

Game AI: The shrinking gap between computer games and AI systems

Alexander Kleiner

Institut für Informatik
Universität Freiburg
79110 Freiburg, Germany
kleiner@informatik.uni-freiburg.de

Abstract. The introduction of games for benchmarking intelligent systems has a long tradition in AI (Artificial Intelligence) research. Alan Turing was one of the first to mention that a computer can be considered as intelligent if it is able to play chess. Today AI benchmarks are designed to capture difficulties that humans deal with every day. They are carried out on robots with unreliable sensors and actuators or on agents integrated in digital environments that simulate aspects of the real world. One example is given by the annually held RoboCup competitions, where robots compete in a football game but also fight for the rescue of civilians in a simulated large-scale disaster simulation.

Besides these scientific events, another environment, also challenging AI, originates from the commercial computer game market. Computer games are nowadays known for their impressive graphics and sound effects. However, the latest generation of game engines shows clearly that the trend leads towards more realistic physics simulations, agent centered perception, and complex player interactions due to the rapidly increasing degrees of freedom that digital characters obtain. This new freedom requests another quality of the player's environment, a quality of ambient intelligence that appears both plausible and in real time. This intelligence has, for example, to control more than 40 facial muscles of digital characters while they interact with humans, but also to control a team of digital characters for the support of human players.

This article emphasizes the current difference between AI systems and digital characters in commercial computer games and emphasizes the advantages that arise if shrinking the gap between them. We sketch some methods currently utilized in RoboCup and relates them to methods found in commercial computer games. We show how methods from RoboCup might contribute to game AI and improve both the performance and plausibility of its digital characters. Furthermore, we describe state-of-the-art game engines and discuss the challenge but also opportunity they are offering to AI research.

1 Introduction

The introduction of games for benchmarking intelligent systems has a long tradition in AI (Artificial Intelligence) research. Alan Turing was one of the first to mention that a computer can be considered intelligent if it is able to play chess [17]. Today AI benchmarks are designed to capture difficulties that humans deal with every day. They are carried out on robots with unreliable sensors and actuators or on agents integrated in digital environments that simulate aspects of the real world.

Within the modern approach, AI is considered as being embedded into an environment and measured by the success or failure of its interactions. The basic idea behind is captured by the model of an intelligent agent [15]. An agent is an entity that is equipped with sensors and actuators. During discrete time steps, the agent receives information from the environment with its sensors and subsequently performs actions that change the environment's current state. From this perspective, there are new challenges for autonomous agents which are similar to challenges humans accomplish every day:

- The outcome of actions is non-deterministic, consequently the agent has to learn a model for their prediction.
- Observations are noisy and ambiguous, consequently the agent has to build an internal representation of the world, known as a *world model*.
- The environment is dynamic, consequently the agent has to interact in real time.
- The environment contains more than one agent, consequently the agent has to compete or cooperate with other agents.

These problems are of particular interest within the RoboCup competitions. RoboCup is an international research initiative that encourages research in the field of robotics and Artificial Intelligence with a particular focus on developing cooperation between autonomous agents in dynamic multi-agent environments. AI systems developed in this context have to provide efficient and robust solutions in order to deal with the difficulties listed above.

In contrast, computer games are not affected by these problems. Digital environments are free of noise and thus deterministic. Also, the problem of cooperation does not necessarily arise since game characters are usually executed on a single computer. Due to this simplicity, digital characters can sometimes completely be described by a set of *trigger mechanisms* that execute pre-defined behaviors if specific conditions are met. In a typical First Person Shooter (FPS) game, for example, digital characters are programmed to shoot and run towards their opponents if they appear in range, and to *freeze* otherwise. They do not account for the structure of the environment as it would be necessary, for example, for reducing their damage. However, the recent development of computer games shows that *modern* games are equipped with realistic physics engines¹, which require sophisticated algorithms for controlling game characters. Furthermore, human players are expecting the opposing Artificial Intelligence to be plausible. In fact, they want to be able to understand and to predict the decisions of their artificial opponents.

¹ A piece of software that performs the physics simulation within a game

Computer games have a large potential for development. In 2001 the US computer game industry's business volume was with 9.4 billion USD for the first time higher than that one of the film industry with 8.3 billion USD. This enabled an exponential growth of the performance of computer graphics boards: In 1995 a bulk graphics card rendered approximately 300.000 polygons per second, in 1999 already 3.000.000 and in the year 2000 even 100.000.000 polygons per second.

