Synthetic Players

A Quest for Artificial Intelligence in Computer Games

Jouni Smed and Harri Hakonen

Synthetic players are the computer-controlled actors in a computer game. In this paper, we describe both the implementation issues and the behavioural expectations related to synthetic players. We recognize the role of synthetic players by analysing the components of games in general and computer games in particular.

In 1962 students at MIT competed against each other in the first real-time graphical computer game *Spacewar* (Graetz 1981). Probably none of them could have dreamt how realistic and complex computer games would turn into in four decades and how large a business would grow around them. However, it took ten years before commercial arcade games such as *Pong* and *Space Invaders* created the business in the 1970s and before home computers brought computer games within the reach of all enthusiasts in the 1980s. Since then game development and programming have turned from small amateur enterprises into a more professional and larger-scale industry. Nowadays, the typical time-span of development from an idea to a finished product is about two years and demands the work contribution of 20–50 persons. Game industry is usually compared to film industry, and in 2001 the global market of computer games finally surpassed global film box office. The current estimates of the annual revenue generated by computer games are around €25 billion and the annual growth is predicted to be over ten per cent over the next few years (Game Developers' Association of Australia 2003).

The game industry has awakened to the possibilities of academic research. International Game Developers Association (2003) lists *game programming* among the eight core topics of game-related academic research. Game programming is defined to cover "[a]spects of traditional Computer Science – modified to address the technical aspects of gaming". This interest in novel solutions and improved methods is understandable, because the marketing of computer games is highly technology-driven. Earlier the selling points were the amount of colours and real-timeliness, then the amount of polygons and the frame update frequency, and now the amount of simultaneous players in a networked game and the realism of the simulation. These features also reflect what programming problems have been in focus of game developers at the time.

Although one might think that computer games are favourable environments for artificial intelligence (AI) or simulation related research, co-operation between academic researchers and game developers has been most lively in graphics programming. In particular for the past two decades the SIGGRAPH community has been the forum where problems and solution methods are exchanged to and fro. Theoretical methods have found their application and eventually ended up as part of the hardware (e.g. in 3D display cards).

In recent years the spread of broadband network connections and the growth of their capacity have geared interest towards problems of distributed multiplayer games (Smed, Kaukoranta & Hakonen 2002). Because computer games are real-time applications, the effect of communication delays are compensated using algorithmic prediction methods, which were originally developed in virtual environment research. Another active field of research is cheating prevention, because the money involved in online gaming is arousing even criminal interest. New methods against virtual attacks are currently being developed, which aim at preventing spying and altering the game data transmitted over a network.

AI related game problems have only recently returned to the focus of academic research (Laird & van Lent 2001). Whereas traditional methods mainly concentrate on turn-based and discrete games, the interest is now in developing real-time methods, which seem to act intelligently, for continuous game worlds. These methods are used in the implementation of computer-controlled actors, or *synthetic players*, for the game.

In this paper, we take a look at the features that synthetic players should have and provide. To understand the role of synthetic players in a computer game, we begin with a general discussion of games, computer games and their constituents. After that, we look at synthetic players from two perspectives: First, we describe the structure of the software components of a synthetic player and describe features that are important in their design and implementation. Second, we analyse the behavioural features that synthetic players should demonstrate.

Defining a Game

Huizinga (1955, 132) defines "play" as

> *an activity which proceeds within certain limits of time and space, in a visible order, according to rules freely accepted, and outside the sphere of necessity or material utility. The play-mood is one of rapture and enthusiasm, and is sacred or festive in accordance with the occasion. A feeling of exaltation and tension accompanies the action, mirth and relaxation follow.*

This definition also captures many of the features present in games. A dictionary definition states that "game" is "a universal form of recreation generally including any activity engaged in for diversion or amusement and often establishing a situation that involves a contest or rivalry" (Encyclopædia Britannica 2004). Crawford (1984, ch. 1) defines game as "a closed formal system that subjectively represents a subset of reality." Accordingly, a game is self-sufficient, follows a set of rules, and has a representation in the real world. These observations are echoed by the definitions of Costikyan (2002, 24), who sees a game as "an interactive structure of endogeneous meaning that requires players to struggle toward a goal", and Salen & Zimmerman (2004, 80) to whom a game is "a system in which players engage in an artificial conflict, defined by rules, that results in a quantifiable outcome."

