7^{TH} INTERNATIONAL CONFERENCE

ON

INTELLIGENT GAMES AND SIMULATION

GAME-ON® 2006

EDITED BY

Lars Wolf

and

Marcus Magnor

NOVEMBER 29-DECEMBER 1, 2006

TU BRAUNSCHWEIG
GERMANY

A Publication of EUROSIS-ETI

Cover art was reproduced by kind permission of Larian Studios, Oudenaarde, Belgium

7TH International Conference on Intelligent Games and Simulation

BRAUNSCHWEIG, GERMANY

NOVEMBER 29 - DECEMBER 1, 2006

Organised by

ETI

Sponsored by

EUROSIS

Co-Sponsored by

Binary Illusions

University of Bradford

Delft University of Technology

Ghent University

Larian Studios

Liverpool John Moores University

The Moves Institute

Simulation First

Hosted by

TU Braunschweig

Braunschweig, Germany

EXECUTIVE EDITOR

PHILIPPE GERIL
(BELGIUM)

EDITORS

General Conference Chair
Lars Wolf
TU Braunschweig
Institute of Operating Systems
and Computer Networks
Braunschweig, Germany

General Programme Chair
Marcus Magnor
TU Braunschweig
Computer Graphics Lab
Braunschweig, Germany

PROGRAMME COMMITTEE

PROGRAMME COMMITTEE

Paolo Remagnino
Kingston University
Kingston, United Kingdom

Marco Rocetti
University of Bologna
Bologna, Italy

Marcos Rodrigues
Sheffield Hallam University
Sheffield, United Kingdom

Leon Rothkrantz
Delft University
Delft, The Netherlands

Leon Smalov
Coventry University
Coventry, United Kingdom

Adam Szarowicz
Kingston University
Kingston, United Kingdom

Ingo Steinhäuser
Binary Illusions
Braunschweig, Germany

Oryal Tanir
Bell Canada
Montreal, Canada

Joao Tavares
University of Porto
Porto, Portugal

Ruck Thawonmas
Ritsumeikan University
Shiga, Japan

Christian Thurau
University of Bielefeld
Bielefeld, Germany

Hans Vangheluwe
McGill University
Montreal, Canada

Clark Verbrugge
McGill University
Montreal, Canada

Tina Wilson
Open University
Milton Keynes, United Kingdom

Michael Young
NC State University
Raleigh, USA

Michael Zyda
USC Viterbi
Monterey, USA

All articles published in this Proceedings have been peer reviewed

EUROSIS-ETI Publications are ISI-Thomson and INSPEC referenced

For permission to publish a complete paper write EUROSIS, c/o Philippe Geril, ETI Executive Director, Ghent University, Faculty of Engineering, Dept. of Industrial Management, Technologiepark 903, Campus Ardoyen, B-9052 Ghent-Zwijnaarde, Belgium.

EUROSIS is a Division of ETI Bvba, The European Technology Institute, Torhoutsesteenweg 162, Box 4, B-8400 Ostend, Belgium

Printed in Belgium by Reproduct NV, Ghent, Belgium
Cover Design by Grafisch Bedrijf Lammaing, Ostend, Belgium
Cover Pictures by Larian Studios, Oudenaarde, Belgium

EUROSIS-ETI Publication

ISBN-NUMMER-10 : 9077381317
ISBN-NUMMER-13 : 9789077381311
EAN : 9789077381311

GAME ON®
2006

Preface

It is our pleasure to welcome you to this event in the city of Braunschweig and at the Technische Universität Carolo-Wilhemina zu Braunschweig, the oldest technical university in Germany.

Since a couple of years, the interest in computer-based games has increased significantly not only on the commercial side but also in research. Several conferences and workshops have been set up, specializing in this exciting new field. GameOn is a well established conference series and looks back already on several successful years. It is a vivid event with interesting presentations and discussions.

GameOn2006 -- the 7th annual European Game-On Conference – brings together researchers and games people in order to exchange ideas on programming and programming techniques, being beneficial to the gaming industry and academia.

The conference will treat various aspects of computer-based games, from graphics, artificial intelligence, to robotics and mobile gaming issues. Two invited talks by Eku Wand and Maic Masuch will provide further insights into current and future game-technology trends. Moreover, tutorials on Java Gaming and the workshop on "Gaming with AIBO Robots", as well as a demonstration of a student strategy game project round up the conference program.

We hope that all attendees will enjoy the conference, the technical program as well as the possibilities to exchange ideas and network with colleagues from various countries to provide a base for fruitful collaborations in future.

Marcus Magnor
Lars Wolf

GameOn2006 Program Chairs

CONTENTS

CONTENTS

SCIENTIFIC PROGRAMME

TUTORIAL

THE MYTHS (AND TRUTHS) OF JAVA GAMES PROGRAMMING

Andrew Davison
Department of Computer Engineering
Prince of Songkla University
Hat Yai, Songkhla 90112
Thailand
E-mail: ad@fivedots.coe.psu.ac.th

KEYWORDS

java, games programming, myths, criticisms

ABSTRACT

This paper examines the commonly-expressed criticisms of Java as a games programming language: that's it's too slow, too high-level, prone to memory problems, too hard to install, not available on games consoles, not used in 'real' games, and not even considered a gaming platform by Sun Microsystems. All of these views are incorrect, aside from the console issue.

INTRODUCTION

Java for games programming: are you joking? No, Java is a great games programming language. When you learnt Java, I'm sure it's many advantages were mentioned: an elegant object-oriented paradigm, cross-platform support, code reuse, ease of development, tool availability, reliability and stability, good documentation, support from Sun Microsystems, low development costs, the ability to use legacy code (e.g. C, C++), and increased programmer productivity (Eckel 2006). That list leaves out my personal reason for programming in Java – *it's fun*, especially when you're programming something inherently good-for-you, such as games.

Most Java-bashers tend to skip over advantages, preferring to concentrate on criticisms. Here's a typical list:

- Java is too slow for games programming;
- Java has memory leaks;
- Java is too high-level;
- Java application installation is a nightmare;
- Java isn't supported on games consoles;
- No one uses Java to write 'real' games;
- Sun Microsystems isn't interested in supporting Java gaming.

Almost all of these objections are *substantially wrong*. Java is roughly the same speed as C++. Memory leaks can be avoided with good programming, and techniques like profiling. Yes, Java is high-level, but it also offers more direct access to graphics hardware and external devices. Installation isn't a nightmare, if you use decent installation software. There's a growing number of excellent, fun Java games, and an enormous amount of support available from Sun and Sun-sponsored sites.

If you're keeping count, I haven't disagreed with the lack of a games consoles port, which is a tad embarrassing for a "write once, run anywhere" language. Things may be changing in this category, as I explain below.

A general point about these objections is that they had more validity in the late 1990s, when the language and its libraries were less sophisticated and slower. The 1990's were a long time ago – Java's user and developer communities are currently burgeoning, producing a plethora of useful tools, online help, and code examples.

Now, back to the criticisms...

JAVA IS TOO SLOW FOR GAMES PROGRAMMING

This is better rephrased as "Java is slow compared to C and C++, the dominant languages for games programming at the moment." This argument was valid when Java first appeared (around 1996), but has become increasingly ridiculous with each new release. Some figures put JDK 1.0, that first version of the language, at *20 to 40 times slower* than C++. However, J2SE 5—the current release—is typically only *1.1 times slower*. Many benchmarks indicate that Java SE 6 is about 20% faster than J2SE 5.

These numbers depend greatly on the coding style used. Java programmers must be good programmers in order to utilize Java efficiently, but that's true of any language. Jack Shirazi's Java Performance Tuning site (http://www.javaperformancetuning.com/) is a good source for performance tips, and links to tools and other resources.

The speed-up in Java is mostly due to improvements in compiler design. The Hotspot technology introduced in J2SE 1.3 enables the run-time system to identify crucial areas of code that are utilized many times, and these are aggressively compiled. Hotspot technology is relatively new, and it's quite likely that future versions of Java will yield further speed-ups. For example, J2SE 5.0 is 1.2 to 1.5 times faster than its predecessor (version 1.4).

Hotspot technology has the unfortunate side-effect that program execution is often slow at the beginning until the code has been analyzed and compiled.

Swing is Slow

Swing often comes under attack for being slow. Swing GUI components are created and controlled from Java, with little OS support; this increases their portability and makes them more controllable from within a Java program. Speed is supposedly compromised because Java imposes an extra layer of processing above the OS. This is one reason why some games applications still utilize the original Abstract Windowing Toolkit (AWT)—it's mostly just simple wrapper methods around OS calls.

Even if Swing is slow (and I'm not convinced of that), most games don't require complex GUIs; full-screen game play with mouse and keyboard controls is the norm. GUI elements maintained by Swing, such as menu bars, button, and text fields, aren't needed, while mouse and keyboard processing is dealt with by the AWT. The latest versions of Java offer a very efficient full-screen mode by suspending the normal windowing environment.

My Program is Slow (Because of Java)

A crucial point about speed is knowing where to lay the blame when a program runs slowly. Typically, a large part of the graphics rendering of a game is handled by hardware or software outside of Java. For example, Java 3D passes its rendering tasks down to OpenGL or DirectX, which may emulate hardware capabilities such as bump mapping. Often the performance bottleneck in network games is the network, not the Java language.

JAVA HAS MEMORY LEAKS

When C/C++ programmers refer to memory leaks in Java, it often means that they don't understand how Java works. Java doesn't offer pointer arithmetic, and typical C-style memory leaks—such as out-of-bounds array accesses—are caught by the Java compiler.

However, these programmers may mean that objects which are no longer needed by the program are not being garbage collected. This becomes an issue if the program keeps creating new objects—requiring more memory—and eventually crashes when the maximum memory allocation is exceeded.

This kind of problem is a consequence of bad programming style, since the garbage collector can only do its job when an object is completely dereferenced, meaning the program no longer refers to the object. A good profiling tool, such as JProfiler (http://www.ej-technologies.com/products/jprofiler/overview.html), can be a great help in identifying code using excessive amounts of memory. JProfiler is a commercial product; many open source profilers are listed at http://java-source.net/; Java SE 6 comes with a great graphical profiler, jhat.

Another memory-related complaint is that the Java garbage collector is executing at poorly timed intervals, causing the application to halt for seconds while the collector sweeps and cleans. The JVM comes with several different garbage collectors, which collect in various ways, and can be selected and fine-tuned from the command line. Information on the performance of the chosen collector can be gathered and analyzed, and Java SE 6 offers many tools for these tasks, including jps, jstat, jhat, and jstack.

JAVA IS TOO HIGH-LEVEL

This complaint is the age old one of abstraction versus speed and control. The details of the argument often include the following statements:

1. Java's use of classes, objects, and inheritance add too much overhead without enough coding benefit;
2. Java's machine independence means that low-level, fast operations—such as direct Video RAM I/O—are impossible.

Statement 1 ignores the obvious benefits of reusing and extending Java's very large class library, which includes high-speed I/O, advanced 2D and 3D graphics, and an enormous range of networking techniques, from lowly sockets to distributed agents. Also forgotten are the advantages of object-oriented design, typified by UML, which makes complex, large real-world systems more manageable during development, implementation, and maintenance.

Statement 2 impacts gaming when we consider high-speed graphics, but it's been addressed in recent versions of Java. J2SE 1.4 introduced a full-screen exclusive mode (FSEM), which suspends the normal windowing environment, and allows an application to more directly access the underlying graphics hardware. It permits techniques such as page flipping, and provides control over the screen's resolution and image depth. The principal aim of FSEM is to speed up graphics-intensive applications, such as games. A lot of the behind-the-scenes speed improvements in Java SE 6 are related to graphics rendering using OpenGL and DirectX.

Statement 2 also comes into play for game peripherals, such as joysticks and game pads; machine independence seems to suggest that non-standard I/O devices won't be useable. Java games requiring these types of devices can utilize JNI, the Java Native Interface, to link to C or C++, and therefore to the hardware. There's also JInput, a very versatile Java-based game controller API (https://jinput.dev.java.net/).

An interesting historical observation is that the gaming community used to think that C and C++ were too high-level for fast, efficient games programming, when compared to assembly language. Opinions started to change only after the obvious success of games written in C, such as Doom and Dungeon Master, in the mid 1980s. Also important was the appearance of cross-platform development tools that supported C, such as Renderware.

JAVA APPLICATION INSTALLATION IS A NIGHTMARE

The general point made here is that a user needs to be a Java expert in order to install and execute a Java application, whereas most game players just want to point and click on a few dialog boxes to get a game up and running. More specific comments include:

1. Java (specifically, the JRE) has to be on the machine before the application will run.
2. Code bloat—even small programs require a 16 MB JRE. Downloading this can be very slow.
3. Frequently changing JVMs make it hard to write code that will work for every possible version of Java.
4. Non-standard components are often required (e.g. Java 3D), causing even more installation problems.
5. It's not possible to compile the application for a specific platform.
6. The .jar extension is commonly hijacked by other software (e.g. by compression programs) at execution time, meaning that the user can't just double click on a JAR to get it to start.
7. The JRE is slower to start up compared to a native compiled application.

All these problems—aside from 2 and 7 perhaps—can be solved by using good installation software. Java applets can be delivered via the Web, and the Java SE 6 plug-in for Internet Explorer and Netscape starts very quickly. Java Web Start (JWS) can be utilized to download applications, and has been improved significantly since J2SE 1.4. There's numerous third-party installers, such as install4j (http://www.ej-technologies.com/products/install4j/overview.html).

The code bloat comment is increasingly irrelevant, with many games weighing in at over 100 MB, and even many graphics and sound card drivers are larger than 15 MB. Adobe Acrobat requires around 25 MB, Real Player 13 MB, and .NET 23 MB. Network speeds are a problem, especially overseas, but broadband usage is growing rapidly.

Sun Microsystems estimates that around 70% of all new PC's come with a JRE pre-installed, although a game installer must still cater for the others.

There's some truth to point 7, but the slow start-up time is fairly negligible compared to the total running time of an average game. Also, Java SE 6's splash screen feature can be employed to 'entertain' the user during start-up.

JAVA ISN'T SUPPORTED ON GAMES CONSOLES

Unfortunately, this criticism has some justification. Video gaming is a multi-billion dollar industry, with estimates placing revenues at $29 billion by 2007—the market will cater to over 235 million gamers. PCs and game consoles account for almost all the income, but only about 10-20% of it is from PCs, the majority coming from three consoles: Sony’s PlayStation 2 (PS2), Microsoft’s XBox, and Nintendo’s GameCube. Sony is the dominant console maker, having nearly twice as many units in homes compared to Microsoft and Nintendo combined. Microsoft accounts for about 95% of the desktop PC market. Arguably, there are only two important games platforms: the PS2 and Windows—and Java isn't available on the PlayStation.

This problem has long been recognized by Sun: back at the JavaOne conference in 2001, Sony and Sun announced their intention to port the JVM to the PS2. Nothing was ever officially released, although it is possible to run Java on Sony's version of Linux, but the OS requires the PS2 to have a hard disk, and only has limited access to the PS2's other hardware.

The difficulties of this approach should be contrasted to the availability of feature rich C/C++ tools and engines for consoles, such as RenderWare (http://www.renderware.com/) and Gamebryo (http://www.ndl.com/). They have a track record of best-selling games, and can port games across the PS2, Xbox, GameCube, and PCs.

In the future, Java may have a better chance of acceptance into the closed-world of console makers because of two trends: consoles are mutating into home media devices, and the meteoric rise of online gaming. Both require consoles to offer complex networking and server support, strong areas for Java and Sun.

The prospects for Java on the PlayStation 3 (PS3) look fairly bright. Both the basic and premium PS3 versions will have 512 MB of RAM, a large hard drive, will support Linux, and use an extended version of OpenGL. Sony's software development chief, Izumi Kawanishi, has spoken of making it easier for individuals to create games on the PS3. Development kits are expected to appear in Spring 2007.

Applications will be written in a high-level, object-oriented language, but currently there's no word on what it'll be. It's likely that a virtual machine will execute the code, utilizing JIT technology.

The PS3 will include a Blu-ray disc for storing high-definition video and data. All Blu-ray drives support a version of Java called BD-J for implementing interactive menus and other GUIs. Also, Blu-ray's network connectivity means that BD-J can be utilized for networking applications such as downloading subtitles, short movies, and adverts.

The lack of Java on consoles **is** a serious issue, but the remaining PC market is far from miniscule. The Gartner Group believes there are 661 million PC users in 2006. The number will hit 953 million by the end of 2008, and cross over the billion mark in 2009.

Games on PCs benefit from superior hardware—such as video cards, RAM, and internet connections—so can offer more exciting game play. There are many more PC games, particularly in the area of multiplayer online games.

Another rapidly expanding market is the one for mobile games, with sales of $530 million in 2003, potentially rising to $2.5 billion in 2007. There are thought to be around 250 million Java-enabled phones at the moment.

NO ONE USES JAVA TO WRITE REAL GAMES

The word "real" here probably means commercial games. The number of commercial Java games is small compared to ones coded in C++ or C, but the number is growing, and many have garnered awards and become bestsellers: Tribal Trouble, Puzzle Pirates, Call of Juarez, Chrome, Titan Attacks, Star Wars Galaxies, Runescape, Alien Flux, Kingdom of Wars, Law and Order II, Ultratron, Roboforge, IL-2 Sturmovik, Galactic Village, and Wurm Online. Many are written entirely in Java, others employ Java in sub-components such as game logic.

Java is used widely in the casual gaming market, where game-play is generally less complex and time-consuming. Implementation timelines are shorter, budgets smaller, and the required man-power is within the reach of small teams. By 2008, industry analysts believe the casual games market will surpass $2 billion in the US alone.

There are numerous Java gaming sites, including a showcase at Sun Microsystems (http://www.java.com/en/games/), community pages at http://community.java.net/games/, a collection of open-source gaming tools at https://games.dev.java.net/, the Java Games factory (http://javagamesfactory.org/), works-in-progress at https://games-forge.dev.java.net/, and many, very helpful forums at http://www.javagaming.org/.

There are several excellent books on Java games programming (Brackeen *et al.* 2003, Clingman *et al.* 2004, Croft 2004, Davison 2005).

SUN MICROSYSTEMS ISN'T INTERESTED IN SUPPORTING JAVA GAMING

The games market isn't a traditional one for Sun, and it'll probably never have the depth of knowledge of a Sony or Nintendo. However, the last few years have demonstrated Sun's increasing commitment to gaming.

J2SE has strengthened its games support through successive versions: version 1.3 improved its graphics and audio capabilities, and version 1.4 introduced full screen mode and page flipping in hardware. Faster I/O, memory mapping, and support for non-block sockets, which is especially useful in client/server multiplayer games, also appeared first in 1.4. Version 5.0 has a decent nanosecond timer at last. Java extension libraries, such as Java 3D, the Java Media Framework (JMF), the Java Communications API, Jini, and JAXP (Java's peer-to-peer API) all offer something to games programmers. Java SE 6 has improved graphics rendering speeds, and offers new features useful for gaming, such as splash screens, scripting, and a desktop API.

Sun started showing an interest in gaming back in 2001, with its announcement of the Java Game Profile, a collaboration with several other companies, including Sega and Sony, to develop a Java gaming API. The profile was perhaps too ambitious, and was abandoned at the end of 2003. However, it did produce three game-focused technologies: a Java binding for OpenGL called JOGL, a binding for OpenAL (a 3D audio library) called JOAL, and JInput.

Part of the 2001 initiative was the creation of the JavaGaming.org website (http://www.javagaming.org), initially manned by volunteers. In 2003, the Game Technology Group was formed, and JavaGaming.org received a substantial makeover as part of the creation of the new java.net portal (http://www.java.net) aimed at the technical promotion of Java. java.net hosts many discussion forums, user groups, projects, communities, and news. The communities include: Java Desktop, Java Education and Learning, Java Enterprise, and Java Games.

The Java Games community pages can be accessed through http://www.javagaming.org or http://community.java.net/games/. The site includes Java games forums, projects, news, weblogs, a wiki (http://wiki.java.net/bin/view/Games/WebHome), and links to games affiliates.

Numerous Java game forums can be accessed from http://www.javagaming.org/forums/index.php. These are probably the best sources of technical advice on Java gaming on the Web, with over 8500 highly opinionated registered users. Discussion topics include Java 3D, Java 2D, Java Sound, J2ME, networking, online games development, performance tuning, JOGL, JOAL, and JInput. There are also sections on projects and code examples.

The project sections (https://games.dev.java.net/) mostly concentrate on JOGL, JOAL, and JInput, but the games-middleware and games-forge sections are wider ranging. The games-forge projects include Chinese chess, jbantumi (a strategic game from Africa), and an online fantasy football management system.

Sun’s substantial presence at http://community.java.net/games/ is mostly as a host for community forums and open source projects (or projects with licenses very close to open source). The projects include JOGL, JOAL, JInput, and Java 3D. Sun is relying on community involvement to move these projects forward, since the Game Technology Group is quite small (Twilleager *et al.* 2004).

One in-house product is Sun's Project DarkStar (http://games-darkstar.dev.java.net), aimed at developing tools for supporting massive multi-player online games. The Sun Game Server (SGS) is it's server-side platform, and there are client APIs for C++, Java SE, and Java ME

REFERENCES

Brackeen , D., Barker, B., Vanhelswue. L. 2003. *Developing Games in Java*, New Riders Games; August.

Clingman, C., Kendall. S., and Mesdaghi . S. 2004. *Practical Java Game Programming*, Charles River Media, June

Croft, D.W. 2004. *Advanced Java Game Programming*, Apress, April.

Davison, A. 2005. *Killer Game Programming in Java*, O'Reilly Media, May.

Eckel, B. 2006. *Thinking in Java*, Prentice Hall, 4th ed., February

Twilleager, D., Kesselman, J., Goldberg, A., Petersen, D., Soto, J.C., and Melissinos, C. 2004. "Java Technologies For Games", *ACM Computers in Entertainment*, Vol. 2, No. 2, April.

BIBLIOGRAPHY

ANDREW DAVISON received his Ph.D. from Imperial College in London in 1989. He was a lecturer at the University of Melbourne for six years before moving to Prince of Songkla University in Thailand in 1996.

His research interests include scripting languages, logic programming, visualization, and teaching methodologies. This latter topic led to an interest in teaching games programming in 1999.

His O'Reilly book, *Killer Game Programming in Java*, was published in 2005, accompanied by a website at http://fivedots.coe.psu.ac.th/~ad/jg/.

GRAPHICS AND RENDERING

OPTIMIZING THE DEFERRED SHADING

Frank Puig Placeres
University of Informatics' Sciences
Cuba
E-mail: fpuig@fpuig.cjb.net

KEYWORDS

Rendering, Deferred Shading, Optimization.

ABSTRACT

Lighting in today's applications is performed by batching light sources together into small groups of 3-8 lights that can be managed by current shaders. For each of these groups the scene is rendered and the light's contribution is added into the framebuffer, which at the end contains the influence of all lights in the world.

A better solution is presented that uses Deferred Shading. This technique simplify the rendering of multiple special effects. It also reduces the overhead of performing several passes on the scene by reducing it to just rendering a full screen quad on each pass instead of the complete scene geometry.

Nonetheless, it is true that Deferred Shading has several disadvantages. It needs a lot of memory to store the Geometric Buffer, it produces a noticeable impact on fill rate and can't handle transparency efficiently, among others.

This article presents solutions to overcome or reduce most of the Deferred Shading drawbacks and give tips on how to implement a scalable system that can run on more graphic cards. The system uses less memory and optimizes the whole process not only in the shaders but also from the application itself by implementing high-level managers that run on the CPU.

MEMORY OPTIMIZATION

In a Deferred Shading system, lighting and other special effects are not computed in the same pass were the scene geometry is processed. Instead, there is a first pass were the scene geometry is rendered and per-pixel's attributes like Position and Normal are saved into several textures composing an auxiliary buffer called a Geometric Buffer (G-Buffer). Those textures are then used on subsequent passes to get the pixel's geometric data without processing the scene again.

Common attributes like Position and Normal can use up to 3 floats each and there's also a need to store material values like specular power, glow factor and occlusion term among others. This can increase the memory footprint between 10 and 40 MB just to store auxiliary values for standard game resolutions. However, cleverly packing those values allows the reduction of the memory footprint.

Normal Vector

Just storing the three components of the normal vector as floats will take up to 12 bytes per pixel. But using the mathematic constraint that normals are unit-length vectors, then it's possible to compute one component given the other two by applying the equation:

$$z = \pm\sqrt{1 - x^2 - y^2}$$

When performing the lighting on View Space the front-faced polygons are always going to have same sign depending on the frame of reference used.

Further optimizations like restraining components into the 16 bits or 8 bits range and using look up texture instead of the given equation to compute the z component, allows reducing the memory footprint down to just 2 bytes and in some case also to accelerate the shader.

Position

Position doesn't have the same unit-length restraint that follows the normal vectors. However, It's possible to just save the distance from the camera to the pixel and then recreate the world space position by creating a ray from the camera to the screen location (which is known) and then multiply it by the stored distance. The above is illustrated in the following code:

G_Buffer.z= *length*(Input.PosInViewSpace);

Then to recreate the pixel position, the vertex shader computes the Screen Position (vSP) and use the previous value as follow:

EyeToScreen=*float3*(vSP.x*ViewAspect, vSP.y, invTanHalfFOV);
PíxelPos = *normalize*(EyeToScreen) *G_Buffer.z;

Material Attributes

While it could be ideal to store all material attributes like specular power, glow factor, occlusion term, etc into the G-Buffer, they consume too much memory. Applications that use one or two attributes could store them directly by first adjusting the number of bits assigned to them. For example, when the specular power is constrained to the values [1, 4, 10, 30] it can be saved using two bits, and the original value can be recreated inside the shader.

In most situations, more attributes are needed, but they don't change per pixel but per surface. In that situation, it's possible to pack the attributes describing the material and only store the index in the G-Buffer.

According to the number of materials, they can be accessed by packing all the attributes in the available unused shader constants, and retrieving them using an indexed look up in the shader, or by packing the values into a texture where each row represent a different material. Also the index can be adjusted to only use the number of bits that fits the maximum value.

FILL-RATE REDUCTION

Given that Deferred Shading systems rely on filtering image pixels, they are likely to become fill-rate limited. Most implementations, after filling the Geometry Buffer, just loop through every light source and apply the shader to each pixel. Instead, it's possible to implement a high level manager, which acts as a firewall by only sending to the pipeline the sources (lights and effects) that influence the final image and executing the shaders only on the pixels that are influenced by the effect or light.

This high level manager receives the list of all sources that should modulate the resulting render and general information like the bounding object and how strong the effect should be applied. (e.g. for a light it could be the brightness). When the list is received, two main stages are executed.

Social Stage

During this first stage, the manager filters the sources on the scene, producing a smaller list by executing the following pseudo code.

1. Visibility and occlusion algorithms are executed to discard lights whose influence is not appreciable.
2. Project visible Bounding Objects into screen space.
3. Combine similar sources that are too close in screen space or influencing almost the same screen area.
4. Discard sources with a tiny contribution due to its projected bounding object being too small or too far.
5. Constraint each pixel to be influenced by no more than a predefined number of sources. Pick the biggest, strongest and closer.

Runtime parameters can be defined to control how small should be the projection of a source to be discarded in step 4 or how much sources can be affecting a single pixel in step 5, etc.

When the average frame rate is high enough, the runtime parameters can be relaxed, and if the frame rate drops, the parameters could be lowered. This produces a fast way to control the fill-rate of the application and allows maintaining smooth frame rates.

Social Stage

This second stage will configure the shaders to reduce the processing cost of each source. First it classify the sources in two main groups, the global sources and the local sources.

Global Sources

These are meant to affect all screen pixels, implying that the shaders must be executed in a complete screen quad. Examples of these sources are big lights illuminating the entire world like the sun when outdoors, or special effects applied to the whole screen like Depth of Field, Fog, etc.

Global sources restrain the number of optimization and heavily depend on how optimized are the shaders being used. Fortunately, most of the time there are just a few of them.

Local Sources

Contrary to global sources, these only affect regions of the scene. Classical examples include small lights spread over the scene, effects like volumetric fog or heat hazes.

One way to process them is to render the bounding object so only the influenced pixels execute the shader. This can work for simple shapes like boxes, but could became slow when using spheres or other objects composed by a lot of polygons.

Another approach involves rendering a full screen quad, just like for global sources, but enabling clipping and rejection features to discard most non-influenced pixels. This should result in less performance impact but is not as accurate as the other approach.

While scissor test and clipping planes, can discard the pixels at the transform level. Dynamic branching can be used to reject them at the fragment level. For example, when dealing with spheres, the individual stage passes the sphere radius and center to the shader and if the pixel's distance to the center is found to be greater than the radius, the pixel is discarded. For Axis Aligned Bounding Boxes, the process is even easier and just involves checking if the position components are inside the box.

When dynamic branching is not supported, the stencil buffer can be used to emulate the behavior. In [ATI05] a technique is described to emulate dynamic branching by creating a pixel mask according to some conditions.

Another non-conservative optimization that can be applied is to use a sort of Level of Detail to decide the shader quality according to how far or small is the source. For example, this can be implemented by computing the full lighting equation on the closest pixels and remove components as the pixels get farther. This way, it's possible to use diffuse and specular contribution for closer pixels and discard the specular component as it goes far.

Special care has to be taken to avoid artifacts and sudden popping that can appear at the transitions points where the equation components are inserted or removed. This can be handled by fading the target component. For example, to discard the specular component, it can be modulated by a factor that start being 1 on the closer boundary to fully show its contribution and gradually fades to 0 according to the distance. From that point on, other shader can be used that doesn't compute the specular influence. This process is described in [fpuig05] and [fpuig06].

OTHER PERFORMANCE OPTIMIZATIONS

Even when the above solutions have being used to reduce the memory footprint and fill-rate, they are likely to improve the general performance as well. For instance, using less memory to store the G-Buffer allows a better use of the texture cache, which reduces texture transfers, thereby reducing the performance impact of packing the pixel attributes in the buffer.

Together with the presented optimizations, the deferred shading system can be further optimized. It's possible to batch sources that are going to be computed with the same shaders. They can even be collapsed into several source shapes and processed with a single shader that computes the contribution of more than one source.

When setting the source shaders, call states can be minimized by first finding those shaders that are going to be used, and sorting the sources according to them. Finally to minimize state changes, each shader is set only once and all sources that use it are processed.

Mixing screen resolutions can make a difference. This way, the texture that stores the pixels colors in the G-Buffer can be rendered at full size, but the texture that holds normals, positions, etc could use lower resolutions. Which implies that the lighting and shading must be computed on a render target which is ½ or ¼ of the viewport dimensions and then modulated by the full resolution color texture.

When processing the scene geometry to fill the G-Buffer, lights, fog and all other unneeded effects at that stage should be disabled. Also it can help to have two different representation of the scene both presenting the same geometry but batched differently. One representation is used to render the color texture in the G-Buffer. It maintains a group of polygons for each texture and each group has to be call with an independent draw call.

The other representation is used for rendering position and normal into the G-Buffer. There's no need to distinct between textures so the complete geometry can be batched and submitted to the pipeline with just one draw call.

On systems where multiple render targets are not supported and a multi-pass approach is used, the cheapest pass should be executed first to handle overdraw with the lowest penalty and then running the more intense passes with the benefit of not reprocessing the same pixels.

Other possibilities are to combine forward rendering with the deferred system. The per-vertex influence of some sources can be written while filling color values in the G-Buffer. This technique can be used when the geometry is highly tessellated, or the light sources are far.

Additionally, it's possible to use most of the general shader optimizations to further enhance the performance in a deferred system. A typical example is the use of half data instead of floats when the extra precision is not needed, etc.

TRANSPARENCY

Alpha Test is easy to integrate with a Deferred Shading system, but alpha blending requires the same pixels to be shaded several times. However the Geometric Buffer only stores information about a single pixel in each Texel. So blending on a deferred shading system is not as simple as on forward rendering.

Still, there are some hacks that can be done to allow blending on a deferred shading system. The easiest is not doing deferred rendering on polygons that need to be blended. In a first pass, the application can perform the deferred path for solid polygons and then forward rendering on all the transparent polygons with alpha blending enabled.

Another approach is to use *Screen Door Transparency*. This technique uses a stippling pattern to mask the transparent polygons in a way that some pixels of the background can be seeing through the mask. For instance, a pattern to represent 50% transparency will skip all the even pixels in one row and all the odd pixels in the next. When applying the stippling pattern to a polygon, the background can be seen through the masked pixels. The holes are so small that they aren't picked up by the eye and in fact the eye blends the nearest pixels giving the illusion of a transparent polygon.

Screen Door Transparency can be implemented directly on the deferred shading pipeline and doesn't require depth sorting. However the screen resolution needs to be relativity high in order to hide the masking pattern. To make it even harder to spot the mask, the pattern can be changed and offset every frame, which also has the advantage of producing better looking results when the transparency has a high depth complexity.

Further approaches can use depth peeling to break complexity into layers, which then can be blended one after the other by using a deferred shading path. Nonetheless, this technique involves executing the complete deferred pipeline for each layer, from filling the G-Buffer to source shading, which can seriously impact performance.

CONCLUSIONS

Deferred Shading is a nice solution to deal with multiple lights influencing a scene. It keeps everything simple, separated and allows handling next generation scenes with a high number of polygons, complex materials and lots of special effects and lights.

This article has presented several techniques to reduce some of its drawbacks, by properly planning the shader implementation and the use of the G-Buffer space while using a high level manger to increase the overall performance and scalability of the system. Also, most of the presented solutions can be controlled in real time, which allows adjusting the image quality according to the average frame.

The combination of these techniques and the potential of a Deferred Shading system to easily handle complex scenes while maintain a great performance, makes this technique a very attractive solution for highly detailed graphics on current and next generation systems.

REFERENCES

[ATI05] ATI, "Dynamic branching using stencil test," ATI Software Developer's Kit, June 2005.

[Delphi3D] "Deferred Shading" available online at www.delphi3d.net/articles/viewarticle.php?article=deferred.htm

[Fabio05] "Deferred Shading Tutorial" Available online at http://fabio.policarpo.nom.br/docs/Deferred_Shading_Tutorial_SBGAMES2005.pdf

[Pritchard04] Pritchard, Matt "Deferred Lighting and Shading" Available online at www.gdconf.com/conference/archives/2004/pritchard_matt.ppt.

[Puig05] Puig, Frank "Fast Per-Pixel Lighting with Many Lights", Graphic Programming Gems 6

[Puig06] Puig, Frank "Overcoming Deferred Shading Drawbacks", ShaderX5

Effective Multi-resolution Rendering and Texture Compression for Captured Volumetric Trees

Christian Linz, Marcus Magnor
Institut für Computergraphik, TU Braunschweig
Mühlenpfordtstr.23
38106 Braunschweig, Germany

Alex Reche-Martinez, George Drettakis
REVES/Inria Sophia-Antipolis
2004 route des lucioles BP 93
06902 Sophia Antipolis, France

KEYWORDS
Three-Dimensional Graphics and Realism

ABSTRACT

Trees can be realistically rendered in synthetic environments by creating volumetric representations from photographs. However, volumetric tree representations created with previous methods are expensive to render due to the high number of primitives, and have very high texture memory requirements. We address both shortcomings by presenting an efficient multi-resolution rendering method and an effective texture compression solution. Our method uses an octree with appropriate textures at intermediate hierarchy levels and applies an effective pruning strategy. For texture compression, we adapt a vector quantization approach in a perceptually accurate color space, and modify the codebook generation of the Generalized Lloyd Algorithm to further improve texture quality. In combination with several hardware acceleration techniques, our approach achieves a reduction in texture memory requirements by one order of magnitude; in addition, it is now possible to render tens or even hundreds of captured trees at interactive rates.

INTRODUCTION

Modeling and rendering trees has been a goal of computer graphics research since the early days of the field (Bloomenthal, 1985; de Reffye et al., 1988; Prusinkiewicz and Lindenmayer, 1990). While most of the effort has been in solutions to generate entirely synthetic trees (e.g., (Prusinkiewicz and Lindenmayer, 1990; de Reffye et al., 1988; Deussen et al., 1998)), an alternative is the approach to capture and render real trees (Shlyakhter et al., 2001; Reche-Martinez et al., 2004). For both synthetic and captured trees, however, polygonal representations (mainly of the leaves) result in objects which are very complex and thus expensive to render. In addition, generating geometric levels-of-detail (LOD) for disconnected triangle meshes, such as the leaves of a tree, is an unsolved problem; the few solutions proposed to date require mixing various different representations(e.g., (Neyret, 1998; Meyer and Neyret, 1998; Behrendt et al., 2005)). However, trees are a good candidate for volumetric representations (Reche-Martinez et al., 2004); one big advantage of such an approach are appropriate multi-resolution LOD structures resulting naturally from the hierarchical data structure representing the volume.

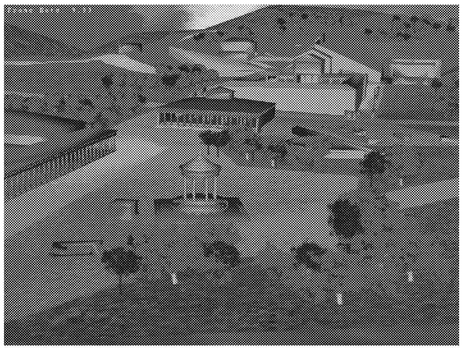


Figure 1: A scene with 290 trees running at 12 fps, and requiring 2.9 MB of texture memory for 3 different types of trees. Using the previous approach (Reche-Martinez et al., 2004), several seconds are required per frame and 641 MB texture memory are needed.

Although Reche et al. (Reche-Martinez et al., 2004) did use a volumetric representation, no multi-resolution solution was presented, and the texture memory requirements were prohibitively high. Despite the realistic renderings provided by the approach, the method remains unusable for all practical purposes (60,000-140,000 polygons and 60-140MB of texture memory per tree).

In this paper we present solutions to both the rendering speed and the texture memory problems. We present an efficient multi-resolution rendering approach, in which we choose the appropriate data-structure by creating textures for each level.In addition, we employ an efficient pruning strategy based on the properties of the generated textures. We then present a modified texture compression approach, choosing an appropriate color space during compression.To improve the results, we introduce a modification to the Generalized Lloyds Algorithm used during codebook generation for vector quantization. Finally, we use several graphics hardware acceleration techniques which allow us to achieve better performance and texture compression rates.

Overall, our technique allows us to render complex scenes containing tens or hundreds of trees at interactive frame rates. Texture memory consumption is reduced by two orders of magnitude. For example, the scene in Fig. 1 shows a scene with three types of trees using a total of 2.9 MB of memory, running at 10 fps. Using the previously existing approach, each frame would take tens of seconds to render, and 641 MB of texture memory would have be required. We believe that with these improvements, captured volumetric trees will become an interesting solution for games and other interactive 3D applications.

PREVIOUS WORK

In the interest of brevity, we will restrict our discussion to a selection of the most relevant previous work. Most previous methods concentrated on entirely synthetic trees based on procedural methods such as grammars (L-systems) (e.g., (Prusinkiewicz and Lindenmayer, 1990; Deussen et al., 1998)) or rule-based plant growing systems which use codified botanical knowledge such as the AMAP system (de Reffye et al., 1988). Such approaches have been used to create highly realistic images of forests and trees, albeit with high polygon counts.
Other than the method of Reche et al. (Reche-Martinez et al., 2004) (described in Sect. in more detail), methods for capturing real trees include (Tadamura et al., 1992), based on two photographs with emphasis on shading, and Shlyakhter et al. (Shlyakhter et al., 2001) who use a visual hull created from photographs of the tree. They then fit an L-system to generate a polygonal model, while leaves are textured by reprojecting the photographs onto the polygons. As was the case for the synthetic trees mentioned above, the resulting models have high polygon counts; in addition, level-of-detail mechanisms are hard to develop for such representations.
Several image or volume-based rendering methods have been proposed for trees. The multi-layer z-buffer method uses precomputed synthetic images of trees (Max and Ohsaki, 1995; Max, 1996). In volumetric texture approaches, the complex tree geometry is represented as an approximation of the reflectance at a distance (Neyret, 1998). An adaptation of this approach to hardware was developed later using textured slices for interactive rendering (Meyer and Neyret, 1998). Meyer et al. (Meyer et al., 2001) presented a hierarchical bidirectional texture solution for trees at different levels of detail, resulting in efficient level-of-detail rendering for trees. Another approach has been developed in (Qin et al., 2003), in which a volumetric approach effects an implicit level-of-detail mechanism, for lighting (both sun and sky) and shadows, using shadow maps. Efficient rendering of trees can also be achieved using point-based methods (Deussen et al., 2002). More recently billboard clouds (Behrendt et al., 2005; Fuhrmann et al., 2005) have been used for rendering trees. All of the above techniques are applied to polygon-based synthetic trees. As such they could be applied to the captured trees of Shlyakhter et al. (Shlyakhter et al., 2001), but it is unclear how these could be applied to volumetric trees.