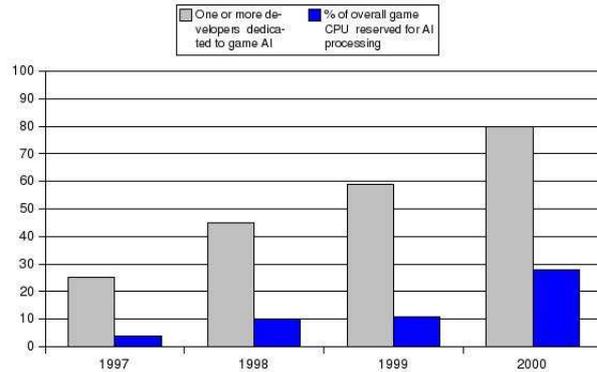


Fig. 1. AI in computer games (data originally collected by Steve Woodcock [20])

Moreover, in the last few years, game developers paid increasing attention to incorporating AI into computer games, as shown by a survey (figure 1) from the Game Developer Conference (GDC) [20], but also AI researches have introduced games as a serious challenge to AI [11,10].

This article emphasizes the current difference between AI systems and digital characters in commercial computer games and emphasizes the advantages that arise if shrinking the gap between them. On the one hand, we sketch some successful solutions of "first-prize-winning" teams, such as the *CS-Freiburg* team [19] and the *ResQ-Freiburg* team [6] and discuss their applicability to computer games. We show how methods from RoboCup might contribute to game AI and improve both the performance and plausibility of its digital characters. On the other hand, we describe state-of-the-art game engines and discuss the challenge but also opportunity they are offering to AI research.

Following we will discuss in section 2 and section 3 some methods utilized by the *CS-Freiburg* [19] team and the *ResQ-Freiburg* [6] team, respectively. In section 4 we describe game engines that are already utilized for AI research and finally we conclude in section 5.

2 AI in RoboCup soccer

In this section we describe techniques utilized by the *CS-Freiburg* team [19], which participated from 1998 to 2001 in the middle size league. In this league a maximum

number of four autonomous robots per team, with a footprint not greater than $2000mm$, compete on a 9×5 meters field and play a game that lasts for 2×10 minutes. The particular challenge in this league is to cover a wide spectrum of research issues ranging from robotic hardware development and low level sensor interpretation to planning and multi-agent coordination. One of the five CS-Freiburg robots is shown in figure 2(a).

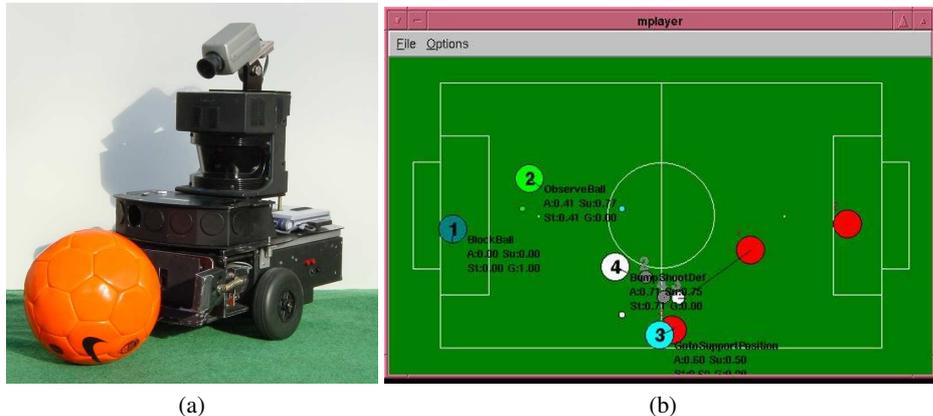


Fig. 2. (a) A CS-Freiburg field player, (b) a world model generated from Laser Range Finder (LRF) and camera data. The small circles represent estimates of the ball position from each robot. Large and red circles represent opponents, whereas large and enumerated circles represent teammates

2.1 Perception and World Modeling

Perception and world modeling are probably two of the most important and also most difficult tasks a robot has to perform. Generally, the correct interpretation of sensor data is difficult due to two major problems. Firstly, sensors return a *noisy* signal, which means that their return is randomly distributed around the true value; in hard cases they might even be completely wrong. Suppose a camera provides images from a soccer field and the task of the robot is to report the position of the ball. An efficient approach is to detect the ball in the image by color, which is pre-defined as red for the ball within RoboCup soccer. However, if taking pictures with a digital camera in such an environment, one can easily see that the ball appears seldom to be red. Due to shadows and reflections of the illumination, the color of the ball can be somewhere between white, black and red. A sensor interpretation module has to be capable of dealing with such noise in order to detect the total shape and position of the ball reliably.

