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

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

Ciência. Inovação 2010
Programa Operacional Ciência e Inovação 2010
MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E ENSINO SUPERIOR

June 2006
(Junho de 2006)
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 Reis\(^1\) (main supervisor), Nuno Lau\(^2\) and Francisco Reinaldo\(^3\).

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

Ciência. Inovação Programa Operacional Ciência e Inovação 2010
MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E Ensino Superior

June 2006
(Junho de 2006)

---

\(^1\) 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 novo domínio. 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. Thanks André Restivo, Luís Paulo and Eurico Carrapatoso. We would be in a lot of trouble, if not for your good will.

Thanks to Alexander Kleiner for creating Freiburg’s 3D Viewer – a great piece of software, with various possibilities hidden “under the hood”. By studying this viewer we gained quite a bit of insight into the inner workings of Rescue log files, allowing us to develop our own tool.

From João:

I would like to thank my family for their continuous support me throughout my graduation. Especially my parents for their tolerance when things wouldn’t go as expected.

A big thanks to 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 take make him such a good friend.

A special kiss to Rita that, despite moving to Morocco, remained 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 to be an even better person as she ages (smoothly), 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


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


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 (Geographical Information System) ..................... 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 .......................................................... Error! Bookmark not defined.
           2.3.6.1 Morimoto Viewer ........................................ 19
CONTENTS

2.3.6.2 Freiburg’s 3D viewer .............................................. 20
2.3.7 System Configuration ............................................. 21
2.4 Communication ...................................................... 22
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
CONTENTS

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

2 Apêndice B: Execução do Simulador ............................................................................. 70
  2.1.1 Execução do Simulador .......................................................................................... 70
  2.1.2 Visualizador em Tempo Real .............................................................................. 71
  2.1.3 Visualização de Logs ........................................................................................... 72
  2.1.4 Execução dos Agentes ........................................................................................ 73
  2.1.5 Scripts alterados e criados ................................................................................. 73
  2.1.6 Ficheiros de Configuração ................................................................................. 75

3 Appendix C – FCPx spreadsheets parameters ............................................................... 76

Annex A ............................................................................................................................... 79
List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Simulation System Functional Outline</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>RoboCup Rescue Simulation System</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>Buildings with Growing Collapse Levels (from 1 to 4)</td>
<td>11</td>
</tr>
<tr>
<td>4</td>
<td>Blocked Roads</td>
<td>12</td>
</tr>
<tr>
<td>5</td>
<td>An Image from Freiburg's 3D Viewer Shows a Building on Fire</td>
<td>13</td>
</tr>
<tr>
<td>6</td>
<td>Civilians with Evolving Status</td>
<td>13</td>
</tr>
<tr>
<td>7</td>
<td>The Refuge as Depicted in the Default Viewer (Left) and Freiburg's 3D Viewer (Right)</td>
<td>14</td>
</tr>
<tr>
<td>8</td>
<td>Civilian Status</td>
<td>15</td>
</tr>
<tr>
<td>9</td>
<td>Ambulance and Buried Civilian</td>
<td>16</td>
</tr>
<tr>
<td>10</td>
<td>Fire Brigades Extinguishing a Fire</td>
<td>17</td>
</tr>
<tr>
<td>11</td>
<td>Police Agents Cleaning Roads</td>
<td>17</td>
</tr>
<tr>
<td>12</td>
<td>Morimoto Viewer Displaying a Simulation in the Foligno Map</td>
<td>19</td>
</tr>
<tr>
<td>13</td>
<td>Morimoto Viewer Displaying a Simulation in the</td>
<td>20</td>
</tr>
<tr>
<td>14</td>
<td>Agents and Their Respective States as Represented in Morimoto Viewer</td>
<td>20</td>
</tr>
<tr>
<td>15</td>
<td>Freiburg's 3D Viewer (Also Displaying Its 2D View)</td>
<td>21</td>
</tr>
<tr>
<td>16</td>
<td>Radio-Communications</td>
<td>23</td>
</tr>
<tr>
<td>17</td>
<td>Object Class</td>
<td>27</td>
</tr>
<tr>
<td>18</td>
<td>Agents State-Machine Implementation</td>
<td>29</td>
</tr>
<tr>
<td>19</td>
<td>Ambulance Team States</td>
<td>32</td>
</tr>
<tr>
<td>20</td>
<td>Fire Brigade States</td>
<td>34</td>
</tr>
<tr>
<td>21</td>
<td>Police Force States</td>
<td>37</td>
</tr>
</tbody>
</table>
List of Tables

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

CHART 1: a) FINAL SCORE EVALUATION; b) CIVILIAN CASUALTIES; c) EXPLORATION AND CIVILIAN DISCOVERY; ................................................................. 55

CHART 2: a) DISCOVERED CIVILIANS BY AGENT TYPE; b) DISCOVERED CIVILIANS (STACKED) .................................................. 56

CHART 3: SCORE; ..................................................................................... 57

CHART 4: a) DISCOVERED CIVILIANS; b) FC PORTUGAL - DISCOVERED CIVILIANS BY AGENT TYPE (STACKED); ......................................................... 57

CHART 5: a) CIVILIANS KILLED; b) FC PORTUGAL - CIVILIANS CASUALTIES (STACKED); .......................................................... 58

CHART 6: a) RESCUE AGENTS BURIED; b) RESCUE AGENTS KILLED .............................................................. 58

CHART 7: a) BUILDINGS ON FIRE; b) BUILDING AREA DESTROYED .......................................................... 59

CHART 8: a) ROADS BLOCKED; b) ROADS OBSTRUCTED .......................................................... 59

CHART 9: a) CIVILIANS RESCUED; b) CIVILIANS RESCUED ADJUSTED .......................................................... 60
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; and 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. "Scribbly", 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 dynamizing the evolution of AI, in particular MAS, the project was launched by Dr. Hiroaki Kitano, an AI researcher that became founder and president 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, mostly 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 in 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;
- Preparation of the team for future developments by new members.
1.2.2 Extended Objectives

After the successful participation in RoboCup 2005 the Project was extended\(^4\). 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 reality 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,

\[ e(t) = f(x(t), u(t), t) \]
\[ x(t + \Delta t) = g(x(t), e(t)) \]

in which:

\[ e(t) \] represents the effects that create change in the city, calculated by \( f \).
\( t \) represents the current time instant.
\( f \) is the function describing how \( x(t) \) \( e \) \( u(t) \) affect the simulated world, changing its status.
\( x(t) \) 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.
\( \Delta t \) is the time step used to forward the simulation discretely.

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

At \( t=0 \), \( x(t) \) represents the initial situation.
From \( x(0) \) 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:

\[
V = \left( P + \frac{S}{S_{\text{int}}} \right) \cdot \sqrt[3]{\frac{B}{B_{\text{int}}}}
\]

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.

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 change 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.](image)

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 viewer\(^5\), 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 scheduled to be revised shortly, as new features – such as earthquake aftershocks – are expected.

\[\text{Figure 3: Buildings with growing collapse levels (from 1 to 4).}\]

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.uoc.ac.jp/~morimoto/rescueviewer/index.html
As seen on Figure 4, the default viewer represents road blocks with crosses, which can be either gray or black. A gray cross (1) means that the road is only partially blocked, while a black one (2) 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 affecting 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 temperature and fuel (buildings), with the supply of oxygen being disregarded. When a building's temperature rises above its material's flash point it bursts into flames, as seen on Figure 5. In contrast, when the temperature drops below this temperature, 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 building 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.](image)

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 2.3.6.1).