We define 'game' by recognizing its main components, the relationships between them, and the aspects that these relationships form, which is illustrated in Figure 1 (Smed & Hakonen 2003).

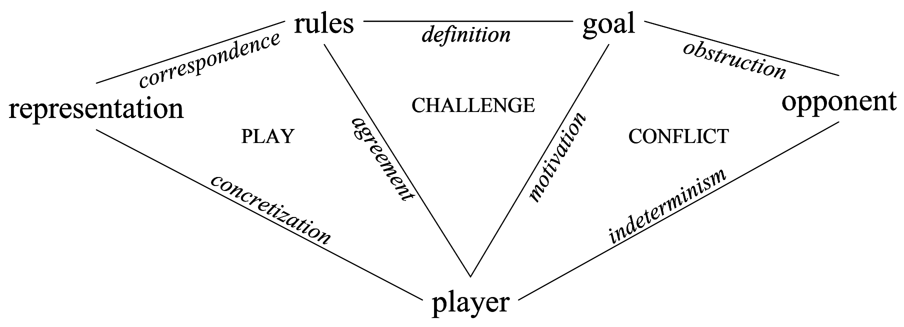

Figure 1. Components, relationships and aspects of a game.

We can immediately recognize three distinct components involved in a game. First, we have *players* who are willing to participate in the game (e.g. for enjoyment, diversion or amusement). Second, we must have *rules* which define the limits of the game. Third, there are *goals* which give rise to conflicts and rivalry among the players. Between these three components we have the following relationships. The players are willing to follow the rules of the game. The rules define the goals of the game. The goals motivate the players to participate in the game and drive the game forwards, and achieving a goal in the game gives a player enjoyment. We call this trio the *challenge* aspect of the game.

The challenge aspect is not enough for a definition, because games are also about *conflict*. For example, a crossword puzzle may be a challenge in its own right but there is hardly any conflict in solving it – unless someone erases the letters or changes the hints or keeps a record of the time to solve the puzzle. Obviously, the conflict arises from the presence of an opponent, which aims at obstructing the player from achieving the goal. The opponent does not have to be a human but it can be some random process (e.g. the throw of dices or the shuffling of a deck of cards). The main feature of the opponent is that it is indeterministic to the player: because the player cannot predict exactly what another human being or a random process will do, outwitting or outguessing the opponent becomes an important part of the game.

Challenge and conflict aspects are enough for defining a game in an abstract sense. However, in order to *play* the game, it needs to be concretized into a representation. This representation can be a cardboard board and plastic pieces as well as three-dimensional graphics rendered on a computer screen. Even the players themselves can be the representtation, as in the children's game of tag. Regardless of the representation there must exist a clear correspondence to the rules of the game.

Let us take the game of poker as an example. The players agree to follow the rules, which state (among other things) what cards there are in a deck, how many cards one can change, and how the hands are ranked. The rules also define the goal, having as good hand as possible, which is the player's motivation. The other players are opponents, because they try to achieve a better hand to win. Also, the randomness of the deck caused by shuffling opposes the player, who cannot determine what cards will be dealt next. The game takes a concrete form in a deck of plastic-coated cards (or pixels on the screen), which represent the abstractions used in the rules.

Apart from the features discussed above, the game play also includes subjective elements such as an immersion in the game world, a sense of purpose, and a sense of achievement from mastering the game. One could argue that the sense of purpose is essential for the immersion. What immerses us in a game (as well as in a book or a film) is the sense that there is a purpose or motive behind the surface. In a similar fashion, the sense of achievement is essential for the sense of purpose (i.e. the purpose of a game is to achieve goals, points, money, recognition etc.). From a human point of view, we get satisfaction in the process of nearing a challenging goal and finally achieving it. These aspects, however, are outside the scope of our current discussion.