Secondly, the information provided by a sensor can be highly ambiguous since a sensor produces the same data within different contexts. Suppose the robot's camera points towards the spectators in order to find the ball. If one of the spectators wears a red T-shirt, the sensor might possibly return the spectators position as a hypothesis for the ball. In most cases the process of sensor interpretation yields more than one hypothesis, each weighted by a probability. Due to ambiguous hypotheses generated by the sensors, it is necessary to combine all of them to a consistent one. We call the consistent hypothesis *world model* which typically includes the positions and velocities

of the robot and objects around it. Generally, it does not suffice to build a world model from sensor perception only. If, for example, parts of the world are not observable by the robot for some time, it becomes necessary to integrate sensor readings continuously in order to memorize perceptions from the past.

Techniques that do integrate observations over time are known, among others, as *Bayesian Filtering* techniques, including the *Kalman Filtering* [13] and the *Markov Localization* [5] approach. Both techniques are commonly utilized by autonomous systems. Figure 2(b) shows a world model resulting from the cooperative integration of sensor information by four CS-Freiburg robots during a soccer match. The figure shows the individual hypothesis of each robot for the ball position (small, grey circles) and the integrated hypothesis (the small, white circle) constructed from the individual hypotheses by a Bayesian approach.

Bayesian world modeling can also be performed by game characters in digital environments. One advantage in this context is that the hypothesis space (i.e. the space of all possible hypotheses) and the sensor input space (i.e. the space of all possible sensor values), can be reduced significantly to a size appropriate to the situation. Additionally the influence of noise can be adjusted or even totally left out. Nowadays most of the games on the market are "cheating" because they equip their characters with supernatural sensors that, for example, are capable of localizing the opponents every time, even they are out of sight or far away. However, human players are generally sensitive to characters with supernatural knowledge, since humans are capable of learning and inferring the knowledge their opponents can have.

A world model built with a Bayesian approach, however, represents exactly the knowledge an agent maximally can have, given all sensor readings from the past. Therefore agents that successively build a world model with a Bayesian approach are more likely to be plausible and realistic to human players. Laird and colleagues did preliminary experiments with a game character, known as *Quake bot*, that was capable of anticipating human players by inferring the position they will reach in the future [10]. Le Hy and colleagues reported first results from teaching game characters to select actions based on a Bayesian approach [7].

2.2 Action selection and execution

Soccer robots are capable of executing primitive actions, such as their rotation and translation or triggering a mechanism for kicking the ball. Due to the complex dynamics of robots, success or failure of simple tasks, such as shooting a goal or dribbling a ball, depends on the execution of a sequence of actions rather than on the effect of a single action. In order to tackle this difficulty, robot control is usually performed by a three-level hierarchy. This hierarchy is composed of primitive actions at the bottom level, of *skills*, which are complex behaviors that execute chains of actions, at the middle level, and of an action selection layer that selects skills with respect to the current situation at the top level.

Skills are small programs that can be initiated in particular situations and terminate after they have succeeded or failed. One can realize the selection of skills by a set of *IF...THEN* clauses that provide for each situation an appropriate skill that the robot has to perform. However, with increasing complexity of the environment, effects of actions