Ambulance Team agent

This is the agent responsible for rescuing Civilians from collapsed buildings. It bears all properties of a Civilian plus the ones specific to its function, namely: unbury Civilians, load them to the Ambulance and unload them in refuges. Figure 9 depicts two Ambulances, with one of them rescuing a Civilian.

![Ambulance and buried Civilian](image)

*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 properties of a Civilian, 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.
**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 properties of a Civilian, plus the ability to remove these obstructions. Figure 11 shows two Police Forces clearing roads. One of them is approaching an obstacle while the other is actively removing another.
2.3.5.3 Center Agents

These agents are represented as ordinary buildings – they appear in a lighter color on the default viewer – possessing all their properties, but having 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 3.2.5 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 Morimoto\(^6\).

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; Fire Brigades red; Police Forces blue; Civilians green and all of them get darker when hurt, turning completely black if they die (see Figure 14). Buildings also have different colors, according to their function or status. While refuges are green and Center (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 are displayed on top of the map. The team name can also be displayed on the top bar, but it is optional.

\[\text{Time: 11} \quad \text{Score: 40.876319}\]

*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.
As previously mentioned, Police and Ambulance activity is denoted with a hallo 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 13: Morimoto Viewer displaying a simulation in the RandomMedium map from RoboCup Osaka 2005.](image)

![Figure 14: Agents and their respective states as represented in Morimoto Viewer.](image)

### 2.3.6.2 Freiburg’s 3D viewer

Also widely used is Freiburg’s 3D viewer\(^7\), mostly due to its appealing look to the public as seen on Figure 15.

This viewer uses familiar forms to represent agents. For example, Fire Brigades are represented by fire trucks, while on roads, and fire fighter helmets, when inside buildings. 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/
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, section 1.6.

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.

<table>
<thead>
<tr>
<th>Entity</th>
<th>Min.</th>
<th>Max.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fire Brigade</td>
<td>0</td>
<td>15</td>
</tr>
<tr>
<td>Police Force</td>
<td>0</td>
<td>15</td>
</tr>
<tr>
<td>Ambulance</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>Civilian</td>
<td>70</td>
<td>90</td>
</tr>
<tr>
<td>Fire Brigade Center</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Police Force Center</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Ambulance Centre</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Refuges</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>Ignition points</td>
<td>2</td>
<td>8</td>
</tr>
</tbody>
</table>
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[8]; 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. The same is true for **field hearing**, also exclusive to 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 same kind. The agents' radio communication scheme can be seen in Figure 16.
There are also other restrictions limiting communications. With the changes introduced to 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** works in the opposite way (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 and improving team strategies. As teams adapt to new features, the time left for the development and perfecting 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 human 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 heavily depended on 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 Rescue agents, and the problems they have to solve. For explanatory purpose we use FC Portugal's implementation which is a modified version of the SOS 2004 team code[14].

It should be kept in mind that each team can have a different solution to the same problem. There is a myriad of different approaches and different strategies; some better, some worse and some just different.

Usually only one solution is presented (ours). Still, since one of the project's objectives was the selection of a base code, 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, on section 3.3.

3.2 Code organization

3.2.1 World State

In a complex domain information is everything and, the better the information is, the better can the decisions based on that information be. 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 used to store world objects' properties is a modification of the one in use by simulators. We can see the object class in Figure 17.
World object data is an instantiation of the class named "Object Pool" composed of hash maps\(^8\) of objects, with the exception of World which is unique, plus some functions to access that data. It should be noted that each data field stored has a timestamp associated, so it is possible to know 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 2.4, message exchange between team agents is limited, both in size and number. For this reason, scrupulous sifting must be made by each agent, carefully selecting which messages should be sent and heard.

One important step is the compression of information. All possible messages between team agents have a number, so that instead of saying "I found a Civilian" a correspondent number is sent. Likewise, IDs (identification number) of team agents, buildings, nodes and roads are also numbered, since at setup all agents receive the IDs in the same order (normal IDs have 9 digits each). An exception is made for Civilian IDs, given that with the

---

\(^8\) The `HashMap` class is roughly equivalent to `Hashtable`, except that it is unsynchronized and permits nulls. *In Java\textregistered TM 2 Platform, Standard Edition, v 1.4.2 API Specification*
rules introduced in RoboCup 2005, those are no longer sent at startup - Civilians need to be discovered before any ID is sent to the agents. Following on the same line of thought, object properties are always sent in the same order, allowing for succinct information exchange.

The next step regards the aggregation of information, which 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 (as many times as the number of tokens), repeating those steps for all the message types that an agent needs to send. When necessary, information can be split into several messages.

When all agents send messages, a selection must be made on which messages to receive. 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, listening only to the center (as seen in Figure 16, field agents can only listen to their corresponding center). As for the centers, they act as repeaters, listening to information from their agents and from the other centers, and resending 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.

Sharing all the information would lead to an exponential growth of messages sent and this growth is avoided by choosing to only send new information. In section 3.2.1 it was mentioned that information was time stamped. Because of this, it is easy to see if the information sensed is newer than the stored one and, if so, store and retransmit it.

### 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 agent's point of 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 2.4, 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 where agents send action commands like move, extinguish, etc. It is 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 2.4, 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. A team agent’s state machine is represented in Figure 18:

![State machine diagram]

*Figure 18: Agents state-machine implementation.*

As previously stated, center agents do not have field actions, so the “send sub-state actions” at the “act” state should be ignored for these agents. 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 amongst agents. The basic behavior of a field agent has even more similarities. For instance, if a field agent is buried it sends a message asking for help or, if it needs to go somewhere and is unable find a possible path to its destination, it reports that fact. Self-preservation is also a shared behavior. If an agent finds its health points decreasing, it stops whatever it is doing and leaves the building (unless, obviously, if it is buried). For field agents, three of the most important blocks are path finding, traffic jam handling and map splitting.

Path Finding

As mentioned in the simulator chapter, in 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 it is not the best solution, it is better than no solution at all. Plus, there may be a need to stop the search and use the current best path since, as previously mentioned, cycle time is limited.
- Efficiency: machine resources are limited, there are usually several agents running on the same machine.

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

Without deeply analyzing search algorithms, which is not the purpose of this section, a brief description of the options available is given. First Best Search is unfit, because it does not guarantee a solution. Floyd’s algorithm was also disregarded, since in most cases it is not as efficient as Dijkstra. So, the remaining algorithms are Dijkstra and A*.

Knowing these two algorithms and perceiving that, in this problem, we have knowledge of origin, destination and its relative positions, one would be inclined to choose the A* algorithm as this is, in a way, a variation of Dijkstra’s algorithm that has an heuristic that exploits 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 oftentimes, especially at the beginning of the simulation, most of the possible paths are blocked
and, in this case (maze like), A* has an 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 which may have obstacles. Because of this, it is common for only one lane to be traversable. Consequently, when two agents try to move through the same lane in opposite directions, neither can pass. This event is called a traffic jam. Please note than this event could arise from a variety of other ways, such as a Fire Brigade extinguishing a building from a street, one of the agents could be a Civilian, etc., but the issue is basically the same. This problem can be aggravated if more than two agents are involved.

