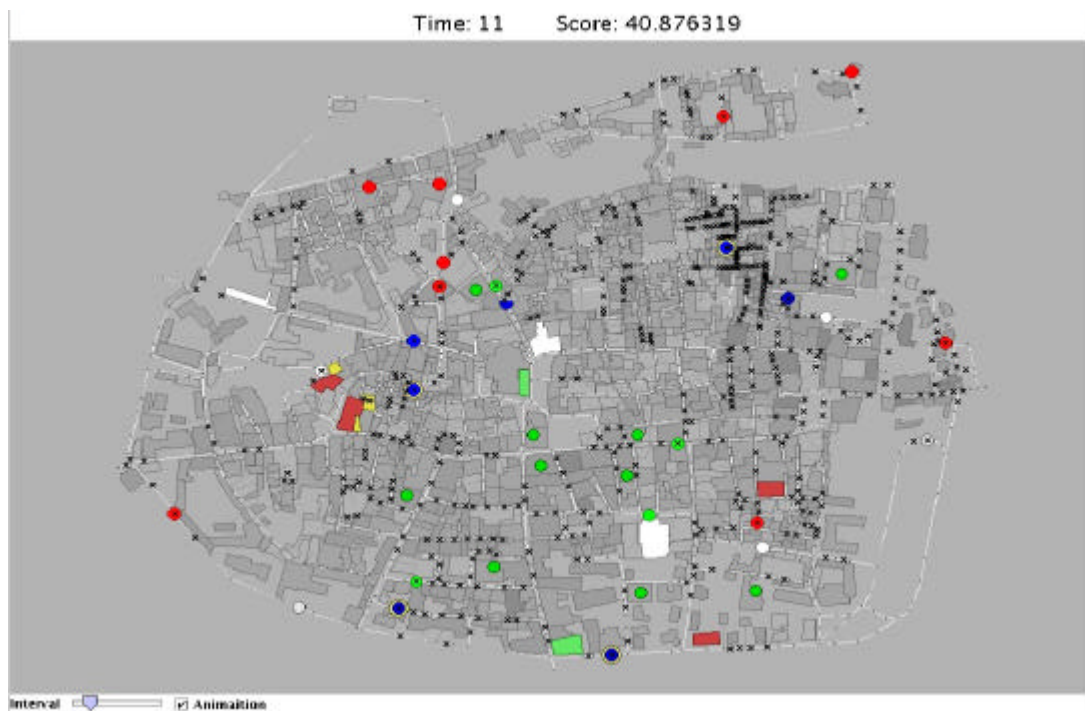


Figure 12: Morimoto Viewer displaying a simulation in the Foligno map.

⁶ Takeshi Morimoto is a PhD candidate at the [Graduate School of Electro-Communications](#) and is affiliated with the [Takeuchi Laboratory](#).

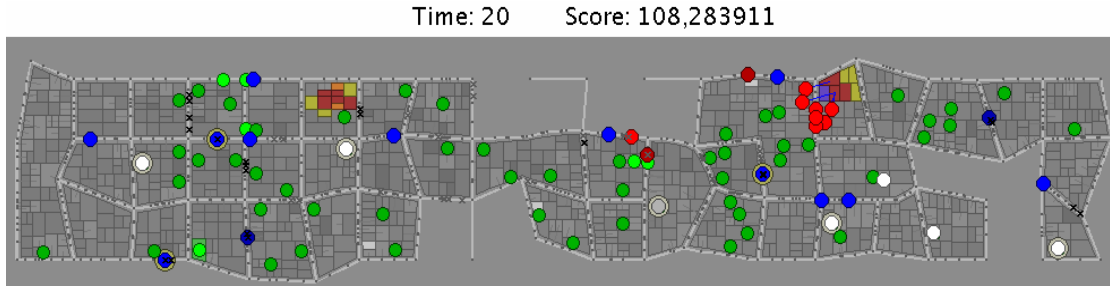


Figure 13: Morimoto Viewer displaying a simulation in the RandomMedium map from Robocup Osaka 2005.

As previously mentioned police and ambulance activity is denoted with a halo around the unit. On the other hand Fire Brigade activity is represented by a blue stream of water shooting from the agent into a building. These agents are represented in Figure 14 along with a civilian and a dead agent. Viewers can also be used to read log files, displaying an offline simulation.

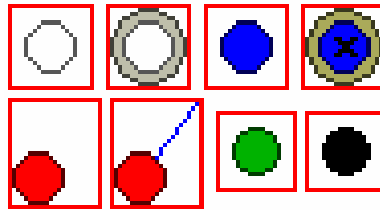


Figure 14: Agents and their respective states as represented in Morimoto Viewer

2.3.6.2 Freiburg's 3D viewer

Also widely used is Freiburg's 3D viewer⁷, mostly due to its appealing look to the public as seen on Figure 15.

This viewer uses familiar forms to represent agents. Fire brigades are represented by fire trucks on roads and fire fighter helmets inside buildings, for example. One further bonus presented is its ability to display statistical data. Although more attractive, Freiburg's 3D viewer is also resource intensive and conveys information in a more cluttered way, which makes it less suitable for developers, that tend to prefer simplicity.

⁷ The 3D viewer, developed by Alexander Kleiner can be found at <http://kaspar.informatik.uni-freiburg.de/~rescue3D/>



Figure 15: Freiburg's 3D viewer (also displaying its 2D view).

2.3.7 System Configuration

By default, the Simulator is configured with the rules for the RoboCupRescue World Championship. However, it is possible to change simulator parameters in order to test strategies, communication options or civilian behaviour, amongst other possibilities. The parameter files are referred in Appendix B, Chapter 4, Section 2.6.

To gain a general notion of the dimensions used in the simulation parameters, default value ranges for some of the most important ones are displayed in Table 1 [12]

Table 1: Parameter ranges in RoboCupRescue 2005 Osaka

Entity	Min.	Max.
Fire Brigade	0	15
Police Force	0	15
Ambulance	0	8
Civilian	70	90
Fire Brigade Center	0	1
Police Force Center	0	1
Ambulance Cente	0	1
Refuges	0	5
Ignition points	2	8

Some other important parameters are:

- simulation time: 300 steps (which corresponds to the 72 hours after the disaster),
- range of agent eyesight: 10m,
- range of agent voice: 30m,

The simulated map area is usually in the order of a few dozens of square Kilometers.

2.4 Communication

As of simulator version 0.47 (2005), communication between modules is done in TCP, although UDP is still supported for legacy reasons. High-level communication between modules is described in the [simulator manual]; however, at this stage, we are only interested in the communications between the Kernel and Rescue Agents.

At each time instant after the initial setup, rescue agents receive sensory information from the Simulator, process it, and send their commands to the simulator.

Sensory information can be received in three forms:

- Visual
- Field hearing
- Radio hearing

Visual information is only sent to field agents. Each agent receives all properties of all objects within the radius of his eyesight. Likewise, **field hearing** is only received by field agents – each agent receives all voice messages sent by an agent within voice range, with sender identification and contents.

Radio hearing is very different. Although messages are still in the sender/contents format, all team agents (field and center) receive such a message, regardless of distance. The following restrictions apply: field agents can only hear messages from their Center or from agents of the same type; center agents can only hear messages from other Centers or from their field agents of the s kind. We can see the agents' radio communication scheme at Figure 16.

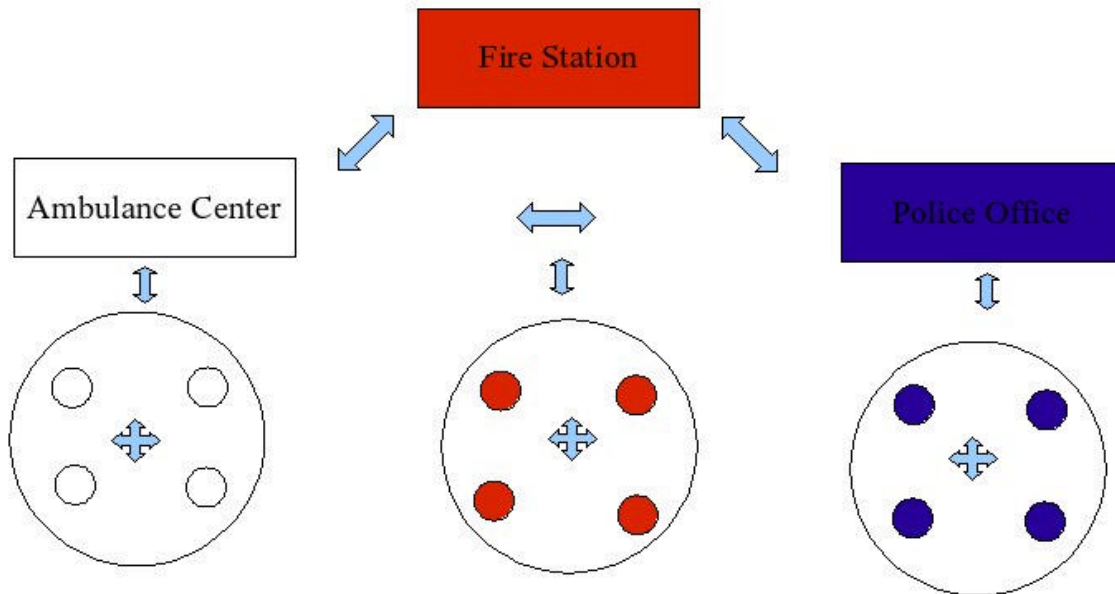


Figure 16: Radio-communications.

There are also other restriction limiting communications. With the Robocup 2004 rules[9], all hearing information between team agents is also limited to the following:

- Field agents can only receive 4 messages per cycle;
- Center agents can only receive a maximum $2*n$ messages, where n is the number of field agents of the Center's type. For example, if there are 10 fire brigades, a fire station can receive 20 messages per cycle, at most.

In appendix B, the process for choosing which messages are received is explained. It should be noted that it doesn't matter if it is a radio or field (voice) message. The restriction applies to the total sum of messages heard.

Sending information is works oppositely to hearing. Each field agent has two forms of sending information. Those forms depend on the destination environment:

- Voice – if the destination is the field within voice range;
- Radio – if the destination is the radio channel.

Like hearing, sending information to team's agents has similar restrictions:

- Field agents can only send 4 messages;

- Center agents can only send a maximum $2*n$ messages, where n is the number of field agents of the Center's type. For example, if there are 10 fire brigades, a fire station can send 20 messages per cycle, at most.
- The data part of any message sent is limited. The current limit is 256 bytes.

2.5 Final considerations

The simulator system is in constant evolution, as new modules are added to the package and the existing ones are updated. The system architecture was designed with this in mind, allowing for independent simulator development and integration. Still, some challenges arise from the hierarchical organization, in which the kernel, being the central module, can become extremely complex[8].

As new features and rules are introduced, teams are required to adapt their code. This leads to a tradeoff between advancing the simulator system towards reality and improving team strategies. As teams adapt to new features, the time left for the development and perfection of high level strategies is significantly reduced. This is a typical problem in which equilibrium between depth and wideness is required, and the situation is actively discussed in the official mailing list.

Another shortcoming of the modular architecture is the fact that, sometimes, specific modules are a lot more developed than others, leading to imbalances. The most current example lies in the fire simulator. This module was greatly improved by a Freiburg university team and some of the original concepts, in regards to communication with the kernel and, through it, to the agents, became outdated. Although, after this update, the spread of fire is directly connected to building temperature, the agents are unable to obtain it. Still, it is a fair assessment that real fire brigades have a notion of temperature in the buildings surrounding them. It is, therefore, a flaw that limits a team's ability to develop tactics considering a useful, easily attained, real world parameter. This knowledge would enable, amongst other things, a more pondered approach to the preemptive watering of endangered buildings.

One example of the simulator package flexibility, and its ability to evolve beyond the predefined architecture, lies also in the fire simulation. Freiburg's fire simulator allows a direct communication to the collapse simulator enabling a new level of interaction and integration between the two modules[11]. When the earthquake hits the city, and the simulation begins, buildings crumble. These collapses can provide the fire simulator with useful information for a realistic creation of ignition points. Furthermore, as fires spread, temperature data can be conveyed to the collapse simulator, as the fire weakens building structure and facilitates collapses. Still, while the fire simulator allows this information exchange, the collapse simulator does not - it is currently outdated and requires a significant upgrade.

As the simulator was studied and tested, some shortcomings of the initialization script proved quite frustrating. This script was heavily dependent of the processor clock speed, requiring manual timing adjustments. Using modern laptops the problem would be augmented, as these adjust their clock speed dynamically depending on several conditions. Different timings were required for the same computer depending whether or not it was plugged into an electrical outlet. A new initialization script was created that addressed these and some other issues, allowing a sequential and synchronized simulator launch, with a failure detection mechanism. Some other minor output changes were made to several simulator modules in order to allow script integration.

Information on this chapter was aggregated in Rescue Technical Report I – Simulator System[13].

Chapter 3

3 Team Implementation

3.1 Initial considerations

In this chapter we take an overlook on the functioning of teams agents and the problems that they have to solve, for explanation purpose we us FC Portugal's implementation which is a modified version of SOS 2004 team code[14].

We have to bare in mind that each team can have a different solution to the same problem, different approaches and different strategies, some better some worse, some just different.

Generally we only present one solution (ours), but one of the project objectives was the selection of a base code. So, as some of the problems and solutions involving team agents are present, some other solutions may be described and compared. The main decision criteria for choosing a base code are discussed in this chapter at section **Error! Reference source not found.**

3.2 Code organization

3.2.1 World State

In a complex domain, information is everything, the better the information, better the decisions may be, based on that information. So, the first step in developing an agent is to properly choose what information it should possess and how that information is stored.

The data structure is a modification of the one in use by simulators. We can see the object class in Figure 17.

- ◆ Rescue::Object
 - Rescue::RealObject
 - Rescue::MotionlessObject
 - Rescue::Edge
 - Rescue::River
 - Rescue::Road
 - Rescue::PointObject
 - Rescue::Building
 - Rescue::AmbulanceCenter
 - Rescue::FireStation
 - Rescue::PoliceOffice
 - Rescue::Refuge
 - Rescue::Vertex
 - Rescue::Node
 - Rescue::RiverNode
 - Rescue::MovingObject
 - Rescue::Humanoid
 - Rescue::AmbulanceTeam
 - Rescue::Car
 - Rescue::Civilian
 - Rescue::FireBrigade
 - Rescue::PoliceForce
 - Rescue::VirtualObject
 - Rescue::World

Figure 17: Object class

So data is an instantiation of class named “Object Pool” composed of hash maps⁸ of objects, with the exception of World which is unique, plus some functions to access that data. Let's note that each data field stored has a timestamp associated, so it is possible to know (and set) when an object property was last updated.

3.2.2 Communication Strategies

Sharing information is vital to agents' performance, as seen in chapter 2 section **Error! Reference source not found.** messages exchanged between team agents are limited both in size and number, so each agent has to choose carefully which message should be sent and heard.

The first step has to do with compressing the information. All possible messages between team agents have a number, so instead of saying “I found a civilian” a correspondent number is sent. Ids (identification number) of team agents, buildings, nodes and roads are also numbered because at setup all agents receive the Ids in the same order (normal Ids have 9 digits each), the only ones that cannot follow this logic are the civilian Ids since as

⁸ The HashMap class is roughly equivalent to Hashtable, except that it is unsynchronized and permits nulls. In *JavaTM 2 Platform, Standard Edition, v 1.4.2 API Specification*

of RoboCup 2005 rules civilian Ids are not sent at startup (civilians have to be discovered). Following this line of thinking object properties are always sent in the same order.

The next step is aggregating the information, this is done by first sending the type of message (i.e. the number corresponding to civilian found), then how many of that type (token), followed by Id and properties * token number, repeating those steps to all the types that an agent needs to send, if necessary information is split into several messages.

But if all agents send messages, a selection of messages to be heard must be made. The solution to this problem is based on the more relaxed number of messages a center can send and hear. So at each cycle field agents will ignore any message from other field agents, they will only listen to the center (as seen in Figure 16, field agents can only listen to the center of their kind). As for the centers they act as repeaters, they listen to information from agents of the same kind and from other centers and resend-it. This way any information takes at most 3 cycles to reach any other agent (agent1 -> center1 -> center2 -> agent2), so it will be available at the beginning of the fourth cycle.

The former solution would lead to an exponential growth of messages sent. The growth is avoided by choosing to only send new information. At section 3.2.1 we noticed that information was time stamped, so it's easy to see if the information sensed is newer than the stored one, and if it is then it's stored and retransmitted.