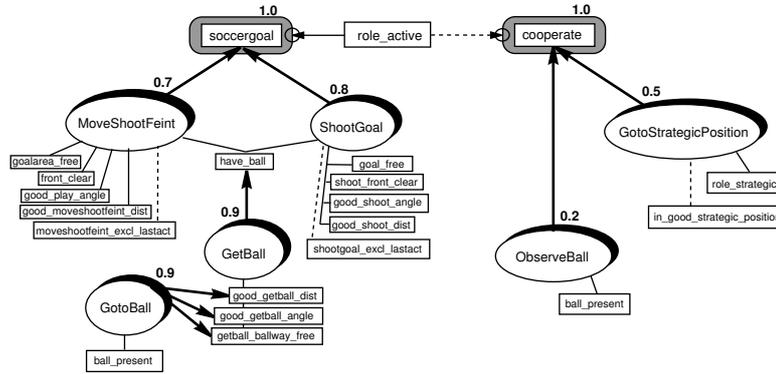


Fig. 3. A part of the behavior network used by the CS-Freiburg robots

become less predictable and less reliable. Therefore it turns out to be crucial to describe those effects by probabilities. The CS-Freiburg team utilized *Extended Behavior Networks* (EBNs), which allow for defining probabilities to the outcome of skills. Figure 3 shows a part of the behavior network used by the robots. One can see the skills (round shapes) and their pre-conditions (boxes) that allow or deny their execution. Given a particular situation, the network returns the skill that leads most likely to a success.

We have also shown that adaptive capabilities, for example gained by online reinforcement learning, can be of advantage, particularly if the world changes rapidly [9]. In our approach the robot improved simultaneously both the action selection and the execution of the skills during its lifetime. If the environment suddenly changes, the robot was capable of adjusting its skills accordingly.

In comparison to real robot systems, computer games are usually based on an approach that does not account for uncertainty and also deals with less complexity than found in real environments. One popular approach is to assemble a large collection of action sequences (comparable to skills) that are executed within a particular situation. This technique is known as *scripting* [16] and is found in most recent computer games. Depending on the game, such a sequence might sometimes even be tailored for a single situation, for example: A player approaches for the first time a particular position in the digital world, subsequently a set of artificial characters around this location starts to talk to each other and leaves afterwards. However, it is clear that this kind of technique can only work within a predictable story line. With the continuous improvement of the physical simulation as well as the increasing depth of human-computer interactions within games, the complexity of digital worlds will grow and become less predictable by simple scripting mechanisms.

In contrast to real robot systems, game environments offer the possibility for collecting a huge amount of real world experiences by exploiting their existing infrastructure. Computer games are usually designed to connect with hundreds of artificial or human characters on the Internet. With this, machine learning, which generally on real robots

lacks sufficient data and learning examples, could reach a new quality and performance in the context of both human-machine interaction and robot control.

2.3 Multi-agent cooperation

The cooperation between robots can be separated into two parts which are cooperative sensing and cooperative acting. The CS-Freiburg robots perform cooperative sensing by communicating their locally generated world models among each other. By this, each robot is able to construct a global world model and to be aware of things that happen outside of its field of view.

Cooperative sensing is in a weak sense also found in computer games. If, for example, a guardian of a castle detects an intruder, other guardians will be alerted by triggering an alarm. However, also cooperation can be "cheated" by game characters. Characters in digital environments are not autonomous in terms of being executed on different computers, so there is no need for communication between them. In fact, they are able to share information on the same machine. However, human players are sensitive to supernatural information exchange between their opponents. They prefer to have an explanation for the cooperation of their opponents, as for example given by the triggered alarm in the guardian example.

More important for game AI are algorithms for cooperative acting. Unfortunately, the complexity of action selection grows significantly with the number of agents and number of actions they can perform. Hence robotic soccer teams make use of role assignments that restrict the set of available actions to those that belong to the selected role. A robot of the CS-Freiburg team fills at any time one of four roles, these include *active* (to play the ball), *strategic* (to defend the team's half of the field), *support* (to support either the defense or the offense depending on the game) and *goalkeeper* (to defend the goal). Except the role of the goalkeeper, the role assignment may change dynamically during a game. This assignment is decided by a simple but effective mechanism: Each robot calculates, with respect to its current knowledge about the world, for each role an utility value that expresses the benefit for the team if the robot would take on this role. For example, the active role assigns high values to robots close to and behind the ball, but lower values to positions that interfere with opponent players. In order to decide the final role assignment, the robots communicate the values they have calculated for each role. Under the assumption that each robot receives the value assignment from all the others, they are able to decide simultaneously a role assignment that maximizes the total utility of the group. A more detailed explanation of the role assignment mechanism is found in the CS Freiburg team description paper [19].