The first step to solve this problem is to detect it, and this is achieved by comparing the current position to the previous one. If it stays the same for more than n cycles (2 in our case), we are facing 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. It should be noted that this solution works, independently of 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) and, with these coordinates, a rectangular form is created and equally divided between the field agents of the desired kind. To that division we call a cell.

The former solution would be 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 most are not. Consequently, some agents could be forced to handle more buildings and roads than others. To resolve this issue, FC Portugal’s team is currently 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 described next. 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 Civilians 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 should be saved. The time taken in travelling, and whether a path to the Civilian exists, are considered.

The next steps for an Ambulance are: go to the Civilian position; unbury him; load him into the Ambulance; travel to the closest refuge and unload the Civilian to safety, moving on to the next one. However, this behaviour can change due to several events, such as receiving updated or new information about a Civilian, which may change rescue order and priorities. As previously stated, behaviour also changes 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 Teams form and act as a single group. This happens because the action of unburying an agent is cumulative, and directly proportional to the number of Ambulance Teams. Therefore, the more Ambulance Teams, the faster the agent will be unburied. There are some exceptions to this tactic, since the moving cost is considered when estimating the Civilian time of death, and sometimes it is more efficient for Ambulances to act individually. Another exception is at the beginning of the simulation given that, due to road blocks, Ambulance Teams don’t usually have possible paths to the same agent, they also act individually.

One final remark on Ambulance behaviour is that, when there are no Civilians to rescue, Ambulance Teams scatter, dividing the map in cells, and search for new Civilians.

The state machine for an 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.](image)

The “Clocked State” is used by some of the other sub states, to keep track of the amount of cycles the Ambulance Team spends in a particular sub state. This is called the duration. From a functional point of view it is not a sub-state by itself, but a common addition to some other sub states.

The “Go to State” is used when the Ambulance wants to move. It handles all moving functionalities, such as the previously described path finding and traffic jam handling modules.
The "Rescue State" is used before, and after, the unburying. It verifies things like if the agent to be rescued is where expected; estimates the time taken for unburying; verifies if the building is on fire and, if so, if it is still possible to rescue the agent, by sacrificing some health points.

The "Search State" is used when there are no known agents to be rescued. In this mode, each agent takes a cell (as 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 and, as stated in 2.3.5.2, their function is to extinguish fires. The strategy for Fire Brigades is the following:

Depending on map size, Fire Brigades are organized into groups - usually 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 considers its assigned fiery region, and prioritizes 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, amongst other factors.

The next step is to choose a suitable neighbor building that is in the water range of the target. The conditions for this are: 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 near the target is used. The disadvantage to this solution 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 without requiring a move action.

After some cycles, water in tanks is depleted and, therefore, Fire Brigades go to the nearest refuge to refill their tanks. As this action takes some cycles, when refilling finishes, Fire Brigades reprioritize between the previous assigned region and the currently unassigned ones - 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 remaining criteria.

Akin to Ambulance Teams, when Fire Brigades run out of fires to extinguish, the groups are scattered and the map is divided into cells, so that new fiery buildings may be found. If none is discovered, Fire Brigades start searching for new civilians.
Additionally, if all buildings have been explored, Fire Brigades keep on visiting all known living, buried, Civilians in order to update the information on their properties, allowing Ambulances to better estimate the Civilian time of death.

The state machine for a Fire Brigade is showed in Figure 20. Because of the very different functioning modes described above, which depend on whether or not there is an active fire, the Fire Brigades have two “Super States”. When there is a fire, the “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” is used when a Fire Brigade wants to move and, akin to the Ambulance Teams, it handles all moving functionalities, such as previously described path finding and traffic jam handling modules.

“Attack State” sub states

The “Clocked State” has the same function that was previously described for the Ambulance Teams (3.2.4.2). It is used by some of the other sub states, to keep track of the amount of cycles the Fire Brigade spends in a particular sub state.
CHAPTER 3: TEAM IMPLEMENTATION

The “Assign And Extinguish Regions” state is where the region is chosen, suitable regions are computed and the path cost to reach them is calculated. Hence, the next state will 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 will not be wasted, as the new target has already been found. As a consequence, in most cases the function to calculate targets doesn’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”.

“Extinguish Building” is where Fire Brigades calculate and choose the neighbouring building from where they 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 large building is too high due to its area, thus it is a common option not to douse such buildings. Instead of extinguishing these, the Fire Brigades let them burn completely, controlling the fire by carefully managing its expansion into neighbouring buildings. However, in certain exceptional situations, such as specific maps with these kinds of buildings in key points, there is a need to consider them as targets. This state exists 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 regions and targets. As such, in order for the Fire Brigade to be useful, only reachable targets are considered. As soon as the assigned region is reachable, Fire Brigades leave this state.

“Maintain Water State” is responsible for keeping track of the group’s water quantity and takes the decision of when to move the group to the refuge, in order to refill the tanks. The decision is made by 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. Contrasting 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, and the map splitting mechanism previously described is also used.
In “Check Civilians” state, the Fire Brigades recursively visit all living, buried Civilians in his assigned map cell.

3.2.4.4 Police Force

As stated in 2.3.5.2, Police Force agents’ function is to clear blocked roads. The strategy for Police Forces 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, and the reason for this is that they only affect the speed of an agent by halving it. The speed reduction isn’t significant, when compared to the time it 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 is 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 options are called tasks. Each of these tasks is given a weight, which is a parameter 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 clearing a specific cell, which is used when no other tasks are requested. When a Police Force receives a task request, it calculates how long each task is going to take, and multiplies it with the weight factor, executing the cheapest one. 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 must move themselves to a certain position, in order to perform a certain action. This means that a Police Force agent, when moving, is always clearing impassable blocks in its way, and no detour of blocks is made.

The state machine for a Police Forced is showed in Figure 21. In this case the “Super State” is called “State Police Force".
The “Clocked State” has the same function that was previously described for the Ambulance Teams (3.2.4.2). It is used by some of the other sub states, to keep track of the amount of cycles the Police Force spends in a particular sub state.

The “Clear Path” state is the Police Force equivalent to “Go to State”, present in Ambulances and Fire Brigades. Accordingly, when the Police Force needs to go to a particular position in order to perform a certain task, it uses this state. The only difference is that it clears all impassable blocks in its way, making no detours to avoid them.
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 was 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 used to enable the usability of other agents, unable to act due to road blocks. When Ambulance Teams or Fire Brigades 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 being that no destination position is provided. Therefore it calculates and chooses the closest blocked road, from his cell, to start clearing.

The "Process Cell" state is used to clear blocks in the Police Force's cell. The agent computes the sequence in which roads should be unblocked, and clears roads in that sequence.

The "Search Cell for Civilian" state is slightly different from the "Find Civilian" state, found in Ambulance Teams and Fire Brigades. The main difference is that, when searching, Police Forces clear impassable road block in their way.

The "Check Buried Civilians" state is similar to the Fire Brigades' "Check Civilians", i.e., when all buildings have been searched for Civilians, Police Forces keep visiting buried alive Civilians. The only difference is that they also clean road blocks in their 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 data is only available at cycle 3). As such, Police Force agents have to wait for cycle 4 to check if they are buried and this state is used for that purpose. As in the "Maintain Health" state, in other types of agent this is implemented in the "Super State". The reasons for this not to happen are the same ones that were previously referred - legacy code.

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 the Police Office.

The "Clear around Building" state is where the task of clearing blocks around a refuge is implemented. Police force agents move around the refuge, clearing every impassable road block in its way. Like with "Clear to Building", request for this task is made by the 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 the center agents is to act as repeaters. As such, the communication module is similar.