Volumetric trees

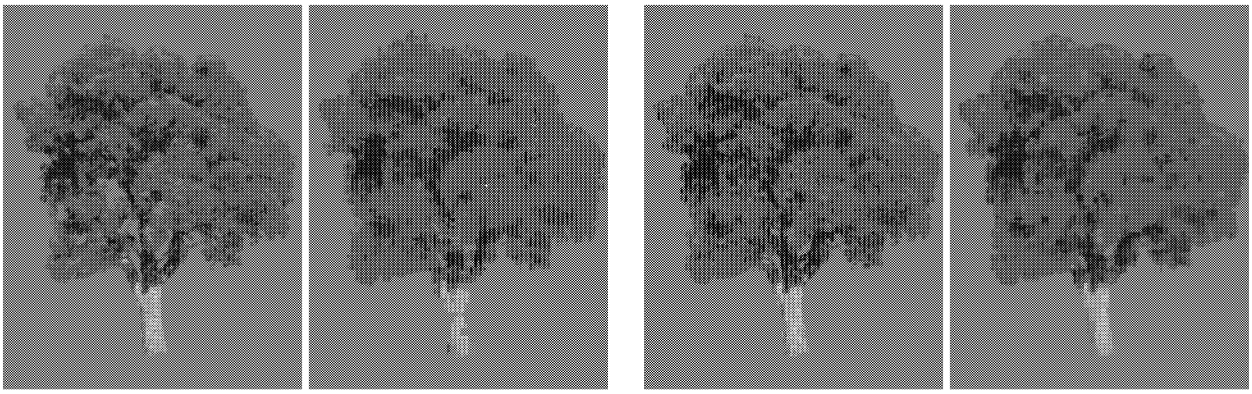
Our rendering and texture compression approach builds on the method of Reche et al. (Reche-Martinez et al., 2004). For clarity, we summarize the method here in more detail.
Tree capture proceeds in three steps. Initially, a set of photographs is taken from around the tree, and the cameras of these images are calibrated. Then, alpha-mattes are extracted from the images, giving an opacity value to each pixel in each view. In a second step, the opacity values are used to perform an opacity estimation on a hierarchical grid, similar to tomography, resulting in the assignment of a density value for each grid cell. The grid used in (Reche-Martinez et al., 2004) was a tri-grid, i.e., each cell is subdivided into 27 children. The degree of refinement of the grid directly influences the quality of the reconstructed volume, where higher refinement allows the reconstruction of finer details. In the final step, textures are generated using a heuristic based on the input images, the depth of the cell in the tree and the alpha/opacity values. The generated textures are then assigned to a billboard in each cell. There is one texture per billboard per input camera position. To render a novel view, the cells are traversed in back-to-front order. The billboards generated from the two closest input cameras are weighted and blended together in the sense of the **over** operator. The two closest cameras are computed once per frame in software.
As mentioned above, despite high-quality tree renderings, this method suffers from high texture memory requirements and the lack of multi-resolution rendering. We address both shortcomings with our new approach.

MULTIRESOLUTION RENDERING

We discuss here two main elements needed to achieve efficient multi-resolution rendering. The first is the choice of the appropriate hierarchical data structure and the generation of the corresponding textures, as well as how to choose the appropriate level of detail. The second is an efficient pruning strategy, based on the properties of the generated textures.

Using an Octree for Multiresolution Rendering

Our goal is to provide a smoothly varying level-of-detail (LOD) mechanism for tree rendering. The volumetric representation is based on a hierarchical data structure. Thus LOD can be achieved naturally by choosing and rendering the appropriate levels of the data structure.
The tri-grid structure used in (Reche-Martinez et al., 2004) is inappropriate for multiresolution rendering since switching from one level to the next involves replacing a single cell (and the corresponding billboards) by 27 sub-cells. This leads to large jumps in the number of primitives, resulting in irregular frame rates. It also produces very visible transition artifacts for the textures which also cannot be avoided by a dissolve in the sense of (Max, 1996). We choose to use an octree; as a result the jumps in number of primitives are not as large as with the tri-grid structure, and the transitions



(a) (b) (c) (d)

Figure 2: Switching from (a) level 4 (51,000 polygons) to (b) level 3 (3,800 polygons) using a tri-grid (average RMS=31.6228). Octree representation with a switch from (c) level 6 (34,000 polygons, pruned 16,000 polygons) to (d) level 5 (6,300 polygons, pruned 3,800 polygons). Note that the transition is less abrupt (average RMS=24.5589).

between different levels of detail are less visible, especially for the lower LODs. Fig. 2(a,b) compares two neighboring levels of the tri-grid hierarchy with neighboring levels of the octree hierarchy (c,d). The artifacts are more clearly visible in the the accompanying video. In the original approach, no provision was made to create billboards and assign textures at *intermediate* nodes of the hierarchy. We use the same texture generation process as in (Reche-Martinez et al., 2004), but at each level of the hierarchy. An alternative would be to average the textures from the lower levels: however, the overhead of texture computation of the intermediate levels corresponds to 37.5% of the total texture generation time. We considered that the tradeoff was worthwhile, since the resulting intermediate level textures are of higher quality.

The selection of the level of detail to be used is based on the distance of each cell to the current camera viewpoint. We set up a fixed number of planes, orthogonal to the camera viewing direction before each rendering pass. During rendering, for each cell of the octree structure, we check whether its center point lies in front of or behind the current LOD selection plane. If it lies in front of the plane, the cell is rendered at the currently active LOD. Else the tree descent stops one level above the currently set LOD, replacing eight cells by their parent cell.

Efficient Pruning Structure

In the original method (Reche-Martinez et al., 2004), rendering speed was hindered by the large number of billboards to be rendered. In addition to the multi-resolution, Sect. , a basic optimization can be performed by better understanding the properties of the textures associated with the billboards attached to each cell. We prune the billboards that do not contribute to the rendered result. For a given cell and a given viewpoint, there is no need to render a billboard if the texture contains no color information. Thus, it can be pruned. After careful study of the generated textures, we realize that this occurs quite frequently using the texture generation heuristic of (Reche-Martinez et al., 2004). During the texture generation process, for each cell we check whether it is visible from a given point of view. We trace a ray through the volume and accumulate the alpha values until we hit the cell. If the accumulated alpha values of the cells hit by the ray exceed a threshold, the cell is essentially invisible from the given viewpoint. The heuristic for texture generation uses the alpha value in its determination of color; as a result, for trees with a dense crown, no colors are assigned to the textures of many of the interior cells or of those cells on the opposite side of the viewpoint. This is clearly illustrated in Fig. 3(a) and (b). In (b) we render a view where we "slice away" the front half of the tree. We can clearly see that most interior cells contain empty textures, and have been pruned. For trees with a sparse crown, Fig. 3(c),(d), this strategy also works, although in a less aggressive manner. The pruning strategy is applied recursively to the entire octree structure.

TEXTURE CCOMPRESSION

The textures generated by the volumetric approach (Reche-Martinez et al., 2004) are of the order of 100-150MB of texture. The packing method reported there did not actually reduce the texture memory required at run time. The reduction reported was for offline storage purposes only.

Since our multiresolution approach makes it necessary to create texture information for the lower levels of detail, more memory may be required, making it even more important to reduce texture memory.

The change from the tri-grid to the octree means that there are typically more levels in the hierarchy, which may result in an increase in the number of billboards. Fortunately, this is partially compensated by the decrease in billboard texture size, from 8x8 (typically used in (Reche-Martinez et al., 2004)) to 4x4. The resulting texture memory is often actually reduced (see Section). Nevertheless, the memory requirements are still too high for the use in computer games or other applications.

(a) (b) (c) (d)

Figure 3: (a) A tree with a dense crown, (b) seen from the side with the foremost half of the volume cut away (camera is to the right). (c) A tree with a sparse crown, (d) again seen from the side with the foremost half of the cells of the volume cut away.

Therefore, we additionally employ an approach for texture compression proposed by Beers et al.(Beers et al., 1996). It is based on vector quantization (VQ) and offers high compression ratios with little loss in quality. We modify their approach to use a perceptually oriented color space such as CIELab for the computation of the texture codebook. We will elaborate on the details in the following subsections.

Texture quantization

The most crucial part of compressing a texture using VQ is designing the codebook. As in (Beers et al., 1996), we employ the Generalized Lloyd Algorithm (GLA), an iterative clustering algorithm which yields a locally optimal codebook for a given set of texture blocks, the training vector set. Our training vector set consists of all 4x4-billboards, encoded as vectors. The algorithm starts with a set of potential codewords from the training set and iterates on the following steps. Each texture block is grouped with the nearest codeword according to a given distance measure. The centroids of the clusters are taken as the new codewords for the next iteration. This process repeats until the set of codewords converges.

The choice of the color space is important at this point. While the use of RGB space with the L_2-norm as a distance measure is simple and fast, GLA often groups textures that are close according to the distance measure but are of different perceptual colours. To counter this problem, we transform the billboard textures to the more perceptually oriented color space CIELab and use the L_2-norm as a distance measure.

The resulting compression has higher overall quality. This can be seen in the comparison in Fig. 4 for the oak tree. In (a) we see the tree with the original uncompressed texture. In (b) we see the compressed texture using RGB space. Clearly, RGB compression results in loss of contrast and overall lower visual quality. In (c) we use the CIELab space. The quality is higher, and contrast is better preserved.

Alpha channel quantization.

The alpha channel quantization is straightforward. We encode each billboard alpha texture into a vector of grey level values and run the generalized Lloyd algorithm (GLA) on this data. Since we quantize the alpha channel independently of the RGB channel, we have to ensure that after quantization, every non-transparent pixel maps to a non-black pixel in the associated color texture. We address this problem by a heuristic that replaces every black pixel in the quantized texture with the color of the brightest pixel, computed from every non-black pixel of the billboard texture. Using the brightest pixel in this heuristic avoids black pixel artifacts in the rendition of the compressed tree.

Color channel quantization.

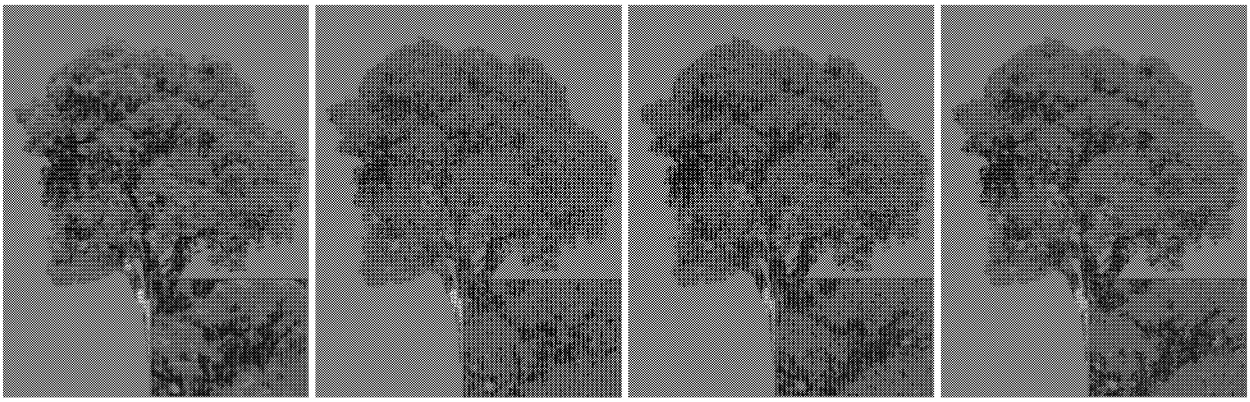
We encode the color channels of each billboard texture into an appropriately sized vector. Afterwards, GLA is run on this vector data. The number of clusters is given by the user and trades compression ratio against quality of the compressed textures.

A drawback of the quantization method of (Beers et al., 1996) is the implicit averaging of colors introduced by the GLA cluster computation, leading to overall darker textures, as well as loss of contrast in the textures. Our solution is to modify the GLA to replace the cluster centroid by the closest original input vector. If we compare Fig. 4(c) to Fig. 4(d), we can see that in the center of the tree certain regions have preserved their bright areas.

To avoid extensive texture context switching, the quantized alpha textures as well as the quantized RGB textures are organized in a texture atlas, typically of size 512x512. Decompression consists of simply computing appropriate texture coordinates in the codebook atlas for each billboard.

RESULTS

All results described here are run on a Linux Fedora PC with an NVIDIA 6800 graphics card with 128MB of texture mem-



(a) (b) (c) (d)

Figure 4: (a) Tree with uncompressed texture. (b) Quantization using RGB space with visible errors (average RMS=30.1971). (c) Quantization with CIELAB space, with original GLA algorithm (average RMS=28.1523). (d) Modified GLA algorithm using CIELAB space (average RMS=28.1017). The RMS is computed with respect to (a) and averaged over the colour channels.

ory, and a 3.06Ghz Xeon CPU. We have tested our approach using three tree models: the oak and pine which were also used in (Reche-Martinez et al., 2004), and the additional small oak model presented in Reche's thesis. We will always refer to these trees in this order unless explicitly stated otherwise. We compare the various improvements to the basic, non-hierarchical tri-grid algorithm of (Reche-Martinez et al., 2004).

Before pruning, the corresponding number of billboards is 154,000, 54,000 and 35,000 for a 7-level octree subdivision. The corresponding numbers for a 5-level tri-grid are 361,000, 152,000 and 114,000. After pruning (Sec.), the *average* number of billboards is 53,000, 43,000 and 33,000, computed by rotating around the tree. In terms of rendering speeds, the average frame rate for the three trees are: 1.5, 2.6 and 2.5 frames per second (fps) for the tri-grid, 4, 6 and 6 fps for the complete octree and 11, 11 and 12 fps for the pruned octree. As we can see, we have an average 78% reduction in the number of polygons and a 86% average speedup in rendering speed, for equivalent quality trees.

The texture memory consumption for a multi-resolution tri-grid version of the above trees is 266, 147 and 228 MB, before any compression (5-level). The corresponding octree texture memory consumption, before any compression is 72, 60 and 86 MB (7-level). Using our compression approach described in Sec. , the memory requirement is 3.1, 2.1 and 6.3 MB. We use a codebook of size 512 for the oak and pine and 1024 for the small pine, which has less contrast overall in the textures. Compared to the uncompressed tri-grid we thus achieve a 64:1 improvement in texture memory consumption (19:1 for the octree).

The performance of the multi-resolution rendering algorithm is harder to compare, since there is no equivalent in the original algorithm. We show in Figs. 1,5 typical interactive “game-like" environments. The “square” scene renders at an average frame rate of 20 fps (max. 24 fps and min. 17 fps), while the scene of the Ancient Greek city of Argos renders

Figure 5: Scene with 12 trees in a square; this scene runs at 20 fps. All three trees are present.

at 10 fps (max. 25 fps and min. 7 fps) for the walkthroughs shown on the accompanying video. On average, 6 trees for the “square” scene and 180 trees for the city of Argos are inside the viewing frustum. All three trees are used. Using the original approach, this would require 641 MB of texture memory, rendering it unusable for common graphics cards. The total texture memory consumption using our approach (with hardware support) is 2.9 MB, or a total compression rate of 240:1 in this case.
Although the improvements in rendering speed and texture memory consumption now allow for rendering hundreds of trees, it is still limited by the traversal of the underlying data structure, needed for correct back-to-front sorting of the billboards. This is done entirely on the CPU and the traversal time hence gives a lower bound for the speed of our method.

CONCLUSION AND DISCUSSION

We have presented a multiresolution rendering approach for captured volumetric trees, together with an efficient texture compression approach. In particular, we use an octree data structure which allows smoother multiresolution level-of-detail control. We employ generated textures at every level. We are able to significantly reduce the number of billboards required to represent the volume by pruning those which do not contribute to the image. For texture compression, we use a vector quantization scheme (Beers et al., 1996) which is modified in two ways: we use a perceptually uniform color space (CIELab), and we modify the GLA algorithm for codebook generation, improving texture quality. The compression does result in a some quality loss (intensity and contrast levels) as evidenced in Fig. 4, but the overall quality is high.. It may be possible to further improve the quality by using alternative heuristics.

The approaches we have introduced, together with the use of a set of graphics hardware optimizations, reduce the texture memory required for the display of the captured volumetric trees by one order of magnitude (e.g., from 72 MB for the oak tree down to 3.1 MB). The multi-resolution algorithm allows the display of large numbers of trees in realistic settings for games or other interactive applications. In the previous approach (Reche-Martinez et al., 2004) a single tree ran at 2 fps; in the examples we have shown, we are able to render environments with 290 trees at 10 fps on average.

The methods we have presented are a significant improvement over previous state of the art for captured volumetric trees. Without these, use of this representation was impractical for all realistic usage scenarios such as games etc. We believe that with the solutions presented here, captured volumetric trees will now be a viable and interesting option for real-world applications such as games etc., since the resulting trees are realistic and convincing, and can be displayed rapidly compared to other approaches.

In future work, we plan to address the two limitations of captured volumetric trees, that is fixed lighting, which is currently embedded in the input photographs used to generate the textures, and the fact that the trees cannot currently be modified, since they are an exact reconstruction of an existing tree. For both of these issues, it will be necessary to create a semantic representation of the tree by identifying the trunk, branches and leaves, allowing their manipulation both photometrically and geometrically. In addition, it would be beneficial to push the back-to-front sorting of the billboards onto the graphics hardware.

REFERENCES

Beers A.C.; Agrawala M.; and Chaddha N., 1996. *Rendering from compressed textures*. In *Proc. SIGGRAPH'96*. ACM Press, New York, NY, USA, 373–378.

Behrendt S.; Colditz C.; Franzke O.; Kopf J.; and Deussen O., 2005. *Realistic real-time rendering of landscapes using billboard clouds*. In *Eurographics 05*. 507–516.

Bloomenthal J., 1985. *Modeling the mighty maple*. In *SIGGRAPH '85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques*. ACM Press, New York, NY, USA, 305–311.

de Reffye P.; Edelin C.; Franson J.; Jaeger M.; and Puech C., 1988. *Plant models faithful to botanical structure and development*. In *Proc. SIGGRAPH 88*. 151–158.

Deussen O.; Colditz C.; Stamminger M.; and Drettakis G., 2002. *Interactive visualization of complex plant ecosystems*. In *Proc. IEEE Visualization 2002*. 219–226.

Deussen O.; Hanrahan P.; Lintermann B.; Měch R.; Pharr M.; and Prusinkiewicz P., 1998. *Realistic Modeling and Rendering of Plant Ecosystems. Proc SIGGRAPH'98*, 275–286.

Fuhrmann A.L.; Umlauf E.; and Mantler S., 2005. *Extreme Model Simplification for Forest Rendering*. In *EG Workshop on Natural Phenomena 05*. 57–66.

Max N., 1996. *Hierarchical rendering of trees from precomputed multi-layer Z-buffers*. In *Proc. 7th EG Rendering Workshop*. 165–174.

Max N. and Ohsaki K., 1995. *Rendering Trees from Precomputed Z-Buffer Views*. In *Proc. 6th EG Workshop on Rendering*. 74–81.

Meyer A. and Neyret F., 1998. *Interactive Volumetric Textures*. In *Proc. 9th EG Rendering Workshop 1998*. 157–168.

Meyer A.; Neyret F.; and Poulin P., 2001. *Interactive Rendering of Trees with Shading and Shadows*. In *Proc. 12th EG Workshop on Rendering, 2001*. 183–196.

Neyret F., 1998. *Modeling Animating and Rendering Complex Scenes using Volumetric Textures. IEEE Trans on Visualization and Computer Graphics*, 4, no. 1, 55–70.

Prusinkiewicz P. and Lindenmayer A., 1990. *The algorithmic beauty of plants. Springer, New York*.

Qin X.; Nakamae E.; Tadamura K.; and Nagai Y., 2003. *Fast Photo-Realistic Rendering of Trees in Daylight*. In *Proc. of Eurographics 03*. 243–252.

Reche-Martinez A.; Martín I.; and Drettakis G., 2004. *Volumetric reconstruction and interactive rendering of trees from photographs. ACM Trans Graph*, 23, no. 3, 720–727.

Shlyakhter I.; Rozenoer M.; Dorsey J.; and Teller S., 2001. *Reconstructing 3D Tree Models from Instrumented Photographs. IEEE CG & A*, 21, no. 3, 53–61.

Tadamura K.; Kaneda K.; Nakamae E.; Kato F.; and Noguchi T., 1992. *A Display Method of Trees by Using Photo Images. Journal of Information Processing*, 15, no. 4, 526–534.

Realistic Rendering of Point Cloud Models with Ambient Shadowing and Environment Lighting

Sushil Bhakar, Feng Liu, Thomas Fevens, Sudhir Mudur

Department of Computer Science and Software Engineering
Concordia University, Montreal, Quebec, H3G 1M8, Canada
sushil@cs.concordia.ca, william7ba@yahoo.com, fevens@cs.concordia.ca mudur@cs.concordia.ca

Abstract: Point clouds acquired using 3D scanners usually need to be converted into mesh models for gaming applications. This involves considerable manual effort. There is however increasing interest in using them directly without conversion. In this paper, we describe the results of our efforts in realistic rendering of point based geometry with real time frame rates. We adapt advanced mesh rendering techniques that help create realistic visual effects. These include self shadowing effects using ambient occlusion, diffuse lighting effects using spherical harmonic representations of irradiance environment maps and specular effects by casting a reflected ray into a pre-blurred environment map. A number of efficiency improvements enable us to handle considerably larger number of sample points in point cloud models – we use an octree hierarchic structure based on feature analysis of the point cloud data to reduce ambient occlusion computations; we push all view independent computations in a preprocessing phase and finally, we perform most of the critical render-time lighting calculations directly on GPU using vertex and fragment shaders.

Keywords: **ambient occlusion**, **point based rendering, environment lighting, GPU programming**

1. Introduction:

Point cloud representations of 3D surface models have been proposed as an alternative to popular triangle mesh representations [16]. Advances in 3D scanner technology have considerably simplified their acquisition. In addition, their representation is much simpler as no connectivity information is required. Yet, most gaming applications usually resort to converting 3D scan data into triangle mesh models involving considerable manual skill and effort. While considerable progress has been made in the development of algorithms for geometric processing and direct rendering of these point-based models, they have yet to be enthusiastically adopted by industries such as gaming and cinema, wherein the concern for high quality realistic images is paramount. An important step for their adoption is to adapt and apply popular techniques for rendering of realistic images of mesh based geometry to produce similar or better rendering results. In this paper, we describe our efforts to adapt mesh rendering techniques of ambient occlusion and environment lighting to achieve realistic rendering of point cloud models with real time frame rates.

There are two main issues that need special consideration when rendering point cloud models. Firstly, rendering techniques must be able to handle the discrete nature of point samples. There is no connectivity information and hence no underlying topology of the surface of the 3D object. Usually, the underlying surface geometry and topology are inferred using neighboring point samples. Secondly, sampling resolutions are usually high to enable capture of fine shape details. The result is that we have a much larger number of points representing the 3D object and any rendering technique must be able to efficiently handle this considerably larger set of point data.

The rest of the paper is organized as follows. In section 2 we provide a brief review of related work on high quality rendering of point based models. Sections 3 and 4 describe adaptation of ambient shadowing and environment lighting techniques respectively. In section 5 we discuss salient details of our implementation, which has been implemented as a plug-in for Pointshop3D [14]. Section 6 presents our conclusions.

2. Related Work:

High quality rendering of point sampled geometry has been the concern of several researchers. Zwicker et al. [17][18]describe a software renderer using elliptic splats for points represented as surfels. A surfel is a surface element represented by a point on the surface, a normal to that surface, an extent for the point given as a radius and other material properties such as color, reflectance, texture coordinates, etc. In a subsequent paper [25], they provided an object space formulation of the EWA (elliptic weighted average) filter and described a hardware implementation using textured polygons. Botsch et al. have described a number of works in which they have tried to improve the performance and rendering capability of EWA renderer. Their improvements include the use of point sprites instead of polygons [26]; this results in a reduced number of vertices being sent to hardware. Other improvements include correction for perspective [15], per pixel shading [27] and deferred splatting [28].

In [14] Zwicker et al. describe a system, Pointshop3D for editing the shape and appearance of point based models. To do this, they gave a formulation for the parameterization of point cloud models. This interactive point cloud parameterization allows distortion free texture mapping. In a follow-up work [21] they enhanced PointShop3D to allow free form deformation and Boolean operations on point cloud models. Pointshop3D is available publicly and supports plug-ins. We too have written our renderer with ambient occlusion and environment lighting as an extension of the native Pointshop3D renderer plug-in, which did not support any of these realistic rendering techniques.

All rendering techniques can be seen as some kind of solution to the rendering equation, introduced by Kajiya in his seminal SIGGRPH 86 paper [29]. The physical basis for the rendering equation is the law of conservation of energy, considering all the light received, emitted and reflected at a particular point. Depending on whether the rendering techniques deal with inter-reflection effects (indirect illumination), these techniques have been classified into two classes -- local illumination and global illumination. A local illumination technique, say using the popular Phong illumination model, requires the computation of two components. There is a Lambertian component, to account for the diffuse reflection behavior, and a specular component to account for surface shininess. It is difficult to produce high quality images with realistic effects using just local illumination due to highly simplified models for lights in the form of distant point or directional sources; they also do not model interreflection effects such as shadows, reflections, refractions, etc. To handle such effects, global illumination techniques such as ray tracing or radiosity are usually employed.

A number of papers have reported efforts in applying ray tracing techniques to point based models to simulate indirect illumination effects. The main problem arises due to the fact that both points and rays are singular geometric entities having no spatial extent. The different methods vary in the way in which this problem is solved and also correspondingly in the acceleration data structures that are employed. Schaufler and Jensen [1] proposed to intersect a ray with a point sampled surface by creating a cylinder around the ray. Adamson and Alexa [2] proposed a number of ray-surface algorithms based on their moving least squares (MLS) implicit surface definition. They initially create a sphere hierarchy and intersect the rays with this hierarchy to find an approximate intersection. Finally they carry out the ray intersection with the MLS surface inside the sphere. In [3] they further improved the efficiency of intersection calculations. Wald and Seidel[4] use a combination of different techniques including an SIMD accelerated intersection code, together with a highly optimized specially built kd-tree data structure. In [5] Adams et al. describe techniques for ray tracing of deformable point sampled surfaces. Once again the emphasis is on clever update of the hierarchical data structure for accommodating the deformation in each frame.

The radiosity method is another powerful and popular graphics method for achieving global illumination effects, particularly for environments with diffuse surfaces. Dobashi et al. [6] describe a very straightforward extension of the standard radiosity technique. For this they consider each surfel as a finite element and calculate inter-reflections among the surfels.

Ray tracing and radiosity based techniques are compute-intensive. In general for large models, it is difficult to obtain real time frame rates using commodity hardware when applying these techniques. As a result, there have been a number of approximation techniques which while not being physically as accurate as the above global illumination techniques still yield realistic effects in rendered images. Ambient occlusion and environment lighting are two such techniques which are quite popular for the high quality rendering effects they make possible with real time frame rates. Below we discuss these techniques, their adaptation to point based models and the efficiency improvements that we have devised taking into consideration the specific characteristics of point cloud representations.

3. Ambient Occlusion:

The ambient occlusion technique [22, 23] tries to attenuate light based on shadowing factors computed for parts of the object. It basically adds shadows to diffuse objects. Ambient occlusion is a crude approximation of the full rendering equation [29]. It takes into account inter-object visibility only. Surfels are shadowed based on whether they are partially occluded to the environment. For this we calculate the accessibility value, which is the percentage of the hemisphere above each surface point that is not occluded by the rest of the geometry [23]. This is done in two passes as described below.

3.1 First pass:

We refer to the surfel that is shadowed as the receiver R and to the surfel that casts the shadow as the emitter E. Amount of shadow that is transferred to R from E is given by an approximate form factor (*cf* Fig. 1) [32]:

$$\frac{A\cos\theta_e\cos\theta_r}{\pi r^2 + A}$$

To calculate how much in shadow is receiver R, we add the form factors treating all other surfels as emitters. We also clamp the value to be less than 1. Therefore initial shadow value from first pass is given by: $T_i = \max(1, \sum_{j \neq i} \frac{A_j \cos\theta_e \cos\theta_r}{\pi r^2 + A_j})$

Since every point (surfel) can be potentially shadowed by every other point, the above is an order N^2 operation. For large point models, this can be very time consuming, even if this is a view independent computation and can be done in a preprocessing phase. For speeding up the computations, we use an octree hierarchy of surfels. The octree construction is based on feature analysis of points within a parent node [20], done based on Eigen value computations. Each node of the octree is labeled as being flat, or as containing an edge or corner. A GPU based hierarchical ambient occlusion computation technique has also been presented in [24] for mesh models. However our work offers an improvement in the manner in which it makes beneficial use of feature analysis as described below.

To compute the ambient occlusion (shadow) factor for each surfel, we consider only those surfels which lie within a cone with geometry defined as follows:

- The cone axis is along the normal associated with the surfel,
- The cone apex is at the 3D point associated with the receiver
- The cone angle has been empirically chosen to be 150 degrees. This angle works for most models. Surface parts that are nearly co-planar contribute very little to this occlusion. A smaller cone angle results in rejecting too many surfels. On the other hand a larger cone angle results in less efficient culling.

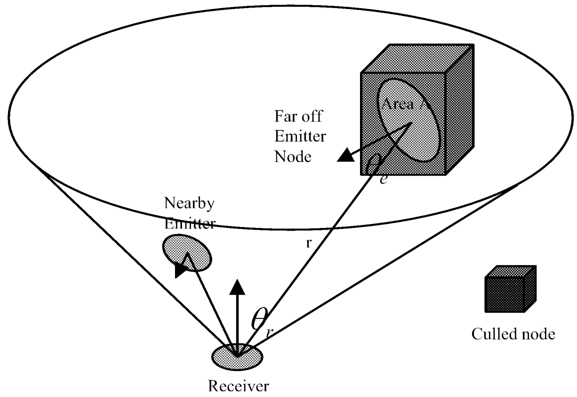


Figure 1: Illustration for Ambient Occlusion

Fast culling of surfels is made possible by top-down traversal of the octree and retaining only those octree nodes lying inside the cone. To further speed up the calculations, distant and nearly flat nodes of octree are treated as single surfel. We have therefore two cases in our form-factor computation:

Form factor for a distant flat node: All the surfels in a given node are merged to form a surfel at the centre of the node. Area of this new surfel is equal to sum of areas of all surfels in node. Since the node is locally planar, we use the normal of any surfel as normal of this new surfel. This way we approximate the computation of form factors for surfels in a flat node.

Form factor for nearby or non-flat node: Form factors are computed for every surfel in node and added up to give the form factor for that node.

3.2 Second pass:

Surfels tend to be too dark after first pass. This is due to ignoring the fact that an emitter which is itself in shadow should not contribute fully again to the shadow factor. We correct this by using the shadow values obtained in first pass in a second pass as follows:.

Final shadow factor

$$S_i = \max(1, \sum_{j \neq i} (1 - T_j) \frac{A_j \cos\theta_e \cos\theta_r}{\pi r^2 + A_j})$$

All ambient occlusion computations are carried out in a preprocessing phase. The results from the occlusion computations are used to modulate the color of every point and then stored with the model data, thus causing no increase in file size. Fig. 2 shows an example to illustrate the results from an implementation of the above technique. For comparison purposes, we have implemented both brute force and feature-analysis octree based ambient occlusion computations. It can be seen that the picture quality in Fig. 2(c) (octree-method) and Fig. 2(d) (brute-force method) is almost the same. But since we have coupled feature analysis with octree hierarchical structure, our method is significantly faster (30 times for this model of about 87,000 points). Depending on the size and complexity of the 3D object's surface, our experiments have shown that we are able to get computation speed-ups by a factor of 25 or more for the models we have worked with.

4. Environment Lighting:

In most real situations lighting is quite complex, as it comes from a variety of sources, such as area light sources, bright reflecting surfaces, skylight, etc. One approach to modeling of more complex lighting is the method of environment maps. An environment map is the incident light at a point in all directions, usually modeled as a spherical light field surrounding the point.

Blinn and Newell were the first to propose environment maps to model perfect mirror like reflection of a small surface illuminated by distant spherical illumination [12]. In more recent work, Wolfgang et al. [9, 10] do prefiltering and determine view independent environment maps. They also applied a physically more accurate model than Phong illumination model for local lighting. For reflections they use their view independent maps. Kautz et al. [11] extended their technique to handle a more general reflectance behavior.

In our work, given an environment (spherical) map of incident light, we compute both diffuse and specular lighting components for each point as described below.

4.1 Diffuse Reflectance with Environment Lighting:

Ramamoorthi et al. [7, 8] have shown that for diffuse lighting, irradiance can be calculated analytically in terms of spherical harmonics. Further, diffuse lighting can be well approximated using only 9 low frequency coefficients. They have very effectively demonstrated the results of their method as applied to mesh representations. Spherical harmonic coefficients [13] are similar to Fourier basis coefficients but defined over a sphere. An excellent tutorial on spherical harmonics lighting can be found in [30].

For the benefit of the readers, we repeat the relevant parts of the formulation given in [7] below. After ignoring shadows and near-field illumination, the irradiance E is a function of surface normal N only and is given by an integral over the upper hemisphere represented using spherical harmonics:

$$E(\theta,\phi) = \sum_{l,m} A_l L_{lm} Y_{lm}(\theta,\phi) \qquad (1)$$

Where Y_{lm} denote spherical harmonic coefficients, L_{lm} denote the spherical harmonic coefficients of the incident light in their expansion, and A_l denotes the dot product of the normal and the direction vector for which analytical formulae have been provided [8].

We can calculate L_{lm} in a preprocessing operation:

$$L_{lm} = \int_{\theta=0}^{\pi} \int_{\phi=0}^{2\pi} L(\theta,\phi)\, Y_{lm}(\theta,\phi)\, \sin\theta \, d\theta d\phi \qquad (2)$$

As shown in [7], just 9 coefficients ($l \leq 2$) are sufficient for a close approximation with average error less than 1%. Also, due to the orthonormal property of spherical harmonics, the above computation is further simplified into a dot product calculation.

A straightforward application of the above method to point sampled surface representations was first presented in [33]. In the present work, we have extended it by using Gaussian blurring of the environment map for glossy reflectance behavior. Further, we have also taken into account self shadowing using ambient occlusion in a computationally efficient manner.

4.2 Specular Reflectance with Environment Lighting:

In order to simulate glossy reflection, rather than perfect mirror reflection, we use a Gaussian blurred version of the environment map. Fortunately, blurring affects the high frequency components more and the low frequency components much less. Hence the 9 low frequency spherical harmonic coefficients do not change significantly from the original environment map. This means that we do not need two different versions of the environment map and we are able to use just the blurred version. Let E denote the eye vector (eye to surface point), L the light source vector, N the surface normal and H the half-vector ([L + E]/2). We calculate Fresnel term F [19] to determine how much light is reflected, assuming that the rest is absorbed by the object,

$$F = \frac{(g-c)^2}{(g+c)^2}\left[1 + \frac{(c(g+c)-1)^2}{(c(g-c)+1)^2}\right],$$

where $c = (E \bullet H)$, $g = \sqrt{n^2 + c^2 - 1}$ and n is the index of refraction. We also compute a value for the distribution term D given by:

$$D = \left[\frac{c_3{}^2}{\cos^2 \alpha (c_3{}^2 - 1) + 1}\right]^2$$

where cos $\alpha = N \bullet H$, and c_3 denotes the shininess of the model (0 for perfect reflection and 1 for total diffuse reflection).

If Ks is the specular reflectance coefficient and L_{in} is the incoming radiance in reflected direction, then the specular component of reflected light is given by *Ks*L_{in}*F*D.*

Figures 3 and 4 show two point models rendered using environment lighting under different lighting conditions and material property settings (without using ambient occlusion calculations.) We are able to render all these models at interactive frames rates varying between 25 and 50 frames a second. It is clear that our modified point renderer incorporates this popular technique very well.

5. Implementation Details

5.1 Preprocessing Computations:

As mentioned earlier, for optimal performance we have carefully partitioned the computations, so that all view independent computations are carried out in a preprocessing. Further as much of the render-time computations as possible have been pushed into the GPU as briefly described below.

In the preprocessing stage we do the following:

- The surfels are organized into an octree with a feature value (flatness measure) computed for each octree node [20]. For computing the flatness, we use the well established Eigen value analysis technique [31].
- The shadow factor is computed for each point by optimally making use of the octree hierarchy and flatness measure as described above. The color associated with each surfel in the data is suitably modulated using this shadow factor.
- The 9 spherical harmonic coefficients of Eq. 2 are computed for the given environment light map. These calculations are done separately for each of RGB channels and stored in an array.
- A blurred version of the environment map is created and loaded as a texture map in the GPU.

5.2 Render-time Computations:

The 9 spherical harmonic coefficients are received by the vertex shader which then computes the diffuse lighting component, vDiffuse, for each point as given in [7]. To further improve implementation efficiency in the GPU, we have reused a clever technique introduced in [34], which uses fewer GPU registers.

The GPU implementation of the specular lighting component is distributed between the vertex shader and the fragment shader programs. First in vertex shader, we calculate mirror reflection direction based on camera position and normal. Using this reflected vector, we pass corresponding 2D texture coordinate to fragment shader. Secondly, in the vertex shader, we calculate the Fresnel and distribution terms and output them to the fragment shader.

In the fragment shader, based on the 2D texture coordinate lookup, we get the color vSpecular. We then compose diffuse and specular lighting components. For example, if diffuse reflectance coefficient is Kd, specular reflectance coefficient is Ks and shadow factor is S_i, then what we get as the final color is given by the following equation:

$$Final\_Color = S_i * (Kd * vDiffuse + Ks * vSpecular * F * D);$$

Figure 5 shows that using ambient occlusion further improves the quality of images produced using environment lighting. This can be clearly noticed by looking at the areas around lip, nose and ears.

6. Conclusions:

In this paper we have demonstrated that high quality mesh rendering techniques can be efficiently adapted for point based models yielding equally good quality images and with real time rendering frame rates. We have presented an efficient method for ambient occlusion computation using a hierarchic structure and a flatness measure based on feature analysis using Eigen value computations on the point cloud data. We also combine ambient occlusion with environment map based lighting supporting both diffuse and specular reflectance behavior. Rendering computation efficiency is improved by partitioning computations so that view independent calculations are done in a preprocessing stage and render-time computations are carried out by the GPU using vertex and fragment shaders. These shaders get ambient occlusion results directly embedded in color channel, thus avoiding any data overhead due to ambient occlusion information. Based on tight integration and adaptation of these techniques, our renderer implemented as a plug-in for Pointshop3D produces final picture quality and frame rates which compare very well with results from mesh based renderers.

8. Acknowledgement:

We thank Paul Debevec for providing light probes. (Available from http://www.debevec.org). We gratefully acknowledge for 3D models Stanford 3D Scanning Repository, Mannequin model by Hugues Hoppe and Gnome model from Pointshop3D site. We also thank PointShop3D community for providing such an excellent tool for experimentation.

References:

[1] G. Schaufler and H. Jensen. Ray tracing point sampled geometry. *Rendering Techniques 2000*. Eds. Peroche and Rushmeier. Springer-Verlag, pages 319-328, 2000

[2] A. Adamson and M. Alexa. Ray tracing point set surfaces. Proceedings of Shape Modeling International 2003, pages 272-279.

[3] A. Adamson, M. Alexa and Andrew Nealen. Adaptive sampling of intersectable models exploiting image and object-space coherence. SI3D 2005. pages 171-178

[4] I. Wald and H. Seidel. Interactive ray tracing of point based models. Proceedings of 2005 symposium on Point Based Graphics

[5] B. Adams, R. Keiser, M. Pauly, L. Guibas, M. Gross, and P. Dutre. Efficient Raytracing of deforming point-sampled surfaces. Proceedings of the 2005 Eurographics conference

[6] Y. Dobashi, T. Yamamoto and T. Nishita, T. Radiosity for point-sampled geometry: 12th Pacific Conference on Computer Graphics and Applications 2004.

[7] R. Ramamoothi and P. Hanrahan. An efficient representation for irradiance environment maps. SIGGRAPH 2001

[8] R. Ramamoorthi and P. Hanrahan. The relationship between radiance and irradiance: Determining the illumination from images of a convex lambertian object. Journal of the Optical Society of America, 2001

[9] W. Heidrich and H. Seidel. View-independent environment Proceedings of the 1998 workshop on Graphics hardware

[10] W. Heidrich, and H. Seidel Realistic, Hardware-accelerated Shading and Lighting. SIGGRAPH 1999

[11] J. Kautz and M. McCool. Approximation of Glossy Reflection with Prefiltered Environment Maps. Graphics Interface 2000

[12] J. Blinn and M. Newell. Texture and reflection in computer generated images. Communications of the ACM 1976

[13] T. MacRobert. Spherical harmonics: an elementary treatise on harmonic functions, with applications. Dover publications 1948

[14] M. Zwicker, M. Pauly, O. Knoll, and M. Gross. Pointshop 3D: An interactive system for Point-based surface editing SIGGRAPH 2002

[15] M. Zwicker, J. Räsänen, M. Botsch, C. Dachsbacher, M. Pauly. Perspective accurate splatting. Graphics Interface 2004

[16] S. Rusinkiewicz, M. Levoy., "QSplat: A Multiresolution Point Rendering System for Large Meshes",*SIGGRAPH, 2000.*

[17] H. Pfister, M. Zwicker, J. van Baar, and M. Gross. Surfels: Surface elements as rendering primitives. SIGGRAPH 2000

[18] M. Zwicker, H. Pfister, J. van Baar, and M. Gross. Surface splatting. SIGGRAPH 2001

[19] R. Cook and K. Torrance. A reflectance model for computer graphics ACM Transactions on Graphics 1982.

[20] S. Bhakar, L. Luo, and S. Mudur. View Dependent Stochastic Sampling for Efficient Rendering of Point Sampled Surfaces. Journal of WSCG 2004.