Computer games are full of situations where team behaviors are beneficial. For example, the classical first person shooter (FPS) game can usually be played in the "Capture the flag" mode, which is the situation of two opposing teams, one defending a fortress with a flag, and another one trying to capture this flag. In some versions of this game, teams are composed of different character types, such as a sniper, a medic and a grenadier. In this particular case, role assignments are quite simple to decide, since they follow directly from the agent's type. For example, one would constraint the medic to automatically healing any injured team member.

However, in any other case, cooperation among game characters is difficult and thus usually performed by human players on the Internet only. We believe that the dynamic role assignment approach might also be applied within digital environments. Digital characters could be programmed to dynamically change their role within the team depending on their health, damage, strength and tactical abilities. Roles can be defined as particular positions of team members within a defense or offense formation. Depending on the attack of an opponent, team members might dynamically change their position assignment during a game.

3 AI in RoboCupRescue

RoboCupRescue is a part of the RoboCup competitions and was originally motivated by the destructive earthquake that happened 1995 in Kobe City, Japan. There exists a real rescue robot competition as well as a simulated rescue robot competition. During the real robot competition, robots have to locate injured people in an unstructured environment, whereas during the simulated competition the challenge is to coordinate multiple agent teams and to locate injured civilians within a simulated large scale urban disaster (see figure 4). We will primarily focus on the later one within this article.

The large scale simulation is carried out by distributed modules that communicate over a network with each other. These modules are: a Geographical Information System (GIS), various disaster simulators (i.e. simulating earthquakes and fires), and agents separated into ambulance, fire and police forces. The task of the AI is to control and coordinate these agents in order to prevent or limit damage to civilians and buildings.

3.1 Perception and World Modeling

One challenge in this domain is to cope with the uncertainty of the global world state, caused by the limited perception range of the agents. Humanoids are simulated with a limited range of perception. They perceive auditory information within a radius of 30 meters and visual information within a radius of 10 meters. To overcome this limitation, agents have to cooperatively build a world model from their percepts that represents the current state of the city.

We developed special algorithms for the coordinated search of civilians in unknown terrain. The difficulty here is to not waste resources by exploring the same area twice and to decide, with respect to the collected evidence, when to expand or to focus the search. This is performed by calculating utility values for each location on the map. These utility values are dynamically updated from the agent's senses, which can be auditory or visual. For example, if an agent heard a civilian, the utilities of locations within the agent's hearing range are increased and otherwise, if locations have been explored completely, set to zero. With this approach it is possible for the agent to focus the search on areas with high evidence, or even to conclude the whereabouts of civilians from evidence given in the past. In order to avoid the exploration of areas that were selected previously by other rescue agents, explored locations are communicated between the agents.