3.2.3 The State Machine

As seen in chapter 2, simulation time is discrete and, as so, done in cycles. A predefined simulation cycle time has 100ms and, in an agents point o view, is divided into the following stages:

- Sense – This is the stage were agents perceive sensory (simulated world) information. The type of information received was described in section **Error! Reference source not found.**, and how that information is treated its explained in section 3.2.2, with the extra indication of knowing that the only information compressed is the team agent's messages, and that civilian messages are typified.
- Act – This is the stage were agents send actions commands like move, extinguish, etc... Its easy to comprehend that this state only applies to field agents
- Send – This is the stage were team agents send messages. Similar to sense action, received was described in section **Error! Reference source not found.**, and how that information is treated it is explained in section 3.2.2.

A possible approach to handle the mentioned stages is resorting to a state machine. So, a team agent's state machine is defined as the following (Figure 18):

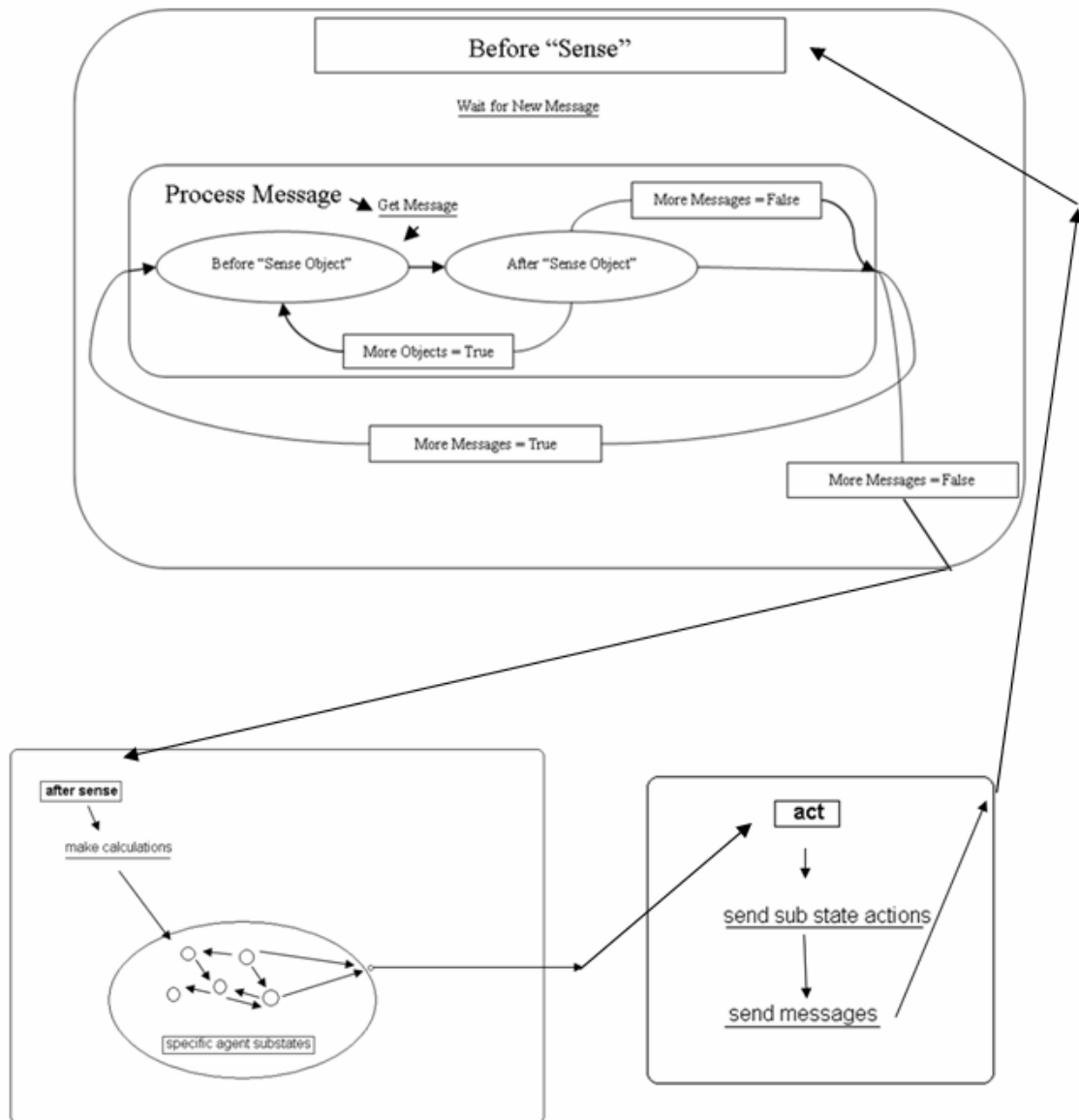


Figure 18: Agents state-machine implementation

As previous stated center agents do not have field actions, so, ignore the “send sub-state actions” at the “act” state. However one of their functions is to handle group strategies, so the “after sense” and “act” states still make sense because strategic decisions will be sent to field agents via message, and as such reflected on their actions. The specific agent implementation of this state machine in FC Portugal’s team is called the “Super State”.

3.2.4 Field Agents

3.2.4.1 Common blocks

Besides data structures, message handling and basic state machine there are other blocks that can be common. The basic behavior of a field agent as even more similarities, for instance if a field agent is buried, he sends a message asking for help, also if an agent need to go somewhere and it cannot find a possible path to its destination he reports that fact. Preserving their own life is also a shared behavior, if an agent finds its health points decreasing he stop whatever he is doing and leaves the building (unless buried). Related to field agents, three of the most important blocks are the path finding, traffic jam handling and the map splitting.

Path Finding

As presented in the simulator chapter section 2.3.4.2, road information comes in the form of a graph. As such, known algorithms for path finding in a graph will be considered.

When choosing a shortest-path algorithm one has to consider the tradeoffs between flexibility, scalability, performance, and implementation complexity. So rescue requirements on path finding were prioritized as the following.

- time cost (speed): cycle time is limited so a fast algorithm is needed
- guarantee of solution: even if its not the best solution, its better than no solution at all, plus there is a need of sometimes stopping the search and use the current best path because, as previous mentioned, cycle time is limited.
- efficiency: machine resources are limited, its common to have several agents running on the same machine.

The most known algorithms to this problem are the First Best Search, Floyd, Dijkstra and the A*.

Without deeply analyzing search algorithms, which is not the purpose of this section, we eliminated First Best Search, because its not granted that this algorithm founds a solution. As for Floyd's algorithm, in some cases is not as efficient as Dijkstra. So the remaining algorithms are the Dijkstra and A*.

Knowing this two algorithm, and perceiving that in this problem we have knowledge of origin and destination and its relative positions, one would be inclined to chose the A* algorithm as this is in a way, a variation of Dijkstra's algorithm that as an heuristic that takes advantage of this situation (e.g. if destination is north of origin, A* will start by exploring paths in the graph that are to the north of the origin). The problem to this solution is that often, especially at start, most of the possible path are blocked, and, in this case

(mazed like), A* has a increased time cost. So the used search algorithm is Dijkstra (see Annex A).

During a simulation, field agents keep track of previous searches, so when traveling a second time to a destination the same path can be used (with no calculations).

Traffic jam handling

As mentioned in the simulator chapter, roads have lanes and roads can have obstacles. Because of this fact, often only one lane is free, so when two agents try to move through the same lane in opposite directions, neither can pass through. This event is called a traffic jam. Please note that this event could arise from other way such as a Fire Brigade extinguishing a building from a building, one of the agents could be a civilian, etc., but the issue is basically the same. This problem can be worsened if more than two agents are involved.

The first step to solve this problem is to detect it. This is achieved by comparing the current position to the current one, if it stays the same for more than n (2 in our case) cycles we are before a traffic jam. Next each agent puts a temporary block (only in his own world state) in the road directly ahead, and a new path is computed. Notice that this solution works no matter the number of agents in the traffic jam.

Map splitting

Sometimes, to perform certain tasks that will be discussed in each kind of field agent, there is a need to split the map equally amongst the simulated field agents of the same kind. In FC Portugal's team this is achieved by calculating the minimum and maximum map coordinates (by comparing apexes of buildings, and node positions). With those coordinates a rectangular form is created and geometrically equally divided between the field agents of the desired kind. To that division we call a cell.

The former solution is good if the road and building density were the same, and uniformly distributed along the map. Some maps are very close to this principle, but other are not, so some agents could have to handle more buildings or roads than others. In view of this fact, currently FC Portugal's team is implementing Voronoi diagrams as a new solution to this problem.

A Voronoi diagram is special kind of decomposition of a metric space determined by distances to a specified discrete set of objects in the space. So, Voronoi cells will be more balanced as they are created taking into account the density of roads and buildings.

Cell attribution is done the following way, at setup the kernel sends all field agents Ids to all field agents in the same order, as the map splitting is done equally in all field agents, each agent takes up the cell corresponding to his relative position in the initial setup data.

3.2.4.2 Ambulance Team

As seen on 2.3.5.2, the main function of an ambulance team is to unbury civilian and take them to a refuge.

The strategy for ambulance teams is the following:

Based on the known civilian properties (mainly buriedness and health points) they estimate the time of death and schedule the order in which civilians will be saved, the time taken in travelling and if whether exists a path to a civilian is considered.

The next steps for a ambulance are to go to the civilian position, unbury him, load the civilian into an ambulance, travel to the closest refuge and unbury and unload the civilian, move on to the next civilian. However this behaviour can change due to several events such as receiving updated or new information about a civilian that changes the priority, the building. As previous stated behaviour also change if there is a fire at the building. Bear in mind that buried field agents will always have higher priority.

In FC Portugal's implementation all ambulance team's form and act as a single group. This happens because the action of unbarring an agent is cumulative and directly proportional to the ambulance teams number, as so the more ambulance team, the fastest the agent will be unburied. There are some exceptions to this tactic because the moving cost is considered when estimating the civilian time of death, as sometimes is more efficient that ambulances act individually. Another exception is at the beginning of the simulation, due to road blocks, ambulance teams don't have possible paths to the same agent, so they also act individually.

One final remark on ambulance behaviour is that when there are no civilians to rescue, ambulance teams split the group, divide the map in cells and search them for new civilians.

The state machine for a ambulance team is showed in Figure 19, as one can see, in this case the "Super State" is called "Decide State".

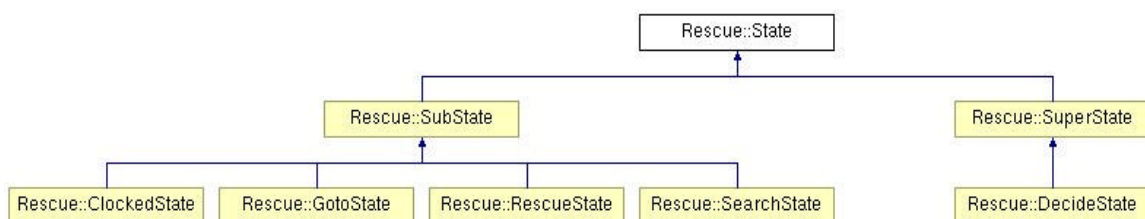


Figure 19 Ambulance Team States

The “Clocked State” is used by some of the other sub states to keep track of the amount of cycles the ambulance team is in a particular sub state, this is called the duration. In the functional point of view is not a sub-state by itself but a common addition to some other sub states.

The “Go to State” is used when ambulance wants to move, it handles all moving functionalities such as the previous described path finding and traffic jam handling.

The “Rescue State” is used before and after the unbarring, it verifies things like if the agent to be rescued is where expected, estimates the time taken for unbarring, verifies if the building is on fire and if even so its possible to rescue the agent by losing some health points but not dieing.

The “Search State” is used when there are no known agents to be rescued. In here each agent takes a cell (was explained in Map Splitting) and searches the unexplored buildings for civilians.

3.2.4.3 Fire Brigade

Fire Brigade agents are the most complex field agents.

As stated in 2.3.5.2, fire brigades function is to extinguish fires. So strategy for Fire Brigades is the following:

Depending on map size, Fire Brigades are organized into groups, usually into to two or three. Based on relative positions, size and proximity to refuges, fiery regions are prioritized and the ones with the highest priority are assigned sequentially to the available groups.

Each group then consider his assigned fiery region and prioritize the burning buildings (from now on they are called targets) to extinguish. Based on relative position in the fiery region, percentage of building area unburned, proximity with building with buried civilian, etc... .

The next step is to choose a suitable neighbor building that is in the water range of the target, the conditions for this are that the building cannot be on fire, it must be reachable, and as close to the target as possible. If no suitable building is found then a road in proximity to target is used, the disadvantage of this is that the brigade occupies a lane which in turn increases the risk of a traffic jam.

After moving to the selected building the fire brigade starts watering the target, note that if for some of the above stated reasons the target changes, and if the new target is in range of the water cannon, the fire brigade simply starts watering the new target, no move action is taken.

After some cycles water in tanks is depleted, so fire brigades go to the nearest refuge to refill their tanks. This action takes some cycles, so when finished, fire brigades reprioritize between the previous assigned region and the currently unassigned regions. The fire brigade proceedings are then repeated. Logically if at some point there are more groups than fiery regions, the available group will be assigned to a fiery region using the remained criteria.

Like ambulances teams when run out of agents to unbury, if fire brigades run out of fires to extinguish, the groups are splitted, and the map divided into cells so that new fiery buildings can be searched.

If no more fiery buildings are found, new civilians can be search. Additionally if all buildings have been explored, fire brigades keep on visiting all known buried alive civilians in order to update information about those civilian properties, so that ambulances can estimate better the civilian time of death.

The state machine for a fire brigade is showed in Figure 20. Because of the two much different functioning modes described about that depend whether or not there is a fire active, the fire brigades have two “Super States”, when there is a fire “Super State” used is the “Attack State”, when there are no active fires fire brigades use the “Help Ambulance” “Super State”.

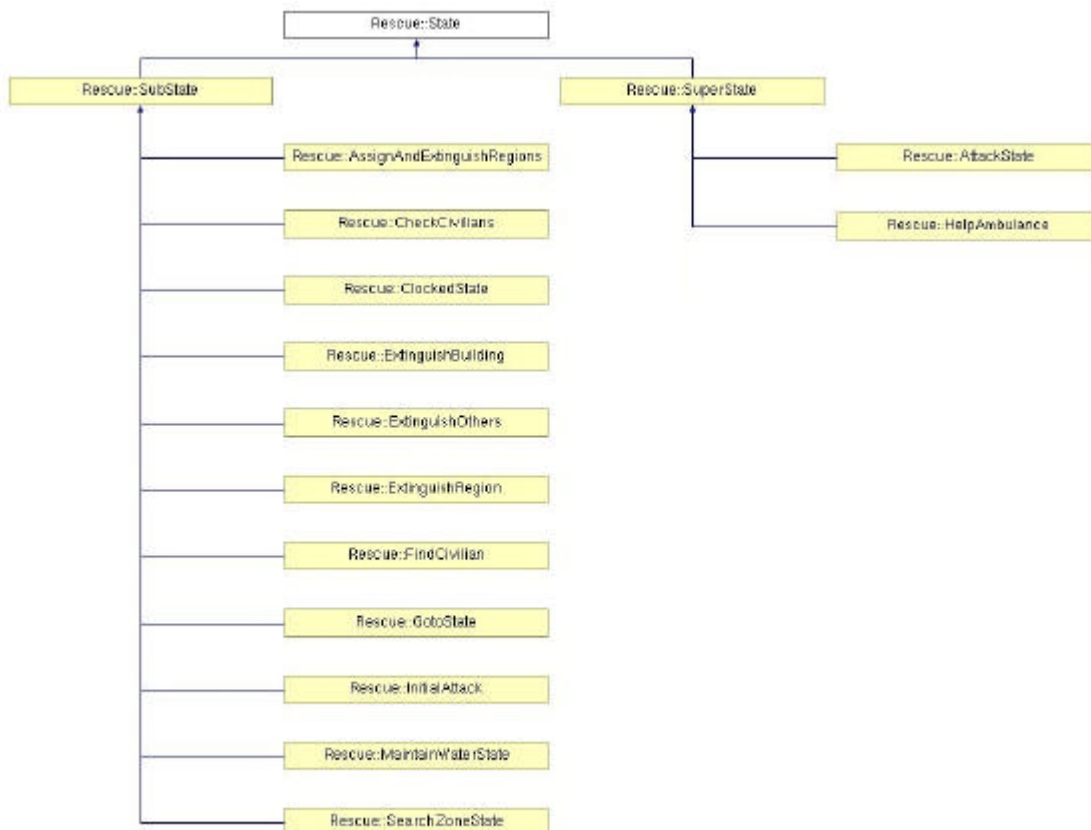


Figure 20 Fire Brigade States

The “Go to State” like in ambulance teams is used when a fire brigade wants to move, it handles all moving functionalities such as the previous described path finding and traffic jam handling.