[21] M. Pauly, R. Keiser, L. Kobbelt, M. Gross. Shape Modeling with Point-Sampled Geometry ACM SIGGRAPH 2003

[22] S. Zhukov, A. Iones, and G. Kronin. An Ambient Light Illumination Model. Rendering Techniques 1998

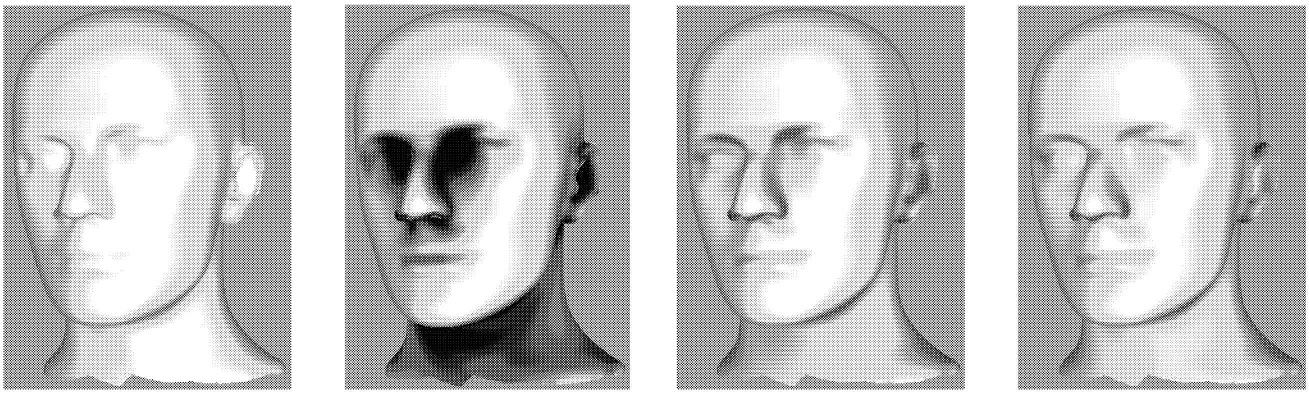
[23] H. Landis. Production-Ready Global Illumination, "RenderMan in Production" SIGGRAPH Course Notes 2002

[24] M. Pharr. Ambient occlusion. In GPU Gems, R. Fernando, Ed., 667-692 Year 2004

[25] L. Ren, H. Pfister and M. Zwicker. Object Space EWA Surface Splatting. A Hardware Accelerated Approach to High Quality Point Rendering. EUROGRAPHICS 2002

[26] M. Botsch and L. Kobbelt. High-Quality Point-Based Rendering on Modern GPUs. Pacific Graphics 2003

[27] M. Botsch, M. Spernat, L. Kobbelt. Phong Splatting. Eurographics Symposium on Point-Based Graphics 2004
[28] M. Botsch, A. Hornung, M. Zwicker and L. Kobbelt High-Quality Surface Splatting on Today's GPUs. Eurographics Symposium on Point-Based Graphics 2005, pp. 17-24.
[29] J. Kajiya The rendering equation SIGGRAPH 1986
[30] R. Green. Spherical Harmonic Lighting: The Gritty Details GDC 2003 Tutorial notes
[31] S. Gumhold, X. Wang and R. McLeod. Feature Extraction from Point Clouds. Proc. 10th Int. Meshing Roundtable, 2001
[32] M. Bunnell Dynamic Ambient Occlusion and Indirect Lighting GPU Gems 2
[33] F. Liu, S. Bhakar, T. Fevens, S. Mudur Environment Lighting for Point Sampled Geometry CGIV 06
[34] P. Pike and J. Sloan. Efficient Evaluation of Irradiance Environment Maps. In Book ShaderX2, shader programming tips & tricks with DirectX 9



(a) **(b)** **(c)** **(d)**

Figure 2: (a) Image without ambient occlusion, (b) after first pass, (c) final result from our method, (d) and brute force result which is almost same as ours.

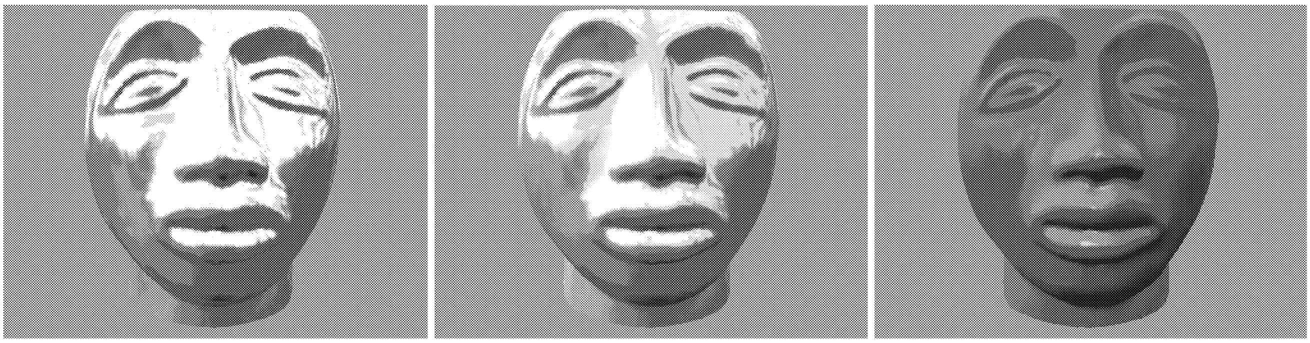


Figure 3: Images rendered with decreasing glossiness.

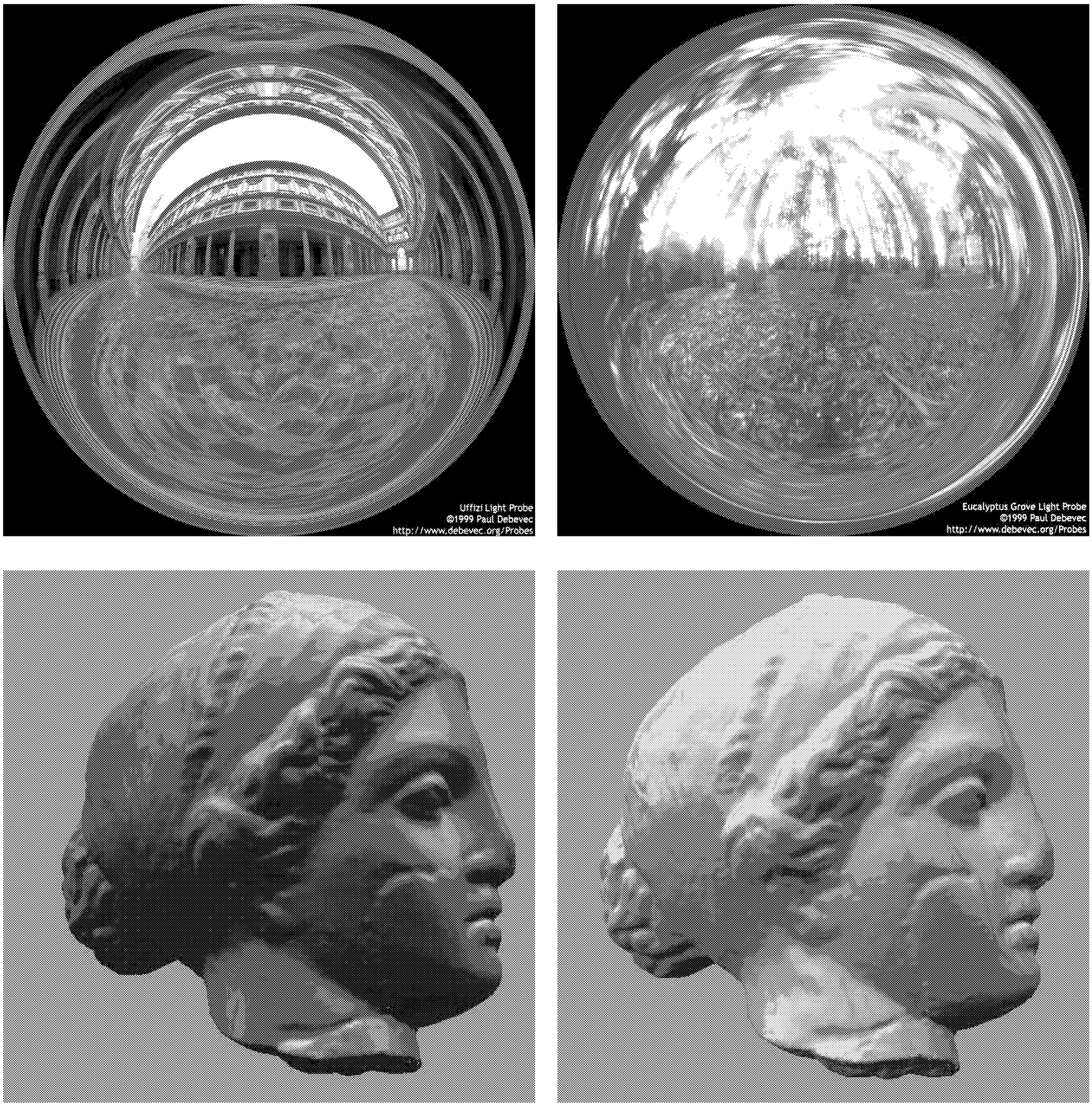


Figure 4: Model rendered with two different environment maps.

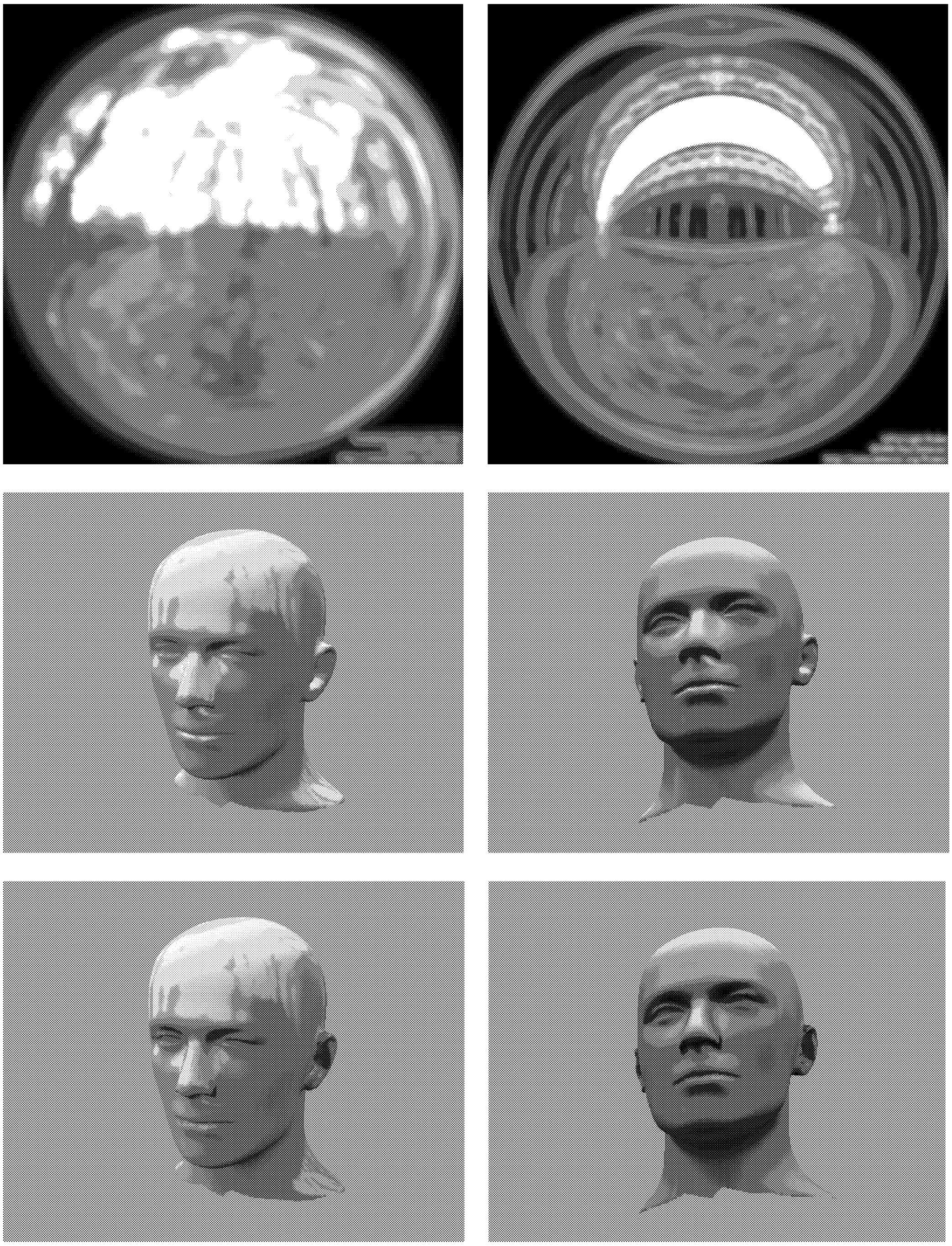


Figure 5: Improved environment lighting by using ambient occlusion. Notice areas around ear, lips and nose. Top row gives environment map used for rendering. Second row gives rendering results without ambient occlusion. Last row shows results with ambient occlusion.

Interactive learning interface for automatic 3D scene generation

Yoshiaki Akazawa[1, 2], Yoshihiro Okada[1] and Koichi Niijima[1]
[1]Graduate School of Information Science and Electrical Engineering
Kyushu University
[2]Reseach Fellow of the Japan Society for the Promotion of Science
744 Motooka, nishi-ku, Fukuoka, 819-0395 JAPAN
{y-aka, okada, niijima}@i.kyushu-u.ac.jp

Keywords: Automatic placement, Semantic database, 3D scene construction, Virtual reality, 3D CG

Abstract
This paper treats an interactive learning interface used to obtain semantic constraints among 3D objects from existing 3D scenes for automatic 3D scene generation. The layout work for 3D scene generation takes a long time because 3D objects have six degrees of freedom (DOF) and are difficult to be positioned by using a standard 2D input device, e.g., a mouse device. To deal with this problem, the authors have already proposed an automatic 3D object layout system based on contact constraints. However, when there are too many kinds of 3D objects to be laid out, it is practically impossible for the user to define semantic constraints among all of them. In this paper, the authors propose an interactive learning functionality and its interface to obtain semantic constraints among 3D objects from already existing 3D object scenes. Using this functionality, the user can generate desirable 3D scenes more easily. This paper presents the learning process to extract semantic constraints from already existing 3D scenes, and delineates its usefulness by showing experimental results.

INTRODUCTION

The layout of 3D objects is inevitable task for developing 3D graphics applications. However, it is not easy task because 3D objects have six degrees of freedom (DOF) and are difficult to be positioned by using a 2D input device on a standard 2D computer display screen. In addition, the 3D layout must satisfy some physical and functional constraints. As a result, the 3D scene generation is time consuming and tedious task. To deal with this problem, we have already proposed an automatic 3D object layout system [1]. Our proposed system automatically generates 3D scenes by the random layout method of 3D objects in keeping with their physical and functional constraints represented as semantic database records. Further more, we have extended our semantic database to support group layout and regular interval layout methods [2]. Although the semantic database is very simple, our extended system can generate realistic 3D scenes. The system needs only a few seconds to generate a 3D scene even if there are many 3D objects in the scene. Our prototype system has to request the user to define semantic database records for all the 3D objects manually. To define such semantics becomes time-consuming task if there are too many kinds of 3D objects to be laid out. So, this paper proposes an interactive learning interface used to obtain semantic constraints among 3D objects from existing 3D scenes. Using this interface, the user can generate desirable 3D scenes more easily.

Many researches on 3D object layout have been made so far. Smith et al. [5] proposed the manipulation of 3D objects using a 2D user interface. This system employs contact constraints among 3D objects to allow the user to position 3D objects using a mouse-device. Bukowski and Sequin [3] also employed a pseudo-physical approach to position objects in a 3D scene using a 2D device. This method maps the 2D mouse motion into vertical and horizontal transformations for determining the position of an object in the scene. However, when there are many 3D objects in a scene, it still takes a long time to lay out them. Coyne and Sproat [4] proposed the WordsEye system which generates 3D scenes composed of 3D objects according to the text description. This method allows a few kinds of 3D scenes to be generated, based on a few lines of text. However, when laying out many various types of 3D objects, the text description approach is not enough for their accurate layouts. Zeng el al. [7] also proposed natural language approach for 3D scene construction. For complex scenes, their system requests the user to write many sentences. Xu et al. [6] introduced an automatic placement system of 3D objects through user interaction. This system drastically reduces time consuming in 3D scene generation. However the system lays out 3D objects only on the floor of a room. It neither consider layout on a ceiling nor on a wall. In contrast, our system considers layouts of 3D objects on the ceiling, as well as the floor by the same framework. Moreover our system can generate 3D scenes including group layouts, in which some objects of the same type are orderly placed close to each other as a group, and regular interval layouts, in which some objects of the same type are orderly placed in keeping with the specific constant interval distance among each other. Most of natural language approaches request the user to define semantic constraints among 3D objects by the text description. If the user wants to generate realistic 3D scenes, he/she has to write complicated text description and needs a lot of time. On the other hand, our interface proposed in this paper overcomes this problem by using the interactive learning. Moreover, our interface can generate more preferable 3D scenes using weight parameters as for semantic constraints among 3D objects those statistically calculated from existing sample 3D scenes. This is the new idea that we propose in this paper.

The remainder of this paper is organized as follows. First of all, we describe placement constraints specified as records of the semantic database. Next, we explain a learning process in detail. After that, we introduce our prototype system and its experimental results. Finally, we conclude the paper.

SEMANTICS FOR AUTOMATIC 3D OBJECT PLACEMENT

Our learning interface proposed in this paper is the new part of our automatic 3D object layout system [1]. This interface provides the functionality to update semantic database records interactively from 3D scenes already generated by the layout system.

Table 1: Semantic database

ot4 (bookshelf)			
Face	Occupancy distance	Parent	Contact constraint
1	1.0		
2	0		
3	0		
4	0	ot1-3	X
5	0		
6	0		ot2-1
ot5 (TV)			
Face	Occupancy distance	Parent	Contact constraint
1	1.0		
2	0		
3	0		
4	0	ot4-3, ot6-3	X
5	0		
6	0		

OT(Object type) No.- Face No.:
{ot1(floor), ot2(wall), ot3(ceiling),,, ot6(desk),...}

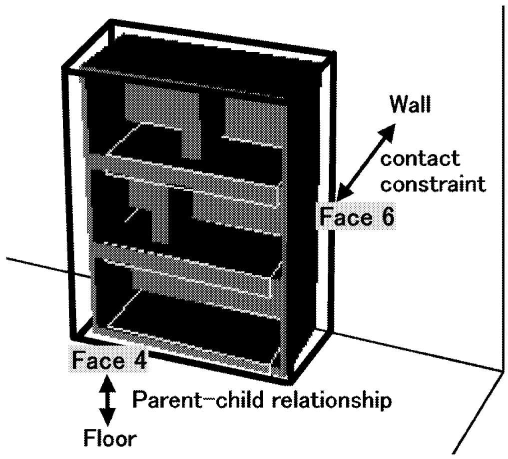


Figure 1: Parent-child relationship and contact constraint of a bookshelf

Semantic database

To define the layout of 3D objects, our 3D object layout system uses a semantic database whose each record called "object info" means placement constraints of the corresponding object class. Among the placement constraints are occupancy distance, parent-child relationship and contact constraints. Table 1 shows two records of the semantic database, i.e., for the bookshelf and the TV. In the following subsections, we describe such placement constraints.

Bounding box

In the real world, every object exists without any collisions. When laying out 3D objects, we have to detect collisions for every object. Detecting accurately collisions for all 3D objects in a scene takes a very long time because each object has its own complex shape. One of the solutions is to simplify such a complex shape. However, even if the 3D shape of each object is simplified, it is impossible to calculate collisions among many 3D objects in real time. To maximally simplify the layout process, we decided to employ the bounding box of each 3D object instead of its original 3D shape as shown in Figure 1. In the real world, every object is in contact with any other object due to the gravity, e.g., a desk rests on a floor and a painting is hung on a wall. While detecting collisions of one object using its bounding box, it is enough to consider only the surface on which the corresponding object lies. In this way, by using bounding boxes, the calculation cost of detecting collisions is drastically reduced. Our prototype system can generate a 3D scene consisting of many 3D objects in an acceptable calculation time.

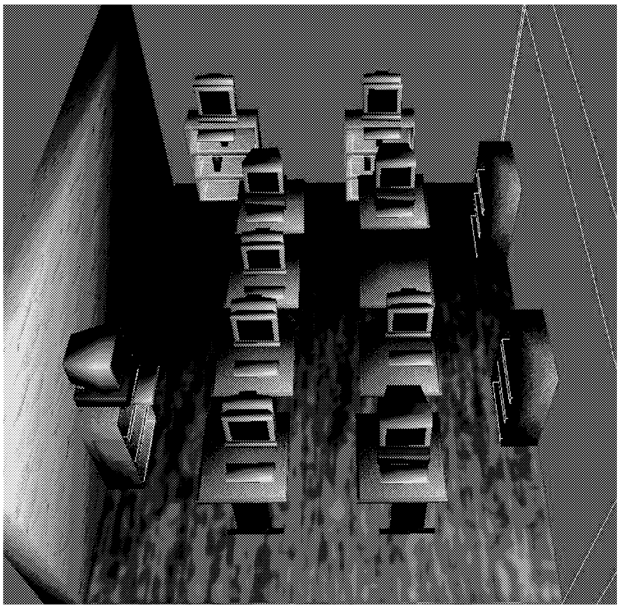


Figure 2: an example of ten TV objects placement

Parent-child relationship and contact constraints

As previously explained, every object in the real world has to touch other objects because of the gravity. At least, one of the faces of a bounding box is defined to be a contact face. For example, for the bookshelf, face 4 of its bounding box must touch the face of the floor as shown in Figure 1. If the user moves the floor object, the bookshelf should move with it. Usually this is treated as a parent-child relationship in 3D applications. Every 3D object has information indicating which of the other 3D objects is allowed to be its parent, and indicating which face of that 3D object and which face of its parent object touch each other. In the semantic database, this information is specified as the parent attribute of the face which is in contact with the parent object. In addition to the above information, we have to specify contact constraints for each face of a 3D object. This indicates whether a face should touch a certain face of another object or not. For example, the bookshelf often touches a wall in addition to the floor. That is, face 6 of the bookshelf in Figure 1 has to touch the wall object. We describe this constraint in the constraint attribute of each face of each record in the semantic database. Using the parent-child relationship and the contact constraint explained above, although these constraints are very simple, the system can generate natural 3D scenes by laying out 3D objects randomly in keeping with those constraints. See the paper [1, 2] for its detail.

INTERACTIVE LEARNING PROCESS

In this section, we introduce an interactive learning process to update the semantic database of 3D objects.

Extension of semantic database for interactive learning

Table 2: extended semantic database

ot4 (bookshelf)			
Face	Occupancy distance	Parent	Contact constraint
1	1.0		
2	0		
3	0		
4	0	ot1-3 (5/5)	X
5	0		
6	0		ot2-1
ot5 (TV)			
Face	Occupancy distance	Parent	Contact constraint
1	1.0		
2	0		
3	0		
4	0	ot4-3 (3/10), ot6-3 (7/10)	X
5	0		
6	0		

OT(Object type) No.- Face No.:
{ot1(floor), ot2(wall), ot3(ceiling),,, ot6(desk),...}

In the real world, strengths of parent-child relationship constraints are not the same. For example, as shown in Figure 2, in the case of a TV class, the probability that bookshelves are chosen as parents of TVs is 3/10. And, the probability that desks are chosen as parents of TVs is 7/10. However, our semantic database records do not have such information so that our layout system chooses a parent-child relationship of a 3D object class by equal probability. To generate more preferable 3D scenes, we added a weight parameter as the strength of a parent-child relationship into each semantic database records as shown in Table 2. The weight parameter means a probability that a parent of an object is chosen. After counting the number of parent samples of 3D object belonging to each class, each weight parameter is statistically calculated from existing sample 3D scenes using the following equation (1):

$$P_i^{OT} = \frac{S_i^{OT}}{\sum_{j=1}^{n} S_j^{OT}} \qquad (1)$$

where both OT and i is the index number of an Object type, n is the total number of Object types, and S_j^{OT} is the total number of j-type objects which are used as the parent of any OT object. So, P_i^{OT}

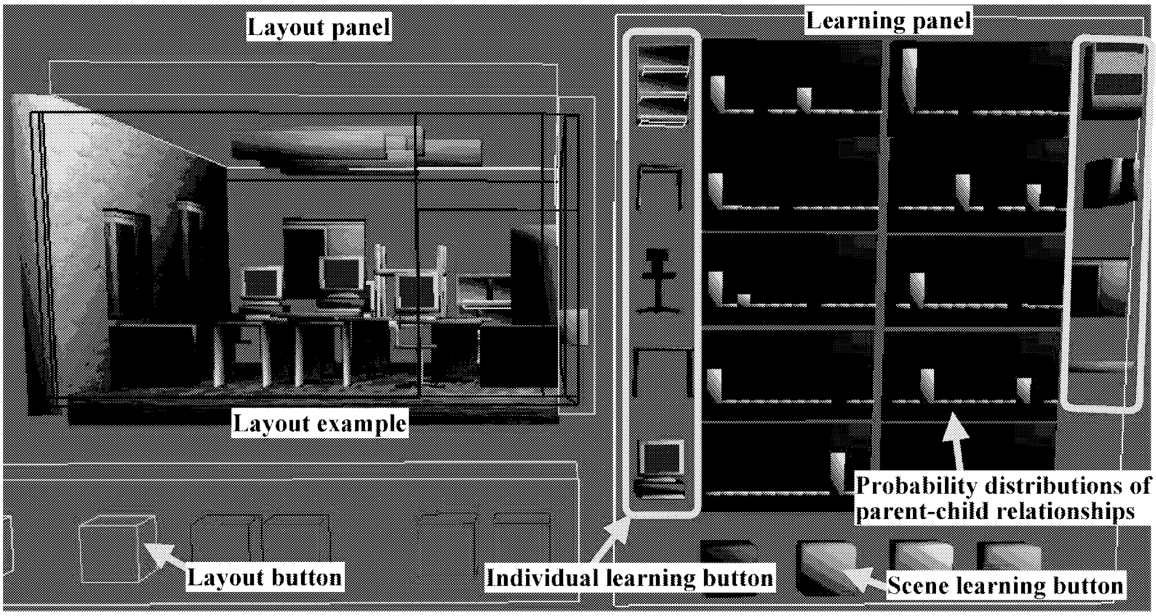


Figure 3: An overview of our interactive learning interface

means the probability that i-type objects are chosen as the parent of any OT object.

By introducing new parameter $P^{OT} = \{P_1^{OT}, P_2^{OT}, \Lambda\, P_n^{OT}\}$, i.e., weights of parent-child relationships between an OT object and the other object, into the semantic database and using them, the system can newly generate more realistic 3D scenes preserving features of already existing 3D scenes.

The interface for interactive learning

Since it is not easy to define weights of parent-child relationships for all of the 3D objects, our system has to provide any functionality that helps the user to calculate such weights and update semantic database records interactively. So, we introduced such an interactive interface, a learning interface, into our automatic 3D object layout system. This interface consists of an individual learning functionality and a scene learning functionality. The individual learning functionality is for defining parent-child relationships of an individual 3D object class in a 3D scene. The scene learning functionality is for defining parent-child relationships of all the existing 3D object classes in a 3D scene.

First of all, we show an overview of the interactive learning interface using Figure 3. Since our automatic 3D object layout system was developed using *IntelligentBox*, the prototype of the interactive learning interface is also realized as some composite components of *IntelligentBox*, which is a constructive visual 3D software development system [8, 9]. This interface mainly consists of a layout panel and a learning panel. The layout panel controls a 3D scene generated using our automatic 3D object layout functionality. If the user pushes the layout button, the system generates a 3D scene according to current semantic database records automatically. The system can also load any 3D scene from scene files. The learning panel is used to update semantic database of 3D objects. This panel changes weight parameters of 3D objects by extracting parent-child relationships from a 3D scene the layout panel provides. This panel has two types of buttons, i.e., individual learning buttons and a scene learning button. If the user pushes one of the individual learning buttons, the system changes the weight parameter of the Object type corresponding to the pushed button. Each chart shown in Figure 3 displays the probability distribution of the weight parameters of a certain 3D object class. The interface puts sets of the probability distribution sequentially according to object types. On the other hand, if the user pushes the scene learning button, the system changes weight parameters about all of the 3D objects existing in the current 3D scene.

In this way, our proposed interface provides two types of learning functionalities. Next, we explain these functionalities in detail. Moreover, we discuss experimental results.

Individual learning functionality

The individual learning functionality extracts parent-child relationships of one individual object class out of a 3D scene. By using this functionality, the user can interactively change weight parameters of child-parent relationships of his/her specified object class.

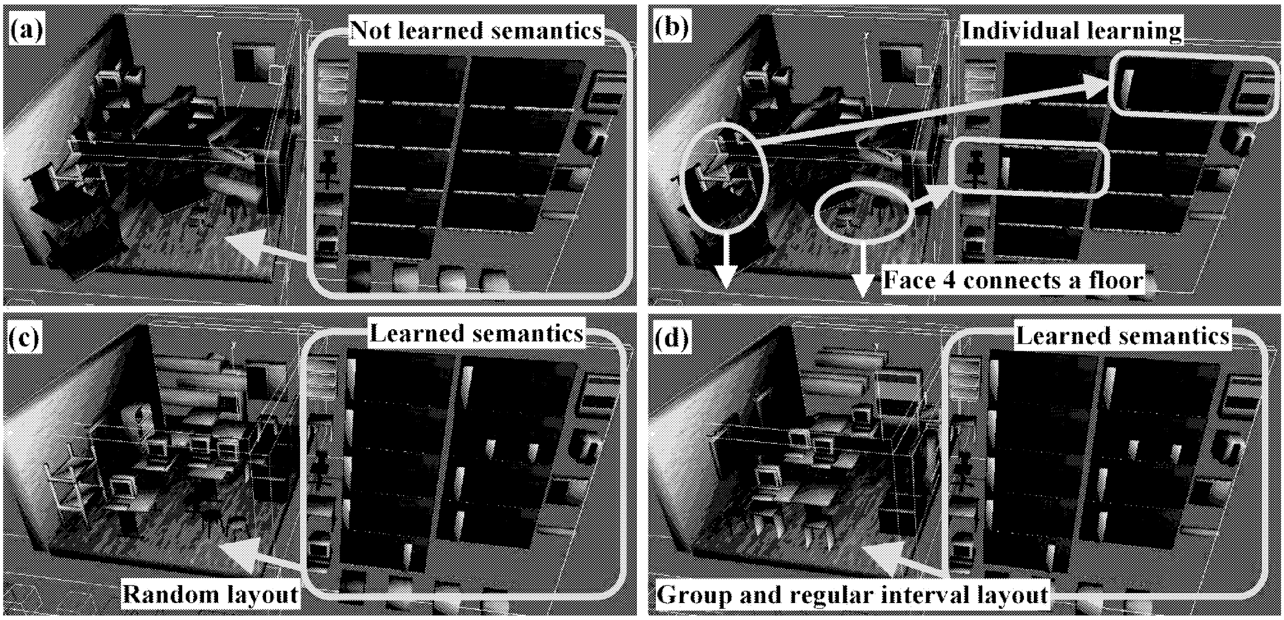


Figure 4: Experiment of the individual learning functionality

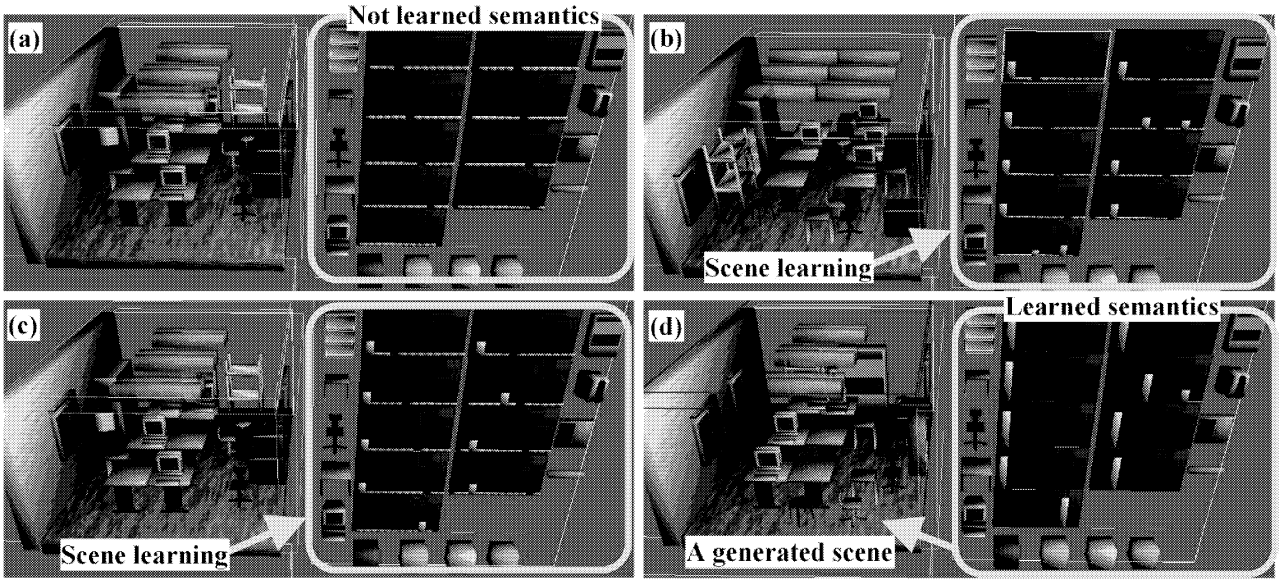


Figure 5: Experiment of the scene learning functionality

Experiment of the individual learning functionality

We introduce an experimental result about the individual learning functionality. At first, our system generates an initial semantic database which has parent-child relationships of all 3D objects that means any objects can be connected to each other. Strictly speaking, weights of parent-child relationships among all 3D objects are the same value as shown in Figure 4 (a). Next, the system lays out 3D objects at random using the semantic database. Since the semantic database initially does not have any user preference information, the generated 3D scene is not guaranteed to be desirable for the user. If the user finds his/her preferable parent-child relationship between any two 3D objects in the 3D scene, for instance, chairs and shelves are correctly laid out on the floor as shown in Figure 4 (b), the user can update weights of parent-child relationships in semantic database records about the chair class and the shelf class using the individual learning functionality by only choosing the chair object and the shelf object. Generally speaking, by counting the number of the same parent-child relationships between any two object types in a 3D scene, the system updates the corresponding weight value in the semantic database using equation (1). This process means the interactive learning. After several interactions of the user using the individual learning functionality, the system obtained the weight parameters of parent-child relationships of the semantic database shown in the right part of Figure 4 (c), and the system can generates the 3D scene shown in the left part of Figure 4 (c). The system also generates the 3D scene shown in Figure 4 (d) preserving semantic constraints of the group layout and the regular interval layout in addition to the user preference about parent-child relationships.

Scene learning functionality

The scene learning functionality extracts parent-child relationships of all the existing object classes out of a 3D scene. Using this functionality, the user can interactively change weight values of parent-child relationships by choosing his/her preferable 3D scenes.

Experiment of the scene learning functionality

We introduce an experimental result about the scene learning functionality. At first, our system generates an initial semantic database which has parent-child relationships of all 3D objects that means any objects can be connected to each other. Figure 5 (a) shows the screen snapshot of this case. After the user loads one sample from existing 3D scenes which he/she likes, using the scene learning functionality, the system extracts parent-child relationships among all 3D objects included in the loaded 3D scene as shown in Figure 5 (b). Every time when the user indicates new other 3D scene as one sample for the learning, the system extracts parent-child relationships about the 3D scene and updates the weight parameters of parent-child relationships of 3D objects in the semantic database as shown in Figure 5 (c). Using the learned semantic database, the system can generate 3D scenes preserving the user preference. Figure 5 (d) shows the screen snapshot of one of the generated 3D scenes.

CONCLUSION

This paper proposed an interactive learning interface to statistically obtain user preference of semantic constraints, i.e., parent-child relationships among 3D objects, from existing samples of 3D scenes. While the natural language approach for 3D scene construction has the serious problem that defining semantic database as the text form is time-consuming, our proposed interface can help the user to interactively define his/her preference about parent-child relationships among 3D objects. To extract such user preferences through our proposed interface, we added a weight parameter about a parent-child relationship into semantic database records of our previous automatic 3D object layout generation system. Consequently, our extended system can generate more realistic 3D scenes according to the user preference.

As the future work, we will apply more weight parameters to other constraints, e.g., a contact constraint, occupancy distance, etc, in the semantic database. Especially, about occupancy distance, we try to use Gaussian distribution determined by calculating the average and the deviation of distances between any two 3D objects in 3D scenes. We also try to use Genetic Algorithm to generate more desirable 3D scenes those preserve multiple different user preferences simultaneously.

ACKNOWLEDGEMENT

This study has been supported by the Research Fellowships of the Japan Society for the Promotion of Science for Young Scientists.

REFERANCES

[1] Y. Akazawa, Y. Okada and K. Niijima, Automatic 3D object placement for 3D scene generation, The European Simulation and Modeling Conference 2003 (ESMC2003), pp. 316-318, 2003.

[2] Akazawa, Y., Okada, Y., and Niijima, K. : Automatic 3D scene generation based on contact constraints, Proc. of the Eighth International Conference on Computer Graphics and Artificial Intelligence (3IA'2005), pp. 51-62, Limoges, France, May 11-12, 2005.

[3] Bukowski, R. W. and Sequin C. H. Object Associations, Symposium on Interactive 3D Graphics, pp. 131-138, 1995.

[4] Coyne, B. and Sproat, R. WordsEye: an automatic text-to-scene conversion system, SIGGRAPH 2001, pp. 487-496, 2001.

[5] Smith, G, Salzman, T. and Stuerzlinger, W. Integration of constraints into a VR Environment, VRIC 2001, pp. 103-110, ISBN 295157300-6, 2001.

[6] Xu, K., Stewart, A. J., and Fiume, E. Constraint-Based Automatic Placement for Scene Composition, Graphics Interface, pp. 25-34, 2002.

[7] Zeng, X., Mehdi, Q. H. and Gough, N. E. 2003 “Shape of the Story: Story Visualization Techniques” Proc. of Information Visualization 2003 (IV03), pp. 144-149, 2003

[8] Y. Okada, and Y. Tanaka,: IntelligentBox: A Constructive Visual Software Development System for Interactive 3D Graphic Applications, Proc. of Computer Animation '95, IEEE Computer Society Press, pp.114-125,1995.

[9] Y. Okada, and Y. Tanaka,: Collaborative Environments of IntelligentBox for Distributed 3D Graphics Applications, The Visual Computer, Vol. 14, No. 4, pp. 140-152, 1998.

ARTIFICIAL INTELLIGENCE IN GAMES

Applying Robotic Techniques to Behavior Recognition in Game Agents

Verena V. Hafner
Berlin
Germany
vvh@ieee.org

Christian Bauckhage, Martin Roth
Deutsche Telekom AG, Laboratories
Berlin, Germany
www.telekom.de/laboratories/

Abstract

Most approaches to behavior synthesis in artificial characters assume a set of elementary behaviors which are activated according to the agent's spatio-temporal context, goals, desires, and needs. Towards the problem of automatically determining a suitable set of elementary behaviors, this paper reports on adopting a successful technique from robotics to game agents. The concept of interpersonal maps has been thoroughly tested in experiments with AIBO robots and from experimenting with recordings of human game-play, we found that it also allows for distinguishing types of interactions between human controlled avatars. Interpersonal maps therefore provide an avenue to automatically mine and learn elementary behaviors from human generated data.

1 Motivation and Background

Recently, academic interest in the problem of programming believable computer game agents has grown considerably. Contributors have studied the problem from the perspectives of deliberative AI (Laird 2001), machine learning (Bauckhage, Thurau & Sagerer 2003, Yannakakis & Hallam 2004, Arvey & Aaron 2005), evolutionary computing (Priesterjahn, Kramer, Weimer & Goebels 2005), and robotics (Adobbati, Marshall, Kaminka, Schaffer & Sollitto 2001, Amir & Doyle 2002, Brown, Barnum, Costello, Ferguson, Hu & Wie 2004). Common to all these works is that they implicitly revert to the concept of *behaviors*.

Computational architectures for synthesizing a suitable mix of *reactive*, *tactical*, or *strategic* behavior, of course require a corresponding vocabulary of elementary behaviors. While in practice this is usually provided by means of preprogrammed scripts, modern computer games also allow for a data-driven approach. Following an idea by Bauckhage et al. (2003), we record matches of multi-player games played by human players. Since these recordings contain detailed information of where the players steered their avatars in the game world and what they had them do, elementary behaviors may be learned directly from human experts. Assuming that behaviors are patterns of frequent activity triggered by spatio-temporal contexts, game data can thus be mined in order to identify the building blocks of reoccurring *human behavior*.

After describing an auspicious technique first introduced for imitation learning in robotics, in section 3, we will turn to behavior mining from game data. Our focus will be on interactions between two players. We shall see that different behaviors manifest in information theoretic distances between sensory recordings of the agents. Experimental results indicate that projecting these distances into lower dimensional spaces allows for distinguishing between different behaviors in the data. A short discussion of our empirical findings will end this contribution.

2 Interpersonal Maps in Robotics

Recently, Hafner & Kaplan (2005) introduced the idea of *interpersonal maps*. Based on information distances between the sensors that can be measured for two agents over a certain time window, these maps are extended abstract body maps which not only represent the body and behavior of one agent, but the interaction between two agents.

2.1 Information Distance

The conditional entropy for two information sources X_i and X_j is defined as

$$H(X_j|X_i) = -\sum_{x_i}\sum_{x_j} p(x_i,x_j)\log_2 p(x_j|x_i) \qquad (1)$$

where $p(x_j|x_i) = p(x_j,x_i)/p(x_i)$. Traditionally, $H(X_j|X_i)$ is interpreted as the uncertainty associated with X_j if the value of X_i is known.

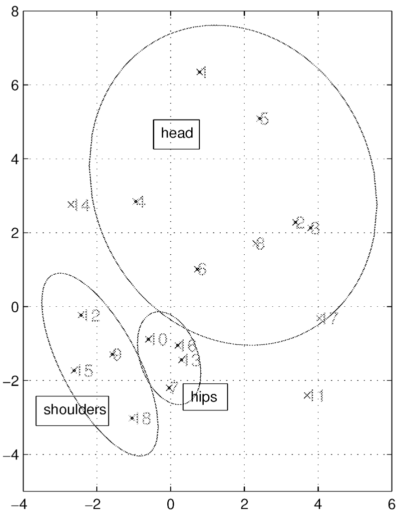
Crutchfield (1990) defines the normalized information distance between two information sources as:

$$d(X_j,X_i) = \frac{H(X_i|X_j)+H(X_j|X_i)}{H(X_i,X_j)} \qquad (2)$$

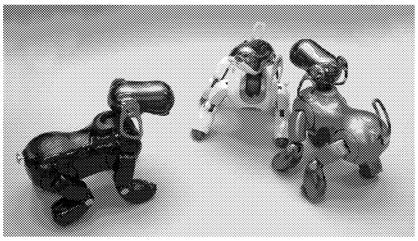
Since d is a metric for the space of information sources, it is preferable to classical mutual information $MI(X_i,X_j) = H(X_i)+H(X_j)-H(X_i;X_j)$. From the well known equivalence $H(X_i,X_j) = H(X_i)+H(X_j|X_i)$, we see that $d \leq 1$; if $d = 1$, the two sources are independent.