3.2.5.2 Ambulance Center

As Ambulance Teams' calculations are relatively simple, there is no need for further calculations at the Ambulance Center and, as such, its main function is to act as a repeater. It should be kept in mind 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

This agent's main function is to act as a repeater; however, some of the calculations computed in Fire Brigade agents are also computed in the Fire Station. This happens both in the prioritize regions case, and in group assignment. As seen, cycle time is limited and, sometimes, field agents have to rush decisions. Fire Stations do not have this limitation, thus it compares the field agent's solution to its own and, if the Station's solution is better, it is sent to Fire Brigades.

The state machine for a Fire Station is composed of a single “Super State” named “Attack State”.

3.2.5.4 Police Office

Besides acting as a repeater, the Police Office computes currently assigned and unassigned Police Force tasks. The assigned tasks are multiplied by a reassign coefficient and the most suitable task for each Police Force is chosen, based on relative position to objectives. As one can easily perceive, the Police Office is, when present, the main responsible for Police Force strategy.
3.2.6 Libraries

One of the most important software engineering techniques is 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. These libraries allow a higher level of programming and this allows different programmers to 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 that programmers may abstract from things like socket management and low level connection protocols. It also possesses structures for the objects of the simulation; constants definitions used 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. It provides a memory interface, with data structures suitable to a team’s agents; 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 and, therefore, 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, centre agents must be able to transmit and receive more than object information. For this reason, there is a different library for every of the three rescue domains. This way, centres can communicate with field agents of the same kind. An example of this situation is the assignment of a task, by the Police Office to a Police Force.

When both field and centre agents do the same type of calculations, the code for these 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 the Fire Station.
3.2.7 The initial parameters

During the description of FC Portugal’s team agents, there were a few references to customizable parameters. Most of those parameters’ values result of human perception, i.e., the value for the parameter is “guessed” based on human knowledge and intelligence and, consequently, 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 gigantic and, therefore, only one limited set of parameters may be adjusted this way. The tool presented and described in chapter 4 represents an attempt to trim this problem. The existing types of parameters are briefly described in this section.

3.2.7.1 Generic

Generic parameters are the ones common to the agents (e.g. minimum acceptable health points when calculating certain tasks). Constant values defined by RoboCupRescue rules that are subject to possible changes are also included here (e.g. the number of bytes on the data part of message).

3.2.7.2 Map Specific

In RoboCupRescue, some maps are known in advance. For those maps, parameters can be tweaked according to map conditions (e.g. the number of Fire Brigades per group) and some pre-calculations can also be performed, like dividing the map in cells.

3.2.7.3 Agent Specific

Many of the 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). Other parameters of this kind are factors such as the 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, the exchange of ideas between teams is strongly encouraged. The main reason for this is that it promotes and hastens the developments in artificial intelligence.

RoboCup teams may opt to start their project from zero, use development kits, or take another team’s code and start from there. It is even possible to use code from several teams taking the best from each.
Starting from scratch or even from a development kit, although better in a structural way, has the disadvantage of taking too long for the team to become competitive and, in the initial years, teams won't 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 these options are, respectively, the following:

- All current 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 eases transitions and fastens development.

From the active rescue teams, the base code that came closer to the specified criteria was the one from the S.O.S. team, from Iran.

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

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

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

- Civilian IDs are not sent at start-up;
- No more than one Fire Brigade may be, simultaneously, inside a building, with the exception of refuges;
The change in the Civilian rule led to more complications than would be expected. The agents' knowledge on whether or not all Civilians were found was lost. So agents must now explore all buildings, in search of Civilians. On low level code this change proved to be a monumental problem, since the allocation of Civilian data structures was no longer made at the beginning of the simulation, and could now be made at any time during the simulation. The new Civilian data could be obtained either by field sense or by message. As was explained in communication strategies (3.2.2), agent IDs are compressed using the initial setup IDs and, therefore, Civilian IDs 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 on the code of the six types of team agents.

As for the Fire Brigade in building limit, the problem was solved by verifying if the chosen building was occupied.

3.5 RoboCup Osaka 2005

In order for 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 time with the paper: “FC Portugal 2005 Rescue Team Description: Adapting Simulated Soccer Coordination Methodologies to the Search and Rescue Domain”[17].

The results achieved in Japan were relatively bad, but not outside expectations. The time used to learn the workings of the simulator, choosing a code, and adapting it to the new rules, left no time for strategy improvements. The solution to the imposed limit of a single Fire Brigade per building revealed to be a poor one, as the paths were calculated prior to the presence of any Fire Brigade in the building. As a result, only one Fire Brigade in the group would enter a building, at each cycle.

During RoboCup, changes were made to solve this problem. The new solution was for the Fire Brigades to consider not only the best building from where to extinguish, but the n best, where n is the number of Fire Brigades in the group. This solution was implemented at the end of the preliminary phase. This led to an improvement on FC Portugal’s team performance, though not enough to make it to the next stage.

---

9 Nuno Lau, Luís Paulo Reis and Francisco Reinaldo
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 easy to perceive that nearly all strategic decisions are taken by field agents. This happens mainly for two reasons, the first of which may be inferred from Table 1 (chapter 2, section 2.3.7). Since center agents are optional, it is possible for a simulation to be ran without center agents - although this has not occurred in competitions in the recent years. The second reason is the possibility of message loss, which is included in the simulation. As a result, it is safer to implement high level strategies in field agents, and use the center agents to improve its results.

Voronoi cells are the next step in map splitting techniques. They are currently being implemented and are expected to improve FC Portugal agents’ performance.

In RoboCupRescue simulations, both known and unknown maps can be used. For the known maps, FC Portugal’s implementation uses map specific parameters which allow higher agent performance.

Participating in the rescue simulation league for the first time was an enriching experience and the knowledge obtained in this environment was vast and enriching. FC Portugal’s rescue team was successfully established with the base code selected, studied, and improved. Finally, the team successfully participated in RoboCup 2005 in Osaka, Japan, marking our team’s first participation in a competition – an important milestone.

Information on this chapter is being used in the elaboration of Rescue Technical Report II – FC Portugal’s Team Agents
Chapter 4

4 FCPx – 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 in the process of implementing minor improvements and tweaking existing 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, since it is possible to see their actual difference by simultaneous parsing 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 built 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 (FCPx) 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 very easy to build, and analyze, tables and comparative charts. From this point, the versatility of this solution gives the programmer endless possibilities. Still, the FCPx 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. In order to properly evaluate a strategy there is a need to run dozens of simulations, comparing the attained results with data from previous approaches. This generates dozens of logs, which take, each, around 5 to 15 megabytes of disk space. 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>
<thead>
<tr>
<th>Table 2: Fire Brigades - Building related data.</th>
</tr>
</thead>
</table>

![Table Image]
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 holds 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.