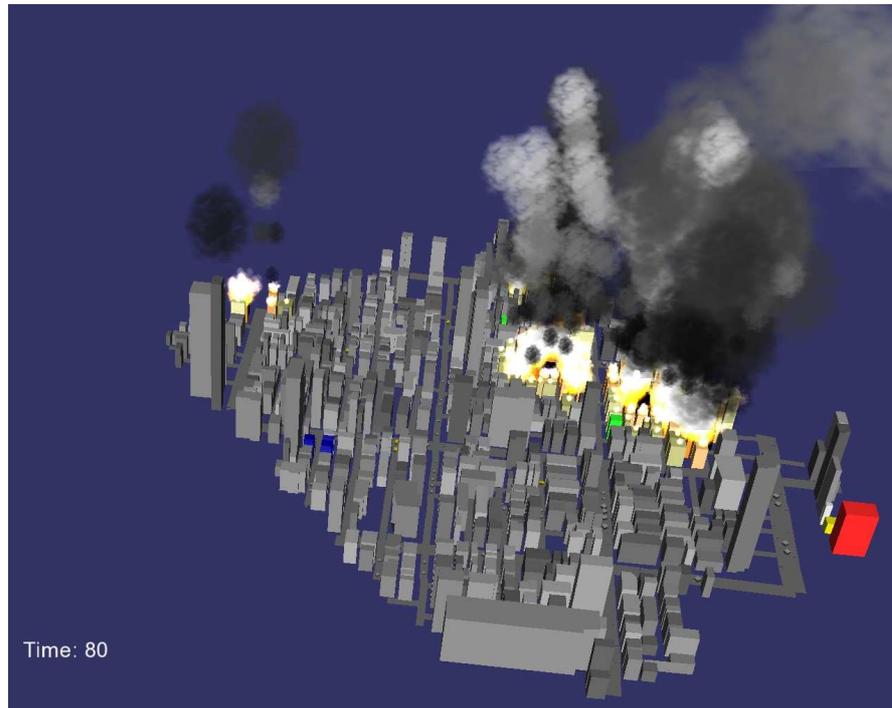


Fig. 4. Simulation of fires on a model of Kobe City, Japan

Evidence based exploration can also be found in computer games. The game *Thief* [12], for example, was one of the first games to provide a simplified visual and acoustic model for its characters. Also the game *Splinter Cell*, one of the best selling games last year, requires human players to keep silent and to hide themselves within darker areas. For example, if a human player causes noise by walking too fast or throwing a can within the hearing range of guardians, guardians are triggered to perform a systematic search. Guardians are also constrained by a limited ability to see. The human player might further reduce this ability by destroying light sources that are part of the ambient illumination. One advantage of games that provide characters based on the agent model, is that they offer the opportunity to win a game by fooling the opponent. The commercial success of the two games mentioned above shows clearly that customers tend to prefer agent like opponents to fully informed and outnumbering opponents.

3.2 Action selection and execution

Within the rescue domain, each agent type focuses on particular targets, e.g. the ambulance teams on civilians and the fire brigades on burning buildings. Typically, targets are selected by a trade-off between their value (depending on the situation) and their costs (depending on the execution time for saving them). In the case of the ambulances,



Fig. 5. (a) A typical town in *Age of Empires 2* (Ensemble Studios) built by a human player, (b) Defending a town against an digital opponent (Images are taken from www.ensemblestudios.com).

utilities are always one² and costs for rescuing a civilian are composed of the expected time for traveling to the target location plus the expected time for digging the target out. However, not only costs and utilities are important in a rescue situation, moreover the fact that civilians might die if rescue teams are too late, plays an important role. The chance of survival of the civilians depends on the degree of their injuries and the extent to which they are buried under a collapsed building. Since survival time and costs are different for each civilian, the chosen rescue sequence has a significant influence on the number of victims an ambulance team is able to save.

In our approach, we utilize a Genetic Algorithm (GA) for calculating a near-to-optimal rescue sequence. This is carried out by representing rescue sequences as DNA strings that are continuously mutated and selected. The fitness of each solution is set equal to the number of victims that will survive due to the corresponding sequence. By continuous mutation and selection, the algorithm converges towards better rescue sequences. The calculation of the score of each solution requires the prediction of the expected life time of each civilian, e.g. the civilians ability to survive with respect to the current situation. This prediction is carried out by a neural network that is trained with a large amount of data collected during previous simulation runs.

The sequence optimization is implemented in terms of an any-time algorithm. This means that the GA can be called multiple times in a short time period rather than for a longer period which might affect the agents performance. Due to this property, the GA and the prediction system are applicable to computer games as well. Particularly in Real Time Strategy (RTS) games, there are situations where an optimized target selection might increase the efficiency and thus the performance of the computer player. One example with comparable conditions might be where engineers within a RTS game have to repair broken defense-buildings during an attack by the opponent. The more buildings are repaired, the more likely the computer player's team will survive.

Another important issue in RoboCupRescue is *path planning*, which is the ability to find a fast and safe path to locations on a city map. In this domain, planning is difficult due to the size of the city and due to the uncertainty of the current situation of the roads. The size of the city makes planning expensive, since it needs a lot of

² Since we consider victims as equally important

computing resources, particularly if the agent needs to know about the travel costs for *each* single target on the map. The uncertainty of the current situation is due to the fact that outside the perception range of the agent, roads might become blocked by debris from collapsing buildings. The continuous changes and uncertainty of the environment forces the agents to revise their plans continuously.

We implemented a special algorithm for path planning that accounts for the difficulties described above. This algorithm is based on the solution from Dijkstra [14] but has been adapted to the specific problem. The algorithm incorporates the structure of the map by combining small road segments into larger ones. With this technique it is possible to find paths on the map in a hierarchical fashion, which is much faster than the standard approach.