2.2 Visualization Through 2D-Sensor-Maps

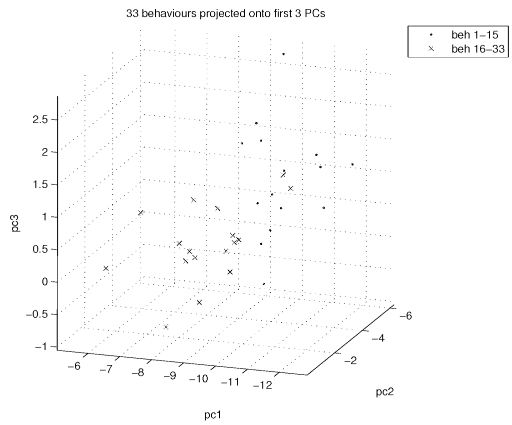
For our purpose, distances between high dimensional sensor recordings, which are captured in a distance matrix **D**,



(a)



(b)



(c)

Figure 1: Experiments with AIBOs. (a) Body map of a walking robot; (b) AIBO robots used for the experiments; (c) clustering of different robotic behaviors (walking, non-walking).

must be transformed into a representation of lower dimensional points $\{\mathbf{p_i}\}$. This can be cast as a constraint satisfaction problem, where each couple of points $\mathbf{p_i}$ and $\mathbf{p_j}$ should satisfy:

$$||\mathbf{p_i} - \mathbf{p_j}|| = d_{ij} \qquad (3)$$

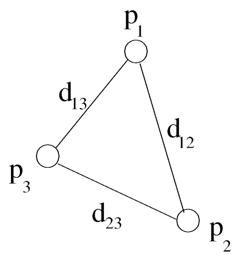


Figure 2: Didactic example for placing three points p_i in a two-dimensional space such that their metric distances correspond to the information distances d_{ij}.

and $||\mathbf{p_i} - \mathbf{p_j}||$ is the Euclidean distance between the position of the ith and jth point and d_{ij} the corresponding distance in the matrix $\mathbf{D}$. This results in a system of $\frac{n(n-1)}{2}$ equations and a set of n points of dimension $n-1$ would permit an optimal solution for these constraints. However, in order to obtain a representation in dimensions lower than $n-1$, one has to resort to approximation. Pierce & Kuipers (1997) describe a statistical method to determine a suitable dimensionality for projecting a given data set. In this paper, we consider two-dimensional projections for illustrative purposes although they may not be the optimal ones.

The information contained in $\mathbf{D}$ can be mapped to a two-dimensional space using an iterative procedure of positioning points in the subspace such that the metric distance between two points in this map approximates the distance in the distance matrix (see Fig. 2). Our algorithm iterates two simple steps. After each information source X_i is randomly assigned a point $\mathbf{p_i}$ in the two-dimensional plane, these steps are:

1. Compute the force $f_i = \sum f_{ij}$ acting on each point $\mathbf{p_i}$, where

$$f_{ij} = (||\mathbf{p_i} - \mathbf{p_j}|| - d(X_i, X_j)) \frac{(\mathbf{p_j} - \mathbf{p_i})}{||\mathbf{p_j} - \mathbf{p_i}||}$$

2. Move each point $\mathbf{p_i}$ according to the force f_i, i.e.

$$\mathbf{p_i} \leftarrow \mathbf{p_i} + \frac{1}{n} f_i.$$

2.3 Experiments with Robot Behaviors

Interpersonal maps are an extension of body maps (see Fig. 1) where the information distances among all sensory pairs of two behaving agents are taken into account and the corresponding 2D representation shows the interaction between two agents.

The idea of interpersonal maps has been applied by Kaplan & Hafner (2006) for a range of different robotic experiments. The robots considered were Sony AIBOs where for each of which 18 sensors were used. One of the experiments showed that when one robot was imitating another robot, the respective interpersonal map was similar to two body maps on top

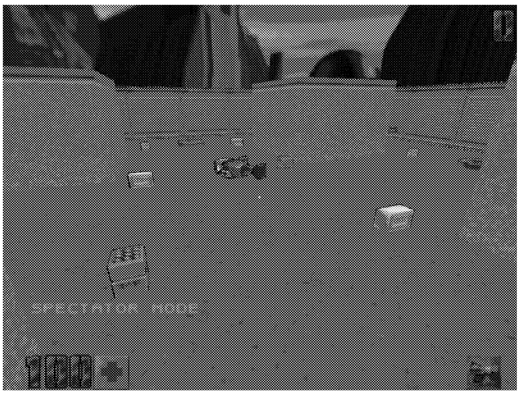


Figure 3: Screenshot of a QUAKE II® environment used in our behavior mining experiments.

of each other. When there was no imitation in the behavior or a long delay in imitation, the interpersonal map showed two clusters.

Other robotic experiments showed that different robotic behaviors could be distinguished and walking and non-walking behaviors were building two clusters in the 3D space of the first principal components of the information distance matrices.

3 Interpersonal Maps for Computer Game Agents

In our work on applying the technique of interpersonal maps to the game setting, initial experiments were carried out using the game QUAKE II®. For accessing the information contained in recorded network data, we applied the QASE API developed by Gorman, Fredriksson & Humphrys (2005).

Considering simple, almost two-dimensional environments as exemplified in Fig. 3, we recorded several sessions of four different types of interactions between two human-controlled agents:

Escaping behavior: One agent tried to escape another agent who was pursuing him directly.

Chasing behavior: One agent was being chased by the other agent who tried to anticipate the motions of his prey.

Following behavior: An agent immediately followed another agent who was not trying to flee but was leading the way.

Random behavior: The two agents were moving independently and randomly so that they only had chance encounters.

The sensors we considered are the positions and velocities of the two agents (see the listing in Tab. 1).

Number	Name/type of sensor
1	x coordinate of agent 1
2	y coordinate of agent 1
3	x velocity of agent 1
4	y velocity of agent 1
5	x coordinate of agent 2
6	y coordinate of agent 2
7	x velocity of agent 2
8	y velocity of agent 2
9	absolute velocity of agent 1
10	absolute velocity of agent 2

Table 1: Numbering scheme of sensors applied in producing the visualizations in Figs. 4 to 6.

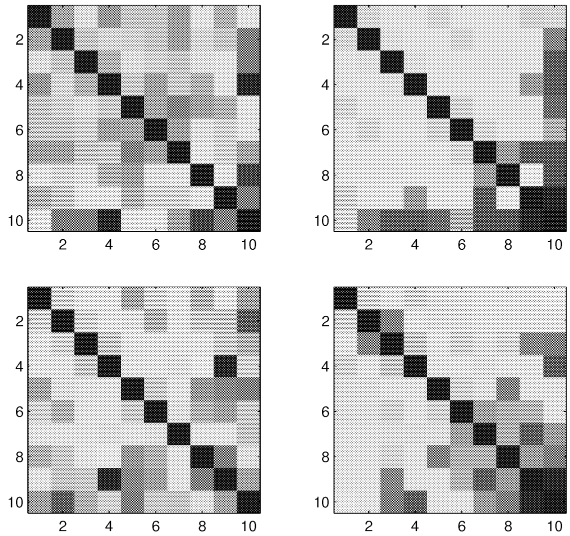
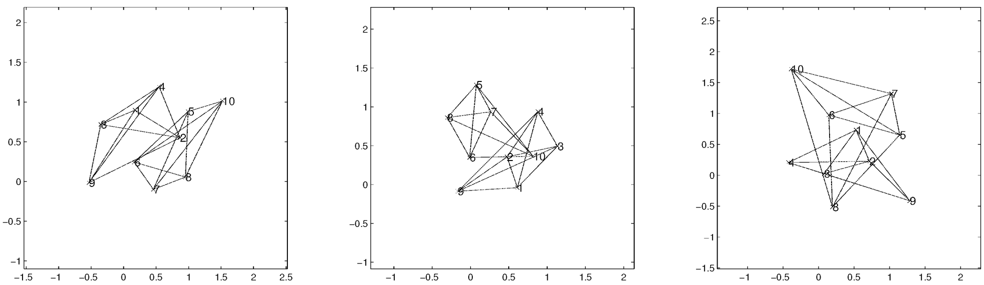


Figure 4: Information distance matrices for chasing, escaping, following and random behavior.

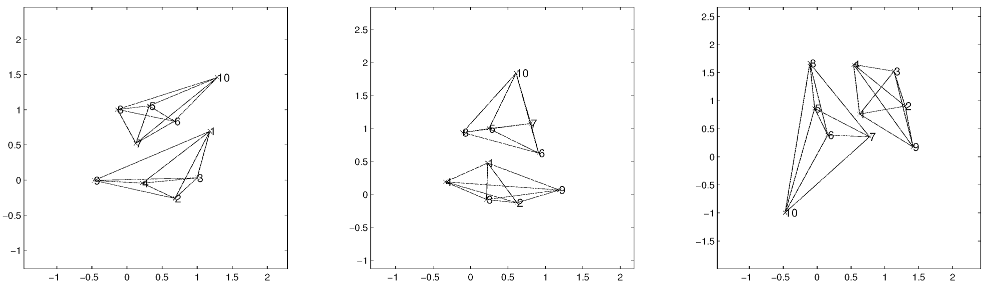
Figure 4 displays the information distance matrices for the four behaviors. The ten items on each axis correspond to the sensors introduced in Tab. 1; the coloring scheme encodes the corresponding information distances: dark blue denotes zero distance as it appears between identical sensors (i.e. on the diagonals of the matrices).

Figures 5 and 6 depict corresponding two-dimensional interpersonal maps resulting from our relaxation algorithm. The ones in Fig. 5 were created from sequences of 600 data samples (60 seconds), the ones in Fig. 6 resulted from 2000 data samples (200 seconds). Note that, in this form, the maps provide a first qualitative idea of the interaction, but should not be confused with a thorough statistical analysis.

None of the behaviors involves the use of a weapon or other items found in the environment. The first interaction is a simple escaping behavior where one agent was trying to flee from the other. The other agent was trying to follow him directly in order to tag him. Accordingly, the maps in



(a) Interpersonal maps for the escaping behavior.



(b) Interpersonal maps for the chasing behavior.

Figure 5: Interpersonal Maps for escaping and chasing behaviors. Sensors belonging to an individual agent are interconnected.

Fig. 5(a) show considerable overlap. The second behavior is a chasing behavior. Here, one agent is chasing another one while trying to anticipate the other's moves. In contrast to the first behavior, the agent therefore rather tried to corner his prey. Concerning the maps in Fig. 5(b), this results in clearly separate clusters. This of course agrees with the intuition behind interpersonal maps. If they show separate clusters, the correlation between the agents' sensors is less strong. The third behavior is again a more imitative behavior: one agent is following the other, who is not running away, but showing the way. Accordingly, the corresponding interpersonal maps in Fig. 6(a) show higher degrees of congruence. Finally, the behaviors mapped in Fig. 6 were totally random; the two agents were moving on the map without noticing each other but had chance encounters.

As with the AIBOs, one can thus find prototypical distance matrices that characterize different types of behavior. We are currently working on a method that tracks the temporal evolution of matrices which are computed online using a sliding window approach. Different behaviors will then be automatically detected by observing transitions between Voronoi cells of attractors in the tensor space of information distance matrices.

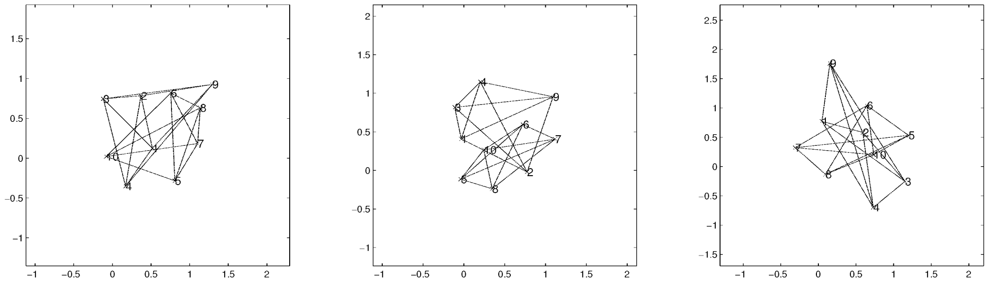
4 Summary and Outlook

In order to create more realistic and life-like acting game agents, it is crucial to synthesize different behaviors in an automated and adaptive way. This requires an architecture that is able to switch between reactive, tactical and strategic behaviors. A game bot based on such a cognitive model should ideally understand and adopt the basic behaviors typical for human gameplay. In order to appear natural, the elementary behaviors should not be provided in form of scripted programs but ideally be mined from observations of genuine human play.

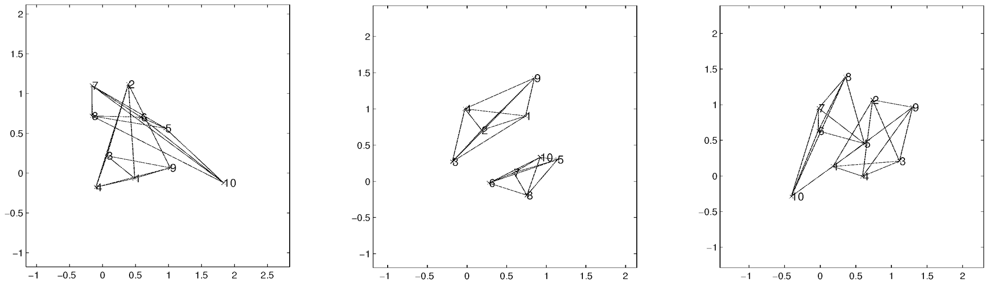
The idea behind the work presented here is to first identify behavioral building blocks used by human players. Reoccurring behaviors extracted from observations of interactions between two human players can then be used as action primitives for the cognitive architecture of an artificial game bot. As a first approach, we apply the concept of interpersonal maps (Kaplan & Hafner 2005) which was inspired by psychological findings. As in the case of experiments with AIBOs, we could show that for the game domain, too, different maps are created for different types of interactions. This allows for identifying and extracting relevant subsequences from streams of recorded human gameplay. Using machine learning techniques such as the ones described by Bauckhage et al. (2003) or Gorman et al. (2005), our next step will be to derive parametric models from these data to proceed towards more convincing behavior synthesis.

REFERENCES

Adobbati, R., Marshall, A., Kaminka, G., Schaffer, S. & Sollitto, C. (2001), Gamebots: A 3D Virtual World Test-



(a) Interpersonal maps for the following behavior.



(b) Interpersonal maps for uncorrelated behavior.

Figure 6: Interpersonal Maps for following and random behaviors. Sensors belonging to an individual agent are interconnected.

Bed for Multi-Agent Research, *in* ‘Proc. Int. Workshop on Infrastructure for Agents and Scalable Multi Agent Systems’, pp. 47–52.

Amir, E. & Doyle, P. (2002), Adventure games: A challenge for cognitive robotics, *in* ‘Proc. Int. Cognitive Robotics Workshop’, pp. 9–16.

Arvey, A. & Aaron, E. (2005), Online Markov Decision Processes for Learning Movement in Games, *in* ‘Proc. Int. Conf. on Computer Games, AI and Mobile Systems’, pp. 48–52.

Bauckhage, C., Thurau, C. & Sagerer, G. (2003), Learning Human-like Opponent Behavior for Interactive Computer Games, *in* ‘Pattern Recognition’, Vol. 2781 of *LNCS*, Springer, pp. 148–155.

Brown, C., Barnum, P., Costello, D., Ferguson, G., Hu, B. & Wie, M. V. (2004), Quake II as a Robotic and Multi-Agent Platform, Technical Report 853, University of Rochester, Computer Science Dept., Rochester, N.Y.

Crutchfield, J. P. (1990), Information and its metric, *in* L. Lam & H. C. Morris, eds, ‘Nonlinear Structures in Physical Systems – Pattern Formation , Chaos, and Waves’, Springer, pp. 119–130.

Gorman, B., Fredriksson, M. & Humphrys, M. (2005), QASE: An Integrated API for Imitation an General AI Research in Commercial Computer Games, *in* ‘Proc. Int. Conf. CGAMES’, pp. 207–214.

Hafner, V. & Kaplan, F. (2005), Interpersonal maps and the body correspondence problem, *in* ‘Proc. Int. Symp. on Imitation in Animals and Artifacts’, pp. 48–53.

Kaplan, F. & Hafner, V. (2005), Mapping the space of skills: An approach for comparing embodied sensorimotor organizations, *in* ‘Proc. IEEE Int. Conf. on Development and Learning’, pp. 129–134.

Kaplan, F. & Hafner, V. (2006), ‘Information-theoretic framework for unsupervised activity classification’, *Advanced Robotics* **20**(10), 1087–1103.

Laird, J. E. (2001), ‘Using a Computer Game to develop advanced AI’, *IEEE Computer* pp. 70–75.

Pierce, D. & Kuipers, B. (1997), ‘Map learning with uninterpreted sensors and effectors’, *Artificial Intelligence* **92**, 169–227.

Priesterjahn, S., Kramer, O., Weimer, A. & Goebels, A. (2005), Evolution of Reactive Rules in Multi Player Computer Games Based on Imitation, *in* ‘Proc. Int. Conf. on Natural Computation’, pp. 744–755.

Yannakakis, G. N. & Hallam, J. (2004), Interactive Opponents Generate Interesting Games, *in* ‘Proc. Int. Conf. CGAIDE’, pp. 240–247.

AI System Designs for the First RTS-Game AI Competition

Michael Buro[†], James Bergsma[†], David Deutscher[‡], Timothy Furtak[†], Frantisek Sailer[†], David Tom[†], Nick Wiebe[†]
[†]Department of Computing Science
University of Alberta, Edmonton, Alberta, Canada
[‡]Tel Aviv University, Israel
email: mburo@cs.ualberta.ca

KEYWORDS
Real-time strategy games, ORTS, real-time AI systems

ABSTRACT

Real-time strategy (RTS) games are complex decision domains which require quick reactions as well as strategic planning. In this paper we describe the first RTS game AI tournament, which was held in June 2006, and the programs that participated.

Introduction

Creating smart computer adversaries and teammates for human players in modern video games is challenging. AI programmers for such games are faced with limited computational resources (because most CPU cycles are still devoted to graphics), real-time constraints, huge state and action spaces, and imperfect information. In addition, the tight release schedule for video games does not leave much room for conducting AI research in games companies. Therefore, a common approach to practically solving these problems is to create an illusion of intelligence (Livingstone, 2006) by scripting actions for nonplayer characters (NPCs) and providing them with more resources including information that is not available to human players. This way it is relatively easy to create NPCs that by having more knowledge of the game state — or bigger virtual muscles — can reach the playing level of human players or even outperform them. There are however problems with this methodology. Scripted action sequences are brittle — they often cannot deal with new situations and are easily defeatable once known. More advanced variations exist (Spronck et al., 2006), whereby script parts are executed probabilistically and probabilities are updated dependent on past performance. But even with such modifications, opponent AI systems still cannot compete with strong players unless they are given unfair advantages.

To overcome this problem, several AI researchers have started to use video games as test applications for their work in recent years. Conferences are now devoted to progress in computer entertainment AI, and the interaction between computer game companies and academia has increased. Another particularly effective way of spuring research in AI is holding competitions. Great examples are the machine-machine and man-machine competitions in the 1980s and 1990s which produced stronger and stronger programs which eventually played on-par or better than the best human players in chess, backgammon, checkers, and Othello. Other examples which have helped to increase the performance of AI systems considerably include the annual planning competition, SAT competitions, and RoboCup. The goal of competitions like the one which we are going to describe here is to repeat the success of classic game AI systems in the area of more complex video games.

In the remainder of the article we first describe the game genre we are interested in — Real-Time Strategy (RTS) games — and the programming framework ORTS we have developed for it. Then, after presenting the tournament game categories, we describe the programs that participated in the first AIIDE RTS game competition, present their tournament results, and conclude the paper with ideas on future RTS game AI competitions.

RTS Games and ORTS

Real-time strategy games are typically tactical simulations engaged in by two or more players. These games are fast-paced and pose several challenging problems such as incomplete information, the need for long-range planning, and a continually changing world with limited time to plan (Buro and Furtak, 2004).

A player can be in control of potentially hundreds of units, each with several possible actions that may be taken several times a second. A naive search of the available action space is clearly intractable. This necessitates potentially several levels of abstraction, for controlling individual units and larger armies.

Games typically involve simplified economies consisting of gathering resources which may be used to construct buildings, research new abilities, and train offensive and support units. Resource usage must be balanced to construct an army capable of effectively exploit opponents' weaknesses while being able to defend against potential threats.

Determining an effective strategy often relies on accurate opponent models. Specifically, determining the types of enemy units that an opponent will likely produce, and how they will be used to attack, at what time, and at which location.

ORTS

The Open Real-Time Strategy (ORTS) game engine, available from `www.cs.ualberta.ca/~mburo/orts`, provides a flexible framework for studying AI problems in the context of RTS games. The ORTS engine is scriptable, which allows for game parameters to be easily changed, and new types of games, or subsets of existing games, to be defined.

Unlike most RTS games, ORTS uses a server-client framework. Instead of each client maintaining a local copy of the entire game state, each frame the ORTS server only sends a client the information actually available to it. This effectively eliminates the ability of clients to cheat by applying simple map-revealing hacks.

Units in ORTS are simple geometric primitive (circles, rectangles, and line segments) located on a fine grid. Objects may travel at an arbitrary heading, with collisions accurately computed by the server.

Map terrain is specified by a grid of tiles, with each tile capable of having arbitrary corner heights and being one of several terrain types. Boundary objects with various collision masks are automatically created along discontinuities between tiles.

Unit vision is tile-based, with different units having a sight range that determines how many tiles away they can see. When the "fog of war" is enabled, a player only has up-to-date information about tiles that are currently seen by an allied unit. The vision model also supports "cloaked" units which can only be seen by "detectors".

All ORTS components are open-source. Along with the server-client framework, this allows users to create their own AI components capable of acting autonomously or to augment a human player.

The AIIDE RTS Game Competition

The RTS Game Competition presented at AIIDE '06 consisted of three separate game categories, arranged in increasing order of complexity. These categories addressed the tasks of multi-unit pathfinding, local combat, and dealing with imperfect information, in that order. Effective solutions in one category relied on implementations from the previous game types.

Game 1: Cooperative Pathfinding

The first game is stated as the task of gathering as many resources as possible within a given amount of time. The player begins the game with one base surrounded by workers. These workers must travel to resource patches randomly positioned on the game field, spend a short amount of time to collect those resources, and finally bring them back to the base.

At the start of the game the entire map and the locations of all resources are known to the player. To complicate the task, the map contains both impassible terrain obstacles, and indestructible mobile "sheep", which randomly travel a short distance, stop, then continue. The entire scenario is perfect information, except for simultaneous actions on the part of the workers and the sheep.

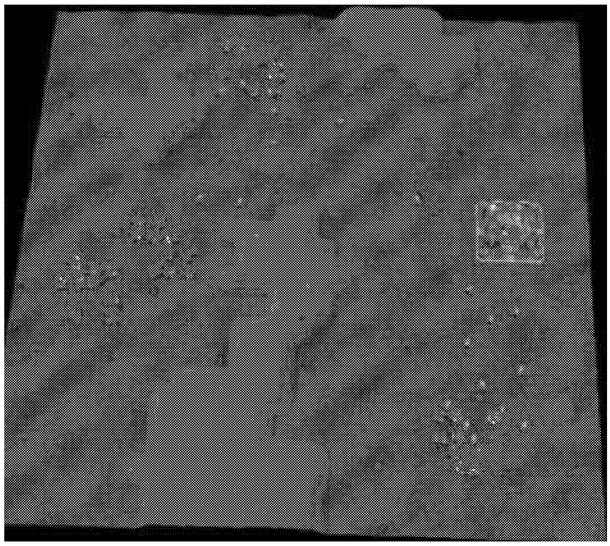


Figure 1: Game 1 client display.

Practically, the task is then to effectively coordinate the motion of the workers to minimize total travel time between the base and the resource patches. Spending a long time to compute near-optimal routes may result in the world having changed to the point where the computed solution is no longer valid.

Game 2: Local Combat

The second game is two-player tank combat, where the objective is to destroy as many of the opposing player's bases as possible within 10 minutes. Each player begins with 5 bases randomly distributed within the playfield, and 10 tank surrounding each base. A game ends immediately if all of one player's bases are destroyed.

As with the first game, each player has full visibility of the entire map. Plateaus, which are impassable and block line-of-sight tank attacks, are randomly placed on the map. Neutral, indestructible sheep also wander randomly.

The focus of this scenario is to effectively engage and destroy enemy squads. Formations which allow one side to concentrate fire on a small number of tanks while exposing themselves to few attackers are preferable. An agent must therefore coordinate the motion of the tanks to bring about these positions while avoiding collisions with other tanks (both allied and enemy) and unpredictable sheep.

Game 3: Mini RTS

The third game is a stripped-down version of a "real" RTS game. Two players begin with one base and several workers located next to a resource patch. The rest of

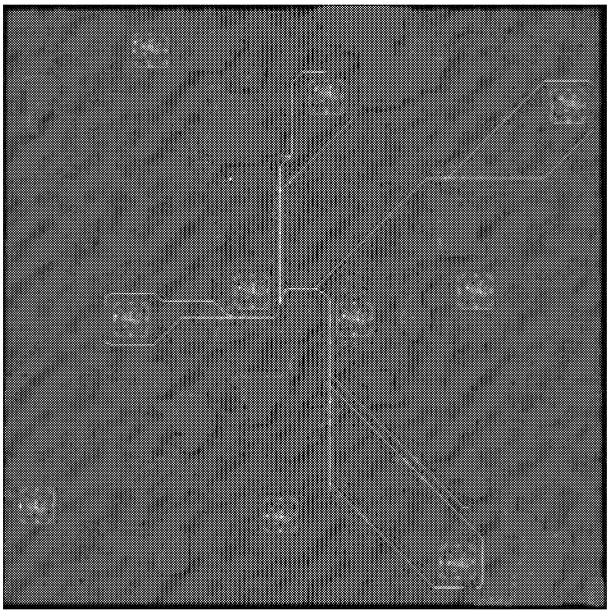


Figure 2: Game 2 client display.

the map and the location of the enemy base is initially unknown. A fog-of-war limits the currently observable parts of the map to those regions that can be seen by allied units.

A player is able to spend minerals and use a worker to construct a barracks and then a factory. Barracks and factories can then be used to train marines and tanks respectively. Tanks have more hitpoints, attack power, and range, but cost more than marines.

The objective of this game is to obtain more points than the opponent before time runs out. Points are awarded for gathering resources, constructing buildings, training units, and for destroying enemy buildings and units. The game ends early if all of one player's buildings are destroyed.

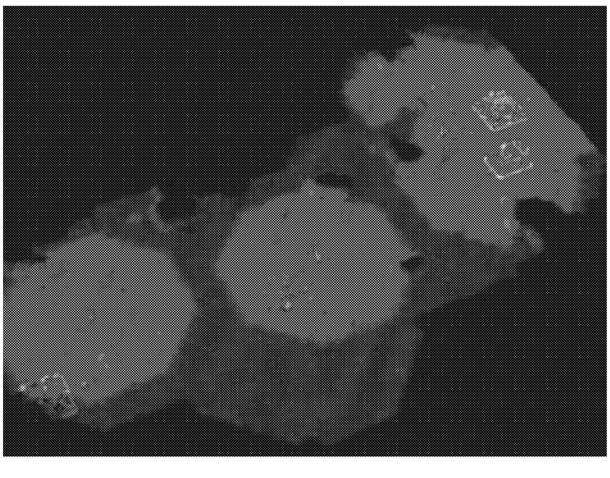


Figure 3: Game 3 client display.

Tournament Setup

All tournament games were played between June 16 and 18, 2006 on 31 undergraduate lab computers in the computing science department at the University of Alberta. Each machine was equipped with a single Athlon XP 1.5 GHz CPU and 512 MB RAM running Linux 2.4.31 and gcc 4.1.1. Shortly prior to the competition a multi-threaded ORTS tournament manager was completed by Krysta Mirzayans. This software greatly simplified running the tournaments and allowed us to play a large number of games.

Authors had access to the tournament computers on which they could upload their programs to test them in individual protected accounts which were frozen just before the tournament commenced. Each participant was asked to send a magic integer to a member of the independent systems group which also set up the tournament accounts. These numbers were then exclusive-or combined to form the seed of the random number generators used for creating all starting positions. This way, no participant was able to know beforehand what games would be played. In order to reduce dependency of game results on starting positions

In what follows we describe all tournament entries in turn and present the results of the tournaments.

Game 1 Entries

brzo1

Author: Michal Brzozowski, University of Warsaw, Poland

Michal's entry used a discrete graph-based terrain representation where neighboring vertices are connected if their connecting edge is traversable. Workers are guided by a finite state machine (FSM) with the following states: move-to, mine, go-back, drop-resources, avoid (entered when hitting a moving obstacle. Avoids obstacles by moving to the left. When it hits a static obstacle, it moves to a random direction), and emergency-path (when hitting a number of obstacles in the avoid state, tries to get back to original path). A coordinator assigns workers to resources based on shortest paths. Each worker picks the closest mineral from its starting point with less than 2 workers assigned already.

creed1

Author: Michal Szostakiewicz (University of Warsaw, Poland)

A search graph is built from nodes representing tile centers. Edge weights depend on mobile objects close by to prevent collisions. Each worker is assigned a random mineral patch and is sent to it. Shortest paths are computed by Djikstra's algorithm. When colliding, workers move to a random location nearby.

umich1

Authors: Joseph Xu and Sam Wintermute (University of Michigan, U.S.A.)

This entry was implemented in the SOAR architecture using a modified version of the standard ORTS pathfinding with an added local obstacle avoidance system. Workers are guided by a mining manager and a FSM. If a worker exceeds its estimated travel time, it requests a new route from the mining manager. The mining manager learns which routes are bad.

uofa1

Authors: David Deutscher (Tel Aviv University, Israel) and Nick Wiebe (University of Alberta, Canada)

This entry is based on three modules:

1. A single-unit path planning algorithm using a simple grid based A* algorithm, which uses a multiple-resolution world representation, pluggable goal definitions (including "touch a target objejct") which can handle variable-sized and shaped objects.

2. A path execution system which calculates the necessary motion at each simulation tick to move a unit along a predetermined path. To do this force fields are used to attract moving units to a point on the path in front of them and to repel them from other objects, buildings, and walls. For each unit traveling along a path, every object, building, and wall whose distance is below a minimum threshold exerts a force on the unit inversely proportional to the square of the distance between them. The movement vector for the unit is the sum of the forces acting on it. This approach solved the problem of path obstruction by sheep and enemy units, as units would just roll off the obstruction. It also is used to give priority to moving units. By calculating and applying these forces to units that were not moving along a path, idle units can be pushed out of the way of moving units. A small randomized vector is added to this pushing force in order to limit the distance that units are pushed in a single direction. Pushing proved to be important for game 2, where it speeds up large group attacks.

3. Dynamic allocation of minerals to workers, based on minimizing a weighted (1:1) combination of the Euclidean distance from the worker's current position and the static path's length between the mineral and the control center (where static means the shortest path found while considering only static obstacles — boundaries, other minerals and the control center itself). Statically-blocked minerals are not assigned and a single worker per mineral is preferred, unless no other option is available. Failures to plan a path or to reach a mineral raises a limited-time flag preventing its use for a couple dozen turns.

Game 1 Results

Initially it was planned to play 300 games per entry lasting 10 minutes each on June 16. But twenty hours into the tournament the tournament manager exceeded its disk quota which was set too low. So, to stay on track, the number of games had to be reduced to 225 per program. The final results were as follows:

rank	name	score	games	ratio
1.	umich1	1458455	225	6482.0
2.	brzo1	1136690	225	(*) 5051.9
3.	uofa1	1136790	225	(*) 5052.4
4.	creed1	559380	225	2486.1

Program brzo1 was leading over uofa1 almost all the time. So, team uofa happily conceded 2AD place to it (*) due to shortening the tournament. Entry creed1 made the server crash several times by referring to fully mined mineral patches which had vanished.

Game 2 Entries

umaas2

Authors: P. Kerbusch, N. Lemmens, M. Urlings, V. Vorsteveld (University of Maastricht, The Netherlands)

This entry creates 5-tank squads in single-file formation. The squad leader plans a path to the nearest base and others follow. When enemy tanks are encountered, a wedge formation is formed and the weakest of all tanks within range is attacked. Tanks move towards the weakest target while firing at the weakest target within range. When no more enemy tanks are in sight, the squad resumes its path in file formation. When a base is destroyed, a new base is located and the squad starts moving towards it. All objects excluding opposing tanks are considered obstacles and each tank reserves one tile.

umich2

Authors: Joseph Xu and Sam Wintermute (University of Michigan, U.S.A.)

This entry is a SOAR agent that attacks tanks before bases. Tanks are grouped by spatial distance, and groups of tanks will try to attack enemy tank groups that are smaller than them. If no such enemy groups exist, smaller groups will try to regroup into larger groups and go for their target then. Unfortunately, a bug was introduced just before the deadline, and most of this behaviour was not realized in the competition.

uofa2

Authors: K. Anderson, J. Bergsma, D. Demyen, T. Furtak, D. Tom, F. Sailer, N. Wiebe (University of Alberta, Canada), D. Deutscher (Tel Aviv University, Israel)

The program first finds a suitable meeting location for all tanks close to the average tank position. Then all tanks are sent there after joining locally first. When the join operation is finished, the entire group starts hunting and attacking the closest enemy tank. When all tanks are destroyed, bases are attacked. The weakest targets are attacked first while minimizing overkill.

The task architecture utilized in this tournament entry (and also in uofa3) was designed to be simple, yet powerful. Each task is composed of a list of units assigned to it, as well as a list of child tasks, and a current line number. Each task also has an execute function which defines the task's behaviour. A task is defined as a series of statements to execute, and the navigation between these statements done by having a variable pointing to the current line number. The statements can range from giving individual units precise orders to creating subtasks for subsets of units. Finally, each task has an identical update function, which is executed whenever an object belonging to that task has completed an order or has been killed. If all the units of a task have completed their orders, the task executes its next statement as defined in the execute function. If the task has reached the end of its execute function, the task itself completes and notifies its parent and also gives control of the units back the the parent. The parent then executes its next statement, and so on. This framework allows complex strategies to be formulated by creating a series of subtasks, and combining them into more complex tasks.

Game 2 Results

Four hundred two-game matches were played for each player pair on June 17. Each game lasted at most 15 minutes. Here are the obtained results:

rank	name	score	matches	ratio
1.	uofa2	390.0	400	0.975
2.	umaas2	210.0	400	0.525
3.	umich2	0.0	400	0

Entry uofa2 won almost all of its games. It crashed in 20 games, but only lost 10 matches in total. The strategy of all tanks meeting near the center first and then hunting tanks with a big group tanks was quite successful. It is also hard to beat in the absence of area effect weapons. Therefore, in subsequent competitions control centers will likely be made weaker to make leaving bases undefended more risky.

Game 3 Entries

umich3

Authors: Joseph Xu and Sam Wintermute (University of Michigan, U.S.A.)

After gathering enough minerals and having built enough marines this SOAR agent sends marines to explore and attack. Defensive behaviour takes precedence, and all units are pulled into battle if the base is under attack.

uofa3

Authors: K. Anderson, J. Bergsma, D. Tom, T. Furtak, F. Sailer, N. Wiebe (University of Alberta, Canada), D. Deutscher (Tel Aviv University, Israel)

Using the task framework described earlier, this entry implemented a so-called turtling strategy which creates a barracks and enough workers such that each visible mineral patch is mined. It then produces as many marines as it can which wait for the opponent to arrive. The squad combat AI described earlier also controls all combat actions in this game and an older version of the mining AI controls gathering minerals.

Game 3 Results

Two hundred two-game matches were played on June 18 — each one lasting for at most 20 minutes. The results were as follows:

rank	name	score	matches	ratio
1.	umich3	124.0	200	0.62
2.	uofa3	74.0	200	0.37

When watching some replays it becomes apparent that there is much headroom in terms of increasing playing strength in game 3. Neither program expanded to other resource locations, nor did they create tanks in later game stages.

Conclusion and Outlook

In this paper we have presented the results of the first RTS game AI competition which was held in June 2006 and described the algorithms used in the tournament programs. Many areas of improvement have been identified, including ORTS documentation, program and server stability, group pathfinding, and high-level AI. We regard this as a promising beginning of a series of many future RTS game AI competitions which hopefully will help elevating the level of real-time AI to new heights.

Acknowledgments

Financial support was provided by the Natural Sciences and Engineering Research Council of Canada (NSERC).

REFERENCES

Buro M. and Furtak T., 2004. *RTS Games and Real-Time AI Research.* In *Proceedings of the Behavior Representation in Modeling and Simulation Conference (BRIMS).* 63–70.

Livingstone D., 2006. *Turing's test and believable AI in games. Computers in Entertainment (CIE)*, Vol. 4(1).

Spronck P.; Ponsen M.; Sprinkhuizen-Kuyper I.; and Postma E., 2006. *Adaptive Game AI with Dynamic Scripting. Machine Learning*, Vol. 63(3), 217–248.

A MULTI-AGENT SOCCER SIMULATOR BASED ON A SIMPLIFIED SOCCER MODEL

Iwein J.J. Borm
L.J.M. Rothkrantz
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
Mekelweg 4, 2628 CD Delft, The Netherlands
E-mail: {iweinb} @ gmail.com

KEYWORDS
Multi-Agent Systems, Cooperative Agents, Soccer Simulation.

ABSTRACT

This paper describes a multi-agent soccer simulator based on a highly simplified soccer model. The simplified soccer model starts on a high level, and is suitable for research on high-level strategies. The simulator features a three-layer model representing the physical environment, robot, and behavior. This layered approach makes the system extensible. The system is fully functional and has been used in an educational environment. Results from this show that it is possible for unexperienced novice programmers to implement cooperative intelligent soccer teams in limited time. The simplified model allows advanced high-level strategies and self-learning elements.

INTRODUCTION

Challenge

Multi Agent Systems (MAS) deal with multiple agents that are collectively capable of reaching goals that are difficult or impossible to achieve by individual agents. One increasingly popular example of MAS is robot or embodied agents. The ultimate challenge of this domain is stated as follows (Kitano and Asada 2000):

"By mid-21st century, a team of fully autonomous humanoid robot soccer players shall win the soccer game, complying with the official rules of the FIFA, against the winner of the most recent World Cup."

The 2050 goal is the successor of the chess challenge, that was accomplished in 1997 (Schaeffer and Plaat 1997). The domain characteristics of soccer vary largely from those of chess. Table 1. illustrates this.

Table 1: Comparison of Chess and Soccer

	Chess	Soccer
Environment	Static	Dynamic
State Change	Turn taking	Real time
Info. Accessibility	Complete	Incomplete
Sensor Readings	Symbolic	Non-symbolic
Control	Central	Distributed

Problem

The high level aspects, such as dealing with cooperation and team work are not dealt with extensively in present literature. The main reason why most research is being done on the low level details of playing soccer is that having a reliable, fast, low level implementation is a necessary requirement for any high level strategies. It is however not a sufficient requirement. There is no use in reasoning about complex strategies to trick the opponent when simple commands are not executed in accordance with the expectation. In both the simulation league, and in all physical robot leagues, a low level implementation of many primitive functions is required before having a fully functional team. Furthermore, there is a large dependance on the quality and stability of the low-level implementation on the performance of high-level strategies. As a result, the problem domain that was already inherently very uncertain is extended with even more uncertainties.

Simulator

Rules governing fair play dealing with, amongst others, free kicks, bookings, penalties, throw-ins, corners, goal kicks and offside, are all necessary elements of human soccer. Including these rules in a simulator however, necessitates their inclusion in the design and implementation of any team. A further dimension of soccer is coaching and allocating players with certain strengths and weaknesses to certain positions. Although attention to all these properties and events must be paid in order to compete with human players, they distract attention from developers towards handling these special cases. For this reason, the aforementioned elements are not included in the world model of the system. Furthermore, all robots are physically identical.

RELATED WORK

As early as in 1994, soccer was introduced as one of the great challenges for Artificial Intelligence (Sahota 1994). A lot of research and experimentation has been going on in the increasingly popular domain since. In the year 2000, the 2050 goal (Kitano and Asada 2000) was formulated as the new challenge for AI for the coming 50 years, in response to the completion of the chess challenge in 1997.

Competing with humans and in accordance with the FIFA rules demands highly sophisticated humanoid robots that have physical abilities comparable to that of humans. Developing such robots can not be done overnight. A set of intermediate targets with regards to the robots, the rules of the game and the environment in which the game takes place have been set out in (Burkhard et al. 2002). Through gradually adding complexity and making competitions ever more humanoid, the 2050 goal is to be met.

Robocup

RoboCup consists of 5 leagues; (1) small-sized robots; (2) middle-sized robots; (3) four-legged robots (Aibo's); (4) humanoid robots; and (5) simulation. The first four leagues deal with actual physical robots – and face the corresponding technical problems, whereas the simulation league has no physical representation.

Simulation League

The soccer simulator was introduced in 1995 (Noda 1995) and has seen a lot of development since. In this league, teams of 11 autonomous software agents compete on a simulator in a match lasting 10 minutes. Unlike the hardware leagues, this league does not face the hardware constraints of current technology. A vast body of literature has been written on various techniques used to create cooperative teams, such as machine learning, multi-agent collaboration, and opponent modeling. There are several variations within the simulation league; A 2D competition, a 3D competition, a 3D-development competition and a coaching competition.