“Attack State” sub states

The “Clocked State” has the same function as described in ambulance teams. It is used by some of the other sub states to keep track of the amount of cycles the fire brigade is in a particular sub state.

The “Assign And Extinguish Regions” state is where the region is chosen, suitable regions are computed and path cost to regions are calculated. The next state will therefore be the “Extinguish Region”.

The “Extinguish Region” is where targets are computed and chosen. Note that in this implementation, two targets are computed: the current target and the next target. This way if a building is extinguished during a cycle, the following cycle a target has already been found. As so in most cases the function to calculate targets don’t have to be rushed into outputting the current best target, it can take more than one cycle calculating the best. The next state will be “Extinguish Building”.

The “Extinguish Building” is where fire brigade calculates and chooses the neighbour building from where it will attack the target. When chosen if already at destination it starts to extinguish the target if not it goes to “Go to State” and waits for the destination to be reached.

The “Extinguish Others” state is a special state designed to handle big buildings. Normally, the cost of extinguishing a big building is too high due to the building’s area, so it’s a common option not to extinguish such buildings. Instead of extinguish a big building, the fire brigades let it burn completely and fire is controlled using buildings around it. However in certain exceptional situations such as specific maps with those kinds of buildings in key points, there is a need to consider that buildings as targets, so this state exist with that purpose.

The “Initial Attack” is as the name suggests, the state used to handle fire fighting at the beginning of the simulation. This happens because at the beginning of the simulation, road blocks prevent fire brigades from reaching the assigned region and target, as such, in order for the fire brigade to be useful, only reachable targets and consequentially reachable neighbours are considered. As soon as the assigned region is reachable, fire brigades leave this state.

The “Maintain Water State” is responsible for keeping track of the group’s water quantity, and the decision of moving the group to the refuge in order to refill the groups’ tanks. The decision is made taking into account the effect that the remaining group water will have on the boundary of the fire.

In “Search Zone State” state, each fire brigade searches a cell for a fire. Opposing the “Find Civilians” state previous described in ambulance teams, in this state there is no need to visit buildings as all fires are visible from roads.

“Help Ambulance” sub states

The “Find Civilian” state is similar to the one described in ambulance teams, as the former, the map splitting mechanism previously described is used.

In “Check Civilians” state, the fire brigades recursively visit all buried alive civilians in his assigned map cell.

3.2.4.4 Police Force

As stated in 2.3.5.2, police force agent’s function is to clear blockades in roads. So strategy for Police Force is the following:

The first strategic decision made is to only clear road blocks at not passable roads, this means that partially obstructed roads will not be cleared. The reason for this is that partial blocks only affect the speed of an agent by halving it. The speed reduction isn’t significant when in confrontation to the time that would take a police force to move to that road and remove the partial block.

The main problem for police forces has to do with the order in which road blocks are removed. On FC Portugal’s implementation there are several ways to do this:

- Clearing blocks until a refuge reached,
- Clearing blocks from a specific point to a refuge,
- Clear blocks around a refuge
- Clear a specific path
- Clear a specific cell

These possible ways are called tasks. To each of those task is given a weight, those weights are parameters specified in a police force configuration file, and can be set for specific maps.

All tasks are requested by other agents with the exception of clear a specific cell, which is used when no other tasks are requested.

When the police force receives a task request, he calculates how long each task is going to take, and multiplies it with the weight factor. Then he executes the cheapest. If the cost of doing a certain task is too high, the task is not considered

Police force agents are always performing tasks unlike the other agents that in order to perform a certain action, they first have to move themselves to a certain position. This means that a police force agent, when moving, is always clearing impassable blocks in its way so no detour of blocks is made.

The state machine for a police forced is showed in Figure 21, as one can see, in this case the “Super State” is called “State Police Force”.

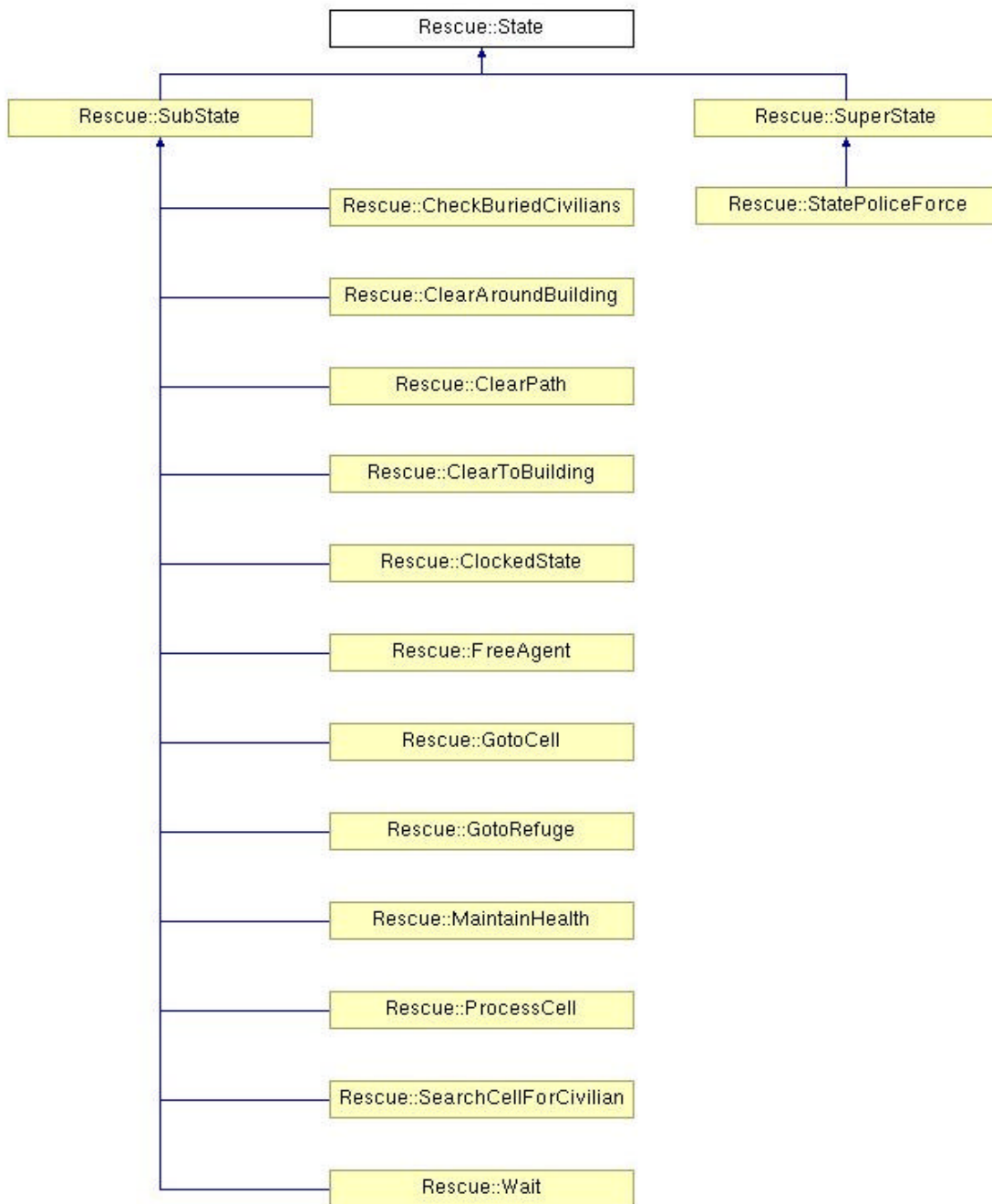


Figure 21 Police Force States

The “Clocked State” has the same function as described in ambulance teams. It is used by some of the other sub states to keep track of the amount of cycles the police force is in a particular sub state.

The “Clear Path” state is the police force equivalent to the “Go to State” in ambulances and fire brigades. So, when the police force needs to go to a particular position in order to perform a certain task, it uses this state. The difference is that it clears all impassable blocks in its way.

The “Maintain Health” state is responsible for avoiding the police force agent’s death. In other agents the state is part of the “Super state” as mentioned when explaining the common blocks. This state originates from older versions of SOS’s code that FC Portugal’s team has not yet integrated in the “Super State”.

The “Free Agent” state is, as the name suggests, used to enable the usability of the other kind of agents. When ambulance team agents or fire brigade agents are unable to execute any useful action, they send a request to be freed. This result is accomplished just by moving the police force agent to a location next to the restrained agent.

The “Go to cell” state is used to go to the police force’s cell when no other agent task is in queue. The difference from the “Clear Path” state is that no destination position is provided, as so it calculates and chooses the closest blocked road from his cell to start clearing.

The “Process Cell” state is used to clear block police force agent’s cell. In here, the police compute the sequence in which roads should be unblocked, and clears roads in that sequence.

The “Search Cell for Civilian” state is a slightly different that the “Find Civilian” state found in ambulance teams and fire brigade agents. The main difference is that when searching, police force clears impassable road block in its way.

The “Check Buried Civilians” state is similar to fire brigade’s “Check civilians” state, i.e. when all buildings have been searched for civilians, police force keep visiting buried alive civilians. The only difference is that he also cleans blocks in the path.

The “Wait” state is a solution to a particularity in the simulator system. Action in the current version of the simulator is only allowed at cycle 4 (because some initial simulator modules date is only available at cycle 3). As such police force agents have to wait for cycle 4 to check if they are buried. This state is used for that purpose. As in the “Maintain Health” state in other kinds of agents this state is implemented in the “Super State”, the reasons for this not to happen are the same.

The “Clear to Building” state is where the task of going to a refuge is implemented. Police force agents move to refuge clearing every impassable road block in its way. The request for this task is made by police office.

The “Clear around Building” state is where the task of clearing blocks around a refuge is implemented. Police force agents move to refuge clearing every impassable road block in its way. Like “Clear to Building” request for this task is made by police office.

3.2.5 Center agents

3.2.5.1 Common Blocks

In comparison to field agents, center agents are much simpler. As explained in communication strategies (3.2.2) the main function of center agents is to act as repeaters. As such, the communication module is similar.

3.2.5.2 Ambulance Center

As explained the main function for this agent is to act as a repeater. As ambulance teams’ calculations are relatively simple, there is no need for further calculations at the ambulance center. Consider that ambulance teams take several cycles to unbury an agent, which is more than enough time to compute the next to be saved.

The state machine for an ambulance center is reduced to a single “Super State” called “Center State”

3.2.5.3 Fire Station

The main function of this agent is to act as a repeater. However some of the calculations computed in fire brigade agents are also computed in fire station. This happens both in the prioritize regions case and in group assignment. As seen cycle time is limited, and some times field agents have to rush decisions. Fire stations do not have the former limitation, so it compares the field agent’s solution to its own and if the station solution is better it, it is sent to fire brigades.

The state machine for a fire station is composed of a single “Super State” named, like in fire brigades’ case the “Attack State”

3.2.5.4 Police Office

Besides acting as a repeater, the police office computes both currently assigned police force tasks and the unassigned. The assigned tasks are multiplied with a reassign coefficient and it is chosen the most suitable task for each police force based on relative position to objectives. As one can apprehend, if active, the police office is in fact the main responsible for police force strategy.

3.2.6 Libraries

One of software engineering techniques has to do with modularization, if part of the code is common to several programs, that code should be available in the form of a library, those libraries should also be organized hierarchically and by function. In this section we present the three libraries used in our code, those libraries allow a higher level of programming. This allows different programmers improve team performance on different layers.

3.2.6.1 The “librescue”:

This library was originally developed for the simulator modules, it handles lower level connections, so we can abstract from things like socket management, low level connection protocol, etc..

It also has data structures for the objects of the simulation, constants definitions use in high level communication protocol, basic simulation parameters, and in our version some handy debugging routines.

3.2.6.2 The “Libadk”

Originally released on a package called ADK (Agent Development Kit) [15], its purpose was to provide the basic tools for agent development, so it provides a memory interface with data structures suitable to a team’s agent, a path finding method and a basic state controller better explained at the State Machine section(3.2.3).

This library was later on developed to allow message exchange between agents, so the code to communicate described in communication strategies (3.2.2) is implemented here.

Thread management and road passage detection is also implemented at this library.

3.2.6.3 Agent Kind Libraries

In order to properly support strategic decisions at center agents, they must be able to transmit and receive more than object information. So, there is a library to every of the three rescue domains. This way, center can communicate with field agents of their kind. An example to this situation is the assignment of a task by the police office to a police force.

When both field agents do the same type of calculations the code for those functions is also placed at the agent’s kind library. An example for this situation is the computation of fiery regions by both fire brigades and fire station.

3.2.7 The initial parameters

During the description of FC Portugal's team agents, sometimes customizable parameters were referred. Most of those parameters values result of human perception, i.e. the value for the parameter is "guessed" based on the human knowledge. As such there is a need to fine tune those parameters. This can be done by trial and error; however, considering the immensity of those parameters, this task is all but impossible. Therefore in this way only one limited set can be adjusted. The tool described in chapter 4, as it is there explained, is a step in solving this problem. For now, only the type of parameters is presented.

3.2.7.1 Generic

Generic parameters are the ones common to the agents (E.g. minimum acceptable health points when calculating a task). Constant values defined by RoboCupRescue rules that are subject to possible changes in the future are also include here (E.g. the number of bytes of a data part of message).

3.2.7.2 Map Specific

In RoboCupRescue, some maps are known. For those maps, parameters can be modified for the simulated conditions (E.g. the number of fire brigades per group). In this case some pre-calculations can be made too, like dividing the map in cells.

3.2.7.3 Agent Specific

Much of agents decisions are made by doing calculations that are affected by coefficients, the majority of those being agent specific (E.g. the relative weight of the police force's task of going to a refuge). Additional parameters of this kind would be factors such as the are threshold from which a building is considered big (used in fire brigades).

3.3 Base code selection and development

The RoboCup project is of open source nature. In addition exchanging ideas between teams is strongly encouraged. The reason behind this line of thinking is that the development in artificial intelligence becomes faster.

Robocup teams optionally start their project from zero, start with development kits, or take other team's code and start from there. It is even possible to use code from different teams.

Starting from scratch or even from a development kit, although better in the structural way, has the disadvantages taking to long for the team to become competitive, and in the initial years teams wont make any scientific progress.

For the above reasons FC Portugal's rescue team, decided to start from an existing team's code.

From this point, and already in the scope of this project, some criteria for choosing the base code were defined:

- Base code would be preferably written in C++ language
- The basic actions of the team's field agents should be efficient
- Base code should allow an easy integration with high-level development
- Base code should be modularized

The reasons behind this selection are the following:

- All team members have more experience in the C++ programming language.
- There is the possibility of adapting some of FC Portugal's simulation soccer team code into the rescue team, especially the parts regarding multi-agent coordination.
- FC Portugal's long term objectives are strategy oriented (High-Level)
- Some of FC Portugal's member will change through time, so modularization fastens the development.

From the active rescue teams, the base code that came more close to the specified criteria was the S.O.S. rescue team code.

3.4 Adapting the chosen code to new rules, towards RoboCup Osaka 2005

The main changes in the RoboCupRescue 2005 was the replacement of the Fire Spread Simulator [16] for the Freiburg's Fire Simulator [11].

As a result, and together with the voting from the rescue community the RoboCupRescue 2005 rules [12] introduced the following main changes:

- Civilian Id are not sent at start-up
- No more than one fire brigade can enter a building at a time, with the exception of refuges.

The change of the civilian rule led to more complications that would be expected. The agents' knowledge on ether or not all civilians were found was lost. So agents now must explore all buildings in search of civilians.

On low level this change was a monumental problem. First the allocation of civilian data structure was no longer made at the beginning of the simulation to me made at any time during the simulation. The new civilian data could be obtained either by field sense or by message.

As it was explained in communication strategies (3.2.2), agent id's information is compressed using the initial setup Ids. So, civilian Id's could no longer be sent in a compressed form. This led to another problem as the size of the civilian id was in number form, frequently as part of the type of message size. This implied modifying all possible civilian messages' functions in order to change the size. This was made on common communication libraries, but also (due to the some programming mal-practices) in the six types of team agent's code.

As for the fire brigade in building limit, the problem was solved by verifying if the building is occupied.

3.5 RoboCup Osaka 2005

In order form a team to qualify for RoboCup it has to submit the team's logs and a Team Description Paper for evaluation.

As the work described in this project only started in March 2005, qualification was previously achieved by FC Portugal's rescue members at the time⁹ with the paper: "FC Portugal 2005 Rescue Team Description: Adapting Simulated Soccer Coordination Methodologies to the Search and Rescue Domain"[17].