Computer games are a subset of games. To be more precise, let us define 'computer game' as a game that is carried out with the help of a computer program. This definition leaves us some leeway, since it does not implicate that the whole game takes place in a computer. For example, a game of chess can be played on a computer screen or on a real-world board, regardless of whether the opponent is a computer program or not. In effect, a computer program in a game can act in three roles: First, it can be used to co-ordinate the game process (e.g. ensuring the participant in a chess game makes legal moves). Second, it can be used to illustrate the situation (e.g. displaying the chess board and pieces on screen). Third, the computer program can participate in the game as a fellow-player. This last role is interesting to us, and we call a computer-controlled participant in a game a synthetic player. To understand what is expected from the synthetic player, we have to begin by taking a closer look at the software components of a computer game.

Software Components of a Computer Game

The three roles – co-ordination, illustration, and participation – for a computer program in a game closely resemble the *model-view-controller* (MVC) architectural pattern for computer programs (Krasner & Pope 1988). The basic idea of MVC is that the representation of the underlying application domain (model) should be separated from the way it is presented to the user (view) and from the way the user interacts with it (controller). Figure 2 illustrates the MVC components and the data flow in a computer game.

The model part includes software components responsible for the co-ordination role (e.g. evaluating the rules and upholding the game state). The rules and basic entity information (e.g. physical laws) form the core structures. These remain unchanged while the state instance is created and configured for each game process. The core structures do not need to cover all the rules, because they can be instantiated. For example, the core structures can define the basic mechanism and properties of playing cards (e.g. suits and values) while the instance data can provide the additional structures (i.e. rules) required for a game of poker (e.g. ranking of the hands, staking, and resolving ties).

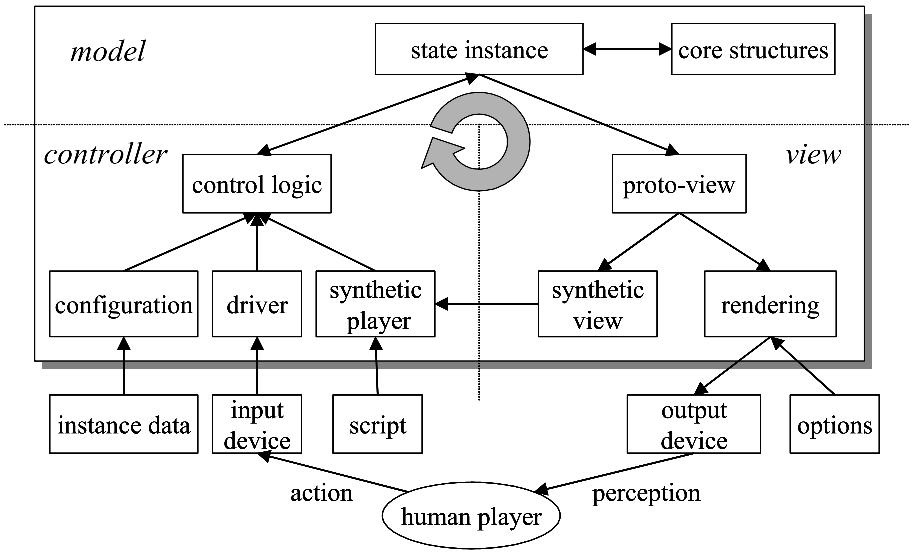

Figure 2. Model, view and controller in a computer game.

The view part handles the illustration role. A proto-view provides an interface into the model. It is used for creating a synthetic view for a synthetic player or for rendering a view to an output device. The synthetic view can be pre-processed to suit the needs of the synthetic player (e.g. board co-ordinates rather than an image of the pieces on a board). Although rendering is often equated with visualization, it may as well include audification and other forms of sensory feed-back. The rendering can have some user-definable options (e.g. graphics resolution or sound quality).

The controller part includes the components for the participation role. Control logic affects the model and maintains the integrity (e.g. by excluding illegal moves suggested by a player). The human player's input is received through an input device filtered by a driver software. The configuration component provides instance data, which is used in generating the initial state for the game. The human player takes part in the data flow by perceiving information from the output devices and generating actions to the input devices. Although Figure 2 includes only one player, naturally there can be multiple players participating in the data flow, each with their own output and input devices. Moreover, the computer game can be distributed among several nodes in a network rather than residing inside a single node. Conceptually, this is not a problem since the components in MVC can as well be thought to be distributed (i.e. the data flows run through a network rather inside a single computer). In practice, however, distributed computer games provide their own challenges (for a more detailed discussion, see Smed, Kaukoranta & Hakonen 2002).