The Soccer Simulator is presently a platform for researching cooperation and team work. It is not restricted by the current state of technology, however there is a large threshold for starting working with the simulator. The simulator is designed from the perspective of being a realistic soccer simulator. As such, many low level functions have to be implemented prior to reasoning on a higher level, and this higher level reasoning depends on the uncertainties that remain after the lower level implementations. Resulting implementations are very complex, and research on a strategic level is difficult.

FLEEBLE

The simulator (Borm 2006) was implemented using a simple agent framework called Fleeble (Pantic et al. 2004). Fleeble is Java-based and provides all functionality required for development and implementation of the simulator and soccer teams. It allows concurrency (multi-threading), multiple agents (and easy communication between these agents) and namespaces.

Namespaces simulate different computers. When loading a child Agent, a certain name space can be appointed and Fleeble will lock the Agent's communication to this namespace. This is particularly useful for multi-agent soccer, as it enables forced namespaces on player agents, such that they can only communicate with the framework and not directly with each other. Figure 1. shows the Fleeble GUI, with the soccer simulator running.

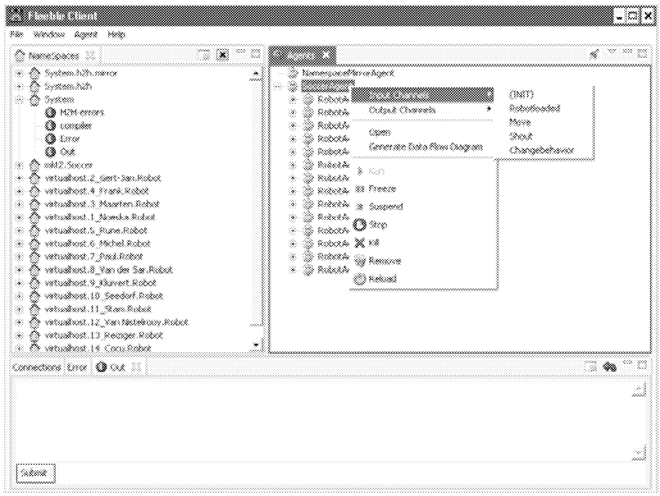


Figure 1: GUI of Fleeble Agent Framework

MODEL

Rather than developing a realistic soccer simulator, the described model is intended to be suitable as a tool for AI research. The most important difference with the official laws of the game is that the ball is always in play in the soccer model, whereas it can be taken out of play by either the referee or by leaving the field in human soccer. There is no centralized control such as a coach or a referee. There are no injuries, bookings, and all players are physically identical. Furthermore, there are no external influences such as the weather or audience. Robots have aural and visual sensors with perfect information in a narrow field. Robots can change their behavior, shout, and move. Robot speed is instant and static, speed penalties are awarded for collisions and shouting. The model features three layers: The player layer, robot layer and framework layer. This is illustrated in Figure 2.

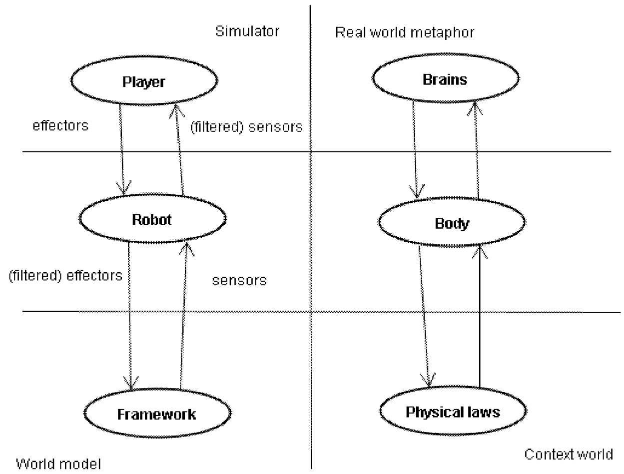


Figure 2: The System Model

The Player Layer

The player layer defines the behavior of the robot. All reasoning of a robot is done in this layer. The player layer receives sensory input from the robot layer. This is by default everything that the robot senses, but a filter can be added to simulate technical imperfections at the robot. The player layer can send its desired actions to the robot layer.

The Robot Layer

The robot layer is the intermediate layer that mediates between the framework (environment) and the player layer (behavior). It will receive commands from the player layer and attempt to execute these, when allowed by the environment. The robot layer senses directly from the environment. To simulate for the physical imperfections of actual robots, such as limited processing power, poor camera quality, imperfect control over all joints, filters can be added over the input and output of this layer. By default the robot layer will sense all visual and aural information within a predefined range without any uncertainty.

The Framework Layer

The framework layer enforces all rules of the game. Rules governing movement, vision, communication, and the game are defined and secured by this layer. The current state of all robots is kept and updated by this layer. The rules of the game and the framework parameters are publicly known by all layers.

Player Capabilities

Basic Player Movement
A robot is a circular solid object. A robot can move either forward, backward, left, or right. The orientation of a robot can be changed by turning either left or right. A robot is able to kick a ball if it is directly in front of him. A robot can not leave the field, as the framework will prevent it from doing so. Robots are physically identical and will move at the same base speed. Certain factors such as recent collisions, the movement direction, or recent communication, can inflict a temporary speed penalty on the robot.

Ball Movement
The ball is also a circular object, but unlike robots, it is not 'controlled' by any higher layer. A ball acquires movement through collision with robots, or through being kicked by a robot (a special type of collision). The ball will move in a straight line, in the direction of the collision or kick. The velocity of the ball decreases with time, as a result of simulated friction with the field. The ball comes to a halt naturally. Ball movement is entirely predictable, except when the ball is kicked. A small random direction element is added when a ball is being kicked.

Visual Model
Vision is the primary sensor of the robot. Although not the only one, intelligent teams can be developed reasoning solely about the visual perception. Robots have a field of vision that is equal in size for each robot, and defined by a given view angle and view range. All objects within this field of vision will be recognized. For each of the objects that are recognized, the name of this object, the team it is playing for, and the exact position and heading are perceived without any errors. Robots will also receive their own position.

Aural Model
To reach high-level strategic behavior by robots, communication is essential. The aural model describes how communication between robots is possible. Robots have a certain shout range. This is a fixed, predefined distance, and objects within this range in the circular area around the robot will receive any messages that the robot decides to send. Any type and amount of data can be sent at any time. The only restriction to communication is that a (severe) speed penalty lasting several seconds is inflicted on the sender of a message. Both friendly and opponent robots within the shout range will receive the message. Apart from the content, the name of the sender and the time at which the message was sent are also included.

Changing Behavior
Since the physical capabilities of robots are identical, substituting them would be a rather useless activity. To allow for adaptive team behavior, an option to substitute the player on a robot is included. Substituting the player on a robot effectively means replacing the player that is currently loaded on the robot by another player. This can be done at any time during the game, and as often as desired. The knowledge base of the existing player can be stored, and as such the newly substituted player will be able to access data collected by the old player. This feature allows for adaptive team behavior.

SOCCER SIMULATOR

Visualization

The soccer simulator that implements the aforementioned model is displayed in Figure 3.
During the development of a team, it is convenient to know the visual and aural areas of a particular robot, and to determine exactly what input a certain robot received. The first is visualized in the user interface, as illustrated in Figure 4. The latter can easily be done in Fleeble, through monitoring the human-readable channel communication.

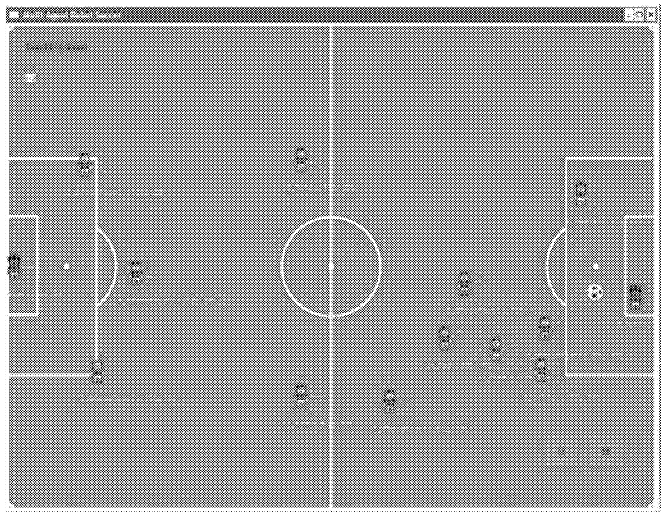


Figure 3: The Multi-Agent Soccer Simulator

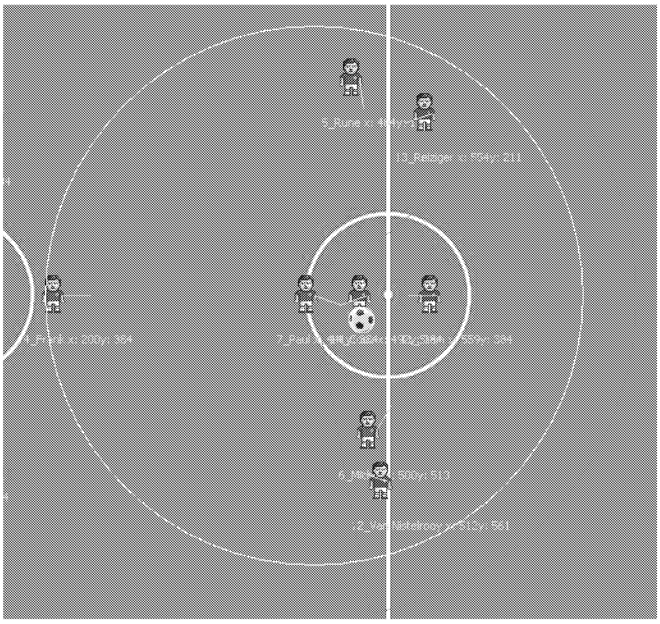


Figure 4: The Visual and Aural Sensor Areas of a Robot

In addition to the visualization of soccer matches, the simulator also features team and tournament creation interfaces. A team consists of 7 players, each with a *Player Agent*, name, base position and an icon. The information is stored in an XML file. The team creation interface (See Figure 5) will automatically create such an XML file.

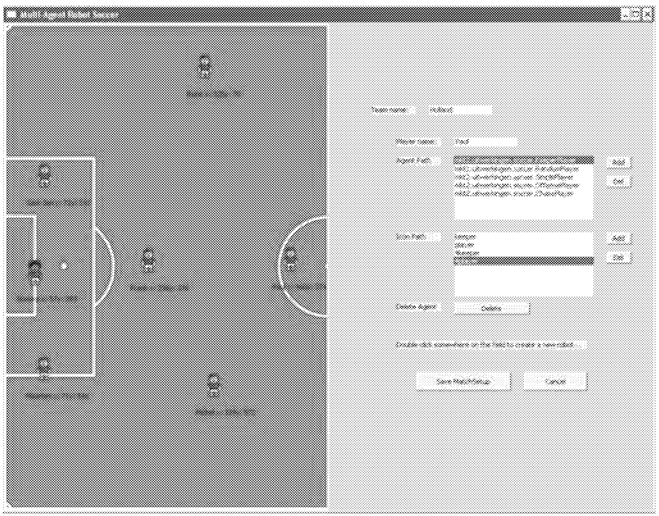


Figure 5: The Team Creation Interface

Randomness

The simulator is mostly deterministic. All parameters are known to all players, and can be used to calculate for instance where a ball will come to a stop, how much time it will take to move to a certain position, etc. There are only two random elements in the game, causing every run to be different. First, there is a random factor when kicking the ball. To prevent lucky shots from a very large distance, and to stimulate strategic behavior, a certain random distortion is added to every shot. Secondly, there is *"Java-induced randomness"*. There are 2x7 Robot Agents, 2x7 Player Agents and a Framework Agent. All Agents have their own thread. These agents use an even larger number of channels, that are also represented with threads. Since the updating of all threads is not synchronous, a message sent on the same time, on the same channel, by two different robots may reach the framework layer at different times. Since positions are not updated continuously but on discrete time intervals, this will cause robots to occasionally be lucky or unlucky. These random variations are equally distributed amongst both teams, and make every game unique.

STRATEGY

In order to develop an intelligent cooperative team in the soccer simulator, a number of design and implementation decisions must be made. A moderately intelligent reference team is included with the simulator that uses rule-based reasoning to reason about the environment. The structure of such teams is laid out in the following list:

1. Roles in the Team: A number of roles have to be identified and included in the design. It is common to distinguish between a goal keeper and other players, but further distinctions such as defensive or offensive are also possible.
2. Formation: The placement of all robots on the field in start position. For more advanced teams, the base positions of all robots may vary among strategies or even within strategies.
3. Player Skills: A wide array of low-, medium-, and high-level player skills have to be designed and implemented. Examples of such player skills include:
    a. Scout the area.
    b. Move to a position.
    c. Avoid opponent.
    d. Pass the ball.
4. High-level information: Raw observations from the environment are not useful for the rulebase. A transformation from sensory input to high level information results in the input for the rulebase. Examples include:
    a. Determine if the Ball is Visible.
    b. Determine if the Ball is Kickable.
    c. Determine whether an opponent is up ahead.
    d. Determine whether there is a teammate standing free.
5. Rule-Based Action Selection: The high-level information can be used to reason about the environment and to determine the best combination of player skills to be used.

Cooperation

The aforementioned elements together do not constitute a cooperative intelligent soccer team. Of utmost importance is the adoption and consideration of social conventions, autonomous reasoning, and the communication model. These have to be embodied in the rule-based action selection mechanism, and taken into account when deciding on what high-level information to elicit.

The soccer model was specifically designed to encourage users to consider the autonomous reasoning (and social laws) versus communication tradeoff. The severely restricted field of vision for robots drastically reduces the amount of information they can reason about. Since any amount and type of data can be communicated through shouting, visual information from neighboring robots can be transmitted and used to draw better conclusions.

To illustrate how powerful communication is in strategy, Figure 6. shows *you*, with the ball, unaware of the presence of two opponents, and the teammate. The teammate sends a message with this information, and suggests a plan. This particular plan consists of the teammate moving towards the flank, and *you* passing the ball to the teammate. *You* will reason about the likelihood of this plan succeeding (based on the newly gained information). If the plan is better than the previous plan, execute this. The teammate will be able to observe by the actions of *you* whether or not the suggested plan is being executed. The teammate will suffer from the communication speed penalty, but the robot with the ball (*you*) does not need to communicate.

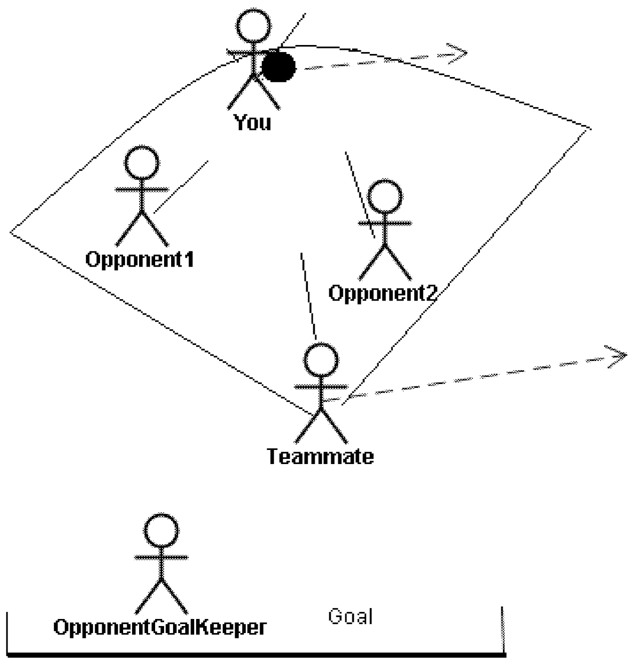
In general, deciding on what plan to execute should be decided by the players taking an active part in the play, as they have the visual information of nearby objects that other passive robots might not have. When it may seem like a good plan to pass the ball to a player by a passive observer, this plan should be *suggested* to the active player. As communication by the active player has a large cost, the passive players should attempt to anticipate based on the active player's movement. After an active player has passed the ball, it can safely communicate and inform neighboring teammates about the plan it has attempted.

RESULTS

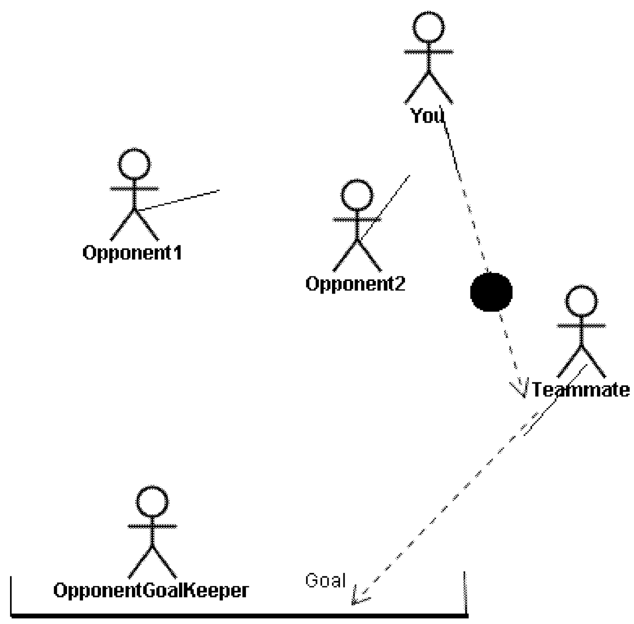
The soccer simulator was used in conjunction with the default moderately intelligent team in a first-year undergraduate introductory AI project (Borm and Rothkrantz 2006). Students were instructed to design and implement an intelligent cooperative team capable of defeating the reference implementation. Students had 3 weeks to complete this assignment. The assignment ended with a competition between all student teams. The results of this competition, held at the 6th of June, 2006 at Delft University of Technology, are shown in Table 2. The fact that first-year students, without any significant aid, are capable of developing such an intelligent cooperative team, is a proof of concept. The simplified soccer model ensures that the focus is not on optimizing skills over players, a coach, taking penalty kicks, or any other non-game quality, but solely on playing the game. Playing the game in term does not require as much effort or code as existing soccer simulators. The results from this assignment were gathered through classroom observations and an anonymous survey.

Table 2: Results. The number indicates the group number, S = SimpleTeam, R = RandomTeam, T = total score. Win = 3 points, Lose = 0 points, Draw = 1 point. [i,j] = {score of i - score of j}

#	2	3	4	5	S	R	T
1	6-0	1-4	0-10	1-4	7-5	13-3	9
2	X	3-15	9-19	4-22	2-20	30-5	3
3	X	X	4-9	6-3	22-9	18-7	15
4	X	X	X	8-4	21-9	34-3	18
5	X	X	X	X	28-5	36-2	12
S	X	X	X	X	X	32-2	6
R	X	X	X	X	X	X	0



(a) Start of Scenario



(b) Get Ready to Score!

Figure 6: Example use of Communication to Outplay Opponents

A similar conclusion can be drawn from both evaluations. Developing a team for the soccer simulator is considered challenging, motivating, and fun.

Adaptive Teams

Existing soccer simulators are plagued by a large amount of uncertainties for attempts at high-level behavior. Attempting to implement a self-learning team in such an environment is prohibitively difficult. An approach was suggested in (Borm 2006) to develop a more advanced team that has a self-learning mechanism using the soccer simulator. This approach will optimize variables governing the execution of high-level actions. For example, the desired result of *dodge opponent* would be to move past a specific opponent player without colliding. Although robots reason about a limited part of the field, they can store and share their observations over a longer period of time. As such, they can reconstruct events and their decisions - given their knowledge at the time. Optimizing decision thresholds through experimenting with their values, and then statistically evaluating the resulting performance measures, will converge to the optimal values against a specific team.

It is also possible to include a probabilistic element to the agent models. Such an approach features both advantages and disadvantages. It makes it more difficult for opponents to predict movement, but it also makes it more difficult for teammates to predict movement, and judge from this by means of social conventions.

CONCLUSIONS

In this paper we have introduced a Multi-Agent Soccer simulator based on a simplified soccer model. The model is composed of three layers, making the system extensible. The model removes all forms of centralized control and uncertainties present in the environment. Players are physically identical, and have limited aural and visual areas with perfect information. An important tradeoff is that of autonomous reasoning and social conventions versus communication. Both types are of utmost importance to successful cooperative teams.

An assignment was constructed for first-year undergraduate students, instructing them to defeat an intelligent reference team. This was considered challenging, motivating, and fun. It also serves as a proof of concept, since a limited background knowledge and programming experience do not obstruct the development of cooperative teams.

FUTURE WORK

An interesting addition to the simulator would be a network option. Users should be able to join or host matches and tournaments with other users. This can for instance be done by setting up a central soccer team database and connecting to this, then downloading the desired team, and finally playing against this team.

Another addition would be to add another layer over the player layer that contains pre-implemented functions for, amongst others, movement. This would effectively reduce the process of creating a team to defining a good rulebase.

To bridge the gap with reality, it is possible to extend the model with uncertainty, to model various states of technological progress, and use this to test the impact of this progress on the performance of a team.

A particularly interesting area of future work is the development of highly advanced teams, and testing their performance. An approach has already been suggested by (Borm 2006). It is also interesting to consider the suitability of various other AI techniques, such as dynamic bayesian networks, for intelligent soccer teams.

REFERENCES

Borm, I.J.J. 2006. “The Design and implementation of a Multi-Agent Soccer Simulator as a Tool for AI Research and Education.” *Technical Report MMI-06-12, Delft University of Technology (Aug)*.

Borm, I.J.J. and L.J.M. Rothkrantz. 2006. “Teaching AI Techniques Using Multi-Agent Soccer.” *In Proceedings of 3rd E-Learning Conference*, Coimbra, Portugal, pp. 113-118. (Sep).

Burkhard, H.; D. Duhaut; M. Fujita; P. Lima; R. Murphey; and Rojas, R. 2002. “The Road to RoboCup 2050”. IEEE Robotics & Automation Magazine, 9(2):3138.

Kitano, H. and M. Asada. 2000. “The RoboCup humanoid challenge as the millennium challenge for advanced robotics”, Adv. Robot., vol. 13, no. 8, pp. 723-737.

Noda, I. 1995 “Soccer server: a Simulator of RoboCup”. In *Proceedings of AI Symposium ’95,* Jap. Soc. For AI, pp. 29-34 (Dec).

Pantic, M.; R.-J Grootjans; and R.Zwitserloot. 2004. “Fleeble Agent Framework for Teaching an Introductory Course in AI”, Proc. Int’l Conf. Cognition and Exploratory Learning in Digital Age, Lisbon, Portugal, pp. 525-530.

Sahota, M.K. and A.K. Mackwordt. 1994. “Can situated robots play soccer?” Proceedings of Canadian AI-94, pp.249-254.

Schaeffer J. and A. Plaat. 1997. “Kasparov versus Deep Blue: The Re-Match”, Journal of the ICCA.” Volume 20, Issue 2, pp.95-101.

A GUIDED GENETIC ALGORITHM FOR THE PLANNING IN LUNAR LANDER GAME

Zhangbo Liu
Department of Computer Science
University of British Columbia
2366 Main Mall
Vancouver, B.C, V6T 1Z4, Canada
email: zephyr@cs.ubc.ca

KEYWORDS
Guided Genetic Algorithm, Reinforcement Learning, Planning, Games

ABSTRACT

We propose a guided genetic algorithm (GA) for planning in games. In guided GA, an extra reinforcement component is inserted into the evolution procedure of GA. During each evolution procedure, the reinforcement component will simulate the execution of a series of actions of an individual before the real trial and adjust the series of actions according to the reinforcement thus try to improve the performance. We then apply it to a Lunar Lander game in which the falling lunar module needs to learn to land on a platform safely. We compare the performance of guided GA and general GA as well as Q-Learning on the game. The result shows that the guided GA could guarantee to reach the goal and achieve much higher performance than general GA and Q-Learning.

INTRODUCTION

There are two main strategies for solving reinforcement learning problems. The first is to search in the space of behaviors in order to find one that performs well in the environment. The second is to use statistical techniques and dynamic programming methods to estimate the utility of taking actions in states of the world (Kaelbling et al. 1996). Genetic algorithms (GA) and Temporal Difference (TD-based) algorithms (e.g. Q-Learning) belong to each of the two categories, respectively.

Both GA and TD-based algorithms have advantages and disadvantages. GA leads to very good exploration with its large population that can be generated within a generation but weak exploitation with elitism selection operator, because its other two operators, the crossover and mutation operators are usually randomly working. TD-based algorithms use two strategies to solve problems with continuous space which are discretization and function approximation. It usually faces the curse of dimensionality when using discretization. With function approximation it is said to be able to alleviate such a problem but might be stuck into certain local optima.

In this paper, we first investigate the GA as well as Q-Learning approach on the Lunar Lander game. Then we propose the guided GA by inserting a reinforcement component into the evolution procedure of GA. Since general GA uses random crossover and mutation operations, its performance is quite unstable. Guided GA is designed to achieve higher efficiency by involving the reward concept of reinforcement learning into general GA while keep all components of general GA unchanged so that the extension from general GA to guided GA is easy to achieve.

The remainder of this paper is organized as follows. In section 2 we introduce research work that is relevant to this paper. In section 3 we describe the Lunar Lander game as well as alternative approaches for the problem implemented with general GA and Q-Learning. In section 4 we present the guided GA approach for the problem. The results of the experiment are shown in section 5 following by the conclusions.

RELATED WORK

Reinforcement Learning for Continuous State-Action Space Problems

The issue of using reinforcement learning to solve continuous state-action space problem has been investigated by many researchers. And game is an ideal test bed.

There are a few well known benchmark problems in the reinforcement learning domain such as Mountain Car (Moore and Atkeson 1995), Cart-Pole (Barto et al. 1983) and Acrobot (Boone 1997). Another popular implementation described in (Ng et al. 2004) made an autonomous inverted helicopter flight.

Two main strategies here are discretization and function approximation. For the first strategy, discretization techniques have been widely pursued and provide convergence results and rates of convergence (Munos and Moore 2002). For the second strategy, several approaches come out on how to configure with multiple function approximators (Gaskett et al. 1999).

Reinforcement Learning + Genetic Algorithm

Some researches on combining the advantages of GA and TD-based reinforcement learning have been proposed in (Chiang et al. 1997), (Lin and Jou 1999). However, both of them use gradient decent learning method which is complex and the learning speed is always too slow to achieve the optimum solution. The idea of guided GA we propose is inspired by (Ito and Matsuno 2002), in which Q-Learning is carried out and fitness of the genes is calculated from the reinforced Q-table. However, in guided GA, instead of using Q-table, we directly insert a reinforcement component into the evolution procedure of the general GA so that the large Q-table and hidden state problem are avoided. In (Juang 2005), Juang proposed another approach to combine online clustering and Q-value based GA for reinforcement fuzzy system design. Compared with the approach described in that paper, guided GA is much simpler in structure and easier to implement while the problem we address in this paper has a higher dimension than that of in (Juang 2005), thus it has the potential to be used in game applications.

THE LUNAR LANDER GAME

The Lunar Lander Game

The Lunar Lander game is actually a physically-based problem in which the controller needs to gently guide and land a lunar module onto a small landing platform.The space is a 400 × 300 pixel rectangle area. It simulates the real environment on the moon that the lunar module has mass and is influenced by the gravity on the moon (1.63m/s^2). The controller here has 5-dimensional state spaces which are: position (x, y), velocity $(\dot{x}, \dot{y})$ and orientation (θ). The controller is able to do four actions: rotate left, rotate right, thrust and do nothing (drift).

When agent becomes the controller instead human player, the problem becomes to an advanced path finding issue. The successful landing requirement consists the checking of the following variables when any part of the lunar module reach the ground:
1) Distance from the pad
2) Speed
3) Degrees of rotation
All of them must be below certain thresholds to achieve safe landing, otherwise it will crash and the game will start from beginning again. The game runs in real time thus it is a good test bed for problems with continuous state and discrete action spaces.

Alternative Approaches

Genetic Algorithm
One alternative approach to this problem is using genetic algorithm (GA) for planning. The GA approach to this problem follows the steps below in one epoch to try to achieve the goal. First, the genome is encoded as a series of genes each of which contains an action-duration pair.The duration here represents the period of time that each specific action is applied. At the beginning, all the actions and durations in those genes in one genome are randomly assigned. A number of genomes will be created together in one generation. Next, the controller starts a trial according to the action-duration series in each genome and uses a fitness function to evaluate their utilities when they crash. There might be many approaches to build the fitness function to this problem. In (Buckland and LaMothe 2002), Buckland and LaMothe suggested the following fitness function:

$$Fitness = w_1 \cdot disFit + w_2 \cdot rotFit + w_3 \cdot airTime \quad (1)$$

where *disFit* and *rotFit* represent the value function of the position and the orientation feature separately. The *airTime* is the time period that the lunar module stays in the air which is defined as $n_a/(v+1)$ where n_a is the number of actions it does ignoring the duration and v is the velocity at landing. w_i are the weights that are applied to balance the function. If the safe landing requirement is satisfied, the fitness value will be assigned with a predefined Big_Number instead of calculating using the equation (1).
After one trial for all genomes of the current generation, the fitness value of each genome will be calculated out and the best n genomes with the highest fitness values will remain and put into the next generation. Other genomes of the next generation are created by using crossover and mutation operators. The crossover operator works by stepping through each gene in its parents' genome and swapping them at random to generate their offspring. The mutation operator runs down the length of a genome and alters the genes in both action and duration according to the mutation rate.
The operator will periodically do one epoch after another until one genome's result reaches the goal or the number of generations exceeds the predefined maximum value. An implementation of a GA solution to this problem can be found in (Buckland and LaMothe 2002).

Q-Learning
Based on our experience, Q-Learning with only discretization won't work for this problem. So we implement a linear, gradient-descent version of Watkins's Q(λ) to this problem with binary features, ε-greedy policy, and accumulating traces described in (Sutton and Barto 1998). Tile coding (Sutton and Barto 1998) is

also used to partition the continuous space into multiple tilings.

THE GUIDED GENETIC ALGORITHM APPROACH

The approaches we mentioned in the previous section both have advantages and disadvantages. The GA is simple to implement and is able to achieve the goal, while its disadvantage is that all its actions are randomly assigned so that its performance is quite unstable. The basic concept of Q-Learning approach is also simple and supposed to be efficient. However, for this game which is a realtime continuous-state problem, Q-Learning with discretization does not work and Q-Learning with function approximation is hard to accommodate. We design the guided GA which incorporates the concept of reward in Q-Learning into GA. Here we call our function "reinforcement function" because unlike the reward function in Q-Learning whose values need to be summed to calculate the Q-value ($Q = \sum rewards$), the reinforcement function here gets the immediate fitness value and will be extended to fitness function at the end of each epoch.

Reinforcement Function Design

To model the reinforcement function is a very challenging work. It has to be smoothly transformed to the fitness function of the general GA (equation (1)) at the end of each epoch so that we can easily extend the general GA to a guided GA without modifying the existing fitness function. On the other hand, it should be properly defined to efficiently guide the agents to perform better. We tried many different versions until finally reaching a solution.

In equation (1) there are 3 parameters and we need to modify two of them which are *disFit* and *airTime* in our reward function. The main difference between equation (1) and the reinforcement function is that in equation (1), all lunar modules reach the ground ($position.y = 0$) and each of them has an accumulator n_a whose value is the number of actions they do during the whole procedure; while in the reinforcement function, the lunar modules are in the air and they only focus on the next action. Based on this difference, we build our reinforcement function as follows:

We use *disFitx* to represent *disFit* in (1), then we build *disFity* which is similar to *disFitx* but for y coordinate. Then our distance function is:

$$disFit' = \sqrt{(disFitx)^2 + \frac{(disFity)^2}{w_y}} \qquad (2)$$

where w_y is used for balancing the weight between $disFitx$ and $disFity$.

airTime, as mentioned in equation (1), is defined as $n_a/(v+1)$. In our reinforcement function, n_a no longer exists, while we find that a single defined function does not work well all the time since on different stages our focuses might be different. For example, when the lunar module is high in the air we would pay more attention on its horizontal position; while when it is close to the ground it needs to slow down to prepare for landing. So instead of simply redefining it as $1/(v+1)$, we take the vertical position into consideration and come with the following definition:

* Defining $disFit'$ and $airTime'$ *\
if $position.y < h_1${
  $disFit' = disFit' \times r$;
  if $position.y < h_2$
    $airTime' = 1/(w_t \times v + 1);$}
else $airTime' = 1/(v+1)$;

where h_1 and h_2 ($h_1 > h_2$) are values of height at which we think should change our strategies and w_t is the weight that can help slow down the velocity of the lunar module to very small values when they nearly reach the ground. r is a scaling factor. Then the reward function we build is:

$$R = w_1 \cdot disFit' + w_2 \cdot rotFit + w_3 \cdot airTime' \qquad (3)$$

where w_i and $rotFit$ are the same as in (1).

Algorithm Description

In each epoch of the GA, the evolution of its genomes is done by three operators: selection, crossover and mutation. The selection is based on elitism, while the crossover and mutation are by random, which leads to the unstable performance of the general GA. In order to better perform the evolution, we insert a reinforcement component whose idea comes from the reward in Q-Learning. There are two strategies to do this. The first one is on-line updating which is similar to other reinforcement learning algorithms. The second one is off-line updating which updates the whole genome at one time before each epoch. We choose the latter based on the consideration of both the standard mechanism of GA and the real time property of the problem. The high-level description of the guided GA is shown below:

algorithm *guided genetic*;
begin
  obtain last generation;
  put a few best individuals directly into new generation;
  use crossover operator to generate new generation;
  use mutate operator on the new generation;
  evolve the new generation;
end

What we add here is the last step whose input is the mutated new generation. Below is the procedure:

procedure *evolve*;
begin

    for each individual i in the generation
        for each gene in i's action-duration series
            get duration d, current state s;
            from state s consider all possible actions a'_i
                with duration d, suppose s'_i are
                possible resulting states;
            select a' and s' based on equation (3);
            if s' satisfies safe landing requirement
                $a \leftarrow a'$;
            else if $a' \neq a$
                $a \leftarrow a'$ with probability $(1 - \varepsilon)$;
            update state;

end

where the greedy rate ε has the same meaning as the ε-greedy policy in reinforcement learning. For any given gene of an individual's genome, there are 4 possible actions and numerous durations (in our implementation for the problem the duration ranges from 1 to 30, which means for any given state there are 120 possible states in the next step). And we would only change the action in action-duration pair so that for any given state there are only 4 possible states in the next step.

We use the ε-greedy policy here, but unlike the so called greedy genetic algorithm (Ahuja et al. 1995) which focuses on greedy crossover, guided GA is inspired by (Ito and Matsuno 2002) in which the authors used Q-table to integrate Q-Learning with GA. However, for our problem using Q-table won't work because of the large state space. Instead, we use the above method to directly insert the reinforcement component into the evolution procedure without saving any previous state or function value in the memory.

EXPERIMENTAL DETAILS

Experimental Design and Results

We conducted an experiment to test the performance of our guided GA and compared it with the general GA and Q-Learning. For guided GA and general GA, we made all variables the same for both of them to ensure fairness. The parameters of our experiment were given as follows:

1. Both of the two contained 100 individuals in one generation. The maximum number of generations was 500. It supposes to be failed if it did not achieve the goal within 500 generations and then would start from the beginning again. The length of chromosome was 50. The crossover rate was 0.7 and the mutation rate was 0.05. The ε was 0.1.

2. The thresholds for the safe landing requirements were:

   (a) Distance = 10.0

   (b) Velocity = 0.5

   (c) Rotation = $\pi/16$

3. To define the values of weights was the most difficult work for the experiment. Below are the best value settings that were selected by empirical study:

   (a) $w_1 = 1, w_2 = 400, w_3 = 4$ (got from (Buckland and LaMothe 2002))

   (b) $w_y = 3, w_t = 6, r = 1.7, h_1 = 100, h_2 = 30$

We also introduced the same feature of guided GA to Q-Learning implementation for building its reward function. To learn to solve a problem by reinforcement learning, the learning agent must achieve the goal (by trial-and-error) at least once (Lin 1992). Testing results showed that general and guided GA were able to achieve the goal almost every time. However, it was very hard for Q-Learning to complete the task. Besides the general reasons such as function approximation strategy often falls into local optimal and Q-Learning converges too slowly, we believed that another important reason was in this realtime problem the control of duration of the action is crucial. GAs could evolve the durations with the crossover and mutation operation. But Q-Learning could not. Adding duration together with action into the state space might make the state space extremely huge, thus lead to Q-Learning's fail. Based on this fact, we only compared the data we got from the testings using general GA and guided GA. The experimental results that we ran both of general and guided GA for 25 trials are shown in Figure 1.

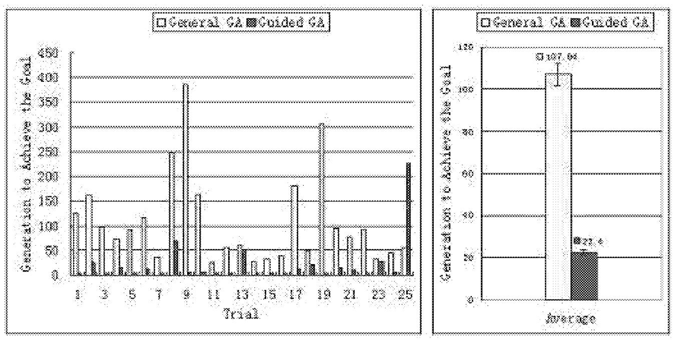


Figure 1: Experimental Results

From the results we can observe that for most of the time the performance of the guided GA were much higher than the general GA except the last trial. Figure 2 shows the fitness that both of them gained during all the generations before the last generation in the 13th trial. According to the data, both the highest and the average fitness of guided GA were higher than general GA.

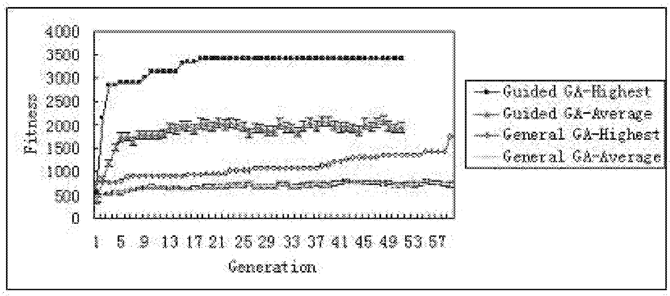


Figure 2: Fitness Gained in the 13th Trial

Analysis

Some questions came out when we observed the data of the results. First, what was the goal's fitness/reinforcement value? Second, why the highest fitness of guided GA was much higher than that of general GA while they achieved the goal in very close steps? Third, why guided GA lost in the last trial while performed much better in previous trials?

First, we used the thresholds for the problem to calculate out the fitness value and found that the fitness value of the goal was just no more than 900. The reason why those individuals with very high fitness values failed to achieve the goal was that there were three parameters in the fitness/reinforcement function. No matter how high the fitness value that certain individual gained, as long as there was one parameter whose value was above the threshold then it failed to achieve the goal. So it was possible that one individual with a low fitness achieved the goal in the next generation by randomly evolving its genome which accidentally hit all the thresholds and triggered a sudden success. And that was the reason that sometimes the individual who achieved the goal was not the one who performed the best in the previous state. Second, both general and guided GA involved randomness that brought the uncertainty to the procedure. So the possible explanation to the third question was that the randomness caused a sudden success to the general GA before the guided GA got out of certain local optimal states. Third, although the highest fitness in each step did not make much sense to us, the average fitness were useful because higher average fitness demonstrated a better chance for the whole generation to achieve the goal. For all the trials we observed, the average fitness of guided GA were much higher than the average fitness of general GA.

CONCLUSION

In this paper we proposed a guided genetic algorithm by adding a reinforcement component into GA. We successfully applied the guided GA for the planning of Lunar Lander game. Based on the experimental results, guided GA achieved much higher performance than general GA and Q-Learning.

REFERENCES

Ahuja, R.; J. Orlin, and A. Tivari. 1995. A greedy genetic algorithm for the quadratic assignment problem. Working paper 3826-95, Sloan School of Management, MIT.

Barto, A.G.; R.S. Sutton, and C.W Anderson. 1983. Neurolike adaptive elements that can learn difficult control problems. *IEEE. Trans. on System Man and Cybernetics.*

Boone, G. 1997. Minimum-time control of the acrobot. *International Conference on Robotics and Automation.*

Buckland, M. and A. LaMothe. 2002 *AI techniques for game programming.* Premier Press.

Chiang, C.K.; H. Y. Chung, and J. J. Lin. 1997. A self-learning fuzzy logic controller using genetic algorithms with reinforcements. *IEEE Transactions on Fuzzy Systems.*

Gaskett, C.; D. Wettergreen, and A. Zelinsky. 1999. Q-learning in continuous state and action spaces. In *Australian Joint Conference on Artificial Intelligence*, pages 417–428.

Ito, K. and F. Matsuno. 2002. A study of reinforcement learning for the robot with many degrees of freedom - acquisition of locomotion patterns for multi-legged robot. In *ICRA '02. IEEE International Conference on Robotics and Automation*, pages (4):3392–3397.

Juang, C.F. 2005. Combination of online clustering and q-value based ga for reinforcement fuzzy system design. *IEEE Transaction on Fuzzy Systems.*

Kaelbling, L.P.; M.L. Littman, and A.P. Moore. 1996. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285.

Lin, C.T. and C.P. Jou. 1999. Controlling chaos by ga-based reinforcement learning neural network. *IEEE Transaction on Neural Networks.*

Lin, L.J. 1992. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Mach. Learn.*, 8(3-4):293–321.

Moore, A.W. and C.G. Atkeson. 1995. The parti-game algorithm for variable resolution reinforcement learning in multidimensional state-spaces. *Mach. Learn.*, 21(3):199–233.

Munos, R. and Andrew Moore. 2002. Variable resolution discretization in optimal control. *Mach. Learn.*, 49(2-3): 291–323.

Ng, A.Y.; A. Coates, M. Diel, V. Ganapathi, J. Schulte, B. Tse, E. Berger, and E. Liang. 2004. Inverted autonomous helicopter flight via reinforcement learning. *International Symposium on Experimental Robotics.*

Sutton, R.S. and A.G. Barto. 1998. *Reinforcement Learning: An Introduction.* Number 206-214. MIT Press, Cambridge, MA.