The results were relatively bad, but not outside expectations. The time used to learn the functioning of the simulator, choosing a code and adapting it to the new rules, left no time for strategy improvement.

The solution to the fire brigade in building limit revelled to be a pour one, as the paths were calculated prior to the presence of any fire brigade in the building. So at each cycle, only fire brigade in the group would only be in the correct building.

During the RoboCup changes were made to solve this problem. The new solution was for the fire brigades to consider no only the best building from where to extinguish, but the n bets where n is the number of fire brigades in group. This solution was implemented(with

⁹ Nuno Lau, Luis Paulo Reis and Francisco Reinaldo

some bugs) at the end of the pre-eliminary stage. This led to an improvement on FC Portugal's team performance; however it was not enough to make it to the next stage.

3.6 Final Considerations

It has been said that center agents are, because of their communication skills, ideal for managing tactics and strategies, however when analyzing FC Portugal's implementation of field agents it is easy to perceive that almost all strategic decisions are taken by field agents. This happens mainly for two reasons: the first can be directly inferred from Table 1 (chapter 2, section 2.3.7) and is that center agents are optional, i.e. it is possible for simulation not to have center agents (although it has not occurred in competitions in the last years), the second reason is to do with the possibility of losing message which is simulated. As such it is safer to implement high level strategies in field agents and, if significant, use the center agents to improve the result of the same strategy.

Voronoi cells are the next hope in map splitting techniques, it is currently being implemented and it is expected to improve FC Portugal's agent's performance

In RoboCupRescue simulations, both known and unknown maps can be used. For the known maps FC Portugal's implementation uses map specific optimized parameters. This way a higher agent's performance can be achieved.

Participating in the rescue simulation league for the first time was an enriching experience, especially for new team members, the knowledge obtained in rescue environment was huge and unvalued.

FC Portugal's rescue team was successfully established. The base code was selected, studied and improved. The team successfully participated in RoboCup 2005 rescue competition, in Osaka, Japan,

Information on this chapter is being used in the elaboration of Rescue Technical Report II – FC Portugal's Team Agents

Chapter 4

4 FCP_x – A Tool for Agent Evaluation and Comparison

4.1 Initial Considerations

A major hindrance in the development of rescue agents is the inability to objectively measure the performance of an agent. This proves to be a big obstacle to the process of implementing minor improvements and tweaking existent parameters. Due to the stochastic nature of the simulator, it is nearly impossible to realize if minor changes in the code improve the overall behavior of the agents, without running several simulations and using different maps. The overall score can be affected by so many variables and starting conditions, that it becomes too vague to describe the improvement or deterioration of the agents' performance in small tasks. It becomes, therefore, essential to extract further data from the simulation. Some teams made an effort in that direction.

Team Damas based their effort on Morimoto's 2D viewer, adding code to enhance the statistic interface, and making it especially useful when used in conjunction with their own "autorun" program. When both these tools were used together, several simulations would be run consecutively. At the end of each, the viewer would save the relevant statistics into a file, killing itself subsequently, and thus enabling a new simulation to begin. This tool provided a limited set of statistics and it is now outdated, no longer supporting recent simulator log files.

Freiburg's 3D viewer tool also has some extra features built-in. A statistic view option is available to measure and evaluate a team's performance in various aspects, such as the team's efficiency in exploring, rescuing civilians or extinguishing fires. This enables an easy comparison of two teams for first-time users, since it is possible to see the actual difference between teams by the simultaneous parsing of two log files. Also possible is the post-processing of simulation log files, allowing the continuous, strenuous, evaluation of time-dependant parameters.

These and other existing tools provide a limited amount of data and some are even currently outdated and, consequently, not working. Comparison can only be presented visually, and on a limited set of parameters, with no further options to record and thoroughly analyze the observed data. It is also, usually, limited to two teams at a time, restricting the user's ability to analyze a certain aspect in a wide variety of teams.

After the successful introduction of team FC Portugal at RoboCup 2005, Osaka, it became apparent that certain future objectives of the FC Portugal project required a data extraction tool. Our graduation project was then extended for another semester, allowing the development of the required software. The primary objective of this program was the ability to extract evaluation parameters, capable of feeding a reinforced learning neural network.

In this chapter we present a new tool that mines data from simulation logs, retrieving a large number of parameters and saving them in a standardized file. Comparison tables or charts can then be generated from this file, and it can also be used to data feed a number of different applications, from simple "performance evaluators" to "learning modules". The FCP eXtended Freiburg 3D viewer (FCPx) is based on Freiburg's 3D viewer, and is in accordance with standardized design patterns and modular software engineering approach. This tool can be extremely useful to the entire rescue community. Our work can be included in future versions of Freiburg's 3D viewer, so that it may, eventually, develop into a standard benchmark for rescue agents' behavior.

4.2 Tool Development

4.2.1 Design Options

There are two possible paths to the extraction of simulation data, which are not, in any way, mutually exclusive. The required data can be logged from the simulator or from the agents and, although some figures may be common, the different nature of both kinds of data gives each one its own perspective on the simulation. Information extracted from the agents is extremely flexible, depending, both in amount and in format, on the implementation of the code for each individual team. Due to the lack of standardization in the Rescue community it is extremely difficult to use this data to compare different teams, as terminology and information extracted can be extremely diverse. Also, since the code is developed, not only by different people, with different strategies and tactics in mind, but also in different development environments and different programming languages, the challenge of creating and maintaining a tool, capable of mining the relevant data in every different team, and to present it in a comparable form, is all but impossible. Still, the figures obtained from the agents are of extreme importance to complement the ones extracted from the simulator, providing the programmers with valuable insight into the behavioral process of each individual agent. It is, nonetheless, a path impossible to take if one of the main

objectives is inter team comparison, due to the above mentioned lack of standardization between teams. The chosen path is, therefore, to collect data from the simulator, benefiting from the fact that a single, common, program is used as the staging grounds for the agents' activities. This enables the gathering of information in a form common to all teams, allowing objective comparisons in any of the obtained parameters.

The FCP eXtended Freiburg 3D viewer (FCP_x) tool was developed over the simple logging features on Freiburg's 3D viewer, extending its ability to analyze and compare different teams. A critical aspect in evaluating and comparing performance is having the information stored in a flexible, adaptable, form. If the data can be easily imported into a spreadsheet, it becomes easy to build, and analyze, tables and comparative charts. From this point the versatility of this solution gives the programmer endless possibilities. Still, the FCP_x tool provides some (initial) customizable spreadsheets to reach a few elaborate outcomes, both as a proof of concept, and as a means to help beginners analyze their data in a predefined way.

4.2.2 Extracting data from log files

Our initial effort went into acquiring as much data as possible from the simulation. A large amount of parameters is directly obtained from the log file, while some other are calculated using those previously gathered. Some values are kept in both absolute and percentage form, so that users may have a relative idea of the action developments, simplifying the comparison between maps and teams. Some of the numbers are, therefore, redundant. However, the effortless comparison achieved justifies that. Also worth mentioning is the focus on expandability. It is possible to add a new parameter with little effort, besides the one required for its characterization.

The information is then saved into a file with the same base name as the log file - therefore using the "date-time-team-map" convention. As was previously mentioned, versatility was the main factor when considering file formats. Keeping that goal in mind, and considering essential the ability to easily export the data into a spreadsheet, CSV (comma separated values) was the chosen file type. Some other advantages arise from this choice, like the potential to reduce the amount of hard disk space occupied by log files. To properly evaluate a strategy there's a need to run dozens of simulations comparing the attained results with data from previous approaches. That generates dozens of logs, which take around 5 to 15 megabytes of disk space, each. Also, in order to examine a team's evolution through time, older logs are stocked for reference. If you add to that the need of comparison to other teams' results, therefore keeping some of their logs in store, the amount of disk space used can quickly amount to dozens, or even hundreds, of gigabytes. A CSV statistical file with our chosen parameters takes around 15 to 50 kilobytes, which is less than one hundredth of the disk space used by each log. Keeping only the statistical files,

and deleting the logs, allows for an extremely larger amount of data to be stored and used, improving the ability to evaluate different teams and strategies. Still, with the deletion of the log file, the ability to reproduce the simulation is lost.

From the CSV files, the data can be effortlessly imported to a spreadsheet. This creates a space where, either through tables or charts, evolutions can be perceived, behaviors studied and conclusion drawn. By this means, the user acquires the extra flexibility necessary for the analysis, and evaluation, of minor changes in the code, and their direct, and indirect, repercussions on the agents’ performance.

4.3 Analyzing the First Statistical Results

Having developed this new tool, the time came for the analysis of our log files from Osaka. We chose to analyze the KobeHard map, an extremely difficult map from the 3rd simulation day, as our agents did not behave as expected and our overall score was low. Using our custom designed spreadsheets, tables were automatically created from the CSV files. These tables contain statistical information and their evolution through time, which should allow us to understand what went wrong.

Table 2: Fire brigades - Building related data

Building Related Data							
Legend							Total Number of Buildings
Burning Buildings	Flooded Unburned Buildings			Number of Times Buildings Re-ignited			Initial Undestroyed Area
Extinguished Buildings	Completely Burned Buildings			Preemptively Watered Buildings that Caught Fire			14895169
							Number of Refugees
							1
Time	Burning	Extinguished	Burned	Flooded	Re-ignited	Flooded That Caught Fire	Bldgs. Undamaged by Fire nor Water
0	N/A	N/A	N/A	N/A	0	0	N/A
1	6	0	0	0	0	0	99,19%
2	6	0	0	0	0	0	99,19%
3	6	0	0	0	0	0	99,19%
4	6	0	0	0	0	0	99,19%
5	6	0	0	0	0	0	99,19%
6	7	0	0	0	0	0	99,05%
7	8	0	0	0	0	0	98,92%
8	8	0	0	0	0	0	98,92%
9	6	2	0	0	0	0	98,92%
10	7	2	0	0	0	0	98,78%
20	5	5	0	0	0	0	98,65%
30	8	4	0	0	2	0	98,38%
40	8	5	1	0	2	0	98,11%
50	9	5	2	0	3	0	97,84%
60	12	6	4	0	3	0	97,03%
70	18	5	6	0	6	0	96,22%
80	23	5	6	0	6	0	95,41%
90	21	5	11	0	6	0	95,00%
100	23	5	13	0	6	0	94,46%
110	20	5	21	0	6	0	93,78%
120	19	5	26	0	6	0	93,24%
130	23	5	30	0	6	0	92,16%
140	29	5	33	0	6	0	90,95%
150	37	4	38	0	7	0	89,32%
160	43	3	43	0	8	0	87,97%
170	51	3	53	0	8	0	85,54%
180	54	3	58	0	8	0	84,46%
190	59	3	73	0	8	0	82,97%
200	53	3	84	0	8	0	81,08%
210	48	3	100	0	8	0	79,59%
220	51	3	112	0	8	0	77,67%
230	58	3	118	0	8	0	75,81%
240	61	2	134	0	9	0	73,38%
250	70	2	143	0	9	0	70,95%
260	70	2	161	0	9	0	68,51%
270	70	2	176	0	9	0	66,43%
280	68	2	191	0	9	0	64,73%
290	61	2	209	0	9	0	63,24%
300	53	2	229	0	9	0	61,62%

The fire brigade relevant data obtained is displayed on Table 2, Table 3 and Table 4. Table 2 presents building data. The most important figure presented is the amount of undamaged

buildings, but all the other values are useful in strategic analysis. The amount of burning buildings is a good indication of whether the fire brigades have fires under control - looking at the displayed data we can easily see that they don't. Two important records, in tactical terms, are the number of re-ignited buildings and of those preemptively flooded that caught fire. With a perfect strategy, when a building is flooded it is done with the right amount of water. Too much will waste both water and time - extremely precious resources; while too little is even worse, allowing the building to catch fire and disrupting the prevention strategy. The same hold true for re-ignition of extinguished buildings. When a building is extinguished, the fire brigades should take their surroundings into account in order to decide whether they should keep pouring water into the building to further lower its temperature or focus their efforts on a different building.

Table 3 Fire brigades – Civilian related data

Civilian Related Data					
Legend					
Percentage of Civilians Discovered		Civilians Killed by Fire		Total Number of Civilians	
Total Number of Civilians Discovered		Pero. of Dead Civilians Killed by Fire		70	
Fraction of Those Discovered by Firebrigades					
Civilians Discovered					
Time	Pero. of Civs. Discovered	No. Civs. Discovered	Discovered by FBs	Civilians Killed by Fire	Pero. of Dead Civilians Killed by Fire
0	5,71%	4	2	0	0,00%
1	5,71%	4	2	0	0,00%
2	5,71%	4	2	0	0,00%
3	5,71%	4	2	0	0,00%
4	7,14%	5	2	0	0,00%
5	7,14%	5	2	0	0,00%
6	7,14%	5	2	0	0,00%
7	12,86%	9	4	0	0,00%
8	22,86%	16	8	0	0,00%
9	22,86%	16	8	0	0,00%
10	22,86%	16	8	0	0,00%
20	32,86%	23	8	0	0,00%
30	45,71%	32	8	0	0,00%
40	47,14%	33	8	0	0,00%
50	51,43%	36	8	0	0,00%
60	54,29%	38	8	0	0,00%
70	57,14%	40	8	1	20,00%
80	60,00%	42	8	2	22,22%
90	60,00%	42	8	2	20,00%
100	61,43%	43	8	2	20,00%
110	68,57%	48	8	2	20,00%
120	72,86%	51	8	2	15,38%
130	72,86%	51	8	2	15,38%
140	72,86%	51	8	2	14,29%
150	74,29%	52	8	2	13,33%
160	74,29%	52	8	3	16,67%
170	75,71%	53	8	3	14,29%
180	75,71%	53	8	4	18,18%
190	75,71%	53	8	5	21,74%
200	77,14%	54	10	6	24,00%
210	78,57%	55	10	6	24,00%
220	80,00%	56	10	7	24,14%
230	80,00%	56	10	8	26,67%
240	80,00%	56	10	8	26,81%
250	80,00%	56	10	10	28,41%
260	81,43%	57	10	11	31,43%
270	81,43%	57	10	12	32,43%
280	81,43%	57	10	12	30,77%
290	82,86%	58	10	12	30,77%
300	82,86%	58	10	12	30,77%

Table 4 exhibits the civilian data relevant to fire brigades. This includes the amount of civilians killed by fire, which is an extremely important figure, due to the high value of every life. By the end of the simulation twelve people were killed by flames, which means over twelve points were lost this way (see score at chapter 2 in section 2.2).

Table 4 Fire brigades – Agent related data

Agent Related Data					
Legend					
	Buried Police Forces Killed by Fire			Fire Brigades Killed by Fire (Not Buried)	
	Buried Ambulances Killed by Fire			Number of Buried Fire Brigades (Alive)	Total Number of Fire Brigades
	Buried Fire Brigades Killed by Fire				13
	Buried Agents Killed by Fire				
Time	Police	Ambulance	Fire Brigade	Fire Brigades Killed by Fire (Not Buried)	No. Buried Fire Brigades (Alive)
0	0	0	0	0	N/A
1	0	0	0	0	N/A
2	0	0	0	0	2
3	0	0	0	0	2
4	0	0	0	0	2
5	0	0	0	0	2
6	0	0	0	0	2
7	0	0	0	0	2
8	0	0	0	0	2
9	0	0	0	0	2
10	0	0	0	0	2
20	0	0	0	0	2
30	0	0	0	0	1
40	0	0	0	0	1
50	0	0	0	0	1
60	0	0	0	0	1
70	0	0	0	0	1
80	0	0	0	0	1
90	0	0	0	0	1
100	0	0	0	0	1
110	0	0	0	0	1
120	0	0	0	0	1
130	0	0	0	0	1
140	0	0	0	0	1
150	0	0	0	0	1
160	0	0	0	0	1
170	0	0	0	0	1
180	0	0	0	0	1
190	0	0	0	0	1
200	0	0	0	2	1
210	0	0	0	2	1
220	0	0	0	2	1
230	0	0	0	2	1
240	0	0	0	2	1
250	0	0	0	2	1
260	0	0	0	2	1
270	0	0	0	2	1
280	0	0	0	2	1
290	0	0	0	2	1
300	0	0	0	2	1

Agent related data is displayed on Table 4. Analyzing these numbers we should notice two important facts. Firstly, there were two fire brigade casualties. Rescue agents' casualties are avoidable and should never take place. In order to accomplish that, agents must be tweaked in order to ensure that only necessary risks are taken. Secondly, there are two buried fire brigades since the beginning of the simulation and only one is rescued. With the uncontrolled blaze present in this map, one more active fire brigade could prove extremely helpful.