A synthetic player is a computer-controlled actor in the game. It can be an opponent, a non-player character which participates limitedly, or a *deus ex machina* which can control natural forces or godly powers and thus intervene the game events. The more open the game world is, the more complex the synthetic players are. This trade-off between the model and the controller is obvious: if we remove restricting code from the core structures, we have to reinstate it in the synthetic players. For example, if the players can hurt themselves by walking into fire, the synthetic player must know how to avoid it. Conversely, if we rule out fire as permitted area, path finding for a synthetic player becomes simpler.

As we can see in Figure 2, the data flow of the human player and that of the synthetic player resemble each other. This allows us to project human-like features into the synthetic player. We may even argue that, in a sense, there should be no difference between the players whether they are humans or computer programs; if they are to operate on the same level, both should ideally have the same powers of observation and the same capabilities. Still, synthetic players usually cheat (e.g. by having outside knowledge or receiving extra resources), and this has been the norm for a long time. Generally, the reason is obvious: a computer program is no match for human ingenuity, and is therefore granted the benefit of playing on its own ground. This is understandable – and we may even forgive it when it seems fair – but, ideally, the synthetic players should be in a similar situation as their human counterparts.

Synthetic Player's Structure

The AI system of a computer game comprises two parts: *pattern recognition* and *decision-making system* (Kaukoranta, Smed & Hakonen

2003). Figure 3 gives a more detailed illustration of the synthetic player component of Figure 2. The world (or rather the synthetic view of the game world) consists of primitive events and states (phenomena) that are passed to pattern recognition and possibly stored for later use in a history buffer. The information abstracted from the current (and possibly the previous) phenomena is then forwarded to the decision-making system. The game world allows a set of possible actions, and the decision-making system chooses the ones to carry out and convey to the control logic component.

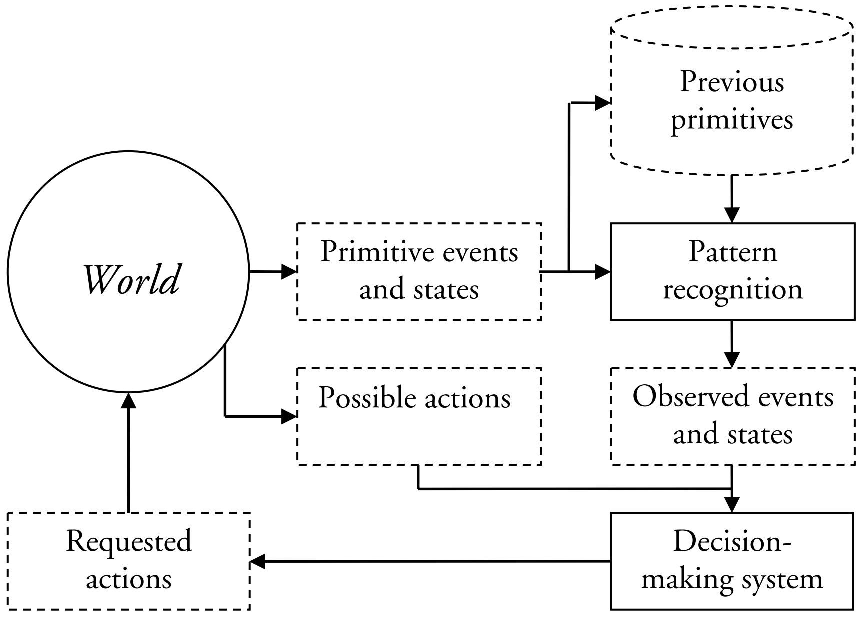

Figure 3. Components of the synthetic player.

The separation between pattern recognition and decision-making is not always clear-cut. Nevertheless, the distribution of responsibilities helps us to recognize features that the methods have. These features also outline the implementation issues that should be addressed already in the design of an AI system for a synthetic player (Smed, Kaukoranta & Hakonen 2003).