<table>
<thead>
<tr>
<th>Civilian Discovered</th>
<th>Time</th>
<th>Fire Brigades</th>
<th>Percentage of Civilians Discovered</th>
<th>Total Number of Civilians</th>
<th>Total Number of Civilians Killed by Fire</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>5.70%</td>
<td>0</td>
<td>4</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>1</td>
<td>5.70%</td>
<td>0</td>
<td>4</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>2</td>
<td>5.70%</td>
<td>0</td>
<td>4</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>3</td>
<td>5.70%</td>
<td>0</td>
<td>4</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>4</td>
<td>5.70%</td>
<td>0</td>
<td>4</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>5</td>
<td>7.40%</td>
<td>5</td>
<td>5</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>6</td>
<td>7.10%</td>
<td>6</td>
<td>6</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>7</td>
<td>7.10%</td>
<td>7</td>
<td>7</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>8</td>
<td>2.80%</td>
<td>8</td>
<td>8</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>9</td>
<td>2.80%</td>
<td>9</td>
<td>9</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>10</td>
<td>2.80%</td>
<td>10</td>
<td>10</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>11</td>
<td>2.80%</td>
<td>11</td>
<td>11</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>12</td>
<td>2.80%</td>
<td>12</td>
<td>12</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>13</td>
<td>2.80%</td>
<td>13</td>
<td>13</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>14</td>
<td>2.80%</td>
<td>14</td>
<td>14</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>15</td>
<td>2.80%</td>
<td>15</td>
<td>15</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>16</td>
<td>2.80%</td>
<td>16</td>
<td>16</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>17</td>
<td>2.80%</td>
<td>17</td>
<td>17</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>18</td>
<td>2.80%</td>
<td>18</td>
<td>18</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>19</td>
<td>2.80%</td>
<td>19</td>
<td>19</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>20</td>
<td>2.80%</td>
<td>20</td>
<td>20</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>21</td>
<td>2.80%</td>
<td>21</td>
<td>21</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>22</td>
<td>2.80%</td>
<td>22</td>
<td>22</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>23</td>
<td>2.80%</td>
<td>23</td>
<td>23</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>24</td>
<td>2.80%</td>
<td>24</td>
<td>24</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>25</td>
<td>2.80%</td>
<td>25</td>
<td>25</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>26</td>
<td>2.80%</td>
<td>26</td>
<td>26</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>27</td>
<td>2.80%</td>
<td>27</td>
<td>27</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>28</td>
<td>2.80%</td>
<td>28</td>
<td>28</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>29</td>
<td>2.80%</td>
<td>29</td>
<td>29</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>30</td>
<td>2.80%</td>
<td>30</td>
<td>30</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
</tbody>
</table>

Table 3 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 ).
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 agent 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.

<table>
<thead>
<tr>
<th>Time</th>
<th>Type of Obstruction</th>
<th>No. of Obstructions Removed</th>
<th>Total Road Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>N/A</td>
<td>N/A</td>
<td>16.247</td>
</tr>
<tr>
<td>1</td>
<td>N/A</td>
<td>N/A</td>
<td>16.247</td>
</tr>
<tr>
<td>2</td>
<td>N/A</td>
<td>0</td>
<td>16.247</td>
</tr>
<tr>
<td>3</td>
<td>N/A</td>
<td>0</td>
<td>16.247</td>
</tr>
<tr>
<td>4</td>
<td>N/A</td>
<td>1</td>
<td>16.247</td>
</tr>
<tr>
<td>5</td>
<td>N/A</td>
<td>3</td>
<td>16.247</td>
</tr>
<tr>
<td>6</td>
<td>N/A</td>
<td>9</td>
<td>16.247</td>
</tr>
<tr>
<td>7</td>
<td>N/A</td>
<td>11</td>
<td>16.247</td>
</tr>
<tr>
<td>8</td>
<td>N/A</td>
<td>16</td>
<td>16.247</td>
</tr>
<tr>
<td>9</td>
<td>N/A</td>
<td>18</td>
<td>16.247</td>
</tr>
<tr>
<td>10</td>
<td>N/A</td>
<td>23</td>
<td>16.247</td>
</tr>
<tr>
<td>20</td>
<td>N/A</td>
<td>52</td>
<td>16.247</td>
</tr>
<tr>
<td>30</td>
<td>N/A</td>
<td>81</td>
<td>16.247</td>
</tr>
<tr>
<td>40</td>
<td>N/A</td>
<td>104</td>
<td>16.247</td>
</tr>
<tr>
<td>50</td>
<td>N/A</td>
<td>134</td>
<td>16.247</td>
</tr>
<tr>
<td>60</td>
<td>N/A</td>
<td>165</td>
<td>16.247</td>
</tr>
<tr>
<td>70</td>
<td>N/A</td>
<td>190</td>
<td>16.247</td>
</tr>
<tr>
<td>80</td>
<td>N/A</td>
<td>211</td>
<td>16.247</td>
</tr>
<tr>
<td>90</td>
<td>N/A</td>
<td>229</td>
<td>16.247</td>
</tr>
<tr>
<td>100</td>
<td>N/A</td>
<td>243</td>
<td>16.247</td>
</tr>
<tr>
<td>110</td>
<td>N/A</td>
<td>257</td>
<td>16.247</td>
</tr>
<tr>
<td>120</td>
<td>N/A</td>
<td>266</td>
<td>16.247</td>
</tr>
<tr>
<td>130</td>
<td>N/A</td>
<td>273</td>
<td>16.247</td>
</tr>
<tr>
<td>140</td>
<td>N/A</td>
<td>279</td>
<td>16.247</td>
</tr>
<tr>
<td>150</td>
<td>N/A</td>
<td>284</td>
<td>16.247</td>
</tr>
<tr>
<td>160</td>
<td>N/A</td>
<td>289</td>
<td>16.247</td>
</tr>
<tr>
<td>170</td>
<td>N/A</td>
<td>292</td>
<td>16.247</td>
</tr>
<tr>
<td>180</td>
<td>N/A</td>
<td>292</td>
<td>16.247</td>
</tr>
<tr>
<td>190</td>
<td>N/A</td>
<td>292</td>
<td>16.247</td>
</tr>
<tr>
<td>200</td>
<td>N/A</td>
<td>292</td>
<td>16.247</td>
</tr>
<tr>
<td>210</td>
<td>N/A</td>
<td>292</td>
<td>16.247</td>
</tr>
<tr>
<td>220</td>
<td>N/A</td>
<td>292</td>
<td>16.247</td>
</tr>
<tr>
<td>230</td>
<td>N/A</td>
<td>292</td>
<td>16.247</td>
</tr>
<tr>
<td>240</td>
<td>N/A</td>
<td>292</td>
<td>16.247</td>
</tr>
<tr>
<td>250</td>
<td>N/A</td>
<td>292</td>
<td>16.247</td>
</tr>
<tr>
<td>260</td>
<td>N/A</td>
<td>292</td>
<td>16.247</td>
</tr>
<tr>
<td>270</td>
<td>N/A</td>
<td>292</td>
<td>16.247</td>
</tr>
<tr>
<td>280</td>
<td>N/A</td>
<td>292</td>
<td>16.247</td>
</tr>
<tr>
<td>290</td>
<td>N/A</td>
<td>292</td>
<td>16.247</td>
</tr>
<tr>
<td>300</td>
<td>N/A</td>
<td>292</td>
<td>16.247</td>
</tr>
</tbody>
</table>

The Police Force relevant data is displayed on Table 5 and Table 6. 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. On the right, it is shown that a buried Police Force was released from debris.
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. Although the rapid spread of fire contributed to this, the amount of Civilians deceased is extremely high.
### Table 7: Ambulance Teams – Agent related data.