Table 5: Police forces – Road related data.

Road Related Data				
Legend				
	Number of Roads With at Least One Blocked Lane			Total Number of Roads
	Number of Roads Blocked (Impassable)			820
	Number of Road Obstructions Removed			Total Road Length
	Percentage of Roads Blocked (Impassable)			16.247
Time	No. Roads Obstruct	No. Roads Blocked	No. Obstructions Remove	Perc. Roads Blocked
0	N/A	N/A	N/A	N/A
1	N/A	N/A	N/A	N/A
2	368	297	0	36.22%
3	368	297	0	36.22%
4	367	296	1	36.10%
5	365	294	3	35.85%
6	359	288	9	35.12%
7	357	286	11	34.86%
8	352	281	16	34.27%
9	350	279	18	34.02%
10	345	274	23	33.41%
20	316	247	52	30.12%
30	287	218	81	26.59%
40	264	194	104	23.66%
50	234	164	134	20.00%
60	203	131	165	15.96%
70	178	104	190	12.68%
80	157	83	211	10.12%
90	139	63	229	7.68%
100	125	48	243	5.85%
110	111	34	257	4.15%
120	102	25	266	3.05%
130	95	18	273	2.20%
140	89	12	279	1.46%
150	84	8	284	0.98%
160	79	3	289	0.37%
170	76	0	292	0.00%
180	76	0	292	0.00%
190	76	0	292	0.00%
200	76	0	292	0.00%
210	76	0	292	0.00%
220	76	0	292	0.00%
230	76	0	292	0.00%
240	76	0	292	0.00%
250	76	0	292	0.00%
260	76	0	292	0.00%
270	76	0	292	0.00%
280	76	0	292	0.00%
290	76	0	292	0.00%
300	76	0	292	0.00%

The police force relevant data is displayed on Table 5, Table 6 and Table 7. Road information present in Table 5 indicates that police forces behaved as expected and cleared out important roadblocks. In Table 6 we can see on the left that police forces were extremely important in the discovery of civilians, and on the right it is shown that a buried police force was released from the debris.

Table 6: Police forces – Civilian related data and Agent related data.

Civilian Related Data			
Legend			
Percentage of Civilians Discovered	Total Number of Civilians		
Total Number of Civilians Discovered	70		
Fraction of Those Discovered by Police Forces			
Civilians Discovered			
Time	Perc. of Civs. Discovered	No. Civs. Discovered	Discovered by Police Forces
0	5,71%	4	1
1	5,71%	4	1
2	5,71%	4	1
3	5,71%	4	1
4	7,14%	5	2
5	7,14%	5	2
6	7,14%	5	2
7	12,86%	9	4
8	22,86%	16	7
9	22,86%	16	7
10	22,86%	16	7
20	32,86%	23	13
30	45,71%	32	20
40	47,14%	33	20
50	51,43%	36	23
60	54,29%	38	25
70	57,14%	40	27
80	60,00%	42	29
90	60,00%	42	29
100	61,43%	43	30
110	68,57%	48	35
120	72,86%	51	38
130	72,86%	51	38
140	72,86%	51	38
150	74,29%	52	39
160	74,29%	52	39
170	75,71%	53	40
180	75,71%	53	40
190	75,71%	53	40
200	77,14%	54	40
210	78,57%	55	41
220	80,00%	56	42
230	80,00%	56	42
240	80,00%	56	42
250	80,00%	56	42
260	81,43%	57	43
270	81,43%	57	43
280	81,43%	57	43
290	82,86%	58	44
300	82,86%	58	44

Agent Related Data		
Legend		
Police Forces Killed by Fire (Not Buried)	Total Number of Police Forces	
Number of Buried Police Forces (Alive)	10	
Agent Related Data		
Time	Police Forces Killed by Fire (Not Buried)	No. Buried Police Forces (Alive)
0	0	N/A
1	0	N/A
2	0	1
3	0	1
4	0	1
5	0	1
6	0	1
7	0	1
8	0	1
9	0	1
10	0	1
20	0	1
30	0	1
40	0	1
50	0	1
60	0	0
70	0	0
80	0	0
90	0	0
100	0	0
110	0	0
120	0	0
130	0	0
140	0	0
150	0	0
160	0	0
170	0	0
180	0	0
190	0	0
200	0	0
210	0	0
220	0	0
230	0	0
240	0	0
250	0	0
260	0	0
270	0	0
280	0	0
290	0	0
300	0	0

The ambulance relevant data is displayed on Table 7 and Table 8. We can clearly see from these statistics that ambulances are underpowered, and a critical piece in this map. From the six present ambulances, three start the simulation buried under debris. Rescuing other ambulances should have been the number one priority, and while this objective was accomplished it took some time. For some reason, the ambulances were unable to rescue many civilians. The rapid spread of fire contributed to this. However, the amount of civilians deceased is extremely high.

Table 7: Ambulance Teams – Agent related data.

Agent Related Data												
Legend											Total Number of Ambulances	
Buried Police Forces Killed by Debris			Buried Police Forces Killed by Fire			Buried Police Forces (Alive)			Ambulances Killed by Fire (Not Buried)		Total Number of Rescued Agents Unburied	
Buried Ambulances Killed by Debris			Buried Ambulances Killed by Fire			Buried Ambulances (Alive)			Total Number of Buried Rescued Agents (Alive)		6	
Buried Fire Brigades Killed by Debris			Buried Fire Brigades Killed by Fire			Buried Fire Brigades (Alive)						
Time	Buried Agents Killed by Debris			Buried Agents Killed by Fire			Living Buried Agents			Ambs. Killed Fire (Not Buried)	Agents Unburied	Buried Rescued Agents (Alive)
	Police	Ambulance	Fire Brigade	Police	Ambulance	Fire Brigade	Police	Ambulance	Fire Brigade			
0	0	0	0	0	0	0	N/A	N/A	N/A	0	0	N/A
1	0	0	0	0	0	0	N/A	N/A	N/A	0	0	N/A
2	0	0	0	0	0	0	1	3	2	0	0	6
3	0	0	0	0	0	0	1	3	2	0	0	6
4	0	0	0	0	0	0	1	3	2	0	0	6
5	0	0	0	0	0	0	1	3	2	0	0	6
6	0	0	0	0	0	0	1	3	2	0	0	6
7	0	0	0	0	0	0	1	3	2	0	0	6
8	0	0	0	0	0	0	1	3	2	0	0	6
9	0	0	0	0	0	0	1	3	2	0	0	6
10	0	0	0	0	0	0	1	3	2	0	0	6
20	0	0	0	0	0	0	1	3	2	0	0	6
30	0	0	0	0	0	0	1	3	1	0	1	5
40	0	0	0	0	0	0	1	3	1	0	1	5
50	0	0	0	0	0	0	1	3	1	0	1	5
60	0	0	0	0	0	0	0	2	1	0	3	3
70	0	0	0	0	0	0	0	2	1	0	3	3
80	0	0	0	0	0	0	0	2	1	0	3	3
90	0	0	0	0	0	0	0	0	1	0	5	1
100	0	0	0	0	0	0	0	0	1	0	5	1
110	0	0	0	0	0	0	0	0	1	0	5	1
120	0	0	0	0	0	0	0	0	1	0	5	1
130	0	0	0	0	0	0	0	0	1	0	5	1
140	0	0	0	0	0	0	0	0	1	0	5	1
150	0	0	0	0	0	0	0	0	1	0	5	1
160	0	0	0	0	0	0	0	0	1	0	5	1
170	0	0	0	0	0	0	0	0	1	0	5	1
180	0	0	0	0	0	0	0	0	1	0	5	1
190	0	0	0	0	0	0	0	0	1	0	5	1
200	0	0	0	0	0	0	0	0	1	0	5	1
210	0	0	0	0	0	0	0	0	1	0	5	1
220	0	0	0	0	0	0	0	0	1	0	5	1
230	0	0	0	0	0	0	0	0	1	0	5	1
240	0	0	0	0	0	0	0	0	1	0	5	1
250	0	0	0	0	0	0	0	0	1	0	5	1
260	0	0	0	0	0	0	0	0	1	0	5	1
270	0	0	0	0	0	0	0	0	1	0	5	1
280	0	0	0	0	0	0	0	0	1	0	5	1
290	0	0	0	0	0	0	0	0	1	0	5	1
300	0	0	0	0	0	0	0	0	1	0	5	1

Table 8 : Ambulance Teams – Civilian related data.

Civilian Related Data								
Legend								Total Number of Civilians
Percentage of Civilians Discovered			Civilians Killed by Debris		No. of Living Civilians Buried Under Debris		Total Number of Civilians	
Total Number of Civilians Discovered			Civilians Killed by Fire		No. of Civilians Rescued by Ambulance Teams		70	
Fraction of Those Discovered by Ambulances			Percentage of Civilians Killed				Number of Refugees	
							1	
Time	Civilians Discovered			Civilians Killed			Living Civs. Buried	Number of Civilians Rescued
	Peroc. of Civs. Discovered	No. Civs. Discovered	Discovered by Ambs.	By Debris	By Fire	Peroc. Civs. Killed		
0	5,71%	4	1	0	0	0,00%	N/A	0
1	5,71%	4	1	0	0	0,00%	N/A	0
2	5,71%	4	1	2	0	2,86%	45	0
3	5,71%	4	1	2	0	2,86%	45	0
4	7,14%	5	1	2	0	2,86%	45	0
5	7,14%	5	1	2	0	2,86%	45	0
6	7,14%	5	1	2	0	2,86%	45	0
7	12,86%	9	1	2	0	2,86%	45	0
8	22,86%	16	2	2	0	2,86%	45	0
9	22,86%	16	2	2	0	2,86%	45	0
10	22,86%	16	2	2	0	2,86%	45	0
20	32,86%	23	3	2	0	2,86%	45	0
30	45,71%	32	5	2	0	2,86%	45	0
40	47,14%	33	5	2	0	2,86%	45	0
50	51,43%	36	5	2	0	2,86%	45	0
60	54,29%	38	5	2	0	2,86%	45	0
70	57,14%	40	5	4	1	7,14%	42	0
80	60,00%	42	5	7	2	12,86%	38	0
90	60,00%	42	5	8	2	14,29%	37	0
100	61,43%	43	5	8	2	14,29%	37	0
110	68,57%	48	5	8	2	14,29%	37	0
120	72,86%	51	5	11	2	18,57%	33	1
130	72,86%	51	5	11	2	18,57%	33	1
140	72,86%	51	5	12	2	20,00%	32	1
150	74,29%	52	5	13	2	21,43%	31	1
160	74,29%	52	5	15	3	25,71%	28	1
170	75,71%	53	5	18	3	30,00%	25	1
180	75,71%	53	5	18	4	31,43%	24	1
190	75,71%	53	5	18	5	32,86%	23	1
200	77,14%	54	5	19	5	35,71%	21	1
210	78,57%	55	5	19	5	35,71%	21	1
220	80,00%	56	5	22	7	41,43%	17	1
230	80,00%	56	5	22	8	42,86%	16	1
240	80,00%	56	5	23	8	44,29%	15	1
250	80,00%	56	5	24	10	48,57%	12	1
260	81,43%	57	5	24	11	50,00%	11	1
270	81,43%	57	5	25	12	52,86%	9	1
280	81,43%	57	5	27	12	55,71%	7	1
290	82,86%	58	5	27	12	55,71%	7	1
300	82,86%	58	5	27	12	55,71%	7	1

The civilian specific information is displayed on Table 9 and Table 10. These tables are mostly redundant, grouping information covered in previous tables. It can be confirmed in Table 9 the major role played by police forces in discovering civilians and Table 10 displays the large amount of casualties. The amount of civilians rescued from collapsed buildings is disappointingly low.

Table 9: Civilians – Civilians discovered

Civilians Discovered						
Legend						
Percentage of Civilians Discovered		Fraction of Those Discovered by Firebrigades			Total Number of Civilians	
Total Number of Buildings Explored		Fraction of Those Discovered by Police Forces			70	
Percentage of Buildings Explored		Fraction of Those Discovered by Ambulances				
*Note: If two or more types of agent find a civilian in the same cycle, they all get credit for it.						
Time	Bldgs. Explored	Civs. Discovered	Discovered by FBs.*	Discovered by Police Forces*	Discovered by Ambs.*	Perc. of Civs. Discovered
0	5,68%	4	2	1	1	5,71%
1	5,68%	4	2	1	1	5,71%
2	5,68%	4	2	1	1	5,71%
3	5,68%	4	2	1	1	5,71%
4	7,70%	5	2	2	1	7,14%
5	8,24%	5	2	2	1	7,14%
6	9,32%	5	2	2	1	7,14%
7	9,73%	9	4	4	1	12,86%
8	10,41%	16	8	7	2	22,86%
9	10,95%	16	8	7	2	22,86%
10	11,35%	16	8	7	2	22,86%
20	16,49%	23	8	13	3	32,86%
30	20,14%	32	8	20	5	45,71%
40	25,41%	33	9	20	5	47,14%
50	28,92%	36	9	23	5	51,43%
60	34,05%	38	9	25	5	54,29%
70	38,24%	40	9	27	5	57,14%
80	41,62%	42	9	29	5	60,00%
90	45,54%	42	9	29	5	60,00%
100	47,57%	43	9	30	5	61,43%
110	51,76%	48	9	35	5	68,57%
120	55,27%	51	9	38	5	72,86%
130	56,89%	51	9	38	5	72,86%
140	58,78%	51	9	38	5	72,86%
150	60,54%	52	9	39	5	74,29%
160	60,81%	52	9	39	5	74,29%
170	61,35%	53	9	40	5	75,71%
180	61,76%	53	9	40	5	75,71%
190	61,76%	53	9	40	5	75,71%
200	62,03%	54	10	40	5	77,14%
210	62,84%	55	10	41	5	78,57%
220	63,78%	56	10	42	5	80,00%
230	64,19%	56	10	42	5	80,00%
240	64,19%	56	10	42	5	80,00%
250	64,19%	56	10	42	5	80,00%
260	64,73%	57	10	43	5	81,43%
270	65,41%	57	10	43	5	81,43%
280	65,54%	57	10	43	5	81,43%
290	65,95%	58	10	44	5	82,86%
300	65,95%	58	10	44	5	82,86%

Table 10: Civilians – Civilians killed and Civilians rescued

Civilians Killed				
Legend				
Civilians Killed By Debris			Final Number of Casualties	
Civilians Killed By Fire			39	
Total Number of Civilians Killed				
Percentage of Civilians Killed				
Time	By Debris	By Fire	Total	Perc. Civs. Killed
0	0	0	0	0.00%
1	0	0	0	0.00%
2	2	0	2	2.86%
3	2	0	2	2.86%
4	2	0	2	2.86%
5	2	0	2	2.86%
6	2	0	2	2.86%
7	2	0	2	2.86%
8	2	0	2	2.86%
9	2	0	2	2.86%
10	2	0	2	2.86%
20	2	0	2	2.86%
30	2	0	2	2.86%
40	2	0	2	2.86%
50	2	0	2	2.86%
60	2	0	2	2.86%
70	4	1	5	7.14%
80	7	9	16	21.05%
90	8	6	14	18.42%
100	8	6	14	18.42%
110	8	6	14	18.42%
120	11	2	13	18.57%
130	11	2	13	18.57%
140	12	2	14	20.00%
150	13	2	15	21.43%
160	15	3	18	25.71%
170	18	3	21	30.00%
180	18	4	22	31.43%
190	18	5	23	32.86%
200	19	6	25	35.71%
210	19	6	25	35.71%
220	22	7	29	41.43%
230	22	8	30	42.86%
240	23	8	31	44.29%
250	24	10	34	48.57%
260	24	11	35	50.00%
270	26	12	37	52.86%
280	27	12	39	55.71%
290	27	12	39	55.71%
300	27	12	39	55.71%

Civilians Rescued				
Legend				
No. of Living Civilians Buried Under Debris			Total Number of Ambulances	
Percentage of Civilians Rescued			6	
Number of Rescued Civilians per Ambulance			Number of Refugees	
Number of Civilians Rescued by Ambulance Teams			1	
Time	Living Civs. Buried	Perc. of Civilians Rescued	Rescued Civs. per Amb.	Number of Civilians Rescued
0	N/A	0.00%	0.00	0
1	N/A	0.00%	0.00	0
2	45	0.00%	0.00	0
3	45	0.00%	0.00	0
4	45	0.00%	0.00	0
5	45	0.00%	0.00	0
6	45	0.00%	0.00	0
7	45	0.00%	0.00	0
8	45	0.00%	0.00	0
9	45	0.00%	0.00	0
10	45	0.00%	0.00	0
20	45	0.00%	0.00	0
30	45	0.00%	0.00	0
40	45	0.00%	0.00	0
50	45	0.00%	0.00	0
60	45	0.00%	0.00	0
70	42	0.00%	0.00	0
80	38	0.00%	0.00	0
90	37	0.00%	0.00	0
100	37	0.00%	0.00	0
110	37	0.00%	0.00	0
120	33	1.43%	0.17	1
130	33	1.43%	0.17	1
140	32	1.43%	0.17	1
150	31	1.43%	0.17	1
160	28	1.43%	0.17	1
170	25	1.43%	0.17	1
180	24	1.43%	0.17	1
190	23	1.43%	0.17	1
200	21	1.43%	0.17	1
210	21	1.43%	0.17	1
220	17	1.43%	0.17	1
230	16	1.43%	0.17	1
240	15	1.43%	0.17	1
250	12	1.43%	0.17	1
260	11	1.43%	0.17	1
270	9	1.43%	0.17	1
280	7	1.43%	0.17	1
290	7	1.43%	0.17	1
300	7	1.43%	0.17	1

Chart 1 and Chart 2 were also generated automatically. In Chart 1 a) it can be seen that only 42% of the total score was kept and most was lost due to casualties. In b) it is shown that most civilians died due to sustained injuries from collapsed buildings, but a rather significant part (17%) were burned alive. From Chart 1 c) and d) and from Chart 2 it is seen that police forces did have a large importance in this map's exploration and civilian discovery. This can either mean that the police forces explored the map in a very efficient way, or that the other agents did not. That information can only be derived from team comparison.