Real-time Response

Whereas in the traditional turn-based games the computer opponent can think (almost) as long as it requires, nowadays games mostly require real-time response. This puts a hard computational strain on the synthetic player, because it can no longer delve into finding an optimal strategy, but has to react immediately. Response is the key-word – even to such extent that game developers tend to think that it is better to have hordes of mindless cannon-fodder than to grant the synthetic players a shred of intelligence. In the past the main reason for this was that the game AI was not slated a fair share of the overall processing resources. Surprisingly, even today the average game AI is granted about ten per cent of the processor capacity (Dybsand 2004).

Distribution has become more important now that games using networking are more common. This may present one solution to the dilemma of achieving both real-time response and intelligence: instead of running the synthetic players on one machine, they can be distributed so that the cumulative computational power of the networked nodes is utilized. For example, *Homeworld* (Relic Entertainment 1999) uses this technique and distributes the computer-controlled opponents among the participating computers.

Autonomy and Communication

Distribution naturally begs the question of how autonomous the synthetic players should be. As long as we can rely on the network there is no problem, but if nodes can drop out and join at any time, distributed synthetic players must display autonomy. This means two things. First, the synthetic player must be persistent, because it can be relocated to another node if the one where it is currently run gets cut off. Second, the synthetic player must be self-sufficient, because it cannot rely on outside processes but should be able to operate on its own. This is not necessarily a drawback, because autonomy can lead to a smaller and better design, and complex behaviour can emerge from seemingly simple autonomous agents.

A corollary of autonomy is that the synthetic players must have a way to communicate explicitly with each other. Because there is no central intelligence controlling them, they have to inform others of their decisions, indicate their plans, and negotiate with each other – just like we humans do in the real world. And, as we shall later see, these communication skills are required also when interacting with the human players.

Levels of Decision-making

Classically, decision-making problems are divided into three levels. On the *strategic level*, decisions are made for a long period of time and are based on a large amount of data. The nature of the decisions is usually speculative (e.g. what-if scenarios), and the cost of a wrong decision is high. The *tactical level* acts as an intermediary between strategic and operational levels. Tactical decisions usually consider a group of entities and their co-operation, and, ultimately, the aim of tactical decisions is to fulfil the plan made on the strategic level. *Operational level* is concrete and closely connected to the properties of the game world. Although the number of decision-making entities on this level is high, the decisions consist of choosing short-term actions among a given set of alternatives.

Let us consider football as an example of the levels of decision-making. On the strategic level, there are the choices of how to win the game (e.g. whether to play offensively or defensively). On the tactical level, the choices concern carrying out the strategy in the best possible way (e.g. whether to use man-marking defence or space-marking defence). On the operational level, the choices are simple and concrete (e.g. where the player is to position himself and, if he has the ball, whether to dribble it, kick it to the goal or pass it to another player). The problem is how to choose what to do (i.e. decision-making) and on what grounds (i.e. pattern recognition). It is fairly simple on the operational level – dribble if you have an opening, pass if you can do it safely – but it gets harder and harder as the level of abstraction rises.

Uses for the Modelled Knowledge

Based on the information provided by pattern recognition, the decision-making system forms a model about the world. The complexity of the world can be simplified with generators, which label the events and states with symbols. For example, the punches in a boxing game can go through a generator that produces the symbols 'jab', 'uppercut', 'cross', and 'hook'. Now, we can construct a model for the behaviour of the generator from the generated symbol sequence. Modelling recognizes the underlying dependencies between symbols, which are typically stronger between symbols that are close to each other. Often a short-term history is sufficient, but the model gets more accurate if we increase the length of the modelling context at the cost of run time.

We can use the model to imitate the actions of a human player (Alexander 2002). For example, we can model the punch series of a real-world boxer, and use the model when selecting the next punch for a computer-controlled boxer. Of course we could construct the model simply by observing the human opponent's moves and start mimicking them. In addition to imitation, the model can be used to predict what will happen next. For example, if we have constructed a model of the opponent's punch series, we can compute the most likely punch the opponent will throw next, and use this prediction to calculate an effect-tive counteraction.