<table>
<thead>
<tr>
<th>Column</th>
<th>Ambulance</th>
<th>Agent</th>
<th>Fire</th>
<th>Living</th>
<th>Unknown</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>500</td>
<td>200</td>
<td>300</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Percent</td>
<td>50%</td>
<td>40%</td>
<td>60%</td>
<td>20%</td>
<td>20%</td>
</tr>
</tbody>
</table>

### Table 8: Ambulance Teams – Civilian related data.

<table>
<thead>
<tr>
<th>Column</th>
<th>Civilian</th>
<th>Civilian Killed</th>
<th>Total Killed</th>
<th>Total Civilian</th>
<th>Total Killed Civilian</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>1000</td>
<td>200</td>
<td>800</td>
<td>800</td>
<td>1000</td>
</tr>
<tr>
<td>Percent</td>
<td>100%</td>
<td>20%</td>
<td>80%</td>
<td>80%</td>
<td>100%</td>
</tr>
</tbody>
</table>
The Civilian specific information is displayed on Table 9 and Table 10. These tables are mostly redundant, grouping information covered in previous tables. Table 9 confirms 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.**

<table>
<thead>
<tr>
<th>Time</th>
<th>Bridge Explored</th>
<th>Civ Discovered</th>
<th>Total Number</th>
<th>% of Civs Discovered by Police</th>
<th>% of Civs Discovered by Ambulance</th>
<th>Total Number of Civilians</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>5.68%</td>
<td>5</td>
<td>5</td>
<td>5.68%</td>
<td>5.68%</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>5.68%</td>
<td>4</td>
<td>4</td>
<td>5.68%</td>
<td>5.68%</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>5.68%</td>
<td>4</td>
<td>4</td>
<td>5.68%</td>
<td>5.68%</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>7.24%</td>
<td>5</td>
<td>5</td>
<td>7.24%</td>
<td>7.24%</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>5.22%</td>
<td>4</td>
<td>4</td>
<td>5.22%</td>
<td>5.22%</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>5.22%</td>
<td>4</td>
<td>4</td>
<td>5.22%</td>
<td>5.22%</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>5.22%</td>
<td>4</td>
<td>4</td>
<td>5.22%</td>
<td>5.22%</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>5.72%</td>
<td>5</td>
<td>5</td>
<td>5.72%</td>
<td>5.72%</td>
<td>5</td>
</tr>
<tr>
<td>8</td>
<td>10.41%</td>
<td>15</td>
<td>15</td>
<td>10.41%</td>
<td>10.41%</td>
<td>15</td>
</tr>
<tr>
<td>9</td>
<td>10.41%</td>
<td>15</td>
<td>15</td>
<td>10.41%</td>
<td>10.41%</td>
<td>15</td>
</tr>
<tr>
<td>10</td>
<td>8.24%</td>
<td>8</td>
<td>8</td>
<td>8.24%</td>
<td>8.24%</td>
<td>8</td>
</tr>
<tr>
<td>11</td>
<td>5.22%</td>
<td>5</td>
<td>5</td>
<td>5.22%</td>
<td>5.22%</td>
<td>5</td>
</tr>
<tr>
<td>12</td>
<td>5.22%</td>
<td>5</td>
<td>5</td>
<td>5.22%</td>
<td>5.22%</td>
<td>5</td>
</tr>
<tr>
<td>13</td>
<td>5.22%</td>
<td>5</td>
<td>5</td>
<td>5.22%</td>
<td>5.22%</td>
<td>5</td>
</tr>
<tr>
<td>14</td>
<td>5.22%</td>
<td>5</td>
<td>5</td>
<td>5.22%</td>
<td>5.22%</td>
<td>5</td>
</tr>
<tr>
<td>15</td>
<td>5.22%</td>
<td>5</td>
<td>5</td>
<td>5.22%</td>
<td>5.22%</td>
<td>5</td>
</tr>
<tr>
<td>16</td>
<td>5.22%</td>
<td>5</td>
<td>5</td>
<td>5.22%</td>
<td>5.22%</td>
<td>5</td>
</tr>
<tr>
<td>17</td>
<td>5.22%</td>
<td>5</td>
<td>5</td>
<td>5.22%</td>
<td>5.22%</td>
<td>5</td>
</tr>
<tr>
<td>18</td>
<td>5.22%</td>
<td>5</td>
<td>5</td>
<td>5.22%</td>
<td>5.22%</td>
<td>5</td>
</tr>
<tr>
<td>19</td>
<td>5.22%</td>
<td>5</td>
<td>5</td>
<td>5.22%</td>
<td>5.22%</td>
<td>5</td>
</tr>
<tr>
<td>20</td>
<td>5.22%</td>
<td>5</td>
<td>5</td>
<td>5.22%</td>
<td>5.22%</td>
<td>5</td>
</tr>
<tr>
<td>21</td>
<td>5.22%</td>
<td>5</td>
<td>5</td>
<td>5.22%</td>
<td>5.22%</td>
<td>5</td>
</tr>
<tr>
<td>22</td>
<td>5.22%</td>
<td>5</td>
<td>5</td>
<td>5.22%</td>
<td>5.22%</td>
<td>5</td>
</tr>
<tr>
<td>23</td>
<td>5.22%</td>
<td>5</td>
<td>5</td>
<td>5.22%</td>
<td>5.22%</td>
<td>5</td>
</tr>
<tr>
<td>24</td>
<td>5.22%</td>
<td>5</td>
<td>5</td>
<td>5.22%</td>
<td>5.22%</td>
<td>5</td>
</tr>
<tr>
<td>25</td>
<td>5.22%</td>
<td>5</td>
<td>5</td>
<td>5.22%</td>
<td>5.22%</td>
<td>5</td>
</tr>
<tr>
<td>26</td>
<td>5.22%</td>
<td>5</td>
<td>5</td>
<td>5.22%</td>
<td>5.22%</td>
<td>5</td>
</tr>
<tr>
<td>27</td>
<td>5.22%</td>
<td>5</td>
<td>5</td>
<td>5.22%</td>
<td>5.22%</td>
<td>5</td>
</tr>
<tr>
<td>28</td>
<td>5.22%</td>
<td>5</td>
<td>5</td>
<td>5.22%</td>
<td>5.22%</td>
<td>5</td>
</tr>
<tr>
<td>29</td>
<td>5.22%</td>
<td>5</td>
<td>5</td>
<td>5.22%</td>
<td>5.22%</td>
<td>5</td>
</tr>
<tr>
<td>30</td>
<td>5.22%</td>
<td>5</td>
<td>5</td>
<td>5.22%</td>
<td>5.22%</td>
<td>5</td>
</tr>
<tr>
<td>31</td>
<td>5.22%</td>
<td>5</td>
<td>5</td>
<td>5.22%</td>
<td>5.22%</td>
<td>5</td>
</tr>
<tr>
<td>32</td>
<td>5.22%</td>
<td>5</td>
<td>5</td>
<td>5.22%</td>
<td>5.22%</td>
<td>5</td>
</tr>
<tr>
<td>33</td>
<td>5.22%</td>
<td>5</td>
<td>5</td>
<td>5.22%</td>
<td>5.22%</td>
<td>5</td>
</tr>
<tr>
<td>34</td>
<td>5.22%</td>
<td>5</td>
<td>5</td>
<td>5.22%</td>
<td>5.22%</td>
<td>5</td>
</tr>
<tr>
<td>35</td>
<td>5.22%</td>
<td>5</td>
<td>5</td>
<td>5.22%</td>
<td>5.22%</td>
<td>5</td>
</tr>
<tr>
<td>36</td>
<td>5.22%</td>
<td>5</td>
<td>5</td>
<td>5.22%</td>
<td>5.22%</td>
<td>5</td>
</tr>
<tr>
<td>37</td>
<td>5.22%</td>
<td>5</td>
<td>5</td>
<td>5.22%</td>
<td>5.22%</td>
<td>5</td>
</tr>
<tr>
<td>38</td>
<td>5.22%</td>
<td>5</td>
<td>5</td>
<td>5.22%</td>
<td>5.22%</td>
<td>5</td>
</tr>
<tr>
<td>39</td>
<td>5.22%</td>
<td>5</td>
<td>5</td>
<td>5.22%</td>
<td>5.22%</td>
<td>5</td>
</tr>
<tr>
<td>40</td>
<td>5.22%</td>
<td>5</td>
<td>5</td>
<td>5.22%</td>
<td>5.22%</td>
<td>5</td>
</tr>
<tr>
<td>41</td>
<td>5.22%</td>
<td>5</td>
<td>5</td>
<td>5.22%</td>
<td>5.22%</td>
<td>5</td>
</tr>
<tr>
<td>42</td>
<td>5.22%</td>
<td>5</td>
<td>5</td>
<td>5.22%</td>
<td>5.22%</td>
<td>5</td>
</tr>
<tr>
<td>43</td>
<td>5.22%</td>
<td>5</td>
<td>5</td>
<td>5.22%</td>
<td>5.22%</td>
<td>5</td>
</tr>
<tr>
<td>44</td>
<td>5.22%</td>
<td>5</td>
<td>5</td>
<td>5.22%</td>
<td>5.22%</td>
<td>5</td>
</tr>
<tr>
<td>45</td>
<td>5.22%</td>
<td>5</td>
<td>5</td>
<td>5.22%</td>
<td>5.22%</td>
<td>5</td>
</tr>
<tr>
<td>46</td>
<td>5.22%</td>
<td>5</td>
<td>5</td>
<td>5.22%</td>
<td>5.22%</td>
<td>5</td>
</tr>
<tr>
<td>47</td>
<td>5.22%</td>
<td>5</td>
<td>5</td>
<td>5.22%</td>
<td>5.22%</td>
<td>5</td>
</tr>
<tr>
<td>48</td>
<td>5.22%</td>
<td>5</td>
<td>5</td>
<td>5.22%</td>
<td>5.22%</td>
<td>5</td>
</tr>
<tr>
<td>49</td>
<td>5.22%</td>
<td>5</td>
<td>5</td>
<td>5.22%</td>
<td>5.22%</td>
<td>5</td>
</tr>
<tr>
<td>50</td>
<td>5.22%</td>
<td>5</td>
<td>5</td>
<td>5.22%</td>
<td>5.22%</td>
<td>5</td>
</tr>
<tr>
<td>51</td>
<td>5.22%</td>
<td>5</td>
<td>5</td>
<td>5.22%</td>
<td>5.22%</td>
<td>5</td>
</tr>
<tr>
<td>52</td>
<td>5.22%</td>
<td>5</td>
<td>5</td>
<td>5.22%</td>
<td>5.22%</td>
<td>5</td>
</tr>
<tr>
<td>53</td>
<td>5.22%</td>
<td>5</td>
<td>5</td>
<td>5.22%</td>
<td>5.22%</td>
<td>5</td>
</tr>
<tr>
<td>54</td>
<td>5.22%</td>
<td>5</td>
<td>5</td>
<td>5.22%</td>
<td>5.22%</td>
<td>5</td>
</tr>
<tr>
<td>55</td>
<td>5.22%</td>
<td>5</td>
<td>5</td>
<td>5.22%</td>
<td>5.22%</td>
<td>5</td>
</tr>
<tr>
<td>56</td>
<td>5.22%</td>
<td>5</td>
<td>5</td>
<td>5.22%</td>
<td>5.22%</td>
<td>5</td>
</tr>
<tr>
<td>57</td>
<td>5.22%</td>
<td>5</td>
<td>5</td>
<td>5.22%</td>
<td>5.22%</td>
<td>5</td>
</tr>
<tr>
<td>58</td>
<td>5.22%</td>
<td>5</td>
<td>5</td>
<td>5.22%</td>
<td>5.22%</td>
<td>5</td>
</tr>
<tr>
<td>59</td>
<td>5.22%</td>
<td>5</td>
<td>5</td>
<td>5.22%</td>
<td>5.22%</td>
<td>5</td>
</tr>
<tr>
<td>60</td>
<td>5.22%</td>
<td>5</td>
<td>5</td>
<td>5.22%</td>
<td>5.22%</td>
<td>5</td>
</tr>
</tbody>
</table>