A FIRST LOOK AT BUILD-ORDER OPTIMIZATION IN REAL-TIME STRATEGY GAMES

Alex Kovarsky and Michael Buro
Department of Computing Science
University of Alberta, Edmonton, Alberta, Canada
email: kovarsky|mburo@cs.ualberta.ca

KEYWORDS

planning, real-time strategy games, concurrent actions

ABSTRACT

Planning is an important cognitive process. We are constantly involved in planning even when executing seemingly simple tasks such as driving to school. The ability to plan becomes essential in unfamiliar environments where we cannot rely on previously learned action sequences, but instead have to generate novel solutions by considering consequences of action choices on the fly. Planning in real-world domains poses a big challenge to current AI systems because of inadequate abstraction and search mechanisms. Human planning ability is therefore still considered superior. A good example is modern video games in which it is apparent that computer characters lack human planning, learning, and reasoning abilities. In this paper we approach the real-time planning problem by considering build-order optimization in real-time strategy games. This problem class can be formulated as a resource accumulation and allocation problem where an agent has to decide what objects to produce at what time in order to meet one of two goals: either maximizing the number of produced objects in a given time period or producing a certain number of objects as fast as possible. We identify challenges related to this planning problem, namely creating and destroying objects, concurrent action generation and execution, and present ideas how to address them.

INTRODUCTION

Planning refers to finding an action sequence that achieves a given goal. People are involved in planning for a variety of tasks in their everyday lives. Most of the time we do not realize that we are actually executing plans, since for most behaviors we use memorized action sequences. For example, we may have memorized a plan to buy groceries which could look as follows: make a list of what you need to buy, drive to the grocery store, buy what is on the list, drive home, and unload the groceries. However, what happens when one of the links in the above chain is broken? If the car does not start, we will have to plan how to achieve the goal of getting the groceries another way. We have many options of getting to the grocery store, including taking a bus, walking, fixing the car, and in turn each of those options requires planning. If a person has never taken a bus to the store before, he will need to find out about the bus's route, schedule, and fare, etc. Thus, in real-life we become engaged in planning when we are facing new situations to which we have no immediate solutions.

Today, most video game AI systems are designed as complex rule-based scripted systems. Specifically, game designers try to foresee with every possible scenario that AI characters can encounter and then write a script of actions for that situation. Such systems are difficult to maintain and expand. Moreover, it is impossible to predict and effectively plan for every potential future state. Most games do not have capabilities to handle unfamiliar situations. In such cases the AI behavior is usually ineffective and predictable. We believe that adding real-time planning capabilities to computer games will result in improved AI behavior. Recently, the commercial success of the action game F.E.A.R. (Orkin, 2005) shows the positive effects automated planning can have on the performance of AI characters. In F.E.A.R., unlike other action games, the AI system performs real-time planning that allows computer characters to adapt their behavior to the current situation.

The focus of our research is creating AI systems for real-time strategy (RTS) games such as Starcraft and Age of Empires. The common objective in RTS games is to eliminate other players through military superiority. Players first instruct workers to gather resources, then use those resources to build more workers and structures that can create military units. These are then sent to battle the enemy in real-time.

Research in RTS game AI can be divided into two branches: higher-level and lower-level AI. Higher-level AI refers to management of resources, decisions on what to build, and strategic decisions such as sending units into battle. Lower-level AI refers to the behaviors of single units and small groups of units that are given commands.

Today, most commercial RTS games are designed as rule-based systems with limited planning capabilities. However, some AI techniques such as influence maps and terrain analysis to deal with new maps have been implemented in commercial RTS games (Pottinger, 2000). Progress has also been made in wall-building (Reid and Davis, 2006), pathfinding (Sturtevant and Buro, 2005) (Demyen and Buro, 2006), and local combat (Kovarsky and Buro, 2005). The improvement of lower-level AI modules is important because without effective solutions in this area, research on higher-level reasoning and planning cannot proceed. Currently, because of the significant advances in lower-level AI the missing link for a complete RTS game AI system is a higher-level AI module responsible for global planning.

Such module could be looked at as an "all knowing" general that makes decisions affecting all global aspects of an RTS game, including resource collection, building decisions, and military expansion. Those general decisions and commands are then passed to lower-level modules which are responsible for the implementation details.

Our focus is build-order optimization in RTS games. We aim to optimize the gathering of resources and the creation of buildings and units in the initial stage of RTS games. In this research we only consider actions of our units, since in the initial game phase there is no or little interaction with the opponent. Our planning domain is defined by a technology tree that specifies the relationships between units, buildings, and resources. For example, in order to build a factory we require a worker, barracks, and sufficient resources. We consider two types of optimization problems: minimizing the time to achieve a certain goal, such as creating 2 tanks and 5 marines, or maximizing the amount of a resource or unit type in a specified time, e.g. gathering the maximum amount of iron within 10 minutes).

The build-order optimization problem presents several challenges that have not been addressed previously in planning research. One problem is the creation of new objects which can act in the world. Another is the concurrent execution of actions in RTS games, which leads to problems such as determining whether a given set of actions is executable concurrently and efficiently generating concurrent action sets.

In the remainder of the paper we first formulate a build-order optimization problem in the planning domain definition language (PDDL, (McDermott and Committee, 1998)). We then discuss the limitations of PDDL with regard to our planning domain. Subsequently, we describe the new challenges we face in build-order optimization in RTS games and then we discuss several approaches to address the unique challenges of RTS games.

```
(define (domain build-order)
  (:types worker building)
  (:predicates
    (canProduceWorkers ?b - building)
    (canProduceMarines ?b - building)
    (activatedW ?x - worker)
    (busyW ?x - worker) )
  (:functions
    (amount-of-resources)
    (num-marines) )

  (:durative-action create-worker
     :parameters (?x - worker   ?b - building)
     :duration (= ?duration 1)
     :condition (and (at start (>= (amount-of-resources) 50))
                     (at start (canProduceWorkers ?b))
                     (over all (canProduceWorkers ?b))
                     (at end (canProduceWorkers ?b)))
     :effect (and (at start (decrease (amount-of-resources) 50))
                  (at end (activatedW ?x))

  (:durative-action create-marine
     :parameters (?b - building)
     :duration (= ?duration 2)
     :condition (and (at start (>= (amount-of-resources) 100))
                     (at start (canProduceMarines ?b))
                     (over all (canProduceMarines ?b))
                     (at end (canProduceMarines ?b)))
     :effect (and (at start (decrease (amount-of-resources) 100))
                  (at end (increase (num-marines) 1))

  (:durative-action gather-resource
     :parameters (?x - worker)
     :duration (= ?duration 5)
     :condition ((at start (at ?x ?l))
                 (at start (not (busyW ?x)))
     :effect ((at end (increase (amount-of-resources) 100))
              (at start (busyW ?x))
              (at end (not (busyW ?x)))))
```

Figure 1: Simple build-order domain Specification in PDDL with three actions: create-worker, create-marine, and gather-resource.

BUILD-ORDER OPTIMIZATION AND PDDL

Research on automated planning has mostly concentrated on classical planning, which in short can be summarized as planning without regard to time. In such planners a plan is an ordered sequence of actions, actions are instantaneous and do not interact with each other. However, in real life very few domains adhere to such restrictions. Therefore, in recent years temporal planning has gained attention. Temporal planners take time into account. Each action requires a certain time to execute and in certain situations several actions are allowed to execute concurrently.

The development of PDDL and subsequently the PDDL 2.1 (Fox and Long, 2003) extension, that incorporates time in its semantics, have further facilitated the research for creating temporal planners. However, the PDDL 2.1 semantics is still too restrictive for our RTS domain. But since PDDL has become a standard in the planning research community, we would like to stay compatible with PDDL.

PDDL was developed to standardize planning domain and problem description in order to enable different planners to compete against one another in interna-

```
(define (problem SimpleBuildOrder)
  (:domain build-order)
  (:objects
     commandCentre barracks - building
     worker1 worker2 worker3 worker4 - worker )
  (:init
     (not (activated worker1))
     (not (activated worker2))
     (not (activated worker3))
     (not (activated worker4))
     (canBuildMarines barracks)
     (canProduceWorkers commandCentre)
     (= (num-marines) 0)
     (= (amount-of-resources) 700) )
  (:goal (and (>= (num-marines) 5)))
  (:metric minimize (total-time)))
```

Figure 2: Build-order problem specification in PDDL. Starting with two buildings, the goal is to create 5 marines as fast as possible using at most four workers.

tional planning competitions. PDDL supports the syntax of STRIPS, ADL, and some other previously used planning languages. A PDDL definition of a planning problem consists of two parts: the domain definition and the problem definition. The domain definition file is where the types, predicates, functions and actions are defined, while the problem definition file is where the objects of types defined in the domain files are declared, the predicates and functions acting on objects are initialized, and the goal conditions for the plans are specified. The following example shows a simplified PDDL RTS domain and problem files.

In the domain file in Figure 1 there are two types (worker and building), four predicates, and two numerical functions. Functions in this context are used to store numerical fluents such as resource amounts. The two actions are temporal actions for which the duration clause specifies the number of steps required for the action to complete. In the condition clause the preconditions for triggering the action are specified. The "at start", "over all" and "at end" expressions specify when a given condition has to hold (i.e. at the beginning of action execution, during, or at the end, respectively). In the effect clause, the effects of a given action are specified. The effect on functions is a numerical change of functions values (increase or decrease), while the effect on predicates is Boolean, i.e. a predicate for a given object or objects can be set to true (e.g. ActivatedW ?x), or to false (not (ActivatedW ?x).

The first action in Figure 1 is create-worker. In order for this action to execute the amount-of-resources value has to be at least 50 and the building has to be able to produce workers (i.e. CanProduceWorkers is true). The effect of this action is a reduction in resources and the activation of a worker (i.e. activatedW is true). The second action in Figure 1 is create-marine. Similarly, in order for it to execute there must be sufficient resources and the building has to be able to produce marines. The effect is a reduction in resources and numerical increase in the number of marines. Unlike workers, marines are not objects. They are modeled numerically, like resources. We do not have to model them as objects since in the domain in Figure 1 they are not involved in any actions. Workers have to be created as objects since workers are involved in the gather-resource action.

In the problem specification in Figure 2 two objects of type building (i.e. commandCentre and barracks) and four objects of type worker are declared. These objects as well as functions are initialized in the :init clause. The worker type objects are set to not activated (since they are not built yet) and the barracks and commandCentre objects are enabled to build marines and workers, respectively. The goal (:goal) of this plan specification is to increase the number of marines to 5 and to do so in a minimal possible time (:metric).

PLANNING IN RTS ENVIRONMENTS

PDDL is quickly becoming the standard input language for planners (Kautz and Selman, 1999) (Hsu et al., 2006). The International Planning Competition (IPC) is an annual event, which is run in conjunction with the ICAPS conference. Currently, IPC is the key test-bed for both classical and temporal planners. However, even the newest extension of PDDL fails to address some of the challenges present in RTS games. First, we divide the issues arising in build-order optimization into two parts: object creation and destruction and concurrent action execution. We then discuss each of the problems, describe the restrictions of the PDDL semantics with respect to each problem and propose ways to relax these restrictions.

Object Creation and Destruction

The typical problems that PDDL aims to address and that are used in the international planning competition are so called closed problems in which the number of objects in the world remains constant. Even the most recent version of PDDL does not allow for the creation or deletion of objects. In RTS games, however, creating objects is key. A typical game starts with a small number of units and a limited amount of resources. Then, those units can create structures that can produce new units, mine resources, or perform other functions. Another important aspect of RTS games is combat. As the RTS game progresses, military units start fighting with their opponents. As a result some units or structures can be destroyed. Currently, PDDL does not provide means for object destruction. Even though no explicit mechanisms for the creation and the destruction of objects exist there is a way to implicitly simulate object creation and destruction in PDDL. All objects that can potentially be created in the future have to be specified in advance in the problem file. Creation and destruction of an object in this setting means switching this object on and off by using a predicate (for example activate (object name)). Refer to Figure 1, where a worker is created by activating (activatedW ?x) a previously defined (Figure 2) object (e.g. worker1) from the problem definition file. This approach is computationally inefficient and awkward to implement. Since all objects have to be specified in advance, at every point the planner has to examine each of the objects when generating possible actions. This examination is inefficient, since in the beginning of the planning process most of the objects are not active. Furthermore, in some scenarios it is difficult to predict the maximum number of objects that are needed in advance. Pre-declaring a large number of objects will lead to the above-mentioned inefficiency, while declaring too few objects might result in a shortage of objects (and possibly in an inability to achieve a given goal). The problem of object creation and destruction should be straightforward to address. We can add an explicit object CREATE and DESTROY capability to PDDL. This can be done by adding a new effect to any given action that will create or destroy a certain object by adding or removing that object from the list of objects. For example, in the domain specification in Figure 1 instead of activatedW worker, we would have a new object create clause. In Figure 2 we will no longer have to specify the worker objects in advance.

Concurrent Action Execution

RTS games are inherently concurrent environments. In many situations objects can execute their actions simultaneously. Thus, when we have a set of actions, we need to determine whether such a set is executable concurrently. Concurrent executability depends on interdependence among actions. If all actions in a set are independent of each other, then they all can be executed concurrently. Sometimes, however, actions in RTS games are dependent. For example, building actions usually require resources, while resource gathering actions produce resources (e.g. create-marine and gather-resource, respectively). Whether a set consisting of resource producing and consuming actions is executable simultaneously depends on the accumulated resource amount, the amount currently produced, and the amount being consumed. Such computation is non-trivial in general. In recent years, some progress has been made on concurrently executing actions with shared resources (or numeric fluents) (Lee and Lifschitz, 2001) (Erdem and Gabaldon, 2005). This research has concentrated on formalizing the semantics for such actions, without much emphasis on the computational effort of generating concurrent action sets and determining concurrent action executability. Another challenge is generating all sets of concurrent actions efficiently given all actions that can be executed individually.

To understand the computational effort required for determining whether a set of concurrent actions is executable, first we need to make a distinction. Two types of concurrent actions are possible: serializable and non-serializable. By serializable we refer to the sets of actions, which when executed serially, one after another, have the same result as when executed concurrently. Non-serializable actions produce different effects. For example, two units may be required to lift an object or two actions are interlocked in such a way that both preconditions require the effect of the other action to be true. For typical RTS game build-order optimization problems considering serializable actions is sufficient.

A second distinction relates to the action execution environment. In the first case one could demand that for any chosen execution sequence the preconditions of all actions in the sequence is met. Alternatively, a set could be called serializable if only one such sequence exists. This approach can lead to faster plans, since more sequences will be declared serializable. However, the first condition is more robust, since in some environments there is no way to ensure that a set of actions will be executed in a given order. This is especially true for RTS game engines.

PDDL only allows for limited concurrency. No two or more actions can simultaneously use a given resource if at least one of the actions is changing its value. In PDDL actions can be executed concurrently as long as they are independent from each other. This is the so-called “no moving targets” rule which is very conservative and will prevent resource related actions from being executed simultaneously in our RTS game environment. For instance, in Figure 1, actions create-worker and create-marine consume the same type of resource. Assuming that their preconditions are satisfied (i.e. there are enough resources to create both unit types), these two actions will not be allowed to execute concurrently in PDDL because they both modify the same resource. Thus, in order for PDDL to work properly in our domain we need to remove the “no moving targets” restriction. This relaxation allows for dependent concurrent actions but also increases the computational effort for checking concurrent executability of an action set and for generating concurrent sets of actions.

Checking Concurrent Executability

One way of reducing the computational effort of planning in RTS environments is through examining the types of actions required (i.e. the expressiveness of their preconditions and effects) for a typical problem specification. By restricting the complexity of preconditions and effects we can greatly improve the runtime speed of determining whether a given set of actions is executable concurrently. We have started to look at constructing a hierarchy of precondition and effect restrictions with growing expressiveness that still allows us to decide robust serializability quickly. The first level of the hierarchy is when all actions are independent of each other.

Therefore, all actions can be executed concurrently if all the preconditions hold. At the second level the actions are no longer independent of each other. In the RTS domain this means that two or more actions can increase and decrease the value of a single resource. At this level we restrict the preconditions to having only logical operators (i.e. no arithmetic operators) and effects to having only commutative operations (i.e. increase and decrease but not set). Here the runtime effort will partly depend on the available amount of the shared resource. At level three of the hierarchy we will allow for a number of resources to be shared between actions. At the next level we will increase the expressiveness once more by allowing preconditions to contain arithmetic operators. Finally, we will examine the general case in which effects are not commutative.

Generation of Action Sets

Another challenge is to generate all possible sets of concurrent actions efficiently. Given the set of actions executable at a certain time point, we need to generate potential sets of such actions efficiently. We should aim towards complexity that is linear with respect to the number of generated sets of actions. In general, given n possible actions there are 2^n ways to choose sets of actions of arbitrary length. Such computation is infeasible in a real-time environment. Thus, suitable approximations need to be found. We could allow objects to execute at most one action at a time. The number of generated action sets will then be reduced since actions performed by the same object can be grouped together and will be left out from further consideration once a single action from that group is selected in a potential action set. Another way to decrease the number of potential action sets is through abstraction. The main challenge for planning in RTS games is that the game is unbounded — meaning the number of units (i.e. workers, buildings, military units) generally grows as the game progresses. As the number of units reaches a certain point, generating all potential action sets will become infeasible in real-time. In most RTS games, however, several units are often assigned to perform similar tasks. Thus, one reasonable approach is to group units by their type into groups, which will be treated as "super-units" and only execute identical tasks during the planning process. Such groups can be flexible. That is, new objects can be added and removed to groups and groups themselves can be merged to form larger groups, or split into smaller groups. How to do such grouping to maximize the use of available computational resources is an interesting research issue by itself. Yet another way of decreasing the number of actions is forcing units to perform a given task for a certain minimum number of time steps. Again this will result in a lower number of units available at a given time and will reduce the number of potential action sets, at the cost of optimality.

CONCLUSION

In this paper we have introduced the build-order optimization problem for real-time strategy games and discussed the following core challenges for creating an automated planning system for this domain: how to deal with object creation and destruction in PDDL, how to decide what actions can be executed simultaneously, and how to generate action sequences efficiently? We have presented initial research ideas on how to tackle these problems. Our hope is that these become the seed for a high-performance RTS game planning system that can be used to aid human players and to improve the playing strength of computer opponents.

ACKNOWLEDGMENTS

Financial support was provided by the Natural Sciences and Engineering Research Council of Canada (NSERC).

REFERENCES

Demyen D. and Buro M., 2006. *Efficient Triangulation-Based Pathfinding.* In *Proceedings of the AAAI Conference.* Boston, 942–947.

Erdem E. and Gabaldon A., 2005. *Cumulative effects of concurrent actions on numeric-valued fluents.* In *Proceedings of the AAAI Conference.* Marina del Ray, 627–632.

Fox M. and Long D., 2003. *PDDL2.1: An extension to PDDL for expressing temporal planning domains. Journal of Artificial Intelligence.*

Hsu W.; Wah B.; Huang R.; and Chen Y., 2006. *Handling Soft Constraints and Preferences in SGPlan.* In *Proceedings of ICAPS Workshop on Preferences and Soft Constraints in Planning.*

Kautz H. and Selman B., 1999. *Unifying SAT-based and Graph-based Planning.* In *Proceedings of International Joint Conference on Artificial Intelligence.* Stockholm.

Kovarsky A. and Buro M., 2005. *Heuristic Search Applied to Abstract Combat Games.* In *Proceedings of the Eighteenth Canadian Conference on Artificial Intelligence.* Victoria.

Lee J. and Lifschitz V., 2001. *Additive Fluents.* In *Proceedings of the AAAI Spring Symposium: Answer Set Programming: Towards Efficient and Scalable Knowledge Representation and Reasoning.* 116–123.

McDermott D. and Committee A., 1998. *PDDL – the planning domain definition language.* In *Technical Report.*

Orkin J., 2005. *Agent Architecture Considerations for Real-Time Planning in Games.* In *Proceedings of the AIIDE Conference.*

Pottinger D., 2000. *Terrain Analysis in Realtime Strategy Games.* In *Computer Game Developers Conference.*

Reid T. and Davis I., 2006. *AI Base Building in Empire Earth II.* In *Proceedings of the AIIDE Conference.* Marina del Rey, CA.

Sturtevant N. and Buro M., 2005. *Partial Pathfinding Using Map Abstraction and Refinement.* In *Proceedings of the AAAI Conference.* Pittsburgh, 1392–1397.

GAMING
WITH
ROBOTS

ROBOT SOCCER STRATEGY DESCRIBED BY TEXT FILE

Václav Snášel, Jan Martinovič, Pavel Krömer, Bohumil Horák
VŠB - Technical University of Ostrava,
Faculty of Electrical Engineering and Computer Science,
Department of Computer Science and
Department of Measurement and Control,
Ostrava, Czech Republic

KEYWORDS

Robotic soccer game, Strategy, Evolution strategy.

ABSTRACT

This paper deals with method of representing robotic soccer game in a simulated and/or real form. This representation is used for controlling robots playing soccer. Our approach to robot soccer is to view it as a local interaction game. We describe our concept of virtual grid and implementation of robot soccer simulator in this paper, as tools for building the strategy and tactical movement database for real game. Strategy learning from game observation is important for discovering strategies of the opponent team and searching of tactical movements groups replaying, simulation and synthesis of anti-strategies.

INTRODUCTION

The typical example of distributed control system with embedded subsystems is the task of controlling physical robots playing soccer. The selection of this game as a laboratory task was motivated by the fact that the realization of this complicated multidisciplinary task is very hard. The task can be divided into a number of partial tasks (evaluation of visual information, image processing, hardware and software implementation of distributed control system, wireless data transmission, information processing, strategy planning and controlling of robots). The task is attractive both for students and teachers, and allows direct evaluation and comparison of various approaches. For the improvement of the game strategy, we develop an abstract description of the game and propose how to use this description for e.g. learning of rules. We also take inspiration from the ant-like systems that reduce the need of complexity of individual robots and lead to robust, scalable systems [?, ?, ?, ?]. We build on our previous work - the hardware implementation and basic control of robots - and we would like to achieve higher level control of the game strategy.

The rest of the paper is organized as follows: First we briefly describe the base hardware and software implementation. Then we describe the representation of the game field using virtual grids. Then we describe possible game strategies. Using the virtual grids and game strategies, we show how to learn rules that describe particular game strategy. Particular attention is paid to the learning using latent semantic analysis. We conclude with the discussion of the presented approach.

The game system can be described as up to twice eleven autonomous mobile robots (home and visiting players), which are situated at the field of the size of $280 \times 220cm$. In our approach we are using software simulator for robot soccer game [?]. Out software part is implemented by decision making and executive agents. The agents corresponding to individual robots are controlled by a higher level agent [?, ?, ?, ?]. The coordinates are saved in the scene database [?], which is common for all agents. Both agent teams have a common goal to score the goal and not to get any goal. For a success, it is also important to extract the strategy of the opponent team. The extraction and knowledge of opponent game strategy is an approach that is known to be successful in other situations as well [?].

The game can be represented as a trajectory in what we call the virtual grid. The virtual grid generally allows us to reduce data volume for easy description of player motion and subsequently for controlling the game or for learning game strategies. The data volume of the description using the virtual grid is obviously smaller than the description using natural coordinates. The exact values depend on the frequency of samples and on the maximal velocity of the mobile robot movement at game field.

GAME STRATEGY

The game strategy can be dynamically changed based on the game progress (i.e. the history and the current position of the players and the ball [?]). The game progress can be divided in time into the following three ground playing classes (GPC):

- GPC of game opening (GPCO)
- GPC of movements in game site (GPCS)
- GPC of game end (GPCE)

The game progress, especially in the GPCS class, can be also divided into the following two game playing situations (GPS):

- GPS of attack (GPSA). The interactions of simple behaviours cause the robots to fall into a V-formation where the ball is in motion roughly towards the opponents goal.
- GPS of defense (GPSD). When the ball is not moving roughly towards the opponents goal, the robots move around it to form an effective barrier and to be in a good position for recovery.

Each GPC has its own movement rules. The classes GPCO and GPCE consist of finite number of possible movements that are determined by initial positions of players and the ball. The class GPCS has virtually unlimited number of possible movements. The movements are determined by the current game situation (GPS) and by the appropriate global game strategy (in next GGS). The movement of the particular robot is determined by the current game class and situation, and also by the robot role. For example, the goalkeepers task is to prevent the opponent to score a goal. His movements are in most cases limited along the goalmouth near of goal line. The preferred movements are in goal line direction. The preference of these movements comes from the particular GGS, where the goalkeeper prevents to score a goal in the way of moving in the position between the central goal point and the ball (or the expected ball position). The preference of other movement directions is created using GPSA, where the movements of goalkeeper secure kicking the ball from the defense zone.

BASIC DESCRIPTION OF STRATEGY SELECTION PROCESS

In this section we describe our approach for learning game strategy from observation. Our goal is to learn an abstract strategy. The main steps of the learning process are:

- Transformation of observations into virtual grids.
- Transformation of observations into strategy grids.
- Learning a strategy based on the observed transitions in the strategy grid.

We adopt definition of strategy [?]: Strategy is the direction and scope of an organization over the long-term: which achieves advantage for the organization through its configuration of resources within a challenging environment...

Strategy application for one movement of players is computed in following steps:

- Get coordinates of players and ball from camera
- Convert coordinates of players into strategic grid
- Convert ball and opponents' positions into virtual and strategic grids
- Choose goalkeeper and attacker, exclude them from strategy and calculate their exact positions.
- Detect strategic rule from opponents' and ball positions
- Convert movement from strategic grid to physical coordinates
- Send movement coordinates to robots

Each strategy is stored in one file and currently consists of about 15 basic rules.

```
.Strategy "test"
.Algorithm "Offensive"
.Author "Vaclav Snasel"
.Date "10.11.2006"
.Size 11 9
.PriorityMine      100 100 100 100 100
.PriorityOpponent  50   50  50  50  50
.PriorityBall      50

.Rule  1  "Attack1"
.Mine a6 c7 d6 e3 f9
.Opponent d3 e7 e8 g2 k6
.Ball i6
.Move a6 g7 f5 j3 i8

.Rule  2  "Attack2"
.Mine a6 c7 d6 e3 f9
.Opponent d3 e7 e8 g2 k6
.Ball i5
.Move a6 g7 g5 h3 h8

...
```

Furthermore the file contains following metadata:

- Information about the name of strategy
- The algorithm to strategy choosing
- The author responsible for current strategy
- The date of last modification
- The size of strategic grid
- Strategic rules

Each strategic rule consists of five records:

- The rule ID and description (e.g. Rule 1 "Attack1"),
- the coordinates of our players in strategic grid (e.g. .Mine a6 c7 d6 e3 f9),
- the coordinates of opponent's players in strategic or virtual grid (e.g. .Opponent d3 e7 e8 g2 k6),
- the ball coordinates in virtual or strategic grid (e.g. .Ball i6)
- strategic or virtual grid positions of the move (e.g. .Move a6 g7 f5 j3 i8).

```
// algorithm for rule selection
// Game.Mine      -- actual positions
// Game.Opponent  -- actual positions
// Game.Ball      -- actual position

maxWeight = 0
selectRule = 0

foreach r in Rule
{
  weight = 0
  ruleTmp = r.Mine
  foreach p in Game.Mine
  {
    s = nearest position in ruleTmp to p
    w = 1 / (distance(s, p) + 1)
    w *= Strategy.PriorityMine
    weight += w
    remove s from ruleTmp
```

```
  }

  ruleTmp = r.Opponent
  foreach p in Game.Opponent
  {
    s = nearest position in ruleTmp to p
    w = 1 / (distance(s, p) + 1)
    w *= Strategy.PriorityOpponent
    weight += w
    remove s from ruleTmp
  }

  w = 1 / (distance(Game.Ball, r.Ball) + 1)
  w *= Strategy.PriorityBall
  weight +=  w

  if weight > maxWeight
  {
    maxWeight = weight
    selectRule = r
  }
}
return SelectRule
```

From observation of opponent's strategy a new set of rules can be written, without necessity of program code modification. Furthermore, there is a possibility of automatic strategy (movement) extraction from running game.

There exist two main criteria in the Rule selection process. The selection depends on opponents' coordinates, mines' coordinates and ball position. The strategy file contains rules, describing three possible formations suggesting danger of current game situation. The opponent's team could be in offensive, neutral or defensive formations. Furthermore, we need to weigh up the ball position risk. Generally, opponent is not dangerous if the ball is near his goal. The chosen rule has minimal strategic grid distance from current

Optimal movements of our robots are calculated by applying minimal distance from strategic grid position. The goalkeeper and attacking player, whose distance is closest to the ball are excluded from strategic movement and their new position is calculated in exact coordinates.

To summarize, the strategy management can be described in the following way:

- Based on incoming data from the vision system, calculate virtual and strategy grid Coordinates of the players and the ball.
- The virtual grid is then used to decide which player has under the ball control.
- This player is issued a kick to command that means that it has to try to kick the ball to a given strategy grid coordinates.
- All other players are given (imprecise) go to coordinates. These coordinates are determined by the current game strategy and are determined for each robot individually. The goalkeeper is excluded from this process since its job is specialized, and does not directly depend on the current game strategy.

EVOLUTIONARY STRATEGY OPTIMIZATION AND VERIFICATION PROPOSAL

In previous section was identified the need to learn opponents strategy from the game and resolve appropriate anti-strategy in response. Also the verification of created strategy is of notable importance. An off-line (out of the gameplay) verification process validating the strategy and ensuring that there are no:

- Contradictory rules, leading to contradictory game situations
- Extra rules, describing immediately or in more steps the same game situations

Such a verified game strategy could provide better and more complex goal-targeted robot behavior (literally).

Genetic Algorithms are powerful and popular optimization and search algorithms inspired by natural evolution introduced by John Holland and extended by David Goldberg. GA are wide applied and highly

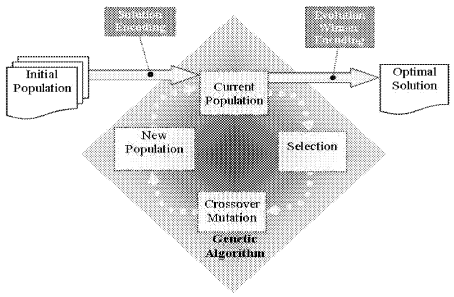


Figure 1: Evolutionary algorithm

successful variant of evolutionary computation [?]. GAs operate over a population of potential solutions encoded into chromosomes. Each chromosome is rewarded with a fitness value expressing its suitability as a solution of given problem. The workflow of GA consists of iterative application of genetic operators on population of chromosomes (see figure ??). Genetic operators are [?]:

- Selection operator: to select the fittest chromosomes from the population to be parents. Through this operator, selection pressure is applied in the population.
- Crossover operator: for varying chromosomes from one population to the next by exchanging one or more of their subparts.
- Mutation operator: random perturbation in chromosome structure; used for changing chromosomes randomly and introducing new genetic material into the population.

A population of chromosomes in particular iteration of evolutionary computation is called generation. When applying evolutionary optimization on robot soccer game, a strategy can be naturally seen as generation of rules encoded as binary chromosomes. The evolution over strategy rules should lead to new improved set of rules that will form better anti-strategy against particular opponent. The challenge of this approach is to find suitable rule encoding (that will allow easy application of genetic operators and respect the nature of investigated problem) and discover useful fitness function to judge the rules against each other.

FUTURE DIRECTIONS

Introduced approach to robot soccer strategy is based on the overall knowledge of global game situation. Next moves of robot players are resolved after an analysis of overall positions of all players and ball. In fact robots are rather being moved (by an omniscient operator) than move independently. More advanced multi agent approach should next to goal targeted strategic manners incorporate also agent behavior based on artificial life methods like ant algorithm and stigmery, a multi-agent coordination and control using techniques inspired by the behavior of social insects. In such a game will the robot players solve simple or trivial gameplay situations according to their own decisions and local knowledge following their individual simple behavior patterns. The patterns could be as follows:

- The attacker is attracted to the ball (when the ball is moving towards opponents goal)
- The attacker with ball is attracted to competitors goal
- The defender is attracted to the ball (when it is moving towards own goal). The defender is extremely attracted to the ball when there is opponents attacker (and no own player is close).
- The goalie is attracted to the ball but also to its goal area (so he goes after ball only in his goal area).

The strategy concept, presented in previous sections, will then be used to solve complex non-trivial game situations (like standard situations in real soccer game) or to incorporate surprising, innovative moves to the game. If the robot activity is partly or mostly independent, there will be no need to evaluate global game situation and search for appropriate move in every TMG. Gained processor time could be used for better strategy learning and in game strategy optimization and optimized game strategy applied in certain gameplay situations (the opponents' team loses ball when attacking by more players; own team can gain advantage by attacking in more players too causing bigger treat to opponents goal) could lead to notably superior results.

CONCLUSION

The main goal of the control system is to enable immediate response in the real time. The system response should be shorter than time between two frames from camera. When the time response of the algorithm exceeds this difference the control quality deteriorates. The method we described provides fast control. This is achieved by using rules that are fast to process. We have described a method of game representation and a method of learning game strategies from observed movements of players. The movements can be observed from the opponents behaviour, or e.g. also from the human players behaviour. We believe that the possibility of learning the game strategy that leads to a fast control is critical for success of the robotic soccer players. Like in chess playing programs, the database of game strategies along with the indication of their success can be stored in the database and can be used for subsequent matches. In future we want to use the modular Q-learning architecture [?]. This architecture was used to solve the action selection problem which specifically selects the robot that needs the least time to kick the ball and assign this task to it. The concept of the coupled agent was used to resolve a conflict in action selection among robots.

Acknowledgements

The Grant Agency of Czech Academy of Science supplied the results of the project No. p. 1ET101940418 with subvention.

References

[(Bernatík et al. 2001)] Bernatík, R., Horák, B., Kovář, P. 2001. Quick image recognize algorithms. In: Proceeding International workshop Robot-Multi-Agent-Systems R-MAS 2001. VSB-TU Ostrava 2001, Czech Republic 2001, 53-58.

[Deneubourg et al. 1991)] Deneubourg, J.L., Goss, S., Franks, N., Sendova-Franks, A., Detrain, C., Cretien, L. 1991 The dynamics of collective sorting: Robot-like ants and ant-like robots. In Proceedings of the First International Conference on Simulation of Adaptive Behavior: From Animals to Animats, pages 356363. MIT Press, 1991.

[(fira)] FIRA robot soccer, http://www.fira.net/ (15-08-2006)

[(Holland and Melhuish 2000)] Holland, O., Melhuish, C. (2000). Stigmergy, self-organisation, and sorting in collective robotics. Artiffcial Life, 5:2:173202, 2000.

[(Horák) et al. 2004] Horák, B., Obitko, M., Smid, J., Snášel, V. Communication in Robotic Soccer Game. Communications in Computing 2004: 295-301.

[(Horák) et al. 2005] Horák, B. Obitko, M., Smid, J., Snášel, V. Strategy and Communication in Robotic Soccer Game. EUROCAST 2005: 565-570.

[(Johnson and Scholes 2001)] Johnson, G., Scholes, K. (2001). Exploring Corporate Strategy: Text and Cases. FT Prentice Hall, 2001.

[(Obitko and Snášel2004)] Obitko, M., Snášel, V. (2004) Ontology Repository in Multi-Agent System. IASTED, International Conference on Artifical Intelligence and Applications (AIA 2004), Innsbruck, Austria, 2004.

[(Mitchell 1996)] Mitchell, M. (1996) An Introduction to Genetic Algorithms. MIT Press, Cambridge, MA, 1996.

[(Park et al. 2001)] Park, K.H., Kim, Y.J., Kim, J.H. (2001) Modular Q-learning based multi-agent cooperation for robot soccer, Robotics and Autonomous Systems, Elsevier, 35 (2001) p. 109122.

[(Sng et al. 2002)] Sng, H.L., Gupta, G.S., Messom, C.H. (2002) Strategy for Collaboration in Robot Soccer. The First IEEE International Workshop on Electronic Design, Test and Applications (DELTA '02) p. 347.

[(Smid et al. 2004)] Smid, J., Obitko, M., Snášel, V. (2004) Communicating Agents and Property-Based Types versus Objects. SOFSEM. Matfyz-Press 2004.

[(Snášel et al. 2006)] Snášel V, Martinovič J., Horák B. Observation for Strategic Control Input Expansion, FIRA RoboWorld Cup 2006

[(Srovnal et al. 2004)] Srovnal, V., Horák, B., Bernatk, R., Snášel, V. Strategy Extraction for Mobile Embedded Control Systems Apply the Multi-agent Technology. International Conference on Computational Science 2004: 631-637.

[(Slywotzky et al. 1999)] Slywotzky, A.J., Morrison, D., Moser, T., Mundt, K., Quella, J.(1999). Profit Patterns: 30 ways to anticipate and profit from strategic forces reshaping your business. 1999.

[(Werger and Mataric 2000)] Werger, B-B., Mataric, M.J. (2000). From Insect to Internet: Situated Control for Networked Robot Teams. Annals of Mathematics and Artiffcial Intelligence. 2000.

[(Kim et al. 2004)] Kim, J., Kim, D., Kim, Y., Seow, K. (2004). Soccer Robotics (Springer Tracts in Advanced Robotics), Springer-Verlag, 2004.

[(Veloso and Stone 1998)] Veloso, M. and Stone, P. (1998) Individual and collaborative Behaviours in a Team of Homogeneous Robotic Soccer Agents, Proceedings of International Conference on Multi-Agent Systems, 1998, p. 309-316.

[(Nakashima al. 2006)] Nakashima T., Takatani M., Namikawa N., Ishibuchi H. and Nii M.: Robust Evaluation of RoboCup Soccer Strategies by Using Match History, 2006 IEEE Congress on Evolutionary Computation, Vancouver, BC, Canada.

AIBO AS A WATCHDOG

Zhenke Yang, Bou Tsing Hau, Leon Rothkrantz
Delft University of Technology, Faculty of Electrical Engineering, Mathematics and Computer Science
Mekelweg 4,
2628 CD Delft
The Netherlands
{Z.Yang, L.J.M.Rothkrantz}@ewi.tudelft.nl

KEYWORDS
AIBO Entertainment robot, Stimulus-response control, Dynamic scripting.

ABSTRACT

This paper presents an ongoing project on an AIBO watchdog. The goal of the project is to add watchdog functions into the AIBO, without modifying its original function of a robot pet. The advantages of using an AIBO for watchdog and surveillance purposes are discussed and an overview of the necessary capabilities to employ an AIBO in this way are given. Finally, two prototypes of the AIBO watchdog are presented. The first prototype uses a simple stimulus-response model for reasoning while the second prototype reasons with dynamic scripts.

INTRODUCTION

Security at home can be greatly enhanced by having several smart robots performing watch functions in and around the house. Toy robots nowadays have become so advanced, that they have the capability to perform common security functions. For example, the AIBO entertainment robot, which was introduced by Sony in 1999, allows people the choice of an electronic pet. However, few know that the same AIBO contains all capabilities of a watchdog. More specifically, the AIBO contains a wide array of sensors (including a camera and microphones), activators (allowing it to move around the house), and wireless LAN capability allowing it to connect to other wireless enabled devices or even the internet. Furthermore, AIBO's open architecture allows programmers to create behaviors for it.
With the increasing availability of these smart toy robots such as AIBO, the vision of multi purpose robots patrolling the home becomes increasingly a reality. In this paper we describe work done to create an AIBO watchdog while retaining all its previous properties. Because, we think the watchdog functionality is a valuable addition to the AIBO's existing functionality, although we have to acknowledge that the main and primary purpose of the AIBO is still that of an entertainment robot. Therefore, AIBO must still remain a robot pet while keeping an eye on the home. Our focus is not on creating a complete home security or surveillance solution from the AIBO, but rather see a, surveillance functionality enabled, AIBO as an addition to existing security systems or a nicety for homes without such an existing security system at all.
The rest of the paper is organized as follows: first we will start with some background and give an overview of the work leading up to this research. Then we continue by defining what makes a watchdog, and argue the suitability and advantages of the AIBO for the job. Then we give the global model of our AIBO watchdog based on a stimulus response model. Next, we extend this model with a more complex situated scripting model. Finally we discuss some of the implementation details and finish with the direction of future work and conclusions.

BACKGROUND

Current home surveillance systems consist of static cameras attached to a ceiling or wall and sometimes even come with microphones for sound registration and sound event detection. Oftentimes these sensors are quite obtrusive and visible in plain view. The sensors are connected to a control unit via either a low-voltage hardwire or narrowband RF signal, which in turn connects to a means for announcing the alarm, hopefully to elicit some response.
Because of AIBOs range of sensors and actuators, it's open architecture, and its's low cost, a lot of researchers around the world have chosen for the AIBO as a research medium for robotic research (Pransky 2001). Using AIBO to help in home surveillance introduces several benefits:

- AIBO is mobile; the mobility of the AIBO has clear benefits above the wall mounted cameras as it allows AIBO to move to other places to investigate at places or maneuver itself to get out of occluded view.
- AIBO is also a robot pet; as we try to add surveillance functionality in the AIBO while retaining the primary function of the AIBO (that of an entertainment robot). The AIBO can carry out the surveillance functions in a less unobtrusive way (compared to wall mounted cameras).
- AIBO provides a flexible platform for multimodal fusion. Cameras and microphones are unimodal sensors, meaning that they can only sense sight or sound. Even though some current surveillance systems contain audio as well as video sensors, generally both modalities are treated separately.

AIBO provides a programming platform to fuse data from the camera and the microphones.

Although there are a lot of benefits by using multimodal systems compared to unimodal system (Llinas and Hall 2001), there are problems that need to be conquered. Using more sensors means more inputs. More input information can be very helpful to decide the most appropriate action but is also more complex.
The current AIBO software provided by Sony uses behaviors that come from a "manually designed database". This software functions well for a robot pet, but is not open for extension. Therefore we used a alternative personality model. This personality is based on an open-ended system or an ever-evolving system by which new behaviors are emerged through the interaction with human and environment (Dobai 2005). This model is more flexible then Sony's database model, but is still under development.