The model does not have to be confined only to the opponent and the game world, but can cover the actions and reactions of the synthetic player itself. Whenever the synthetic player makes a decision, the outcome produces feed-back – positive or negative, direct or indirect – which can be used in learning (Evans 2002). For example, in *Black & White* (Lionhead Studios 2001) the computer-controlled pet creature learns from other entities' reactions, from feed-back from the human player, or from its own experiences. Hence, the rule "Do not eat trees" can be derived either from the villagers' disapproval for wasting resources, from a sharp slap by the owner, or from the resulting stomach-ache.

Synthetic Player's Behaviour

The game world is anthropocentric, because everything in it revolves around the human player. Regardless of the underlying method for decision-making, the synthetic player is bound to show certain behaviour in relation to the human player, which can range from simple reactions to general attitudes and even complex intentions. The list of features we provide here is by no means comprehensive, but points out some details that are relevant to the design of the synthetic player. These are the things that a casual player is most likely to notice first, whereas the structural details we discussed earlier are of more interest to the game developers and programmers.

Humanness

The success of networked multiplayer games is partly due to the fact that human players can provide something synthetic players still cannot: human traits and characteristics. These include flaws as much as (or even more than) strengths: fear, rage, compassion, hesitation, and emotions in general. Even minor displays of emotion can make the synthetic player appear more human. For instance, in *Half-Life* (Valve Software 1998) and *Halo* (Bungie Software 2003), the synthetic players who have been taken by surprise do not act in superhuman coolness but show fear and panic appropriate to the situation. We, as human beings, are quite apt to read humanness into the decisions even when there is nothing but naïve algorithms behind them. Sometimes a game such as *NetHack* (DevTeam 2004) even gathers around a community that starts to tell stories of the things that synthetic players have done and to interpret them in human terms.

A computer game comprising just synthetic players could be as interesting to watch as a movie or television show (Charles, Mead & Cavazza 2002). In other words, if the game world is fascinating enough to observe, it is likely that it is also enjoyable to participate in – which is one of the key factors in games like *The Sims* (Maxis 2000) and *Singles* (Rotobee 2004), where the synthetic players seem to act (more or less) with a purpose and where a human player's influence is, at best, only indirect.

There are also computer games that do not have human players at all. Already back in the 1980s *Core War* demonstrated that programming synthetic players to compete with each other can be an interesting game itself (Dewdney 1984). Since then some games have tried to use this approach, but, by the large, AI programming games have been only by-products of "proper" games. For example, *Age of Empires II: The Age of*

Kings (Ensemble Studios 1999) includes a possibility to create scripts for computer players, which allows for the organization of games where programmers compete in creating the best AI script. The whole game is then carried out by a computer while the humans remain as observers. Although the programmers cannot affect the outcome during the game, they are more than just enthusiastic watchers: they are the coaches and the parents, and the synthetic players are the protégés and the children.

Stance

The computer-controlled player can have different stances (or attitudes) towards the human player. Traditionally, synthetic player has been seen only in the role of an *enemy*. As an enemy the synthetic player must provide challenge and demonstrate intelligent (or at least purposeful) behaviour. Although the enemies may be omniscient or cheat when the human player cannot see them, it is important to maintain the illusion that the synthetic player is at the same level as the human player.

When the computer acts as an *ally*, its behaviour must adjust to the human point of view. For example, a computer-controlled reconnaiss-ance officer should provide intelligence in a visually accessible format rather than overwhelm the player with lists of raw variable values. In addition to accessibility, the human players require consistency, and even incomplete information (as long as it remains consistent) can have some value to them. The help can even take the form of concrete operations as in *Battlefield: Vietnam* (Digital Illusions 2004) where the computer-controlled fellow-soldiers respond to the player's commands.