Path planning is also a serious issue in RTS games, since the implementation significantly influences the quality of the game. RTS games suffer under the same two problems mentioned above. The size of the map depends roughly on the number of players participating in a game. For example, there are RTS games which are played with more than eight players. During the game, each player builds up one or more towns, castles, and more than 100 units (see figure 5). It is interesting to note that David Pottinger (Ensemble Studios) once wrote: "Pathfinding is one of the slowest things most RTS games do, e.g. *Age of Empires 2* spends roughly 60 to 70% of simulation time doing pathfinding."

Additionally, RTS games are suffering under the uncertainty on the situation of the map, which has two reasons: Firstly, a map can initially be generated by random, i.e. it is not possible to pre-compute all possible plans. Secondly, users are allowed to modify the map during the game. In *Age of empires 2*, for example, human players are allowed to build walls all over the map in order to block the path of their enemy.

Due to the similarity between problems in computer games and problems in RoboCup, solutions developed in the context of RoboCup are well suited for computer games and might improve their quality.

4 Simulation and Game Engines

The development of autonomous systems is in some cases nearly impossible without a simulation software. There are several reasons, one is obviously the fact that experiences in real world environments are expensive to acquire, either due to the physical damage they might cause or simply due to the time it costs to complete them. For example, the machine learning performed on our robots (see 2.2) was only possible by utilizing a simulation software tailored for RoboCup soccer [8]. This is due to the fact that learning was carried out by a *reinforcement learning* approach that had to be applied in more than 10000 learning episodes³ until the robot was able to successfully shoot goals against a virtual goalkeeper.

However, the design of a good physics simulation is tedious and probably not something an AI systems developer would like to invest time in. Fortunately, there are many simulation software packages available now. The "Open Dynamics Engine" ODE [4],

³ An episode is a series of actions that terminates if either the robot succeeds or fails

for example, is a freely available software library that offers a rigid body simulation. Besides open source development, there are also companies developing physics engines for commercial computer games. These engines are typically used for more than one game, whereas a single game is sold more than a million times. Due to the high sales quantity of games, physics engines are available for low prices. For example, the physics engine of the Swiss company *NovodeX*, which will be utilized for the latest versions of the game *Unreal*, is available for less than 300USD [3]. The *Nimbro* humanoid robot group in Freiburg uses since 2004 the *NovodeX* game engine for the simulation of their humanoid robot [1].

Instead of using a physics engine, it is also possible to buy a game and to use its game engine directly. This is known as *modding*, a technique that allows users to modify certain parts of the game, such as the geometry and physical behavior of objects. These objects can be static, such as landscapes and buildings, but also dynamic, such as digital characters. Furthermore the game allows for the control of these characters internally or externally by a network connection. We are currently employing the game *Unreal Tournament 2003* together with an extension [18] developed for the NIST⁴ USAR⁵ test facility. The extension offers a rich set of virtual robots that can be plugged into the simulation and allows the simulation of user specific robot types.

It is easy to see that game engines are already offering a degree of efficiency, detail and diversity which is clearly beyond the means any "hand-made" simulator of research groups focusing on AI or robotics could ever offer. This is apparent with the impressive *Havok* [2] game engine that will be part of the game *Half Life 2*. *Half Life 2*, which has first been demonstrated at the Electronic Entertainment Expo 2003, provides a preview of the close-to-reality physics models future game engines will take on. Characters can initiate complex chain reactions of items in the environment. For example, items thrown into the water interact as one might expect them to do. Barrels move like corks for a moment then sink to the bottom. Wood floats on the surface and even supports other items tossed on top of it until the load becomes unstable or too heavy and then sinks.

As well as the environment, the detail level of digital characters, particularly their bodies, is also improved. The body is modeled with a complex skeleton that can be manipulated by virtual muscles. In *Half Life 2* there are approximately 40 muscles for modeling just the face animation. This makes it possible for game characters to express many individual human emotions (see figure 6). The game company *Valve* developed a special editor, known as the *Half Life Face Poser*, for authoring facial animations consistent with phonemes pronounced by the characters.

Since such tools are improving in terms of complexity and diversity, it seems straight forward that they will also be of interest to developers of humanoid robots and human-robot interactions.

5 Conclusion

We have shown that there are several good reasons why computer games will positively influence AI research in the future: Firstly, the computer game market provides a high

⁴ National Institute of Standards and Technology

⁵ Urban Search and Rescue



Fig. 6. Image taken from the facial animation of the "g-man" in Half Life 2

potential for development, since games sell in great numbers. Tools developed for a particular game might even be reused for other games (the concept of game engines). Secondly, game engines have improved significantly during the last few years. They provide outstanding physics simulations and tools for designing and studying human-machine interactions. Thirdly, due to their Internet connectivity, games are providing a great deal of data from human-machine interactions, which is useful, for example, in the context of machine learning.