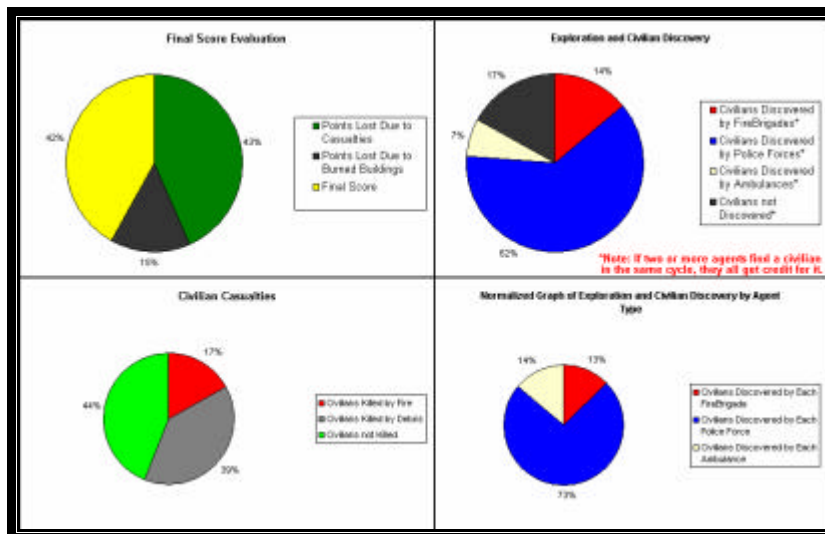


Chart 1: a) Final score evaluation; b) Civilian casualties; c) Exploration and civilian discovery; d) Normalized graph of exploration and civilian discovery.

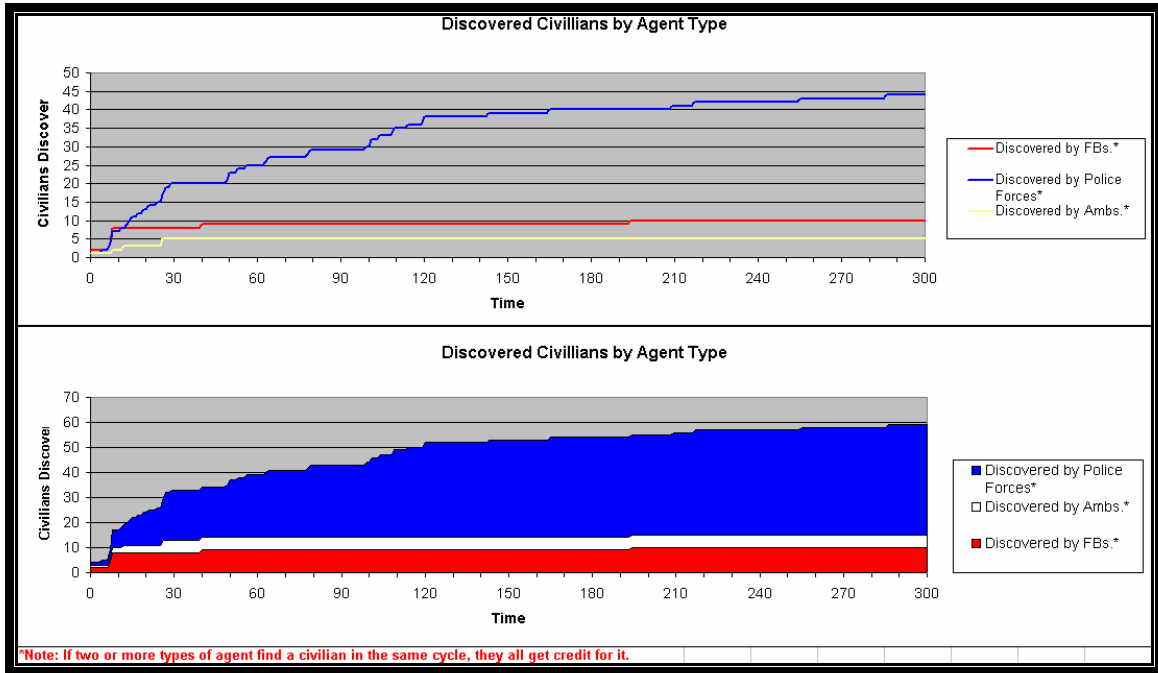


Chart 2: a) Discovered civilians by agent type; b) Discovered civilians (stacked)

In Table 11 a summary of the team’s performance is presented, resuming most of what was seen in this chapter. As can be perceived inferred from previous tables and charts, our tool makes it fairly easy to analyze any team’s behavior. There is still a lot to improve on our agents, and much effort must be put into code development in order to obtain better scores.

Table 11: Simulation summary.

Team: FCPortugal		Map: KobeHard	
Score Data		Civilians Discovered	
Total Number of Agents	99	Civilians Discovered by FireBrigades*	10
Number of Agents not Killed	58	Civilians Discovered by Police Forces*	44
Percentage of Building Area Destroyed	31,19%	Civilians Discovered by Ambulances*	5
		Civilians not Discovered*	12
Initial Score	100,00	*Note: As explained in the civilians tab, these parts may not add up to the total number of civilians.	
Points Lost Due to Casualties	43,18		
Points Lost Due to Burned Buildings	14,98		
Final Score	41,84		
Casualties		Civilians Discovered (Normalized by Agent Type)	
Total Number of Civilians	70	Total Number of FireBrigades	13
Civilians Killed by Fire	12	Total Number of Police Forces	10
Civilians Killed by Debris	27	Total Number of Ambulances	6
Civilians not Killed	31		
Total Number of Rescue Agents Killed	2	Civilians Discovered by Each FireBrigade	0,77
Simulation date: 07 15 10:29:20		Civilians Discovered by Each Police Force	4,40
		Civilians Discovered by Each Ambulance	0,83

4.4 Team Comparison

The tables from the previous section provide us only rough guidelines of the team's flaws and do not make it obvious what the coding priorities should be. In order to acquire that knowledge, we should compare our team with the current RoboCupRescue World Champions – team Impossibles, from Iran. This will allow us to perceive the most important differences. Since the competition has an open source policy, it also allows any team to analyze its weaknesses and, finding a team which performs better, adapt their strategies. For this purpose another custom spreadsheet was developed – this one oriented towards team comparison.

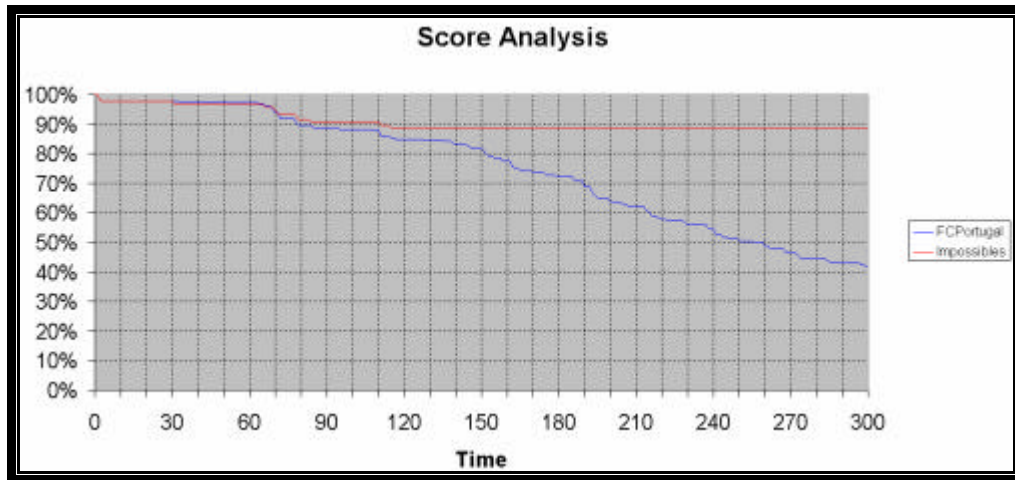


Chart 3: Score.

Simulation score for both teams and its evolution through time is displayed on Chart 3. As can be seen, the overall score from team Impossibles is much higher. Although in the early stages of the simulation FC Portugal held a small lead, the champions quickly gained a sizeable advantage.

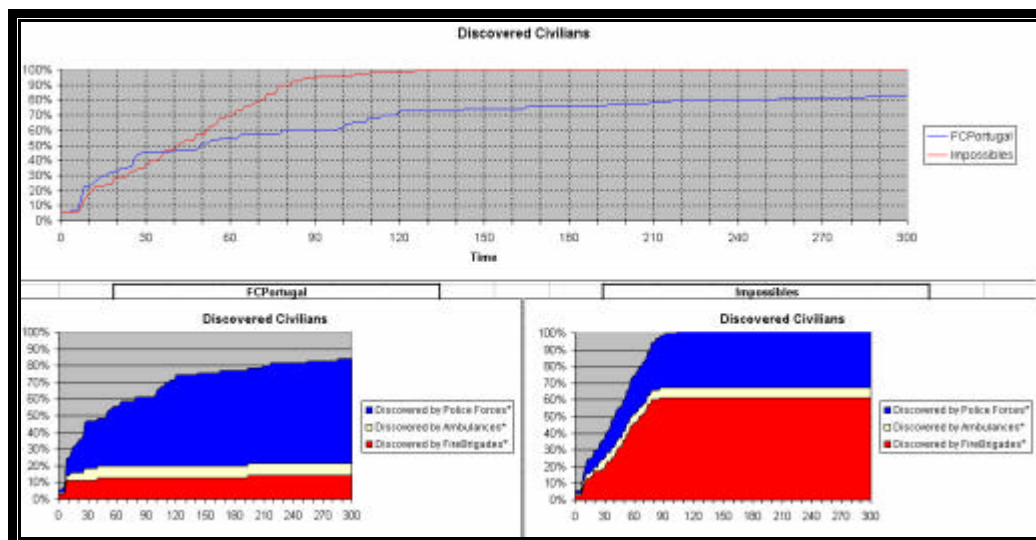


Chart 4: : a) Discovered Civilians; b) FC Portugal - Discovered Civilians by Agent Type (stacked); c) Impossibles - Discovered Civilians by Agent Type (stacked).

On Chart 4, civilian discovery is compared between both teams. Once again, although FC Portugal held a small edge in the first cycles, the Impossible quickly gained a wide margin. The main difference in exploration strategy can be easily identified. While the champions' fire brigades were used as scouts, ours were too busy trying, unsuccessfully, to control the fire expansion.

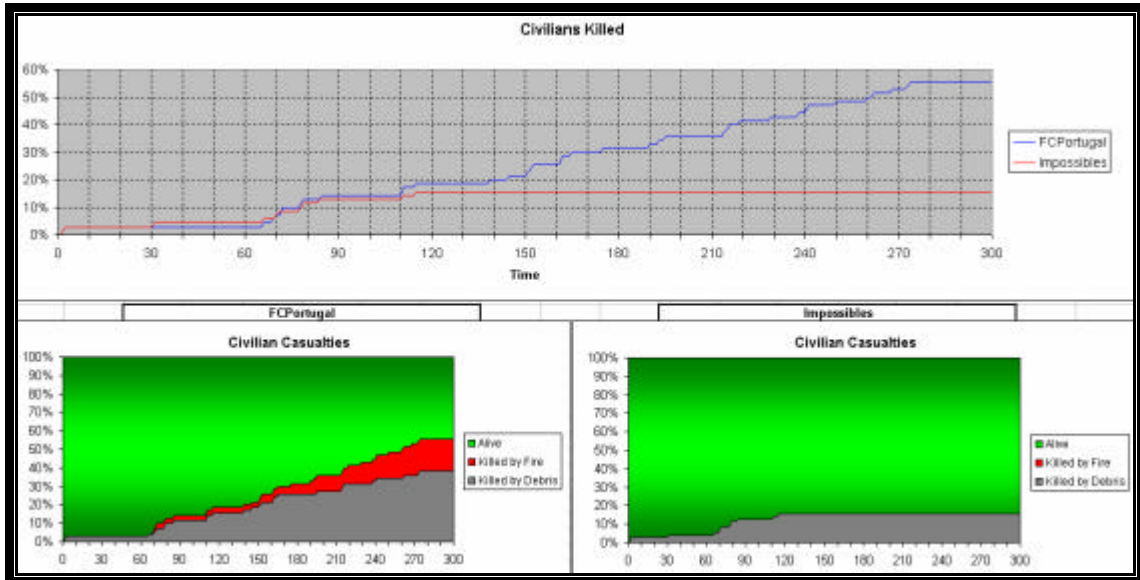


Chart 5: a) Civilians Killed; b) FC Portugal - Civilians casualties (stacked); c) Impossible - Civilians casualties (stacked).

The enormous difference in civilian casualties becomes clear on Chart 5. While no civilians were burned to death on our opponent's simulation, around 17% of the civilians present in FC Portugal's simulation died by exposure to fire. The Impossible Ambulances were also undoubtedly more productive as the number of casualties caused by collapses was also extremely lower.

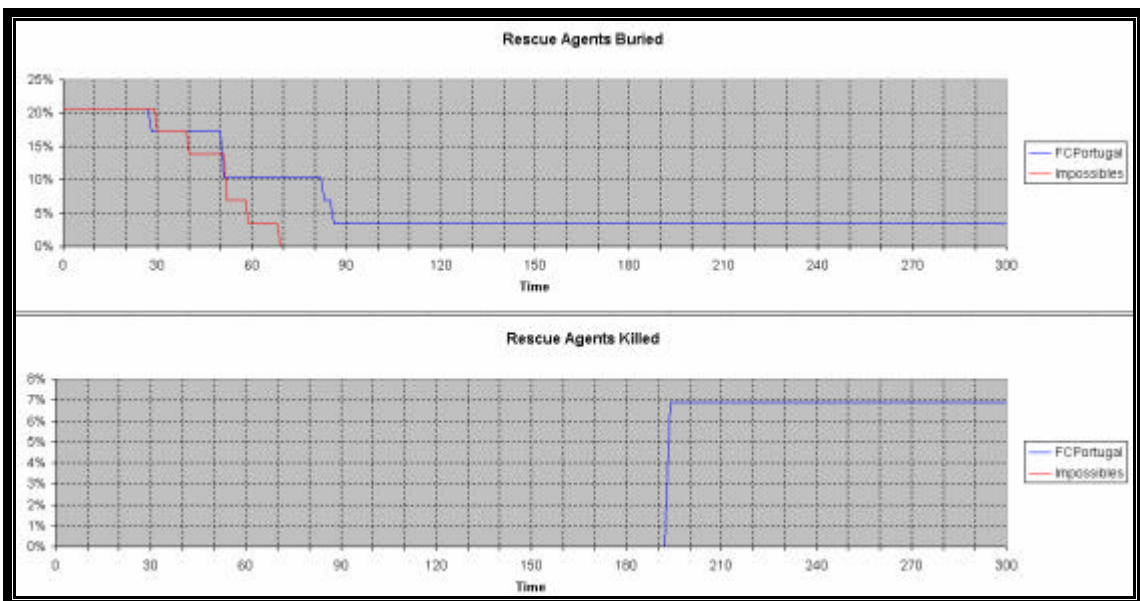


Chart 6 : a) Rescue agents buried; b) Rescue agents killed.