The computer has a *neutral* stance when it acts as an observer (e.g. camera director or commentator) or a referee (e.g. judging rule violations in a sports game) (Siira 2004). Here, the behaviour depends on the context and conventions of the role. In a sports game, for example, the camera director program must heed the camera placements and cuts dictated by the television programme practice. Refereeing provides another kind of challenge, because some rules can be hard to judge. Finally, non-player characters (NPCs) can be used to carry out the plot, to provide atmosphere, or simply to act as extras. Nevertheless, as we shall see next, they may have an important role in assisting immersion in the game world and directing the game play.

Story-telling

Story-telling is not about actions but reasons for actions. Human beings use stories to understand intentional behaviour and tend to "humanize" the behaviour of the characters to understand the story (Spierling 2002). While "traditional" story-telling progresses linearly, a game must provide an illusion of free will (Costikyan 2002). According to Aylett & Louchart (2003) computer games differ from other forms of story-telling in that the story time and real time are highly contingent, whereas in traditional story-telling forms (e.g. cinema or literature) this dependency can be quite loose. Another differentiating factor is interactivity, which is non-existent or rather restricted in other forms of story-telling. Bringsjord (2001) lists four challenges to interactive story-telling: First, plot and three-dimensional characters are not enough to produce a high-quality narrative: there must be themes (e.g. betrayal, self-deception, love or revenge) behind them. Second, something is needed to make sure the story stays dramatically interesting. Third, apart from being robust and autonomous, the characters (i.e. synthetic players) have to be memorable personalities by themselves. Fourth, a character should understand the players – even to the point of inferring other characters' and players' beliefs based on its own beliefs.

Anthropocentrism is not only reflected in the reactions but also in the intentions of the synthetic players. As a form of entertainment, amusement or pastime, the intention of games is to immerse and engulf the human player fully in the game world. This means that the human player may need guidance whilst proceeding in the game. The goals of the game can become blurred, and NPCs or events can lead the human players so that they do not stray too far from the intended direction set by the developers of the game. For this reason the game developers are quite eager to include a story into the game. The usual approach to include story-telling into commercial computer games is to have "interactive plots" (International Game Developers Association 2004). A game may offer only a little room for the story to deviate – as in *Dragon's Lair* (Sullivan Bluth 1989) where, at each stage, the players can choose from several alternative actions, of which all but one lead to a certain death. This linear plot approach is nowadays replaced by the parallel paths approach, where the story-line is divided into episodes. The player has some freedom within the episode, which has fixed entry and exit points. At the transition point the story of the previous episode is concluded, and new story alternatives for the next episode are introduced.

Research on story-telling computer systems is mainly motivated by the theories of Propp (1968), because they help to reduce the task of story-telling to a pattern recognition problem; for example, see Fairclough & Cunningham (2002), Lindley & Eladhari (2002), and Peinado & Gervás (2004). This pattern recognition approach can even be applied hierarchically to different abstraction levels. Spierling *et al.* (2002) decompose the story-telling system into four parts: story engine, scene action engine, character conversation engine, and actor avatar engine. These engines either rely on predefined data or act autonomously, and the higher level sets the outline for the level below. For example, based on the current situation the story engine recognizes an adaptable story pattern and inputs instructions for the scene action engine to carry out. This resembles the approach used in the Façade system (Mateas & Stern 2002), where a drama manager guides an autonomous simulation world from above. In addition to these implementation-oriented approaches, other methodological approaches to interactive story-telling have been suggested in the fields of narratology and ludology, but we omit a detailed discussion of them here.

The main problem of the often used top-down approach is that the story-generating program must act like a human dungeon master. It must observe the reactions of the crowd as well as the situation in the game, and recognize what pattern fits the current situation: is the game getting boring and should there be a surprising plot twist, or has there been too much action, would the players like to have a moment's peace to rest and regroup? Since we aim at telling a story to the human players, we must ensure that the world around them remains purposeful. We have general plot patterns that we try to recognize in the history and in the surroundings of a human player. This in turn determines how the synthetic players will act.

Instead of a centralized and omnipotent story-teller or dominant dungeon master, the plot could be revealed and the (autobiographical) "story" of the game (as told by the players to themselves) could emerge from the interaction with the synthetic players. However, this bottom-up approach is, quite understandably, rarely used because it leaves the synthetic players with a grave responsibility. They must provide a sense of purpose in a world of chaos.