WATCHDOG FUNCTIONS

Before we start presenting the design of the surveillance robot, we have to determine what the robot is actually supposed to be capable of doing. The most common ways to secure a house can be divided in four categories (MinBuza 2006, Utrecht 2006).

1. Discourage – under this category fall measures to give outsiders the impression that there is someone in the house, even if there is no one present. A lively house discourages intruders.
2. Prevent - time and safety of the intruder are inversely proportional. The more time it costs to break in, the less safe the intruder feels. Prevention measures include the use of good locks etc.
3. Detect - detection system give an alarm when e.g. the door has been opened abnormally, the detection system can alarm the police. But the false alarms can become a nuisance as well as potentially dangerous.
4. Register - surveillance cameras which can be monitored by a professional security company can provide the surveillance guards or police the needed information to track and identify the intruder.

After mapping these four categories on the AIBO watchdog the resulting AIBO watchdog capabilities are as follows:

- Capability to discourage the intruder by making sounds randomly and turn on the lights by wireless intranet/internet or touch.
- Capability to prevent intrusion by locking and opening doors and switch lights on or off when necessary to delay/prevent intrusion.
- Capability to detect the intruder by its sounds sensor and its camera.
- Capability to register the intruder by its camera and possibly its sounds sensor.

THE PROCESS MODEL

Figure 1 shows the global model of the watchdog illustrating the process interactions.

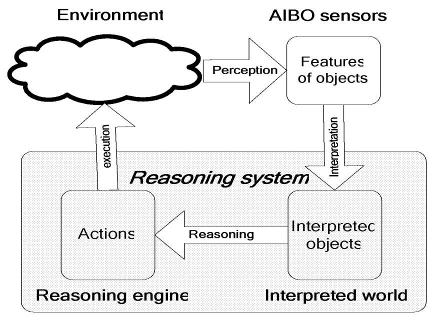


Figure 1: Global model of the watchdog illustrating the process interactions

In this model we distinguish several data processing steps.

- Perception; the environment contains a collection of physical objects which are spread across the environments area, e.g. table, door or fire. All objects have features which can be perceived by the AIBO sensors. The AIBO robot dog has 4 types of sensors which can be interact with its environment: sound sensor, image sensor, distance sensor and touch sensor. The inputs of these sensors create a feature list at a certain time and location.
- Interpretation; from the data gathered in the first step, AIBO is able to extract particular features which are located in space and time. The features in the feature list will be used for reconstruction of the objects in the environment. The correctness of this mapping is heavily depending on the correctness of the sensor's signal and the quantity of features that the sensors can distinguish. Furthermore due to ambiguity of the features the interpreted objects will not always match the original object. This problem can partly be solved by introducing more detectable features to decrease the ambiguity level. The interpretation process will result to the interpreted objects.
- Reasoning; the reasoning part includes the process of creating and adapting the internal world map, once the objects are located, AIBO can continue to update its internal world map. The world map consists of a snapshot of the location of objects that the AIBO has sensed in space. This enables the AIBO to compute a path from one point to another based on this world map. In order to perform watchdog functions, the AIBO should be sensitive to changes in the environment. These changes are reflected in the internal world map, but the reasoning process also has to determine whether the changes constitute dangerous situations, situations that need further investigation or adaptation of the internal world model.
- Actions; an outcome of the reasoning process can be that a certain action has to be executed. In the simplest form the actions can be generated via a stimulus response model. However, more sophisticated models such as stimuli trigger scripts which include a predefined series of actions to

achieve a certain goal, or BDI models (Rao 1995) can be used. In the case that there are multiple possible actions a process of action selection to determine which action to execute has to be carried out. After execution of the selected action the environment will provide the sensors with new data and this process will continue again.

THE REASONING SYSTEM

As discussed in the previous section, the reasoning component in our model is responsible for determining what actions to take given the (interpreted) sensor information arriving in the course of time. The following sections describe two possible reasoning approaches to achieve this goal.

Stimulus-response approach

In the stimulus response approach hard rules are given to the AIBO when it encounters some event. Table 1 shows some examples of stimulus-response rules The first prototype of the watchdog is based on this reasoning system.

Table 1: Stimulus response rules related to watchdog functions

Stimulus	Response
Unknown sound	Go to the direction of the sound source and find the cause.
Missing television	Stand still and alarm people / Capture image and send it by email.
Flames visible	Sound alarm send alarming email and escape from room.

The degree of intelligence is determined by the reactions that the reasoning system comes up with after reviewing the input data. Giving the AIBO hard rules to execute when it encounters a certain object will not make the AIBO act intelligently. This stimulus response approach therefore is not a suitable approach for the AIBO watchdog. To give the AIBO more intelligence it is needed that AIBO can memorize the objects that it encountered before and prioritize the actions that need to be carried out.

Dynamic scripting approach

As a better alternative to the stimulus response approach, the second prototype uses dynamic scripting as reasoning model. In this approach, the AIBO has in its memory a list of predefined script. From sensing the environment, detected features are stored in a memory and can contribute to the activation of a script. If a script passes a certain activation threshold, this script becomes dominant and the actions in this script are executed (see Figure 2). In time features are removed from memory and scripts can loose their domination.

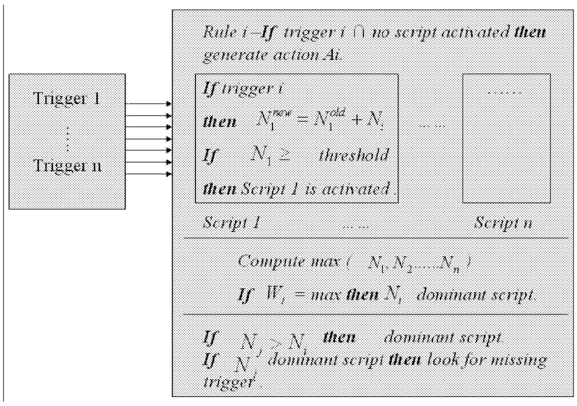


Figure 2: Reasoning scheme for a dynamic scripting system

In the dynamic scripting approach, we can distinguish between two kinds of memory components, the short term memory and long term memory (Figure 3). The short term memory component contains the object and event that AIBO is dealing with at the moment.

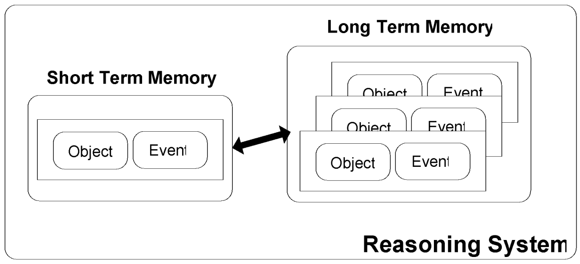


Figure 3: The two memory components, their contents and their relationships

First, when an object is recognized, it will be compared with the object in the short term memory. When the new object and its related event has a lower priority than the current object in the short term memory, the new object will be placed in the long term memory. Otherwise the new object will be placed in the short term memory and the old object which was in short term memory, will be placed in the long term memory. Duplication in the long term memory is not allowed. When AIBO encounters a lower priority object on its way to the goal point and also encounters it at the way back, the oldest object will be removed from the long term memory and a new one will be placed in the long term memory.

This last-in-first out approach for the long term memory is chosen above the approach based on the priority of the objects. The explanation for this choice is that when the current event consumes a lot of time the event with the next highest priority will probably not exist any more when the current event is completed. Therefore it is no use to leave that event in memory. Except the time issue there is also another reason to choose for the last-in-first-out approach, the place issue. When AIBO has just finished its last mission, the event of the last object is also the closest object to its current point. There is one exception: when AIBO has not encountered an object for a long time and AIBO is walking in a circle. Then the closest object can be the oldest detected object. Figure 4 shows this situation. In general the last-in-first out approach will be more efficient.

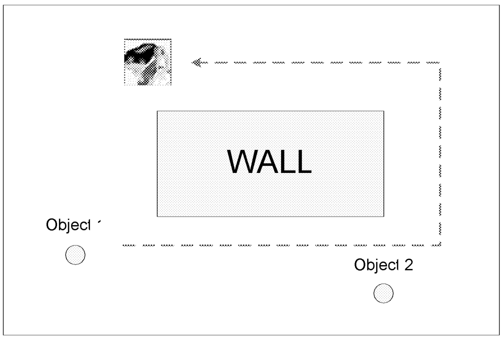


Figure 4: AIBO has not encountered an object for a long time and AIBO is walking in a circle. The closest object is the oldest detected object

The best approach for the long term memory issue is a combination of the two approaches. We can set some threshold to determine which of the two approaches will be used. This can be based on the nearest place to satisfy a certain long term event or based on the priority level which has arrived within the last 5 minutes.

IMPLEMENTATION

The watchdog functionality is built on top of the modular architecture introduced by (Dobai 2005) (which is itself based on another architecture developed in the URBI Project at ENSTA Laboratories. URBI (Universal Robotic Body Interface) (Baillie 2004) is a scripted language designed to work over a client-server architecture in order to remotely control a robot. URBI is released under the GNU General Public License. Our present prototype runs the server on AIBO (the original URBI server) and the client on a PC (developed by us), therefore the reasoning is done on the PC and AIBO is in charge of sending raw data information from the sensors and executing complex commands written in the URBI scripting language. Our long term goal is to have both client and server running on AIBO and in this way having an autonomous robot.

The sensor data received from the AIBO consist of

- Raw images from the AIBO camera. From these images visual waypoints, indicated by special cards, can be extracted allowing the AIBO to navigate and position itself. The recognition of the waypoint cards is however very sensitive to lighting conditions. Therefore, the waypoint cards need to be calibrated under current lighting conditions beforehand (see Figure 5). At this moment the software of the prototype is not advanced enough to recognize arbitrary objects such as televisions or tables, so these objects are currently entered manually in the system via the GUI on the PC (Figure 6).
- Raw stereo waves from the AIBO microphones. From the sound data, the direction of the sound source is calculated using the algorithm described in (Datcu 2004). By applying a simple volume thresholding algorithm on the wave samples, loud sounds can be easily detected. The automatic recognition of relevant characteristic sound events such as the sound of breaking glass are not implemented yet, these sound events can be manually added through the GUI on the PC (Figure 6).
- Infrared distance reading. With these readings, the AIBO can determine the existence of obstacles on its direct path.

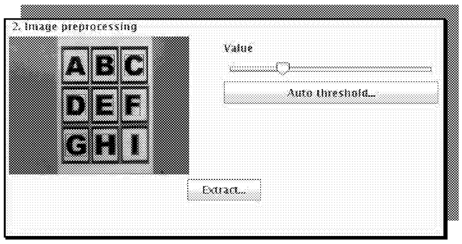


Figure 5: AIBO can recognize waypoint cards, but the recognition software has to be calibrated beforehand

An external event generator module which is part of the GUI module simulates special events that the AIBO is currently not able to detect automatically. Technically, these special events are stored in nodes on the map. When the AIBO arrives at that node, all these events are unleashed upon the AIBO. The events at each node can be modified by the user using the GUI module (Figure 6).

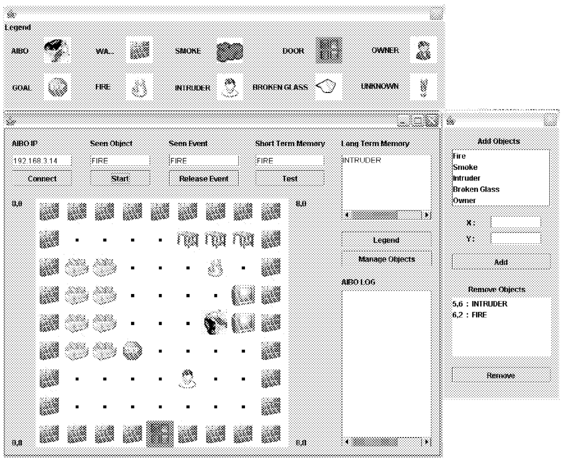


Figure 6: GUI to create special events for AIBO to handle. For each node events can be specified. These events are sent to AIBO when it reaches that node

To test the system, we have built an artificial home environment of which a map is shown in Figure 7. There are objects such as furniture from the real world in the test environment. As a first scenario, we introduced a patrolling behavior consisting of a simple waypoint system for the AIBO to follow.

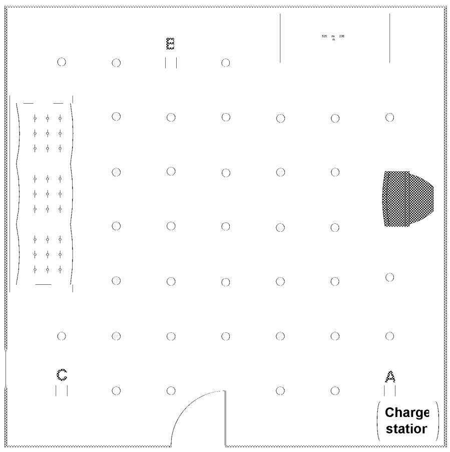


Figure 7: A map of the home, with furniture and waypoints

For simplicity, the start point and end point are given to the AIBO beforehand. The navigation goals are represented as the small squares with a character above it. The next destination node is default the next character in alphabetic order. When special situations arise, the goal node will be adapted. The path between the two nodes will freely be chosen by the route planner, but it will take the node with the shortest distance to the destination node by default. Except when there are objects on its way, it will take another route to avoid collision. Since this project is not dealing with the pre processing of the sensors input, the input of the sensors are simulated and stored in the nodes of the waypoints.

CONCLUSIONS

The AIBO entertainment robot has the ability to perform surveillance functions. Its mobility and its array of sensors make it ideal for investigating in places where surveillance cameras have no view of vision. Because of its primary function of robot pet, its presence is non obtrusive to the robot owner. Furthermore, its open architecture allows for the implementation of data fusion algorithms to get more reliable and robust conclusions then are possible with a single modality.
In this paper we described the model and two reasoning approaches of our AIBO watchdog. We also described the artificial room in which the AIBO performs its watchdog duties. Because of the lack of a suitable object detection module, we created a GUI in which the user can manually add objects the AIBO will detect. Also the localization method is based on artificial waypoint marks instead of real landmarks, but this also is a result of the lack of an object detection module.

REFERENCES

Hall, D. L., Llinas, J., "Handbook of Multisensor Data Fusion," CRC Press, 2001.

Baillie, J.C., "URBI: A UNIVERSAL LANGUAGE FOR ROBOTIC CONTROL", International Journal of Humanoid Robotics, 2004, World Scientific Publishing Company

Dobai, I., Rothkrantz, L., van der Mast, C. "Personality Model for a Companion AIBO", ACM SIGCHI International Conference on Advances in Computer Entertainment Technology, Broadway New York, June 2005, pp. 438-441.

Datcu D., Richert M., Roberti T., de Vries W., Rothkrantz L.J.M., "AIBO Robot as a soccer and rescue game player", Proceedings of GAME-ON 2004, ISBN 90-77381-15-5, pp. 45-49, November 2004.

Joanne Pransky, "AIBO- the Nr. 1 selling service robot.", Industrial Robot: an international journal, Volume 28, Number 1 2001, pp24-26.

MinBuza, Dutch department of foreign affairs "Wijs op reis - Je huis veilig achterlaten." https://minbuza.nl/default.asp?CMS_ITEM=MBZ458731

Utrecht, Gemeente Utrecht. "Utrecht steeds veiliger.", http://www.utrecht.nl/smartsite.dws?id=131024

Rao, A. et al., "BDI Agents: From Theory to Practice", Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95), San Francisco, USA, June 1995.

MOBILE GAMING

ON GUARANTEEING EQUITY TO MOBILE PLAYERS IN A METAVERSE

Stefano Cacciaguerra
Department of Computer Science
University of Bologna
Via Sacchi 3, Cesena (FC), Italy
E-mail: scacciag@cs.unibo.it

KEYWORDS

Multi-Player Online Games, Pervasive Entertainment, Agent-based Entertainment, Virtual Environments.

ABSTRACT

In the 1992, Neal Stephenson described his vision of how a virtual reality-based Internet should evolve in the near future coining the word Metaverse. The access to the Metaverse is possible from any location adopting portable devices. With the advent of virtual environments like World of Warcraft, Second Life, Project Entropia and Sociolotron many prototypes of Metaverse are born. Further, leader companies in the entertainment promote portable consoles that exploit wireless hotspots. In this sense, wireless infrastructures extend the Internet allowing the participation in virtual environment even from mobile devices. Unfortunately, wireless infrastructures belong to the unreliable networks. Therefore, the movement of a mobile player under the coverage area of these networks might cause interruptions and lags in the participation. In this context, our aim is to ensure good playability to all players apart from the kind of networked infrastructures and devices used, or, at least, to try to upper bound the critical situations guaranteeing equity among all users.

INTRODUCTION

In the 1992, Neal Stephenson invented in his Snow Crash, a science fiction novel, the word Metaverse. The Metaverse constitutes his vision of how a virtual reality-based Internet should evolve in the near future. The Metaverse world is a black sphere 65536 km in circumference where Street runs around the equator. In this world, there are 256 Express Ports located at 256 km intervals beside the Street. Here, it is possible to buy a development license and build anything is approved by the group which manages the Street protocol. In this novel, a user can access the Metaverse through a private terminal that allows to customize the avatar and, overall, to participate from any location adopting portable devices. In accord with the Stephenson's vision, one of the main possible future scenarios of Internet shows its evolution as a global cyberspace where each user can interact through his own avatar. From this standpoint, the avatar becomes a tool that allows the humans to create a virtual world (e.g. as shown in the Matrix trilogy). This synthetic world should be so accurate to build up the ideal environment for the artificial life of the user. Hence, it promotes to study the social behavior of humans and supports the basis of a new form of communication (Schell J. 2002). The immersive reality and the possibility to interact with the virtual world without constrains allow theirs use as a support to the decision making (Cacciaguerra et al. 2005). The entertainment industry is investing money in implementing the concept of the Metaverse. Many commercial Virtual Environments (VE) have an own economy, an own government, an own currency and many synthetic inhabitants are born and die, daily. Participants from across the globe can leave the real world behind and undertake heroic quests exploring virtual lands. In this evolutional shared reality, denizens can explore, build on virtual land, socialize, or vie for status. The main examples are:

- World of Warcraft, a massively multiplayer online role play game that enables over five millions of subscribed players;
- Second Life, a massive virtual universe where there are hundred shops that sells items from virtual i-Pod up to synthetic Ferrari;
- Project Entropia, a massive virtual universe where the total economic turnover amounted to more or less 165,000,000 US dollars in 2005;
- and Sociolotron, a multiplayer online role play game for adults that takes pride in being a highly politically incorrect game.

In this scenario, as described by Stephenson, it becomes interesting to allows users to share interactive activities while they are moving with theirs mobile devices under a set of wireless adapters spread on a building, a district or a town directly connected to the Internet (Cacciaguerra et al., 2006). Since movements cause interruptions and lags in the participation, our aim becomes to upper bound these critical situations ensuring a good playability and, overall, equity to all players even if they use portable devices under unreliable networks. The reminder of the paper is organized as follows. Section 2 illustrates the problems in participating in a metaverse from a mobile device. Section 3 highlights design issues to ensure equity to all players. Section 4 describes the system architecture. Section 5 presents some results. Finally, Section 6 concludes this work with an open question.

PARTICIPATION OF MOBILE PLAYERS

Many public places, such as, parks, airports, railway stations and megastores are increasing the availability of network connectivity through wireless adapters. Furthermore, leading companies such as Nintendo, Sony and Nokia are competing in the handheld entertainment market promoting respectively Dual Screen, Play Station Portable and N-Gage, which can take advantage of multi-playing through the wireless connections. In this sense, it becomes possible to promote exciting interactive experiences with other users worldwide, exploiting ubiquitous connectivity by means of integrated wired and wireless infrastructures. Although wireless technologies suggest new possibilities, such as, mobility, location and context awareness to multi-player gaming, problems arise when they try to extend the wired Internet. From a technical viewpoint, Bluetooth, Wi-Fi, Wimax and UMTS technologies allow to communicate with nearby access points in less than a hundred of milliseconds. According to this consideration, the main problem in wireless communication arise from the mobility of user, which creates a complex networked scenario to investigate. This complex scenario emerges questions, like:

- what happens when a mobile user needs to participate in an Internet game, but is too far from the nearest access point?
- Or when he wants to continue playing while he is moving from one access point to the others?

In these cases, handovers, transmission errors and temporary link outages occur causing lags. Lag refers to noticeable period experienced in computing communications to the user caused as a result of extended or unexpected delay. In this sense, the packet coming from the mobile device to the VE might be lost because no coverage area is available or delayed because the users is moving from an area without coverage to another with it. Furthermore, it is well-known that increasing the distance from the access point to the mobile device, the bandwidth decreases and there is a higher probability to find more obstacles that dissipate the signal. Moreover augmenting the number of users more collisions among packets occur and, in this condition, increasing the quantity of access points amplifies interference in the communications. From a technical viewpoint, all this means that vertical or horizontal handovers, transmission errors and temporary link outages might cause a momentary interruption of the network communication. Instead, extended link outages might cause long interruption, while a disconnession caused by failed handover or an application shutdown could even cause a permanent interruption. At a higher level, this means that a user may not be able to send or may send with a significant delay his actions to the VE, losing some turns of the match and, worst, having fewer chances to win the game. In this sense, playing from a mobile device over an unreliable network could disadvantage users making unfair theirs participation in the game.

ON DESIGNING A PARTICIPATORY FRAMEWORK

Our approach promotes a mechanism able to handle the communication on both sides of the networked infrastructure when problem due to the movements of the players occur. This mechanism should be able to guarantee a transparent playability for the players maintaining interactivity, and coherence in the game realizing a playing session. The playing session sustains the equity/fairness for all players. Our idea suggests to place the entire game system on the wired Internet side and to allow the participation from a graphical shell on a mobile device. In this scenario, we design a Participatory Framework (PF) accommodated on both sides able to monitor the network infrastructures and to manage the game when problems due to the movements occur, implementing the playing session over the stack TCP/IP. On the mobile side, this framework is able to detect if the game system is unreachable or reachable with a certain delay. In accordance with these considerations, the framework is in charge of waiting for the next game ack or of trying to connect again the system. On the game side, the framework is able to detect if the application on the mobile device had some problems or if the movement of the player caused problems. In this case, the framework gives the control of the player's avatar to a Mimicking Mechanism (MM) that plays it until the problems are solved. In this way, the system is not affected by the fault of a single player and the game continues its evolution as if nothing had occurred.

Interactivity

In line with these considerations, the game system should maintain the interactivity under a sensorial perceptivity threshold, providing a realistic "look and feel" and bounding low quality scenes or jumpy shots on mobile devices as limited as possible. To reach an adequate "look and feel" is inevitably reflect on the subjective impression about the scores in the game. In particular, we do not support with the fact that a better player loses with weaker one because he is at a disadvantage due to network impairment in the communications. In this context, interactivity is related to the latency and jitter between action generation on a mobile device and the next event visualization in its graphical shell (after its execution on the game system). In (Cacciaguerra et al. 2006), we tested the effectiveness of the management of interactivity in the PF over a networked scenario where mobile players participate in the game, when two link outages occur. Our measurement was related to the performance of the whole game engine when players begin to have problem in the communications. This measurement is different from the player's perspective approach adopted in (Dick et al. 2005) where a normalized value represents the score of each player in comparison with the others and a new metric, the degree of subjective quality in audio and video contents. Our idea is to upper bound the time (in relation to the human perceptivity threshold) to deliver an action to the game system. To reproduce in our lab a scenario with real networks, we involved a synthetic emulative platform to route IP packets (Cacciaguerra 2005) according to the proposed scenario, linking real mobile devices and workstations to the game server. In one case, the synthetic emulative platform was configured to reproduce a static transmission delay between a workstation and the server simulating a reliable and uncongested channel. In the other case, the transmission delay between a mobile device and the game server was calculated through a function that simulates two consecutive outages. In order to study the performance of our approach, we compared the game evolution when five different managements of the human perceptivity threshold. Results confirmed the effectiveness of our approach.

Coherence

Coherence is related to the uniformity of the evolution of the game obtained as if all players participate through reliable and effective network communications. This means that each player participates in the game, correctly (i.e. without wasting time), in a perfect world, from the viewpoint of the network communications, and, of course, under the hypothesis to have not any other problem coming from the game system or from the interface on the mobile device. In order to maintain coherence, it is important to implement a MM able to substitute the player when his action does not arrive to the game system within the perceptivity threshold. In particular, the mechanism must not play better than the user, making unuseful his future actions, or, worst, binding his possibility to win the match. As an interesting side effect, each player should be not able to understand who has some problems in communication looking the behavior of the avatars of the other participants. Although the research of this mechanism is not our responsibility, we propose some considerations on its working to improve the effectiveness of our framework.

Unfortunately, the believability of an avatar is subjective, influenced by the culture and the skills of the other players (Mac Namee 2004). Further, to show an adequate degree of humanness (Livingstone 2005), the avatar should adopt human-like reaction and decision times, avoid to give superhuman capabilities and realize some tactical/strategy reasoning (McGlinchey and Livingstone 2004). Hence, which is the best action the MM has to play when it is in charge of the orphan avatar? The key point is what means the best action (Cacciaguerra and Roffilli 2005). From the player's viewpoint the MM should choose the most predictable action while from the others' perspective it should try to reproduce the strategy of the user.

In the former case, the default action could be the natural consequence for the correct evolution of the game and could be easily predicted. This approach is successful if the loss of control is limited to a low percentage of the whole actions and, if possible, with a low frequency. The main advantages are the easy implementation using a lookup table of state-action pairs. Unfortunately, if the avatar uses a hard-coded default action for a long period will be recognized quickly.

If the loss of control is a substantial percentage of the whole actions or its frequency is too high, the avatar will most likely start to show non-human behavior. In this latter case, keeping coherence should take into account the stochastic/strategic behavior of human beings. Traditionally, this effect was limited increasing the complexity of the algorithms that control the avatar. The development of these algorithms is problematic because they must take into consideration many combinations of events and situations. An alternative idea is to monitor each user during the game evolution in order to recognize the typical pattern of his behavior and to instruct the MM on his strategy. The MM must be adaptive to capture the essence of the user's strategy. In this sense, we need techniques (Dietterich 1997) that analyzes a collection of task pairs (instance, solution) without knowing the dynamics of the solution (i.e. without formalizing the algorithm). With the collection of an adequate number of situation, a MM should be able to resemble the user's strategy/ability with an appropriate level of mimesis and with a good generalization. Another issues is related to the computational effort to produce a good action in a short time (i.e. shorter than the perceptivity threshold). Finally, the last issue is associated to produce a model knowledge: collecting data off-line or performed on-line during the game evolution.

Equity/Fairness

The good management of the interactivity and of the coherence together improve the equity/fairness to all players. In distributed systems, fairness is related to guarantee to avoid starving any process in order to assure to all of them the same priority in allowing their accesses to the shared resource. In this sense, all processes in a distributed system should have the same chance to make progress. Similarly, our aim is to guarantee equity among all players allowing them to play a correct number of actions with the same frequency inside the metaverse. In particular, this mean that all users are able to participate correctly in the game and cannot take advantage respect the others because they play more. Unfortunately, the problem coming from the network communications might disadvantage some of them. From our viewpoint, we can consider avatars as separate processes (independently from the fact to be played by a software code or by a human being). Therefore, the possibility to play the same number of actions with the same frequency in the match becomes our measure of the equity in this procedure. Hence, if the framework is able to detect the problems in the network communications and, simultaneously, to substitute a lost action with an adequate level of mimesis, it becomes possible to play under fair conditions. Of course, this means that not all players will take the same number of actions at the end of the match, but at least the number of events is the same for each avatar! In this sense, our framework shall give all players the same chances to participate in the game and the same possibility to do not lose.

SYSTEM ARCHITECTURE

In order to support our thesis, we developed a prototype that integrates a Multi-Agent System with our PF. From this standpoint, the System for Parallel Agent Discrete Event Simulation (Riley 2003) implements the game environment setting up the rules of the synthetic world and programming the behavior of its agents. On the other hand, we realize a playing session layer on the stack TCP/IP between the user and his avatar implementing the PF. Hence, if a short interruption makes the player unable to deliver an action to his avatar within the perceptivity threshold, the PF detects the problem and tries to maintain the interactivity in the game evolution forcing the MM to control the orphan avatar guaranteeing the coherence with an adequate level of mimesis. Or worst, if the interruption lasts along becoming a permanent one, the PF recovers the communication allowing users to resume the control of his avatar while the MM produces an adequate strategy for the avatar. As consequence, the PF (see Fig. 1) is splitted into two coupled modules residing, respectively at the user side (called Shell Participatory Framework; SPF) and at the game side (called Avatar Participatory Framework; APF). In particular, the APF accommodates the MM.

Avatar Participatory Framework

The APF manages either prolonged failures due to any critical problem, or brief temporary malfunctions due to short interruption. At the beginning of a match, it checks for the TCP connection to its SPF, while the MM controls the avatar until it receives an action from its player. Then, it accepts the connection to the avatar from a SPF, or, eventually, it recovers a previously instantiated communication after the occurrence of a long or permanent interruption. When the connection is active, the APF checks up two different timeouts continually. The former (called Action Timeout) is useful to prevent low interactivity from slowing down the evolution of the game, monitoring the responsiveness of its SPF within an upper bound period of time. Clearly, when the upper bound is overcame, the APF allows the MM to play an action in place of his user. If the MM plays a number of consecutive times exceeding a maximum value (called TCP Timeout), the APF leaves to it the control of the avatar setting the state of the communication as "broken". As a consequence, the APF shuts down the connection, jumping to the listening phase, ready to recover this communication.

Shell Participatory Framework

Every time, the SPF checks if its avatar is reachable or if a new connection is necessary. When the SPF connects the first time its APF, it starts a new playing session. Each turn, this protocol forces each APF to send a RequestAction event to its own SPF. The SPF waits for a RequestAction event until a period long an Action Timeout and checks up its incremental identifier. In this way, it is possible to know if there is some delays or if some communications get lost. If the RequestAction comes too late, it buffers the last generated action, waiting for the next slot time. If, in the meantime, the player generates another action, the SPF will buffer the last one so as not to deliver an old action related to a previous situation in the game. If, unfortunately, the SPF does not receive any request within an amount of time equal to the value of the maximum admissible number of consecutive Action timeout (called in this case TCP timeout), it sets the state of the communication as "broken". In this case, it shuts down the connection trying to recover the previously instantiated communication by reconnecting to its own avatar.

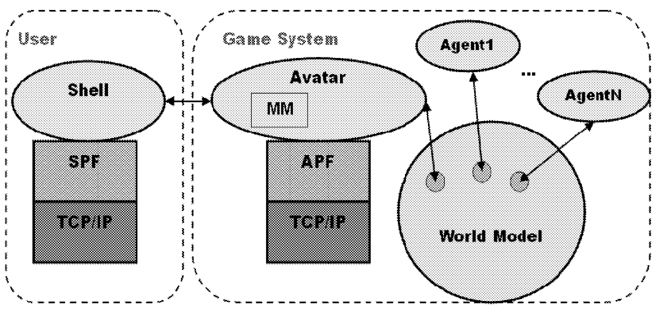


Figure 1: The Participatory Framework

CASE STUDY

This paper tries to demonstrate the effectiveness of the playing session implemented through our PF. The previous work (Cacciaguerra et al. 2006) showed that the framework is able to guarantee interactivity, but it does not prove that this system architecture lets unchanged the chances of victory in the game. Along with this, it does not demonstrate that, at the higher application level (i.e. at the game level) the playing session does not alter the progress of the strategy of an user and, as consequence, does not modify the whole evolution of the match. To demonstrate it, we realize a case study implementing over our architecture a clone of the game Tron. The environment of Tron is an arena where players challenge each other driving a synthetic motorbike that leaves behind, as wake, a wall. Each motorbike must avoid to crash in any wall during its movement. When a motorbike hits one of the walls or the borders of the arena, it explodes leaving its player out of the game. The aim is to remain the last one alive avoiding the walls and trying to close the ways of the other players. A motorbike can never stop: it could accelerate from the minimum speed to the maximum and vice versa. Turning left or right decreases the speed of the bike. We choose Tron because is a fast-paced multi-player game with a low complexity in its implementation but that allows the performance of sophisticated strategies. In fact, it has just one kind of agent (i.e. the motorbike) that can go, each turn, straightforward, left or right. As being a fast-paced game, it allows to conduct an adequate number of trails in few time.

Experimental Campaign

With the aim of studying the performance of our PF, we reproduce in a lab two scenarios executing the same trials under either reliable or unreliable networks. In particular, in the scenario with unreliable networks the mobile player passes, cyclically, through three different phases. At the beginning, he stays near the access point exploiting good coverage area. After, he moves toward the border while the coverage area becomes worst. Finally, he goes out and comes back. The measure of performance is to verify how much the outcomes of the same match in both the scenarios diverges. In order to demonstrate that our PF is not invasive we believe that the percentage of victory of a strategy respect another one should remain the same, also if we conduct the same trails under different scenarios. In order to produce an adequate number of trials for the comparison in the two scenarios, we automate the process reproducing the behavior of expert players, studying the main strategies to win a match in Tron. Each strategy has been implemented through an apposite code accommodated at the mobile side over SPF.

I. The first strategy mimics a newbie player: when it arrives near a wall, it checks up both the right and the left side and chooses that one without an obstacle.
II. The second tries to move each turn in the direction where it is possible to cover the longest practicable way before crossing a wall.
III. The third goes on the direction that guarantees the highest number of practicable way in the next turn.

IV. The fourth is equal to the previous but it checks more than the first next turn.

We study the capabilities of each strategy conducting 30 trails where each agent adopts the same behavior. To understand the difference among the strategies we collect data regarding on the number of cell taken up from the wake of each avatar and theirs lifetime (see Tab. 1). In order to reduce the number of confrontations, only the winning strategies are compared between them. As we can see in the Tab. 2 the lifetime of each strategy is more important of the taken up space. For example, strategy III is successful with I e II also if they occupy a greater percentage of the arena with theirs wake. This means that III is a more adaptable strategy that exploits the ways among the wakes of its opponents.

In order to reproduce the behavior of a player, each MM runs the same code, accommodated in the SPF, implementing the strategy. In line with this, when an action generated from the code on the mobile device does not reach the avatar, the same code inside the MM generates the event. By means of this trick, we reproduce with an adequate level of mimesis the behavior of the player (being the same code). Hence, if we are able to show that the percentage of victory does not change in both the scenarios, as shown in Tab. 2, it is high probably that our PF guarantee equity for all players. In particular, the difference in Tab. 2 are due to the adoption of random components inside the code that implements the strategies. For example, strategies II chooses randomly if more paths have the same length. The high number of trials conducted should support this opinion. Further, Fig. 2 show the snapshots of the playlogs of the same match where are engaged avatars adopting strategies without random components, under the same conditions, apart from the networked scenario. The evolution of the match is the same! Hence, we can claim that the tendency to win a match from a better strategy is reported also if our PF interact hardly with the game to guarantee equity among players.

CONCLUSIONS

This approach prevents that problems coming from unreliable networks decrease the chances of victory of mobile users guaranteeing equity among all players of a metaverse. The experimental campaign has shown that coupling an improved network framework with a mechanism able to reproduce the behavior of a user with an adequate level of mimesis allows to promote fair matches guaranteeing equity among all players. The trick to accommodate the same code either over the SPF and inside the MM has supported this theory as it is highlighted in Fig. 2. In particular, the outcomes of this approach show that the percentages of victory due to the adoption of a specific strategy do not changes also if the PF interact hardly with the game. In essence, the PF does not alter the progress of the strategy of an user and, as consequence, does not modify the whole evolution of the match. If we admit that our framework substitutes the player an acceptable number of times not invasive for his entertainment, a main question still remains: which is the best way to reproduce the behavior of a user? This open question represents in our opinion, one of the hottest topic in the literature, while our implemented prototype a way to test it. As side effect, it promotes also, the interesting possibility to allow a user to challenge against him-self, for a better comprehension of his own errors and of his good actions.

Table 1: Study of the Behavior of Strategy

	% AVG taken up space	б	AVG lifetime (sim time)	б
I	6.4	0.8	145	35.9
II	6.9	0.7	127	32.1
III	4.3	0.4	442	133.1
IV	6.5	0.8	561	133.6

Table 2: % of Victory Changing Scenario

Match	unreliable	reliable
I vs III	0-100	0-100
II vs III	4-96	0-100
III vs IV	0-100	8-92

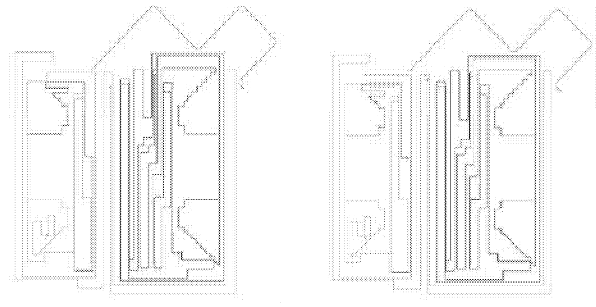


Figure 2: Snapshots Coming from two Videos Showing the Same Match, Left) Under Unreliable Networks, Right) Under Reliable Network

REFERENCES

Cacciaguerra S. 2005. "Experiences with Synthetic Network Emulation for Complex IP based Networks", *Ph.D. dissertation*, University of Bologna (Italy).

Cacciaguerra S., Cagneschi C., Fabbri R., 2005 "The Architectonical Design of Virtual Environments Fuels a new Form of the WWW" *in proc. of ESMc* (October), Porto (Portugal).

Cacciaguerra S., Mirri S., Pracucci M., Salomoni P., 2006. "Wandering about the City, Multi-Playing a Game" *in proc. of IEEE International Workshop on NIME* (January), Las Vegas (NV-USA).

Cacciaguerra S., and Roffilli M., 2005. "Agent-based Participatory Simulation Activities for the Emergence of Complex Social Behaviors" *in proc. of AISB* (April), Hatfield (England).

Dick M. Wellnitz O. and Wolf L. 2005. "Analysis of Factors Affecting Players' Performance and Perception in Multiplayer Games" *in proc. of NetGames ACM conference* (Oct), Hawthorne (NY - USA).

Dietterich T. G., 1997. "Machine Learning Research: Four Current Directions" *AI Magazine*, 18 (4), 97-136.

Livingstone D. 2006. "Turing Test and Believable AI in Games" *in Computers in entertainment ACM*, Vol. 4, No 1 (Jan), 1-13.

Mac Namee, B. 2004. "Proactive persistent agents: Using Situational Intelligence to Create Support Characters in Character-Centric Computer Games" *Ph.D. dissertation*, University of Dublin (Ireland).

Mc Glinchey, S. and Livingstone D. 2004. "What Believability Testing Can Tell us" *in proc. of the CGAIDE*, Redding, (WA-USA).

Riley P., 2003. "SPADES: System for Parallel Agent Discrete Event Simulation" *AI Magazine*, 24(2):41-42.

Schell J., 2002. "Understanding Entertainment: Story and Gameplay are One" *The human-computer interaction handbook: fundamentals, evolving technologies and emerging applications archive*, Lawrence Erlbaum Associates, pp. 835 – 843.

ON BRINGING ADVENTURE GAMES INTO THE MOBILE GAMING SCENARIO

Marco Furini
Computer Science Department - University of Piemonte Orientale
Via Bellini 25/G - 15100 Alessandria, Italy
Email: furini@mfn.unipmn.it

KEYWORDS

Mobile Gaming, Game on low-resource devices, embedded games, 3GP.

ABSTRACT

Videogames developed in 80s are very popular in the current mobile scenario. These games don't require several system resources and hence they are well suited for the limited system resources of current cellphones. In this paper we focus on a genre that was very popular in the past, but is not receiving much attention today: the adventure games. These games are proper for the current mobile scenario as they require very little system resources. We propose a software architecture to potentially transform an audio mobile device into an adventure games console. The adventure game is described through MPEG7 and is embedded into a 3GP file. Our approach ensures interoperability, transparency, lightness and security. The simplicity of our approach along with the large availability of audio devices may bring new life to adventure games, which can become an exciting game genre also for today's users.

INTRODUCTION

A recent research indicates that the mobile-gaming market has grown, since its first appearance in 1997, to a total of $590 millions in revenue in 2005, with a forecast of $1.5 billion by 2008 (see, ABI (2006)). The interest around this market is so high that giants of the game industry, like Electronic Arts and Ubisoft, are entering the mobile-gaming market.

Looking at the best sellers video games of the Verizon wireless provider, it is interesting to note that several titles come from mid 1980s: Tetris, Pac-Man and Pong are some examples. This popularity is due to the multimedia features of current cellphones, which are comparable to the one of the gaming console of the mid 80s. This is why, the gaming industry is using old video games for the current mobile scenario.

In this paper we focus on a particular genre of videogames: the adventure games. These games were very popular in the 80s and 90s and although the adventure game market suffered from the introduction of high performing game consoles, today, some adventure games are still released, and many fans develop their own adventure games with tools like Adventure Game Studio and Visionaire.

The main characteristic of these games is that the player has to deal with problem-solving rather than to shoot against something; Being focused on a narrative story, instead of being based on reflex-based challenges, adventure games are well suited for the mobile scenario, as they don't require several system resources (some adventure games are text-based and don't even require a graphical screen). Famous adventure games included Zork, King's Quest and The Secret of Monkey Island.

The contribution of this paper is to propose a simple, transparent, inter-operable and secure mechanism to potentially transform any portable music player into an adventure game console. The idea is to describe the script of an adventure game through a text-based description and then to store this description inside a media stream without modifying its structure. For interoperability reasons, our approach involves standard mechanisms like the Advanced Audio Coding, the MPEG7-DDL for the multimedia content description and the 3GP media container format as the file format.