Chart 6 confirms what was commented on section 4.3, in regards to trapped rescue agents. Rescuing these agents must be a top priority, as the champions demonstrate. By cycle number seventy all the 'Impossible's' rescue agents have been released, while FC Portugal takes longer to free its agents and maintains a trapped fire brigade throughout the whole simulation. As for agent casualties, it was already referred in section 4.3 that agent's must be programmed with better safeguards to prevent their own demise.



Chart 7: a) Buildings on fire; b) Building area destroyed.

The analysis made on Chart 4 proves correct, as it can be seen on Chart 7 that our opponents are able to control the fire within the first sixty simulation cycles. This allows their fire brigades to pursue other tasks: namely exploration towards civilian discovery.

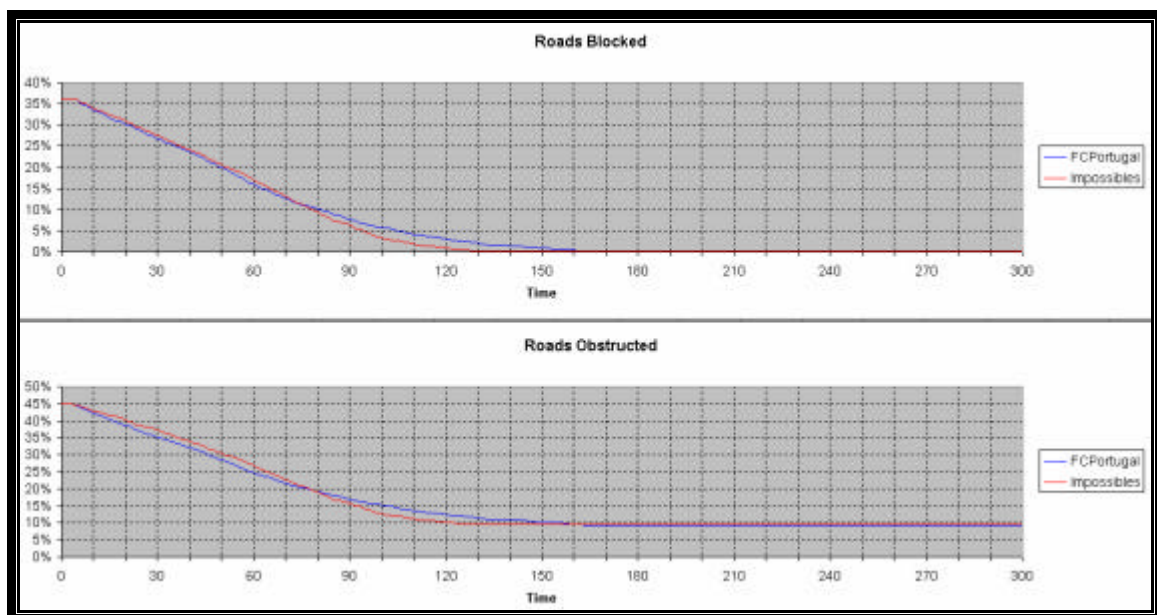


Chart 8: a) Roads blocked; b) Roads obstructed.

From the data in Chart 8, it is reasonable to assume that there is no large difference in police force capabilities. Both teams' agents perform their tasks as expected.

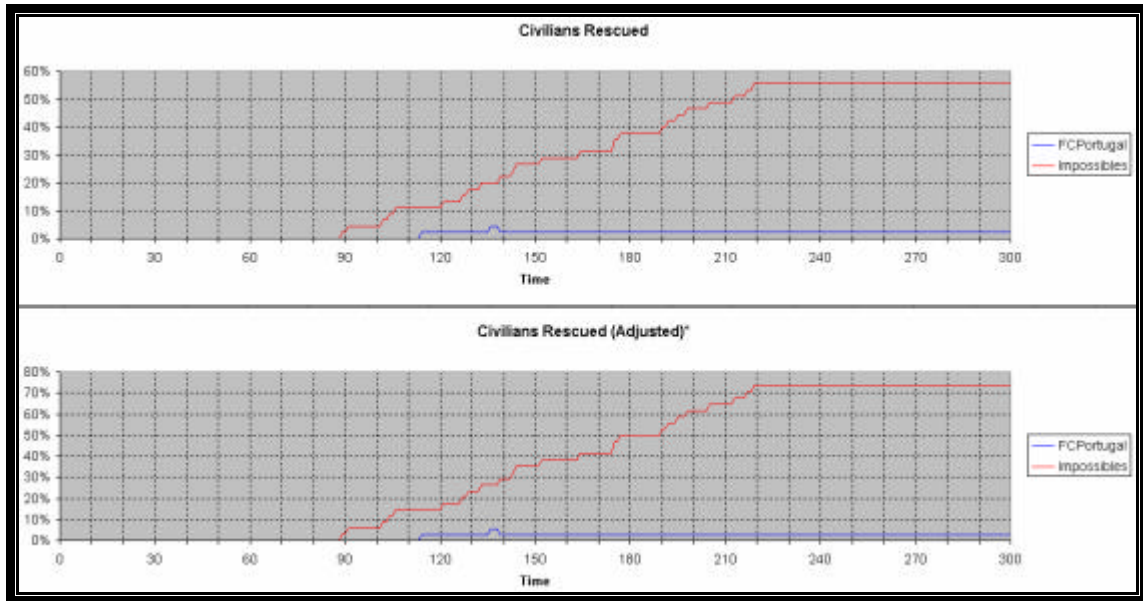


Chart 9: a) Civilians rescued; b) Civilians rescued adjusted.

On the other hand, the Impossible ambulances are extremely more successful as was predicted in the analysis of Chart 5. From Chart 9 we can see that, not only do their ambulances work in an efficient way, in order to save the maximum number of civilians, they also choose wisely which civilians not to rescue. This can be discerned by comparing Chart 9 a) and Chart 9 b). In chart b) buried living civilians at the end of the simulation are discarded from the “unsaved civilians” count. This is because of a small flaw in current simulation rules, which allows these unsaved civilians not to influence the score negatively.

Table 12: Comparison summary.

Display Name:	FCPortugal	Impossible	*You may change the Display Names as you please		
Team:	FCPortugal	Impossible			
Map:	KobeHard	KobeHard			
Date:	07/15	07/15			
Time:	10:29:20	10:58:55			
Score Data			Civilians Discovered		
	FCPortugal	Impossible	FCPortugal	Impossible	
Total Number of Agents	99	99	Civilians Discovered by FireBrigades*	10	43
Number of Agents not Killed	58,59%	88,89%	Civilians Discovered by Police Forces*	44	26
Percentage of Building Area Destroyed	31,19%	0,39%	Civilians Discovered by Ambulances*	5	4
			Civilians not Discovered*	12	0
Initial Score	100,00	100,00	*Note: As explained in the civilians tab, these parts may not add up to the total number of civilians.		
Points Lost Due to Civilian Casualties	43,18%	11,29%			
Points Lost Due to Burned Buildings	14,98%	0,19%			
Final Score	41,84%	88,51%			
Civilian Casualties			Civilians Discovered (Normalized by Agent Type)		
	FCPortugal	Impossible	FCPortugal	Impossible	
Total Number of Civilians	70	70	Total Number of FireBrigades	13	13
Civilians Killed by Fire	17,14%	0,00%	Total Number of Police Forces	10	10
Civilians Killed by Debris	38,57%	15,71%	Total Number of Ambulances	6	6
Civilians not Killed	44,29%	84,29%	Civilians Discovered by Each FireBrigade	0,77	3,31
			Civilians Discovered by Each Police Force	4,40	2,60
			Civilians Discovered by Each Ambulance	0,83	0,67

Table 12 provides us with a summary of team's comparison. Some of the facts commented on before are assembled here, allowing for some conclusions to be consolidated. The necessity to improve the fire brigades becomes obvious for three primary reasons: reduce the amount of points lost to burned buildings; reduce the amount of points due to burned civilians; and allow the fire brigades to act as scouts when they are able to rapidly control the fires. On the other hand ambulances require better delineated rescue priorities and an improved algorithm to estimate civilian life expectancy.

4.5 Final Considerations

The FCPx tool introduction was a success. Its usefulness was proved and its integration into Freiburg's 3D viewer future releases is expected. From section 4.3 and 4.4 it is shown how conclusions can be drawn and directions set by a simple analysis of tables and charts. What could, before, take several simulations and great attention to be determined, can now be done in minutes.

The tool's webpage has seen quite some traffic for such a restricted community. The main page had over four hundred visits and the download page has been seen over a hundred times. It was, nevertheless, expected, since FCPx fills a gap which had been previously mentioned several times in on-line discussion.

Information on this chapter was aggregated in Rescue Technical Report II – FCPx , A Tool for Agent Evaluation and Comparison [18].

Chapter 5

5 Conclusion and Future Work

The simulator package is an incredible piece of software, but being supported by such an extremely complex environment makes the RoboCupRescue competition prone to a myriad of problems. Issues about which direction to take are constantly debated, and maintaining an equilibrium is sometimes extremely difficult. This was proven during the final month approaching the 2005 Robocup competition, in which development problems delayed the planned introduction of new rules. Even during the competition, there were instances of simulations being repeated due to simulator bugs. The current discussion (December, 2005) lies with the timings on feature implementation, and it appears that new features will now be introduced only once a year, although nothing has yet been decided. A new issue will surely arise after this one is settled.

The lack of standardization is also a problem in the RoboCupRescue environment. Since every team uses its own method for code integration, and most codes have little to no modularity, a great part of the benefits of an open source project are lost. As an example, if strict communication standards were to be created, that would enable the fire brigade agents from a team to cooperate with police forces from a second team and ambulances from another, with little to no modifications of the code. Allowing code portability in this way would lead to faster developments and more complex strategies, since less time would be required to adapt ideas, previously developed by different teams. Presently, if a team wishes to adapt an improved path finding algorithm, from a different team, a lot of code has to be adapted or even rewritten. This simple task involves a lot of time and isn't usually worth the effort due to the modest gains achieved in such a change. However, if there was standard interface with path finding modules, any team could just adapt a new approach that proved to be even narrowly better as long as the standard was respected.

Related with the core objectives of the project is one other major concern – conformity with reality. Some of the current simulation rules have no connection to the real world. It has been discussed in Section 2 that agents are unable to obtain the temperature of their

surroundings, and this problem is deeply rooted in the system. Others exist, and some have existed unnoticed for a long time. One accepted “feature” on the system is completely dissociated from reality – the fact that civilians may survive trapped during the entire simulation length. The three day duration was chosen, for this is considered the essential period after a major disaster. After this deadline, the probability of finding survivors drops drastically. This fact is connected, amongst other things, with the human biological need for water – a life limiting factor in catastrophes. While it is acceptable that a civilian may survive under debris for a period of over three days, it makes no sense that there is no incentive to save these people since, if they survive till the end of the simulation, their contribution to the score is kept. This artificial deadline disconnects the simulated world from a continuous reality, in which these people would eventually die.

The participation in RoboCup Osaka 2005 was a success from several angles, allowing a deeper understanding of the competition and its community.

On a more personal note, this entire project was a great endeavor, of which we are proud to have been a part. Although intense and time consuming, it was rewarding to start a team that we hope will eventually achieve great results, and to contribute to the Rescue community, which is filled with very nice, and extremely intelligent people.

Robots developed for the RoboCupRescue Robot League were recently used in the city of New Orleans. We hope to one day see practical applications to the simulation competition in real world scenarios – and to know that we helped develop this immense project to save lives.

Bibliography

1. Kitano, H., S. Tadokoro, I. Noda, H. Matsubara, T. Takahashi, A. Shinjou, and S. Shimada, *RoboCup Rescue: search and rescue in large-scale disasters as a domain for autonomous agents research*," in *Systems, Man, and Cybernetics, 1999. IEEE SMC '99 Conference Proceedings. 1999 IEEE International Conference on*, Vol. 6, pp. 739-743, 1999
2. Reinaldo, F., J. Certo, N. Cordeiro, L.P. Reis, R. Camacho, and N. Lau, *Applying Biological Paradigms to Emerge Behaviour in RoboCup Rescue Team*," in *Progress in Artificial Intelligence: 12th Portuguese Conference on Artificial Intelligence, EPIA*, Springer-Verlag, LNCS, Vol. 3808, pp. 422 - 434, Covilha, Portugal, 5 - 8 December, 2005
3. Paquet, S., N. Bernier, and B. Chaib-draa, *DAMAS-Rescue Description Paper*," in *RoboCup-2004: Robot Soccer World Cup VIII*, Springer Verlag, Berlin, Vol. 3276, pp., 2004
4. Reis, L.P., *Coordination in Multi-Agent Systems: Applications in University Management and Robotic Soccer*, Department of Engineering, 2003, University of Porto, Porto
5. Russell, S. and P. Norvig, *Artificial Intelligence: A Modern Approach*. 1995: Prentice Hall.
6. Rich, E. and K. Knight, *Inteligência Artificial*. 2a. ed. 1994, São Paulo: Makron Books.
7. Reis, L.P. and N. Lau. *FC Portugal Rescue Team Site*. 2005 December 2005 [cited 2005 December]; Available from: <http://paginas.fe.up.pt/~lpreis/fcportugal/rescue/>.
8. Committee, R., *RoboCup Rescue simulator manual v0.4*, 2000, The RoboCup Rescue Technical Committee
9. Akin, H.L., C. Skinner, J. Habibi, T. Koto, and S.L. Casio, *Robocup 2004 Rescue Simulation League Rules V1.01*. 2004, Robocup 2004 Rescue Simulation League Technical Committee.
10. Takahashi, T., I. Takeuchi, K. Tetsuhiko, S. Tadokoro, and I. Noda, *RoboCup-Rescue Disaster Simulator Architecture*," in *Robocup 2000: Robot Soccer World Cup IV*, Springer-Verlag GmbH, Vol., pp. 379-284, 2000
11. Nüssle, T., A. Kleiner, and M. Brenner, *Approaching Urban Disaster Reality: The ResQ Firesimulator*, 2004, Universität Freiburg, Institut für Informatik
12. Skinner, C., M. Brenner, S. Paquet, N. Ito, and A. Rahimi, *Robocup 2005 Rescue Simulation League Rules*. 2005, Robocup 2005 Rescue Simulation League Technical Committee.
13. Cordeiro, N., J. Certo, F. Reinaldo, L.P. Reis, R. Camacho, and N. Lau, *Rescue Technical Report I - Simulator System*, 2005, LIACC(NIAD&R) and IEETA
14. Amraii, S.A., B. Behsaz, H. Gheibi, M. Izadi, H. Janzadeh, F. Molazem, A. Rahimi, M.T. Ghinani, and H. Vosoughpour, *S.O.S. 2004: an attempt towards a multi-agent rescue team*, 2004, Amirkabir Univ. of Technology
15. Bowling, M., *Robocup Rescue: Agent Development Kit*, Dept. of Comp. Science, Univ. of Alberta
16. KANEDA T., M.F., TAKAHASHI H., MATSUI T., ATSUMI M., HATAYAMA M., TAYAMA K. CHIBA R., TAKEUCHI K., *Simulator Complex for RoboCup Rescue Simulation Project - As Test-Bed for Multi-agent Organizational Behaviour in Emergency Case of Large-Scale Disaster*. p. 6.
17. Lau, N., L.P. Reis, and F. Reinaldo, *FC Portugal 2005 Rescue Team Description: Adapting Simulated Soccer Coordination Methodologies to the Search and Rescue Domain*, 2005, Univ. of Porto
18. Certo, J., N. Cordeiro, F. Reinaldo, L.P. Reis, R. Camacho, and N. Lau, *Rescue Technical Report II - FCPx, A Tool for Agent Evaluation and Comparison*, 2005, LIACC(NIAD&R) and IEETA

Appendixes

1 Apêndice A: Instalação do Simulador

1.1 Processo de Instalação do Simulador

O simulador pode ser encontrado em: <http://sourceforge.net/projects/roborescue/>

Foi utilizada a versão 0.47.

Após descomprimir o ficheiro descarregado, ir à pasta “program” e executar o comando make. No caso de ocorrerem problemas ir a cada uma das sub-pastas e executar o comando make. Se a mensagem for do tipo: make: “blockadessimulator” está actualizado passar para a subpasta seguinte.

Para o desenvolvimento do projecto, o simulador foi instalado em três terminais diferentes onde surgiram diversos problemas.

1.1.1 Problemas encontrados:

1.1.1.1 Primeira Instalação

Inicialmente foi utilizado um computador de secretária, com um processador Pentium III.