*Note: If two or more types of agent find a civilian in the same cycle, they all get credit for it.*
### Table 10: Civilians – Civilians killed and Civilians rescued.

<table>
<thead>
<tr>
<th>Total Number of Casualties</th>
<th>Total Number of Ambulances</th>
<th>Total Number of Fatigues</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Chart 1 and Chart 2 were also generated. 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 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 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 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.

<table>
<thead>
<tr>
<th>Team:</th>
<th>FCP ortugal</th>
<th>Map:</th>
<th>KobeHard</th>
</tr>
</thead>
<tbody>
<tr>
<td>Score Data</td>
<td></td>
<td>Civilians Discovered</td>
<td></td>
</tr>
<tr>
<td>Total Number of Agents</td>
<td>99</td>
<td>Civilians Discovered by FireBrigades*</td>
<td>10</td>
</tr>
<tr>
<td>Number of Agents not Killed</td>
<td>58</td>
<td>Civilians Discovered by Police Forces*</td>
<td>44</td>
</tr>
<tr>
<td>Percentage of Building Area Destroyed</td>
<td>31.19%</td>
<td>Civilians Discovered by Ambulances*</td>
<td>5</td>
</tr>
<tr>
<td>Initial Score</td>
<td>100.00</td>
<td>Civilians not Discovered*</td>
<td>12</td>
</tr>
<tr>
<td>Points Lost Due to Casualties</td>
<td>43.16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Points Lost Due to Burned Buildings</td>
<td>14.98</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Final Score</td>
<td>41.64</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Casualties</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total Number of Civilians</td>
<td>70</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Civilians Killed by Fire</td>
<td>12</td>
<td>Total Number of FireBrigades</td>
<td>13</td>
</tr>
<tr>
<td>Civilians Killed by Debris</td>
<td>27</td>
<td>Total Number of Police Forces</td>
<td>10</td>
</tr>
<tr>
<td>Civilians not Killed</td>
<td>31</td>
<td>Total Number of Ambulances</td>
<td>6</td>
</tr>
<tr>
<td>Total Number of Rescue Agents Killed</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Simulation date:</td>
<td>07/15/10.29.20</td>
<td>Civilians Discovered by Each FireBrigade</td>
<td>0.77</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Civilians Discovered by Each Police Force</td>
<td>4.40</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Civilians Discovered by Each Ambulance</td>
<td>0.83</td>
</tr>
</tbody>
</table>
4.4 Team Comparison

The tables from the previous section provide 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 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.

![Score Analysis Chart](chart3.png)

*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.

![Discovered Civilians Chart](chart4.png)

*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 Impossibles 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) Impossibles - Civilians casualties (stacked).](image)

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 Impossibles' 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.](image)
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 time cycle 70, all the Impossibles’ 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, such as 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 Impossibles’ 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 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.