We have also shown that there are several good reasons why AI methods developed in the context of RoboCup are optimally fitting the needs of modern computer games: Firstly, algorithms from that context are fast and reliable since they have to enable the agent to act under sensor noise and uncertainty in real time. Secondly, these algorithms are particularly based on the agent model and thus provide the basis for digital characters to model their environment. Digital characters that build models of their environment are leading to realistic and plausible behaviors.

One can expect that the development potential of computer games and the growing focus on game AI will have an accelerating effect on AI research. We expect that cooperation between game companies and AI researchers will increase in the future. From this symbiosis many useful tools might arise. For example, the successive development of physics engines and improvement of game AI will continue to demand more performance from the computer. This will probably enforce the development of specialized hardware for AI or physics simulation. A similar process has already been noticed in the past. The introduction of the standardized interfaces *DirectX* and *OpenGL* for the

rendering of 3D graphics initiated the development of efficient hardware boards that exponentially improved each year.

References

1. S. Behnke, N. Mayer, J. Müller, and M. Schreiber. Nimbro 2004 team description. In *8th RoboCup International Symposium*, 2004.
2. HAVOK Physics Engine. <http://www.havok.com>, 2004.
3. NovodeX Physics Engine. <http://http://www.novodex.com/index.html>, 2004.
4. ODE Open Dynamics Engine. <http://ode.org/ode.html>, 2004.
5. D. Fox, W. Burgard, and S. Thrun. Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research (JAIR)*, 11, 1999.
6. ResQ Freiburg. Team homepage. <http://www.informatik.uni-freiburg.de/~rescue>.
7. R. Le Hy, A. Arrigoni, P. Bessire, and O. Lebeltel. Teaching bayesian behaviours to video game characters. *Robotics and Autonomous Systems (Elsevier)*, 47:177–185, 2004.
8. A. Kleiner and T. Buchheim. A plugin-based architecture for simulation in the F2000 league. In *Proc. Int. RoboCup Symposium '03*. Padova, Italy, 2003.
9. A. Kleiner, M. Dietl, and B. Nebel. Towards a life-long learning soccer agent. In *Proc. Int. RoboCup Symposium '02*. Fukuoka, Japan, 2002.
10. J. E. Laird. It knows what you're going to do: adding anticipation to a quakebot. In *Proceedings of the fifth international conference on Autonomous agents*. ACM Press New York, 2001.
11. J. E. Laird and M. van Lent. Human-level AI's killer application: Interactive computer games. *Computer*, 34(7):70–75, 2001.
12. T. Leonard. Building an ai sensory system: Examining the design of thief: The dark project. In *Game Development Convergence (GDC 2003)*, 2003.
13. P. S. Maybeck. The Kalman filter: An introduction to concepts. In I.J. Cox and G.T. Wilfong, editors, *Autonomous Robot Vehicles*, pages 194–204. Springer-Verlag, 1990.
14. T. Ottman and P. Widmayer. *Algorithmen und Datenstrukturen*. Spektrum Verlag, 2002.
15. Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach, Second Edition*. Prentice-Hall, Englewood Cliffs, NJ, 2003.
16. Inc. Tim Sweeney, Epic MegaGames. Reference of the unreal scripting language. <http://unreal.epicgames.com/UnrealScript.htm>, 1998.
17. Turing. Computing machinery and intelligence. In *George F. Luger (Ed.), Computation and Intelligence: Collected Readings, AAAI Press/The MIT Press*. 1995.
18. J. Wang, M. Lewis, and J. Gennari. A game engine based simulation of the nist urban search and rescue arenas. In *Winter Simulation Conference*, 2003.
19. T. Weigel, J.-S. Gutmann, M. Dietl, A. Kleiner, and B. Nebel. CS-Freiburg: Coordinating robots for successful soccer playing. *IEEE Transactions on Robotics and Automation*, 18(5):685–699, 2002.
20. S. Woodcock, J. E. Laird, and D. Pottinger. Game AI: The state of the industry. *Game Developer Magazine*, August, 2000.