O sistema operativo utilizado foi o Mandrake 10.0 (Sistema operativo actualizado automaticamente - 24/03/2005) com ferramentas de desenvolvimento C++ incluindo o gcc 3.3.

Foram encontrados os seguintes problemas:

civillian

Problema:

```
/usr/bin/ld: cannot find -lstdc++
```

```
collect2: ld returned 1 exit status
```

```
make[1]: ** [civilian] Erro 1
```

make[1]: Leaving directory `/home/joao/projecto/rescue-0_46alpha-unix/program/civilian/Civilian'

make: ** [all] Erro 2

Solução:

instalar pacote:

libstdc++static-devel

Problema:

/usr/bin/ld: cannot find -lm

collect2: ld returned 1 exit status

make: ** [civilian] Erro 1

Solucao:

instalar pacote:

glibc-staic-devel

viewer:

problemas:

viewer/Viewer.java: In class `viewer.Viewer':

viewer/Viewer.java: In constructor `()':

viewer/Viewer.java:31: error: No method named `setExtendedState' in scope.

```
    setExtendedState(JFrame.MAXIMIZED_BOTH);
```

^

viewer/Viewer.java: In method `viewer.Viewer.addDoAnimateCheckBox(javax.swing.JPanel)':

viewer/Viewer.java:71: error: Can't find constructor `javax.swing.JCheckBox(Ljava/lang/String;Z)' in type `javax.swing.JCheckBox'.

```
    final JCheckBox doAnimateCheckBox = new JCheckBox("Animaition", true);
```

^

viewer/Viewer.java: In method `viewer.Viewer.statusPanel()':

viewer/Viewer.java:82: error: Can't find method `deriveFont(II)' in type `java.awt.Font'.

```
m_statusLabel.setFont(m_statusLabel.getFont().deriveFont(Font.PLAIN, 20));
```

^

3 errors

Solução:

instalar pacotes:

jikes

kaffe

Problema: (versão antiga de javac)

viewer/Viewer.java:12: error:Cannot find class "JFrame" [JLS 8]

make: ** [all] Erro 1

Solução:

instalar:

Java(TM) 2 SDK, Standard Edition

editar ficheiro Makefile inserindo o path do novo javac

antes da variavel

ex: /usr/java/j2sdk1.4.2_07/bin/(\$JAVAC) ...

Problema:

SimpleSexp2Lexer.cc:540: error: syntax error before `::' token

SimpleSexp2Lexer.cc:543: error: register name not specified for `char*yy_cp'

SimpleSexp2Lexer.cc:551: error: syntax error before `if'

SimpleSexp2Lexer.cc:572: error: ISO C++ forbids declaration of `

yy_load_buffer_state' with no type

.

.

.

SimpleSexp2Lexer.cc:213: warning: `void yy_flex_free(void*)' declared `static'

```
but never defined
make[2]: *** [SimpleSexp2Lexer.o] Error 1
make[2]: Leaving directory `/home/joao/Projecto/rescue-0_46alpha-unix/program/civilian/itk'
make[1]: *** [itk/libitk.a] Error 2
make[1]: Leaving directory `/home/joao/Projecto/rescue-0_46alpha-unix/program/civilian/Civilian'
make: *** [all] Error 2
```

Solução: Instalar pacote flex (instalado 2.5.4a)

traffic:

Problema:

```
make: javac: Comando não encontrado
make: ** [all] Erro 127
```

Solução:

Instalar suporte de Java para gcc :
ex: gcc-java-3.3.2-6mdk

Viewer 3D

Esta aplicação não é compilada quando se executa o comando make da pasta programs.

Para compilar é necessário ir à pasta “program/viewer/3Drescue”

Problema:

```
checking for main in -logsg... no
configure: error: Can't find OSG library
See `config.log' for more details.
```

Solução:

Fazer download do pacote OpenSceneGraph (e dependencias: Producer e OpenThreads)

Página do Projecto:

<http://sourceforge.net/projects/openscenegraph>

Download(versão usada 0.9.7-3): http://prdownloads.sourceforge.net/openscenegraph/OSG_OP_OT-0.9.7-3.tar.gz?download

seguir instruções de instalação incluídas no ficheiro README.txt incluídas no pacote.

1.1.1.2 Segunda Instalação:

O simulador foi também instalado num portátil Compaq Presario X1000, com uma versão customizada da distribuição Gentoo de nome x1000v3 onde surgiram as mesmas dificuldades, sendo resolvidas da mesma forma.

1.1.1.3 Terceira Instalação:

Instalação em Suse 9.2 num Portátil Acer Aspire 1694 wlmi

Pacotes instalados:

Ferramentas de desenvolvimento C e C++

Java sdk

Flex

e OSG+OT+OP (que requer: X11-devel; glx; libs-devel de : jpeg,gif,tiff, png)

1.2 Apêndice B: Execução do Simulador

1.2.1 Execução do Simulador

Os scripts do simulador estão todos na pasta boot. Os scripts foram alterados ou criados para a versão 0.47. Para correr o simulador existem 4 scripts:

all.sh
all2.sh
run.sh
game.sh

—
all.sh
pode ser executado simplesmente:
ex:
./all.sh

ou só com o parametro mapa (tal e qual o nome do directorio)
ex:

`./all.sh Foligno`

ou com dois parametros (mapa e nomedoficheiro log)

O nome do arquivo a ser gerado com extensão `.log` pode ser passado por parametro no arquivo `all.sh`, através da linha de comando:

`./all.sh nomedomapa nomedoarquivolog`

ex: `./all.sh Kobe avaliacao`

arquivo gerado: `0621-111517-avaliacao-Kobe.log`

O nome do arquivo é `mmdd-hhmmss-nomedoarquivolog-mapa.log`

—
`all2.sh`

idêntico a `all.sh` mas guarda logs de cada um dos modulos para `ficheiros.log` com o respectivo nome do modulo (ainda não laterado para funcionar)

—
`run.sh`

idêntico a `all.sh` mas sem correr o modulo `viwer` (corre só a simulação guardando num ficheiro log)

—
`game.sh`

chama `all.sh` excepto se o numero de argumentos seja diferente de 2 apresenta um modo de uso do comando

1.2.2 Visualizador em Tempo Real

Para correr o visualizador 2D, temos:

- `all.sh`
- `all2.sh`
- `game.sh`

Para correr o visualizador 3D, pode-se fazer de duas formas:

- executar o `run.sh` seguido do script `2viewer3D.sh`.

Chama o visualizador 3D configurado com as opções de visualizar o formato 2D + 3D com o gráfico estatístico. É configurado para máquina local com opções de desenhar fumo e fogo.

- executar o script fcp_3D.sh (sendo um versão alterada do all.sh para uma versão 3D). Este script acumula script 2viewer3D.sh mais o run.sh. A vantagem é poder correr tudo de uma vez.

Estes dois scripts foram desenhados pelos integrantes do grupo deste trabalho.

Os detalhes adicionais do visualizador 3D são conseguidos através do comando `./3Drescue` na pasta `program/viewer/3Drescue`, (PASSAR ISTO PARA O PORTUGUES)

```
-2D          Use the 2D viewer alone
-3D          Use the 3D viewer alone
-H <hostname> Connect to kernel on host <hostname>
-T <teamname> The teamname to be displayed
-a <filename> Use the action.log from <filename>
-both       Use the 2D and the 3D window (default)
-cf <filename> Use <filename> as secondary datasource for statistics
-d <directory> Load rescue.log and action.log from <directory>
-f <filename> Load from rescue logfile <filename>
-fast       Calculate statistics only
-fd         Draw Fires
-fh         Hide Fires
-h or --help Display this information
-p <port>    Portnumber to connect to kernel, defaults to 6000
-s <number>  Number of graphs, defaults to 1
-sd         Draw Smoke
-sh         Hide Smoke
-t <timepoint> Start from timepoint
```

O processo de navegação no visualizador é como segue abaixo:

3D-View:

Hold the left mousebutton and drag the mouse to rotate the view.
Hold the middle button and drag to pan the view.
Hold the right button and move up/down to zoom.

2D-View:

Click on a building in order to center the 3D-View on this building.

Keyboard controls:

3D/2D-View:

```
q    Quit
,    Reduce speed
.    Increase speed
t    toggle building transparency
f    toggle fires
s    Toggle smoke
```

3D-View:
<space> Center view

Graph window:
<left> cycle trough graphs backwards
<right> cycle trough graphs forwards

1.2.3 Visualização de Logs

O arquivo gerado pelo processo de simulação permite uma análise *offline* do progresso da simulação. Voce pode utilizá-lo para executar com vários visualizadores 2D ou 3D. Isto é interessante, pois podemos analisar todo o processo de simulação de uma equipa que salva seu .log. Os ficheiros .log estão normalmente disponíveis nas páginas das equipas Rescue. Para ver o log funcionar dentro de um visualizador, deve-se digitar:

Visualizadores 2D

O script logviewer permite uma visualização offline do log. Por exemplo, na pasta boot com o log na pasta boot:

```
./logviewer.sh 0621-111517-avaliacao-Kobe.log
```

Visualizadores 3D

O script logviewer permite uma visualização offline do log. Por exemplo, na pasta boot com o log na pasta boot:

```
./logviewer3D.sh 0621-111517-avaliacao-Kobe.log
```

1.2.4 Execução dos Agentes

Para executar os nossos agentes, deve ir a pasta raiz (fc_portugal) e correr o script start.sh com os parâmetros da simulação em causa (num. de agentes). O comando é usado da seguinte forma:

```
USAGE: ./start.sh [na [nf [np [nc [ns [no [host]]]]]] #
# WHERE: na = number of ambulance teams #
# nf = number of fire brigades #
# np = number of police forces #
# nc = number of ambulance centers #
# ns = number of fire stations #
# no = number of police offices #
```

```
# host = host name or ip
```

O host não é obrigatório e se não for preenchido, deve-se usar o local host.

Exemplo:

```
./start.sh 5 10 10 1 1 1
```

Para terminar a execução dos agentes a qualquer momento durante a simulação, deve executar o seguinte script: kill_agent.sh.

1.2.5 Scripts alterados e criados

A seguinte lista de scripts alterados na pasta boot são:

- all.sh

parametros Foram adaptados, variáveis foram acrescentadas, temporizações foram ajustadas para garantir uma correta ordem de execução de módulos e permitir a ligação dos módulos java.

- all2.sh

parametros Foram adaptados, variáveis foram acrescentadas, temporizações foram ajustadas para garantir uma correta ordem de execução de módulos e permitir a ligação dos módulos java.

- run.sh

parametros Foram adaptados, variáveis foram acrescentadas, temporizações foram ajustadas para garantir uma correta ordem de execução de módulos e permitir a ligação dos módulos java.

- 2viewer.sh

Foi alterado para permitir a passagem de host como parâmetro, por ser um módulo java.

- 4morimototrafficsimulator.sh

Foi alterado para permitir a passagem de host como parâmetro, por ser um módulo java.

- 5firesimulator.sh

Foi alterado para permitir a passagem de host como parâmetro, por ser um módulo java.

A seguinte lista de scripts criados na pasta boot são:

- 2viewer3D.sh

Executa o visualizador 3D em tempo real, ligado a máquina local com as opções de nome de equipe, fogo e fumo. Para a opção de nome de equipe, o valor padrão é fc_portugal.

- fcp_3D.sh

Este script acumula script 2viewer3D.sh mais o run.sh

- logviewer3D.sh

Executa o visualizador 3D em offline, com as opções de nome de equipe, fogo e fumo. Para a opção de nome de equipe, o valor padrão é fc_portugal.

- kill_sim.sh

Termina a execução do simulador e dos seguintes agentes: ambulancia, policia, bombeiros, Centro de ambulancias, Esquadra de policia e Quartel de bombeiros.

O seguinte script foi criado na pasta fc_portugal:

- kill_agent.sh

Termina a execução dos seguintes agentes: ambulancia, policia, bombeiros, Centro de ambulancias, Esquadra de policia e Quartel de bombeiros.

1.2.6 Ficheiros de Configuração

do Simulador

Os seguintes ficheiros encontram-se na pasta boot e são apresentados abaixo:

- config.txt

Este é o principal ficheiro de configuração do simulador. É bastante flexível, podendo ser configurado pelo utilizador. A configuração é dada por <parametro:valor>.

- civilian_rules.txt

Define o comportamento dos civis.

Dos subsimuladores

- Pasta firesimulator

- config_addition.txt é o ficheiro de configuração que se encontra na subpasta res_q_firesimulator. A configuração é dada por <parametro:valor>.

- Pasta miscsimulator

- config.txt é o ficheiro de configuração do Misc. A configuração é dada por <parametro:valor>.

1.3 Appendix C – FCPx spreadsheets parameters

//// (FIREBRIGADE)

Buildings

Burning Buildings

Flooded Unburned Buildings

Extinguished Buildings

Number of times buildings reignited

Completely Burned Buildings

Perc. of Bldgs. Undamaged by Fire nor Water

///

Civilian Related Data

Percentage of Civilians Discovered

Civilians Killed by Fire

Total Number of Civs Discovered

Perc. of Dead Civilians Killed by Fire

Fraction of Those Discovered by Firebrigades

///

Agent Related Data

Buried Police Officers Killed by Fire

fire brigades Killed by Fire (Not Buried)

Buried Ambulances Killed by Fire

Number of Buried fire brigades (Alive)

Buried fire brigades Killed by Fire

/// (POLICE)

Road Related Data

Number of Roads with at least one Blocked Lane

Number of Roads Blocked (Impassable)

Number of Road Obstructions Removed

Percentage of Roads Blocked (Impassable)

//

Civilian Related Data

Percentage of Civilians Discovered

Total Number of Civs Discovered

Fraction of Those Discovered by Police Forces

//

Agent Related Data

Police Forces Killed by Fire (Not Buried)

Number of Buried Police Forces (Alive)

/// (AMBULANCE)

Agent Related Data

Buried Police Forces Killed by Debris

Buried Police Forces Killed by Fire

Buried Police Forces (Alive)

Ambulances Killed by Fire (Not Buried)

Buried Ambulances Killed by Debris

Buried Ambulances Killed by Fire

Buried Ambulances (Alive)

Total Number of Rescue Agents Unburied

Buried fire brigades Killed by Debris

Buried fire brigades Killed by Fire

Buried fire brigades (Alive)

Total Number of Buried Rescue Agents (Alive)

//

Civilian Related Data

Legenda

Percentage of Civilians Discovered
Civilians Killed By Debris
No. of Living Civilians Burried Under Debris
Total Number of Civilians
Total Number of Civs Discovered
Civilians Killed By Fire
No. of Civilians Rescued by Ambulance Teams
Fraction of Those Discovered by Ambulances
Percentage of Civilians Killed
Number of Refuges
/// (CIVILIANS)
Civilians Discovered
Percentage of Civilians Discovered
Fraction of Those Discovered by Firebrigades
Total Number of Civs Discovered
Fraction of Those Discovered by Police Forces
Percentage of Buildings Explored
Fraction of Those Discovered by Ambulances
//
Civilians Killed
Civilians Killed By Debris
Civilians Killed By Fire
Total Number of Civilians Killed
Percentage of Civilians Killed
//
Civilians Rescued
No. of Living Civilians Burried Under Debris
Percentage of Civilians Rescued
Number of Rescued Civilians Per Ambulance
Number of Civilians Rescued by Ambulance Teams

Annex A

Dijkstra's algorithm

The algorithm works by keeping for each vertex v the cost $d[v]$ of the shortest path found so far between s and v . Initially, this value is 0 for the source vertex s ($d[s]=0$), and infinity for all other vertices, representing the fact that we do not know any path leading to those vertices ($d[v]=\infty$ for every v in V , except s). When the algorithm finishes, $d[v]$ will be the cost of the shortest path from s to v -- or infinity, if no such path exists. The basic operation of Dijkstra's algorithm is edge relaxation: if there is an edge from u to v , then the shortest known path from s to u ($d[u]$) can be extended to a path from s to v by adding edge (u,v) at the end. This path will have length $d[u]+w(u,v)$. If this is less than the current $d[v]$, we can replace the current value of $d[v]$ with the new value.

Edge relaxation is applied until all values $d[v]$ represent the cost of the shortest path from s to v . The algorithm is organized so that each edge (u,v) is relaxed only once, when $d[u]$ has reached its final value.

The algorithm maintains two sets of vertices S and Q . Set S contains all vertices for which we know that the value $d[v]$ is already the cost of the shortest path and set Q contains all other vertices. Set S starts empty, and in each step one vertex is moved from Q to S . This vertex is chosen as the vertex with lowest value of $d[u]$. When a vertex u is moved to S , the algorithm relaxes every outgoing edge (u,v) .