<table>
<thead>
<tr>
<th>Display Name</th>
<th>FCPortugal</th>
<th>Impossibles</th>
<th>FCPortugal</th>
<th>Impossibles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Score Data</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total Number of Agents</td>
<td>99</td>
<td>99</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of Agents not killed</td>
<td>96.99%</td>
<td>96.99%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Percentage of Building Area Destroyed</td>
<td>31.19%</td>
<td>31.19%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total Time</td>
<td>10:29:30</td>
<td>10:29:30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Points Lost Due to Civilian Casualties</td>
<td>43.98%</td>
<td>43.98%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Points Lost Due to Burned Buildings</td>
<td>14.98%</td>
<td>14.98%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Final Score</td>
<td>41.84%</td>
<td>41.84%</td>
<td>80.15%</td>
<td>80.15%</td>
</tr>
</tbody>
</table>

Civilian Casualties FCPortugal Impossibles

<table>
<thead>
<tr>
<th>Civilian Casualties</th>
<th>FCPortugal</th>
<th>Impossibles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Number of Civilians</td>
<td>70</td>
<td>70</td>
</tr>
<tr>
<td>Civilians killed by Fire</td>
<td>37.14%</td>
<td>37.14%</td>
</tr>
<tr>
<td>Civilians killed by Burns</td>
<td>28.57%</td>
<td>28.57%</td>
</tr>
<tr>
<td>Civilians not killed</td>
<td>44.29%</td>
<td>44.29%</td>
</tr>
</tbody>
</table>

Note: As explained in the Civilians tab, these parts may not add up to the total number of civilians.

Civilians Discovered (Normalized by Agent Type) FCPortugal Impossibles

<table>
<thead>
<tr>
<th>Civilians Discovered</th>
<th>FCPortugal</th>
<th>Impossibles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Civilians Discovered by Fire Forces*</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>Civilians Discovered by Police Forces*</td>
<td>44</td>
<td>44</td>
</tr>
<tr>
<td>Civilians Discovered by Ambulances*</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Civilians not Discovered*</td>
<td>12</td>
<td>12</td>
</tr>
</tbody>
</table>

*You may change the display names as you please.

Table note: As explained in the Civilians tab, these parts may not add up to the total number of civilians.
Table 12 provides us with a summary of the teams’ 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; allow the Fire Brigades to act as scouts when they are able to rapidly control the fires; and reduce the amount of points lost due to burned Civilians. 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 integration into Freiburg’s 3D viewer future releases is expected. From sections 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 Police Force agents from a team to cooperate with Fire Brigades 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 with such a change. However, if there was a 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 simu-
lation 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.


4. Reis, L.P., Coordination in Multi-Agent Systems: Applications in University Management and Robotic Soccer, Department of Engineering, 2003, University of Porto, Porto


8. Committee, R., RoboCup rescue simulator manual v0.4, 2000, The RoboCup Rescue Technical Committee


15. Bowling, M., Robocup Rescue: Agent Development Kit, 2000, Dept. of Comp. Science, Univ. of Alberta


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

Appendixes

Appendixes in this document were written in Portuguese. They were written especially so that new FC Portugal’s team members could effortless start to interact with the system.

1 Appendix A: Simulator Installation

1.1 Processo de Instalação do Simulador

O simulador pode ser encontrado em: http://sourceforge.net/projects/roboescue/

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:

\textit{civillian}\n
Problema:

\texttt{/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

**Solução:**

instalar pacote:
glibc-static-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(java.awt.JComponent)`:

viewer/Viewer.java:71: error: Can't find constructor `java.awt.JCheckBox(java.lang.String,Z)` in type `java.awt.JCheckBox`.

    final JCheckBox doAnimateCheckBox = new JCheckBox("Animation", 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/Projeto/rescue-0.46alpha-unix/program/civilian/itk'

make[1]: *** [itk/libitk.a] Error 2

make[1]: Leaving directory `/home/joao/Projeto/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 -l osg... 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
Following the installation instructions included in the README.txt included in the package.

### 1.1.1.2 Segunda Instalação:

The simulator was also installed on a Compaq Presario X1000 laptop, with a customized version of the Gentoo distribution named x1000v3, where the same difficulties occurred, which were resolved in the same way.

### 1.1.1.3 Terceira Instalação:

Installation in Suse 9.2 on a portable Acer Aspire 1694 wlimi

Package installed:

- Development tools C and C++
- Java sdk
- Flex
- OSG+OT+OP (requires X11-devel; glx; libs-devel: jpeg, gif, tiff, png)
2 Appendix B: Simulator Execution

2.1.1 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 nome do ficheiro log)

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

./all.sh nome do mapa nome do arquivo log

ex: ./all.sh Kobe avaliacao

arquivo gerado: 0621-111517-avaliacao-Kobe.log
O nome do arquivo é mmdd-hhmmss-nome do arquivo-log mapa.log

__
all2.sh
idêntico a all.sh mas guarda logs de cada um dos modulos para ficheiros.log com o respec-
tivo nome do modulo (ainda não alterado para funcionar)

__
run.sh
Idêntico a all.sh mas sem correr o modulo viewer (corre só a simulação guardando num ficheiro log)

game.sh
Chama all.sh excepto se o número de argumentos for diferente de 2, apresentando um modo de uso do comando

### 2.1.2 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 corer tudo de uma vez.

Estes dois scripts foram desenvolvidos pelos membros deste grupo de trabalho.

Os detalhes adicionais do visualizador 3D são conseguidos através do comando ./3DRescue na pasta program/viewer/3DRescue,

- **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)
- **-of <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
r  Reduce speed
i  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

2.1.3 Visualização de Logs

O arquivo gerado pelo processo de simulação permite uma análise offline do progresso da simulação. Você 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 equipes 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

2.1.4 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.

2.1.5 Scripts alterados e criados

A seguinte lista de scripts alterados na pasta boot são:

- all.sh

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

- all2.sh

  Parâmetros foram adaptados, variáveis foram acrescentadas, temporizações foram ajustadas para garantir uma correcta ordem de execução de módulos e permitir a ligação dos módulos java.
- run.sh

Parâmetros foram adaptados, variáveis foram acrescentadas, temporizações foram ajustadas para garantir uma correcta 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.

- 4morimotrafficssimulator.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 à 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: ambulância, policias, bombeiros, Centro de ambulâncias, Esquadra de polícia e Quartel de bombeiros.

O seguinte script foi criado na pasta fc_portugal:

- kill_agent.sh

Termina a execução dos seguintes agentes: ambulância, policias, bombeiros, Centro de ambulâncias, Esquadra de polícia e Quartel de bombeiros.
2.1.6 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 sub-simuladores:

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

///

Agent Related Data
Burried Police Officers Killed by Fire
fire brigades Killed by Fire (Not Burried)
Burried Ambulances Killed by Fire
Number of Buried fire brigades (Alive)
Burried 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 Burried)
Number of Buried Police Forces (Alive)

(AMBULANCE)
Agent Related Data
Burried Police Forces Killed by Debris
Burried Police Forces Killed by Fire
Burried Police Forces (Alive)
Ambulances Killed by Fire (Not Burried)
Burried Ambulances Killed by Debris
Burried Ambulances Killed by Fire
Burried Ambulances (Alive)
Total Number of Rescue Agents Unburried
Burried fire brigades Killed by Debris
Burried fire brigades Killed by Fire
Burried fire brigades (Alive)
Total Number of Burried 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 Fire Brigades
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 $(u,v)$.