Our proposal is composed of a script manager, a scene manager and an interaction manager, which cooperate to produce the adventure in a transparent and secure way. Transparency ensures that every audio player can potentially become an adventure game console. In fact, the only requirement is the presence of an enhanced player able to read and understand the stored adventure game description. Hence, only a simple software update is necessary. To achieve transparency, we consider the 3GP file format, which is a container of different multimedia resources. Security is essential to avoid any malicious alteration of the adventure game description and is achieved through the development of a security mechanism that ensures that only a legal 3GP file owner can enjoy the full features of an adventure game and also ensures that alterations cannot be done to the 3GP file. This mechanism is developed with classic security tools (encryption, watermarking and hash functions).

The characteristics of our approach, along with the never-ended interest around adventure games and considering the large availability of audio devices, may bring

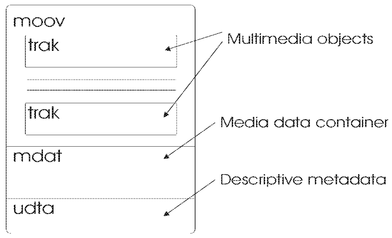


Figure 1: The 3GP ISO Base Media File structure.

new life to adventure games, which can become an exciting game genre of current cellphones.
The remainder of this paper is organized as follows. First, we review the 3GP, AAC, MPEG7 and the used security tools and then we present our proposal.

PRELIMINARIES

In the following we present basics of the standards used in our mechanism: i) the 3GP (the media file container of the adventure game); ii) the AAC (the audio stream format used to encode the narrator voice) and iii) the MPEG7-DDL (the markup language that describes the adventure game). We also review the tools used by our security mechanism.

The 3GP File Format

Our approach introduces the adventure games inside 3GP files. This file format is based on the ISO Base Media File Format (readers can refer to MPEG (2005) for a detailed description) and can contain time-based audio-visual information with an object-oriented structure (each object represents a media object). In particular, the file is made of data structures called boxes, which may contain actual media data or metadata (information to define the media properties). By combining these boxes, different multimedia objects can become a single *presentation* (called *movie* in 3GP files).
Figure 1 shows a simple example. The file has three main boxes: Movie (`moov`), Media Data (`mdat`) and User Data (`udta`). The `moov` box includes all the data related to the presentation; it may contain several `trak` boxes, each of them represents the container of an individual object. The `mdat` box is the media data container, while the `udta` box is used to store descriptive metadata (in MPEG-7 format) related to the whole presentation. As we show in the following, our proposal uses a `trak` box for the audio and a `udta` box for the game description.

The MPEG7-DDL

Our approach uses MPEG7-DDL to describe the adventure game. MPEG7-DDL is a markup description language based on XML and is provided with a set of tags to describe multimedia contents. It produces a description of the spatial layout of different media objects (video, audio, graphics, text) as well as the temporal order in which these object will be played out during the presentation. These tags have attributes and values and are usually enclosed between angle brackets in the form `<tag attribute=value>`, with the exception of tags that do not have attributes.
Among the several pre-defined tags, it is worth presenting some tags that will be used to describe data in this paper. The `<AudioSegment>...</AudioSegment>` tags allow decomposing an audio stream into several audiosegments, by specifying the showtime and the duration (through the tags `<MediaTime>...</MediaTime>`) of each audio-segment. The `<Image>...</Image>` tags allow specifying an image, which is located through the `<MediaUri>` tag. The `<Label>...</Label>` tags allow labeling a portion of the MPEG7 description. The `<TextAnnotation>` and `<FreeTextAnnotation>` tags are used for simple text description. For a detailed MPEG7 description, readers can refer to Hunter (2001).

The AAC File Format

Our approach uses the Advanced Audio Coding (AAC) format to encode the vocal narration of the adventure game. AAC is an MPEG standard audio encoding algorithm and can be used inside 3GP files. This format provides high audio quality at low bit-rates and is gaining wide adoption in the marketplace. Readers can refer to Brandenburg (1998), Herre and Purnhagen (2002) for further details about this format.
Here, we simply highlight that an AAC stream is composed of a set of self-encoded audio blocks. This is the main difference with other encoding algorithms, where to decode an audio block, it is necessary to have several (adjacent) audio blocks. Conversely, an AAC player can jump from one block to another without any problem, and, as we better show in the following, this is a fundamental characteristic for our proposal. In fact, in an adventure game, the player might jump from one audio portion to another, depending on the user's choices.

Security Tools

The goal of a content protection mechanism is to disclose the material only to authorized users. Today, this protection is achieved through a digital right management system that wraps the media file with a control mechanism that is usually based on classic security techniques like encryption and information hiding.
With encryption, the player has to know the cryptographic key (which should be kept secret to the user to avoid unauthorized usage or distribution) to play out the media file. Information Hiding is a technique that hides information inside a media file. It is achieved

through watermarking techniques as explained by Cox et al. (1997), where a watermarking key is used to generate a random sequence during the embedding process. The random sequence is used to spread out the hidden information (the watermark) into the media file. The knowledge of the watermarking key is essential to extract the watermark. The security of the watermark relies on the fact that it has to be: hidden, imperceptible, directly connected to the media content, statistically invisible, robust and tamper resistant. If all these properties are met, extracting and altering a watermark is hard with no knowledge of the watermarking key.

OUR PROPOSAL

In this section we present details of our proposal that aims at describing and storing an adventure game inside a 3GP file in a transparent, light, inter-operable and secure way, so that adventure games may be potentially played over any mobile audio device. Our focus is on adventure games for two main reasons: i) old games are getting new life thanks to their use over modern cellphones and adventure games were the most popular games in the 80s and ii) adventure games are well suited for the mobile scenario where system resources are limited, and where several devices are still released without any graphical screen (think of portable MP3 players).
The architecture we propose aims at transforming an audio mobile device into an adventure game console. As we mentioned, we seek for: *lightness*, *interoperability*, *transparency* and *security*. Lightness is essential for a scenario with limited resources and is achieved through MPEG7-DDL, which allows describing and synchronizing different media streams through a simple and light text-based description; Interoperability ensures that the produced file can be used over several different devices and is achieved through the usage of standard tools like MPEG7-DDL and the AAC format (for the encoding of the vocal narration); Transparency is ensured by the usage of the 3GP files. Security ensures that only a legal owner of the media file can play out the adventure game and also that alterations of the adventure data (or a part of it) are not allowed. It is guaranteed by a security mechanism designed using classic security tools such as watermarking and hash function.
In the following we show how a text-based adventure game is produced, protected and played out.

Production of the Adventure Games Script

Let us consider a simple *Knight Adventure*: *A knight enters into the castle. He is looking for a treasure. Three doors are before him: one leads to a death-trap; another to lion cage and only one is the door that has to be opened in order to reach for the treasure.* Although very simple, this story present a character (the knight) that has to take decisions (which is the right door?) in order to accomplish a task (to find the treasure).

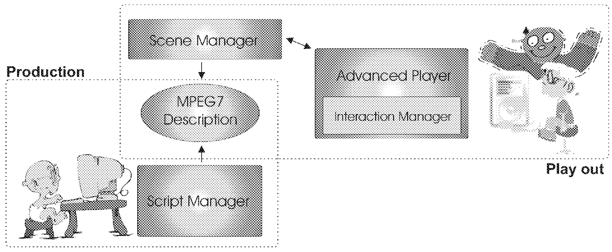


Figure 2: The porposed software architecture to produce and protect adventure games.

Users can affect the story development by interacting with the story. Since user's choices are unknown in advance, the adventure game plot has to contain all the possible story developments that can happen. Hence, multiple storyline and multiple user's choices have to be described in advance. Figure 2 shows the proposed software architecture where the script, scene and interaction manager cooperate to produce an adventure game.

Script manager
Since the story development depends on the user's choices, the story, along with all the possible story developments, is divided into basic audio chapters.
An audio chapter contains the vocal narration of a portion of the adventure and can be of four possible types: initial, interactive, sequential and ending. The initial chapter is the first audio part that is played out; only one initial chapter per adventure game is allowed. The interactive chapter allows users to interact with the story. The sequential chapter does not allow interaction (it just presents information to the user) and the ending chapter is the one that ends a storyline (note that multiple ending chapters may be present). For instance, our *Knight Adventure* may be composed of six different chapters:

- Ch 1. (*Initial*) A knight enters the castle. He is looking for a treasure.
- Ch 2. (*Interactive*) He has three doors before him. Left, right or center door?
- Ch 3.1. (*Ending*) He opens the door and fifty swords hit him.
- Ch 3.2. (*Ending*) He opens the door. A lion appears and roars at him. He tries to escape, but the Lion is already upon him.
- Ch 3.3. (*Sequential*) He opens the door and he takes the stairs up to the roof.
- Ch 4. (*Ending*) A box is partially hidden, but he spots it and he finally finds the treasure.

```
<AudioSegment>
 <label>"41"</label>
 <MediaTime>
   <MediaTimePoint>00:22:30</MediaTimePoint>
   <MediaDuration>00:00:10</MediaDuration>
 </MediaTime>
 <TextAnnotation><FreeTextAnnotation>
   He opens the door and fifty swords hit him.
   </FreeTextAnnotation></TextAnnotation>
</AudioSegment>
```

Table 1: MPEG7 description of an audio chapter.

Although very simple and with only one interactive chapter, three different stories with three different endings are possible (1-2-3.1, 1-2-3.2 and and 1-2-3.3-4).
Once the plot has been divided into several audio chapters, an MPEG7-DDL description has to take place. This description has to specify: i) all the audio chapters information, and ii) the choices offered to the user. Table 1 shows an example of an audio chapter description. The `audiosegment` tag is used to define an audio chapter; every audio chapter (identified with a unique label) specifies the beginning (`mediatimepoint`) and the duration of the audio segment (`mediaduration`). If available, a text description of the audio chapter may be specified through the `textannotation` tags.
In addition to the audio chapters description, all the possible user's choices have to be described too. To this aim, the script manager defines a table (the scene transition table); Table 2 shows how a text-based description can be used to handle possible user choices. Each entry is identified by the Chapter IDentifier number (num="X") and includes a possible question (for interactive chapters) and the possible chapter destinations. If the chapter is interactive, there are at least two possible audio chapter destinations; if the chapter is of an ending type, no chapter destination is present. If the chapter is sequential, a single destination is present.
Once the script manager has produced the text-based description, content protection has to be added.

Adventure Game Protection

The security mechanism is in charge of protecting the content with the following goals: i) every user can listen to a pre-defined story and ii) only a legal owner can enjoy the adventure game. To this aim, the vocal narration (AAC stream) is produced with the following rules:

- A single pre-defined storyline is linearly encoded so that the audio chapters that compose the story can be sequentially found from the first to the last (no interactive chapters are present). This story is in clear and any player can play it out.
- All the other audio chapters are encrypted and stored in the second part of the file. Without the decryption key, the play out produces audio noise.
- The first and the second part are separated by 120 seconds of silence, so that, if played with an ordinary player, the pre-defined storyline would not be immediately followed by audio noise.

```
<ROWSET>
     <ROW num="2">
        <QUESTION>What door would you like to open?
                  Left, center or rigth?</QUESTION>
        <DEST1>3.1</DEST1>
        <DEST2>3.2</DEST2>
        <DEST3>3.3</DEST>
    </ROW>
    <ROW num="3.3">
        <QUESTION></QUESTION>
        <DEST1>4.</DEST1>
        <DEST2></DEST2>
        <DEST3></DEST>
    </ROW>
    ...
 </ROWSET>
```

Table 2: A text-based description of the possible choices a user can take.

The audio stream (AAC format) and the adventure description (MPEG7-DDL format) have to be stored inside the 3GP file. As previously described, a 3GP file is organized through boxes. Hence, the AAC stream is stored inside a `trak` box, while the static MPEG7 description is stored inside the `udta` box. In this way, transparency and lightness and interoperability are achieved: thanks to the use of the 3GP file format, every player that can handle 3GP files can handle the AAC stream and the MPEG7 description.
To ensure security, the 3GP file has the following data watermarked in it, as shown in Figure 3:

- α, the key for the audio stream decryption; The key is watermarked in the first audio chapter and the player can retrieve it while playing it out. The key is watermarked as it has to be provided to the media player, but not the the user.
- k, the decryption key for the audio description A_{desc}; Again, the key is needed by the player to decrypt the adventure game description, but it has to remain unknown to the user. The key is watermarked inside the first audio chapter.
- $WID = H(E_K(A_{desc}))$ is watermarked to protect the audio from unauthorized alteration. It ensures integrity via a lightweight verification procedure that compares the hash of the whole adventure game description (encrypted) against the WID. Note that, for performance reasons, the use of stronger cryptographic tools is avoided.

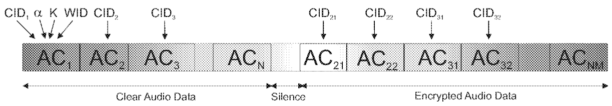


Figure 3: Information watermarked inside the 3GP file.

- CID, the audio chapter identifier (watermarked inside any audio chapter, with the exception of ending scenes); This ID uniquely identifies each chapter.

Data are hidden into the file using classical spread-spectrum techniques as the ones proposed by Cheng et al. (2002). These techniques require a watermarking key to spread out the data in the media file. Since the software player must know the watermarking key to read the watermarked data, the key is hidden in the player's code with suitable software engineering techniques (hence users don't know the key). To avoid illegal copies, each 3GP file is released for a specific instance of a software player.

Adventure Game Play Out

To enjoy the adventure game an enhanced player is necessary. In fact, if played with an ordinary player, only the unencrypted audio stream is rendered. The enhanced player is in charge of: i) playing the audio stream, ii) decrypting the encrypted data, iii) interacting with the user and iv) jumping from one audio chapter to another depending on the user's choices. As shown in Figure 2, to perform all these tasks, the player cooperates with the scene and with the interaction manager. The player retrieves the hidden data during the play out of the first audio chapter: the embedded watermark WID (for weak integrity verification), k (for decrypting the adventure game description), and α (for decrypting the second part of the audio stream). Right after the extraction, the player checks the integrity by computing $H(E_K(A_{desc}))$ and by comparing it with the retrieved WID. If the integrity check fails, reproduction is interrupted, otherwise the audio description and the second part of the audio track are decrypted.
Once the security check has passed, the rendering of the adventure game is done in cooperation with the interaction and the scene manager as described in the following.

Interaction manager
The interaction manager is in charge of handling the interactions between the user and the system. It is activated when an interactive audio chapter is played out; by using the scene transition table, it identifies the question to pose to the user and, using the answer and the scene transition table, it gives the next audio chapter CID to the scene manager. Note that the interaction interface depends on the available hardware: in cellphones the interactions are done through the keyboard, but more complex systems may be available (e.g., a pad, a voice or visual recognition system).

Scene manager
The scene manager is in charge of identifying the audio chapter to play out. It controls both the player and the interaction manager. By cooperating with the interaction manager, it gets the CID of the chapter that has to be played out; using this CID, it accesses the MPEG7-DDL description and finds out the corresponding audio segment in the AAC stream; it analyzes the audio-segment description and gives the play out timing information to the enhanced player.

CONCLUSIONS AND DIRECTIONS

In this paper we presented a software architecture to transform an audio device into an adventure game console. The proposed mechanism is provided with features like: i) Transparency (no modification to the 3GP file), ii) interoperability (3GP files can be managed by mobile device), iii) lightness (the MPEG7 description requires limited system resources), iv) security (the digital content is protected from illegal usage and modifications). Our proposal can be expanded with multi-modal recognition system (e.g., a voice recognition system, a point-and-click interface) and with multiplayer capabilities.

ACKNOWLEDGEMENTS

This work has been partially supported by the Italian M.I.U.R. under the MOMA initiative.

REFERENCES

ABI, 2006. *ABI Research - Market Research Report.* In *www.abiresearch.com [online].*

Brandenburg K., 1998. *Perceptual coding of high quality digital audio.* In *Applications of digital signal processing to audio and acustics.* Kluwer, 39–83.

Cheng S.; Yu H.; and Xiong Z., 2002. *Enhanced spread spectrum watermarking of MPEG-2 AAC audio.* In *Proceedings of the IEEE Int. Conf. on Acoustics, Speech, and Signal Processing.* 3728–3731.

Cox J.; Killian J.; Leighton F.T.; and Shamoon T., 1997. *Secure spread spectrum watermarking for multimedia.* In *IEEE Trans. Image Process.* vol. 6, 1673–1678.

Herre J. and Purnhagen H., 2002. *General audio coding.* In *The MPEG4 Book.* Prentice Hall.

Hunter J., 2001. *An overview of the MPEG-7 description definition language (DDL). IEEE Trans Circuits Syst Video Techn*, 11, no. 6, 765–772.

MPEG, 2005. *Information technology Coding of audio-visual objects Part 12: ISO base media file format. ISO/IEC 14496-12:2005.* In *www.chiariglione.com [online].*

Using Mobile Multi-Hop Ad-hoc Networks for Multi-Player Games

Oliver Wellnitz Lars Wolf
IBR
Technische Universität Braunschweig
Mühlenpfordtstrasse 23, 38106 Braunschweig, Germany
{wellnitz|wolf}@ibr.cs.tu-bs.de

ABSTRACT

Over the last couple of years, multi-player games have become more and more popular. Additionally, new mobile devices now have sufficient resources to play these multi-player games in mobile and wireless networks. While commercial mobile game consoles start to emerge, we take a step forward and take a look at future mobile gaming.

In this paper, we discuss the concept of playing multi-player games in mobile ad-hoc networks. Such networks are created spontaneously, can be used free-of-charge and have the ability to communicate beyond the range of a single wireless device. Here, we describe a general architecture for mobile gaming and analyse its challenges in three phases: Before the game starts, during the game and after the game. We then discuss possible solutions to these problems and present and refer to related work where possible. Furthermore, we introduce four reference scenarios for mobile multi-player gaming which can be used for a future evaluation of our approach.

1. INTRODUCTION

During the last decade, computer games evolved from single-user games to multi-player games which allows people to play with or compete against each other. This development contributed to the huge success of computer games for the last couple of years. Today, small and mobile devices have become very popular and the trend towards mobile devices also has an impact on computer games. Some examples are handheld gaming devices like Sony's Playstation Portable (PSP) or Nintendo's Dual-Screen (DS). Both devices contain wireless communication equipment that allows for multi-player gaming. In addition to games that exist today, multi-player games on mobile devices allow their users to play anywhere at any time. They also have the chance to introduce new kinds of games to the market such as mixed reality or location-based games that make use the player himself in the real world instead of relying on avatars in a virtual world.

To achieve communication between mobile devices, several different technologies such as GSM, UMTS, Bluetooth, Wireless LAN, or others can be employed. However, all wireless technologies share common problems like attenuation, fading, shadowing, reflection, scattering, and diffraction.

There are two different basic forms of mobile communication. Infrastructure-based communication requires often fixed base stations which relay messages to other mobile nodes and also act as a coordinator for the wireless network. Secondly, ad-hoc communication allows for direct communication between the mobile devices. Today, mobile gaming in infrastructure networks mainly uses cellular networks to play games on mobile phones or a private WLAN access point at home to connect laptops or consoles.

In this paper, we propose the use of a mobile multi-hop ad-hoc network to connect the players of a multi-player game without the need for any network infrastructure. Mobile ad-hoc networks are created in a spontaneous manner whenever two devices move within communication range of each other and can be used free of charge. Furthermore, mobile ad-hoc networks can support communication to remote devices by using intermediate nodes as relays to forward data to their destination. Thus, multi-hop ad-hoc networks can grow beyond the range of a single wireless transmitter by using a fair cooperation between mobile devices.

In general, multi-player games for mobile devices often do not take the different networking environment into account. Wireless networks are more error-prone than wired networks like the Internet. Thus, existing approaches like a central game server for a multi-player game are unsuitable for networks with such a dynamic environment. On the other hand, a fully distributed approach like peer-to-peer networks does not make efficient use of the available bandwidth, makes cheating in a game easier and does not take differences in terms of resources of the mobile devices into account. By using a distributed server architecture, the game can create the necessary redundancy while keeping network requirements at a minimum. Each game server is responsible for game clients in his vicinity to which it can communicate efficiently. In [10], we proposed a game architecture that introduces the idea of zone servers, which are responsible for a certain area (zone) of the network. Zone servers are not independent devices. The game server software runs on a player's mobile device in addition to the game client. Zone servers communicate in a peer-to-peer fashion with each other. Although, due to the knowledge of the game rules, they are able to make local decisions on their own and delay and aggregate or omit information to other servers. We proposed that these game servers should be determined from the group of players and that this selection should be made based on the capabilities and performance of the mobile device as well as its position in the ad-hoc network. Other possible factors may include mobility information, battery level and energy consumption or the player's trustworthiness to act as a game sever.

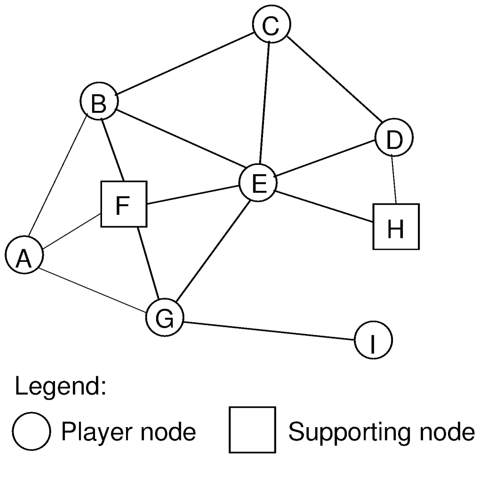


Figure 1: Mobile Ad-hoc Network

Multi-player games can be distinguished in several categories. First-Person Shooters (FPS) or sports games like car racing are fast, reaction-based games with high network requirements regarding delay or latency. Role-playing Games (RPG) are usually round-based games with low network requirements and Massively Multi-Player Online Role-Playing Games (MMORPG) support tens of thousands of concurrent players. Their problem mainly lies with scalability issues on the server side. Our goal is to support all kinds of multi-player games in mobile environments that can be played in the Internet today and establish a game architecture that also allows for the creation of new kinds of mobile games such as mixed reality games. The only exception is made for MMORPGs because we do not expect that an infrastructure-less network will be able to cope with thousands of users at the same time. For the purpose of this paper, we focus on mobile ad-hoc networks only and do not rely on any infrastructure network or the Internet. Although, it is possible to integrate such a connection in the future.

Figure 1 shows an example of a mobile ad-hoc network in which vertices represent mobile nodes and edges denominate the possibility of a direct wireless communication between two nodes. For this paper, we assume that every link between two nodes is bidirectional.

We distinguish two different types of nodes participating in the game: Players participate in the game (player nodes) while other nodes cooperatively forward traffic in the network but otherwise do not have knowledge about the game (supporting nodes). Supporting nodes may use the network for other applications and may also rely on player nodes to forward their traffic to its destination. For purpose of this paper, we use the terms user and node interchangeably.

The remainder of this paper is structured as follows: Section 2 discusses likely scenarios for mobile gaming in multi-hop ad-hoc networks. Section 3 shows the three phases of a mobile game and examines the problems in each phase. Finally, Section 4 concludes this paper.

2. MOBILE GAME SCENARIOS

It is safe to assume that not in every situation players will rely on mobile multi-hop ad-hoc networks for their connectivity. If only a few players in close vicinity play together, they may choose to directly communicate with each other in a peer-to-peer fashion. The same is true when they can use exisiting infrastructure free of charge, like a wireless access point at home. But if the number of players gets higher and/or the players are distributed over a larger area, the use of mobile multi-hop ad-hoc networks is more likely to be beneficial.

Figure 2: An InterCityExpress at a train station

In general, the network characteristic between mobile nodes depends on various properties of the mobile nodes like their number, position, or speed as well as the attributes of their ambient environment such as noise/background traffic, the possibility of line-of-sight communication, or the attenuation of the materials found there.

In order to ascertain the feasibility and to determine the performance of protocols and mechanisms for games in mobile ad-hoc environments, all these different characteristics have to be taken into account. Hence, specific and realistic scenarios which allow us to identify all parameters mentioned above are mandatory. In the following, we define four reference scenarios where mobile games will most likely be played. All scenarios are based on the assumption that players remain nearly stationary while playing games. We only allow one exception to this rule for new types of mobile games where players have to move around in order to interact with the game.

2.1 Scenario A: Train station

This scenario involves a medium-sized area, e.g. $100 \times 250\, m^2$. People in train stations usually walk around, wait in designated waiting areas, or wait on a platform. The group of players only comes from the group of waiting people while supporting nodes may also be carried by people roaming around. This scenario contains a number of attractors such as the rail service center, the shops, the entries and exists, and the plattforms. When moving around, people usually travel on foot, hence we assume a maximum speed of $2\,{}^{m}/_{s}$. Meanwhile, other mobile nodes in trains that pass through the train station can have speeds up to $25\,{}^{m}/_{s}$ Players as well as other people carrying supporting nodes walk in the hallways or on the plattforms.

When discussing wireless communication at train stations, some particular problems have to be taken into account. Besides the usual obstacles for wireless communication such as

Figure 3: The inside of an InterCityExpress

walls or other people, a commercial wireless infrastructure may exist at train stations (e.g. one or more WLAN access points that can be used to access the Internet). Any communication with this infrastructure interferes with game transmissions if both utilize the same frequency. Wireless communication between platforms is impaired if a train blocks the line of sight between two devices. Additionally, electromagnetic noise from the overhead line and other electric equipment also has a negative effect on wireless communication.

Generally, people do not stay very long at train stations. They either get on or off the train. Therefore, we expect the game time for one player usually to last no longer than 15 minutes. However, train delays could increase that number.

2.2 Scenario B: Train ride

On the train, the scenario size is approx. $3 \times 175\,m^2$. In this scenario, two different situations should be distinguished. First of all, during the train ride, people usually sit at their seats and do not move often. The second situation is a stop at a train station when people get on or off the train and in the following minutes while they make their way through the coaches. So again, players are not expected to move during their game and the mobility of supporting nodes is also low as long as the train is on the move. Again, people travel on foot, so we assume a maximum speed of $2\,{}^{m}/_{s}$. Generally, we conclude as far as node mobility is concerned the train scenario is of rather static nature.

The area in this scenario is quite long but not very wide which means from a communication point of view that data always has to travel along the axis of the train. There is usually no alternate route available so that e.g. game traffic may have to be routed through congested areas in order to reach the players. Another problem for wireless networking is that carbodies are made of metal mostly aluminium or steel. This leads to attenuation of the wireless signals to stations in other coaches. To communicate between nodes at both ends of the train could require up to 10 hops. Again, commercial access points may be present in trains which may lead to an increased usage of ISM-band frequencies. Finally, other nodes outside the train may pass by at relative speeds of up to $100\,{}^{m}/_{s}$.

A train ride usually lasts longer than ten minutes with exceptions for local trains or subways. In trains, games could also last several hours with some users playing the whole time while other leave or join late in the game as the get on or off the train.

Figure 4: A school yard

2.3 Scenario C: School yard

For the school yard scenario, we assume a larger rectangular area of $250 \times 250\,m^2$ and define it as an open outdoor space. The main difference between the previously mentioned scenarios is the lack of supporting nodes and the grouping of players. For this scenario, we assume that students who turned on their mobile nodes are playing games. So there are no additional supporting nodes. Secondly, we think that students will stand or sit in groups together rather than playing from different positions distributed across the school yard. We define these groups as clusters of players or mobile nodes. On a school yard, several player clusters can evolve for two different reasons: Different clusters of players can play different games or clusters are formed up of members of the same team which compete against other teams in the same game. With players remaining stationary and no supporting nodes, this scenario is static. However, students can decide to switch on their mobile devices and join the game or players may decide to leave the game.

2.4 Scenario D: Mixed reality and location-based games

Mobile devices allow the player himself to be integrated into the game thus leading to new kinds of game types. Mobile games are able to integrate the player's location as well as his heading, speed, position, or other properties in the gameplay. Examples of these new type of games are the human pacman game[3] or a first person shooter in which players use sensor-equipped plastic guns to hunt virtual monsters in a park[6]. A scenario for these new types of games could range from small parks with an edge length of a few hundred meters to the size of a larger city covering several square kilometers. To help interaction between the players, it may be sufficient if mobile devices are able to directly communicate with each other in a peer-to-peer fashion. However, the game may require some kind of global game state, e.g. the players position, that must be transmitted to all players in the game. Hence, an infrastructure network and/or mobile ad-hoc network may be necessary. Players in this scenario can move at various speeds as they walk, take the bus or even drive a car.

3. MOBILE GAME CYCLE

In this section, we will split up a game into three phases. Each phase has distinct problems which we will discuss in detail. In the pre-game phase, players have to find each other, agree on a game to play and set up the necessary software and infrastructure to play the game. During the game, a game architecture has to deal with communication errors and link failures as well as game server redundancy and the mobility of the players. After the game, the results of the game such as the high-score can be uploaded to the Internet.

3.1 Creating the Game - The Pre-Game Phase

The pre-game phase is the first of three phases of a game. We assume that at least two players are equipped with mobile devices and participate in the same wireless ad-hoc network. All devices within this network can either communicate directly with each other or are able to forward data through supporting nodes.

3.1.1 Player Discovery

Before a game can be played, a user has to find one or more other users in his neighbourhood with whom he agrees on a game. This neighbour discovery can be implemented as a broadcast or multicast request which is sent into the ad-hoc network. A mobile node should either send a request using a low, fixed maximum hopcount (TTL) or by employing a mechanism which increases the hopcount for each subsequent discovery request if it did not receive an answer to its previous request in time. Both strategies are suitable to keep the overhead low if the ad-hoc network is large. Whenever a discovery request is received by a mobile node whose user is interested in playing a multi-player game, the mobile node should send a discovery reply message to the originator of the request. This reply should at least contain information about the sender but may also contain information and preferences about other mobile nodes which users share the same interest in gaming. We believe that a player discovery mechanism should focus on finding players in general rather than specific games because while a user might not find teammates for the game of his choice, a group of users may find a common interest and agree on a game to play. Konark[5] was developed as a discovery protocol for mobile ad-hoc networks and could be used for this purpose. Also, a simple chat application which is already used in today's multi-player games to let the player's discuss the parameters and settings of the game should be implemented to help users to find suitable partners for a game.

3.1.2 Game Software Distribution

Another problem is the availability of the game software itself at the mobile nodes. Unlike with multi-player games on the Internet, a user will probably find only a few other users in his network, e.g. in a long-distance train. While they could all possibly agree on the type of game to play to spend their time during the long ride, the group may have a disjunctive set of games installed on their mobile devices. One possibility to solve this issue is to buy the needed software over an wireless infrastructure connection, e.g. UMTS, from a vendor on the Internet. However, this method relies on network infrastructure to be present and available. But the most important downside of this idea is that the cost of such transaction which includes not only the costs for the game software itself but also the costs for communicating through the wireless infrastructure network is most likely to be considered to high for a regular player. Another idea is to include several licenses with a multi-player game, so that the owner of the game software is allowed to invite a certain number of other players to his game. The software could then be copied freely in the ad-hoc network up to the maximum number of available licenses but may only be used if the owner of the software is also present.

3.1.3 Assigning Game Servers

We now assume that our user has found a group of other players, that they agreed on a game and that the game software is installed on all mobile nodes. Before a game can be started, suitable nodes should be assigned as servers for the game. As previously discussed in Section 1, we do not consider peer-to-peer games for mobile ad-hoc networks. In [11], we introduced a distributed dominating set algorithm for game server selection which uses local information to determine well-suited mobile nodes from the group of players. This algorithm calculates an initial game server set and can adjust to network changes during the game. With our algorithm only players of that game are eligible to play the role of game server because we believe that a user not participating in the game will not be willing to spend battery power and other resources of his mobile device to support the game. Also, each instance of a game will have its own game servers for the same reason. In a mobile ad-hoc network, mobile nodes can fail for various reasons. Nodes can move out of the wireless range of other nodes, their connection could be temporarily impaired, or a mobile device could simply run out of battery. Hence, for redundancy reasons at least two nodes should act as server for the game. However, in static scenarios such as the school yard scenario mentioned in Section 2, a single game server might also be sufficient if no other suitable game server can be found.

After the game server infrastructure is set up, the game can be started.

3.2 Playing the Game - The In-Game Phase

During the game, communication between mobile devices may be impaired for several reasons. A robust game architecture should be able to cope with mobility of users, temporary network congestion, and changing background noise. The mobility of devices can either apply to players which currently have the role of a game server, players that have no special role in the game, or supporting nodes. Our definition of mobility also includes significant changes in the environment that have an effect on the characteristic of the network such as bit error rate or signal strength as this change is indistuingishable from node movement by a mobile node measuring network connectivity to other nodes.

3.2.1 Mobility Issues

One should keep in mind that with our zone server architecture, game servers and their respective clients usually reside in close vicinity of each other. Players are able to switch to another game server whenever they experience degradation from the network. As game servers exchange information to keep the game synchronised, players should have little problem in restoring their network connection by using another server and continue playing the game. To ensure a smooth transition from one server to another, players

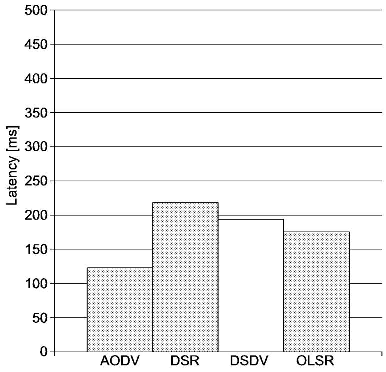


Figure 5: Ad-hoc Routing Protocol Latency

should continuously monitor the network conditions to their servers. Whenever the connection degrades below a threshold, the node should find a suitable new server and establish a backup connection. This way, the player can speed up the handoff to another server. In order to be helpful, the threshold mechanism must ensure that a player does not switch back and forth between game servers.

Game server movement effects not only a single but all players connected to that server. Additionally, it can also influence the synchronisation between the moving game server and the other game servers. If a game server moves away, e.g. its user gets off a train, all players attached to this game server will either notice a decrease in network performance when communicating with their game server or in case of fast fading, the game server will vanish from the network. In both cases, players will use the method described above to reconnect to another server of the game. The number and the location of players can change during the game if players decide to leave while others may decide to join during the game. Hence, the number of game servers should also adjust to the number of players, their location in the network, and the connectivity inside the ad-hoc network. For this reason, our game server algorithm[11] runs continuously during the game and determines new game servers if necessary. Game servers must give up their server role when they have no more clients. The game itself is responsible for defining how many game servers are necessary as this requirement varies from game to game.

The movement of supporting nodes does not affect the game directly. But in a mobile ad-hoc network players and game servers may need to use a number of supporting nodes to transmit their game data to its destination. While we can not prevent that supporting nodes move, a robust architecture must ensure that a new path between communication partners can be reestablished swiftly. In mobile ad-hoc networks, this responsibility lies with the ad-hoc routing protocol. Several papers have analysed the network requirements for games regarding packet loss, latency and other parameters[1, 4, 7, 9]. They found that the packet loss ratio should be generally kept under 5 % and latency should be below $100 - 200\,ms$ although these values depend on the type of game. We have done an initial evaluation with the network simulator NS-2 that suggests that the reactive Ad-hoc On-Demand Vector routing protocol (AODV)[8] as the best candidate for mobile gaming (see Figure 5).

It goes without saying, that prediction of mobility using a threshold based mechanism cannot deal with all kinds of wireless problems. The previously mentioned fast fading, for example, is hard to forecast and any mobile game may introduce additional game-specific mechanisms to hide such problems from the user. Also, extended time periods without any connection to the game must be dealt with accordingly so that the remaining players can continue to play the game in a fair and uninterrupted way.

3.2.2 Quality of Service

Game traffic can also be influenced by other traffic which is sent by supporting nodes through the ad-hoc network. The rules of a multi-hop ad-hoc network state that a node that expects other nodes to forward its data should also in return be willing to forward theirs. So while the supporting nodes forward our game traffic, they might also want to send data of their own. This data could come from other applications as well as other games. Examples of other applications are accessing the Internet through an access point or exchanging documents with a business partner. Contention in the network can lead to increased latency, jitter, and data packets being dropped which can have a huge impact on the playability of a game. Applying quality of service (QoS) mechanisms in these situation such as priority queuing and rate control, as well as broken link detection can help to deliver real-time game traffic more reliable and on-time while preventing other applications from starving. In [2], QoS modifications for multi-player games in mobile ad-hoc networks were discussed in detail. It shows that such modifications can be used to ensure that the packet loss ratio is kept below 5 % and latency and jitter is kept below $25\,ms$ for connections up to three hops even if there is significant background traffic.

3.2.3 Game Splits & Merges

During a game, two or more groups of players can become separated from the network. This can happen if a group of players gets off the train at a train station or a cluster of players on a school yard looses their network connection to the rest of the network. In cases where a group does not contain a game server, the players in that group cannot continue to play the game. However, if two or more groups can still communicate with a game server in their neighbourhood, the game in one group continues independently of the games in the other groups. Such game splits are easy to detect with a simple time-out-based mechanism for the game server synchronisation. If the game is permanently split, it is possible for the game to continue with a reduced number of players. For temporary game splits, it may be a nice feature to be able to merge two or more games again.

A game merge would heavily rely on the type of the game and the game data structure that must be synchronised. However, a game synchronisation mechanism has to deal with small inconsistencies in the game state anyway and it may be able to support game merges up to a certain point.

Finally, it should be pointed out that for games that last longer than a few minutes, game developers should ensure that players are allowed to join a game that is already in progress because of the volatile nature of mobile networks.

3.3 After the Game - The Post-Game Phase

After the game has finished, players can decide to continue playing another round of that game using the same players and game infrastructure that is already established, they can pick another game to play or they could choose to quit playing. Nevertheless, storing the outcome of the game such as the high score in a persistent way can prove useful. Players can utilize existing infrastructure to store the results of the game, such as the high-score, on the Internet. In situations where no wireless infrastructure network is available, the mobile device can store this information and try it again later, e.g. by sending it through the player's WLAN access point at home.

Besides high scores, massively multi-player games (MMORPGs) could be enhanced with small mobile multi-player games. For example, one could allow the user to download his MMORPG character to a mobile device and take it with him. A separate mobile multi-player game could then allow players to trade items or fight with each other during a train ride without any connection to the online game itself. The results of this fight or the trade could be then fed back into the MMORPGs allowing the player to virtually participate in an online world even without a permanent connection to the Internet.

4. CONCLUSION

In this paper, we proposed a concept for mobile multi-player games by using multi-hop ad-hoc networks. For this, we have introduced four reference scenarios where we believe multi-player games are most likely to be seen in the future. We also examined the challenges of mobile gaming in all three phases of a game and presented algorithms for these problems based on related work where possible. Otherwise, we suggested practicable solutions.

Creating network support for fast-paced games in mobile ad-hoc environments is not an easy task. This remains true even if you only restrict yourself to the games that are played on the Internet today. Nevertheless, we are convinced and believe to have shown that ad-hoc networks are a viable and cost-effective method for communicating data for mobile multi-player games.

In the future, we will further refine our ideas and design algorithm and protocols that will reflect the suggestions presented in this paper. Further on, we plan to evaluate our concept using the mobile gaming reference scenarios from this paper.

5. ACKNOWLEDGEMENT

The photo in Figure 2 was taken by Greg O'Beirne. The photo in Figure 3 was taken by Reinhard Kraasch. Used with permission.

6. REFERENCES

[1] T. Beigbeder, R. Coughlan, C. Lusher, J. Plunkett, E. Agu, and M. Claypool. The Effects of Loss and Latency on User Performance in Unreal Tournament 2003. In *Proceedings of the 3rd Workshop on Network and System Support for Games*, pages 144–151, Aug. 2004.

[2] D. Budke, K. Farkas, O. Wellnitz, B. Plattner, and L. Wolf. Real-Time Multiplayer Game Support Using QoS Mechanisms in Mobile Ad Hoc Networks. In *Proceedings of the 3rd Annual Conference on Wireless On demand Network Systems and Services (WONS 2006)*, Les Ménuires, France, Jan. 2006.

[3] A. D. Cheok, S. W. Fong, K. H. Goh, X. Yang, W. Liu, and F. Farzbiz. Human pacman: A sensing-based mobile entertainment system with ubiquitous computing and tangible interaction. In *Proceedings of the 2nd Workshop on Network and System Support for Games*, May 2003.

[4] M. Dick, O. Wellnitz, and L. Wolf. Analysis of Factors Affecting Player's Performance and Perception in Multiplayer Games. In *Proceedings of 4th International Workshop on Network and System Support for Games (Netgames 2005)*, Hawthorne, USA, Oct. 2005.

[5] S. Helal, N. Desai, V. Verma, and C. Lee. Konark – A Service Discovery and Delivery Protocol for Ad-hoc Networks. In *Proceedings of the Third IEEE Conference on Wireless Communication Networks (WCNC)*, New Orleans, Mar. 2003.

[6] K. Mitchell, D. McCaffery, G. Metaxas, and J. Finney. Six in the city: Introducing real tournament: A mobile ipv6 based context-aware multiplayer game. In *Proceedings of the 2nd Workshop on Network and System Support for Games*, May 2003.

[7] L. Pantel and L. Wolf. On the impact of delay on real-time multiplayer games. In *Proc. of the 12th international Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, pages 23–29, New York, USA, 2002.

[8] C. E. Perkins, E. M. Belding-Royer, and S. R. Das. Ad hoc On-Demand Distance Vector (AODV) Routing. RFC 3561, Nokia Research Center, University of California, University of Cincinnati, July 2003.

[9] P. Quax, P. Monsieurs, W. Lamotte, D. D. Vleeschauwer, and N. Degrande. Objective and subjective evaluation of the influence of small amounts of delay and jitter on a recent first person shooter game. In *Proc. Workshop on Network and System Support for Games*, pages 152–156, 2004.

[10] S. M. Riera, O. Wellnitz, and L. Wolf. A zone-based gaming architecture for ad-hoc networks. In *Proceedings of the Workshop on Network and System Support for Games (NetGames2003)*, Redwood City, USA, May 2003.

[11] O. Wellnitz and L. Wolf. Assigning Game Server Roles in Mobile Ad-hoc Networks. In *Proceedings of the 16th ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'06)*, Newport, USA, May 2006.

AUTHOR LISTING

AUTHOR LISTING