Concluding Remarks

The work on synthetic players is still in its early stages. At the moment, the research efforts are mainly concentrating on the algorithmic and methodological problems, which we discussed by analysing the synthetic player's structure. However, we predict a shift of interest to the behavioural aspects of synthetic players, because they form the contact between humans and computers playing the game. Without a doubt the future promises us tougher challenges and meaner villains to beat, but then again, this calls for a more meaningful co-operation as well as coexistence with the synthetic players.

Jouni Smed has a Ph.D. in Computer Science from the University of Turku, Finland. Currently he acts as a post-doctoral researcher at the Turku Centre for Computer Science (TUCS) and lecturer at the Department of Information Technology, University of Turku, Finland. His research interests include algorithms and networking in computer games.
E-mail: jouni.smed@it.utu.fi
URL: http://staff.cs.utu.fi/staff/jouni.smed/

Harri Hakonen acts as a lecturer at the Department of Information Technology, University of Turku, Finland. Apart from computer games, his research interests include string algorithms and object orientation.
E-mail: harri.hakonen@it.utu.fi
URL: http://staff.cs.utu.fi/staff/harri.hakonen/

References

ALEXANDER, THOR (2002). "GoCap: Game Observation Capture." *AI Game Programming Wisdom*. Ed. Steve Rabin. Hingham, MA: Charles River Media. 579-589.

AYLETT, RUTH & SANDY LOUCHART (2003). "Towards a Narrative Theory of Virtual Reality." *Virtual Reality* 7.1: 2-9.

BRINGSJORD, SELMER (2001). "Is It Possible to Build Dramatically Compelling Interactive Digital Entertainment?" *Game Studies* 1.1.
<http://www.gamestudies.org/0101/bringsjord/> [2002-11-20]

BUNGIE SOFTWARE (2003). *Halo: Combat Evolved*. Microsoft Games.

CHARLES, FRED, STEVEN J. MEAD & MARC CAVAZZA (2002). "Generating Dynamic Storylines Through Characters' Interactions." *International Journal of Intelligent Games & Simulation* 1.1:5-11.
<http://www.scit.wlv.ac.uk/~cm1822/ijigs11.htm> [2004-11-18]

COSTIKYAN, GREG (2002). "I Have No Words & I Must Design: Toward a Critical Vocabulary for Games." *Computer Games and Digital Cultures Conference Proceedings*. Ed. Frans Mäyrä. Tampere, Finland. 9-33.

CRAWFORD, CHRIS (1984). *The Art of Computer Game Design*. Berkeley, CA: Osborne/McGraw-Hill.
<http://www.vancouver.wsu.edu/fac/peabody/game-book/Coverpage.html> [2004-11-18]

DEVTEAM (2004). *NetHack*. <http://www.nethack.org> [2004-11-18]

DEWDNEY, A. K. (1984). "Computer Recreations: In the Game Called Core War Hostile Programs Engage in a Battle of Bits." *Scientific American* 250.5: 14-22.

DIGITAL ILLUSIONS (2004). *Battlefield: Vietnam*. Electronic Arts.

DYBSAND, ERIC (2004). “GDC 2004 AI Roundtables Moderator Report.” <http://www.gameai.com/cgdc04notes.dybsand.html> [2004-11-18]

ENCYCLOPÆDIA BRITANNICA (2004). "game." *Encyclopædia Britannica Online*. <http://www.eb.com> [2004-04-23]

ENSEMBLE STUDIOS (1999). *Age of Empires II: The Age of Kings*. Microsoft Games.

EVANS, RICHARD (2002). “Varieties of Learning.” *AI Game Programming Wisdom*. Ed. Steve Rabin. Hingham, MA: Charles River Media. 567-578.

FAIRCLOUGH, CHRIS & PÁDRAIG CUNNINGHAM (2002). “An Interactive Story Engine.” *Proceedings of the 13th Irish International Conference on Artificial Intelligence and Cognitive Science*. Eds. Michael O’Neill et al. (Lecture Notes in Computer Science, 2464). Berlin: Springer. 171-176.

GAME DEVELOPERS’ ASSOCIATION OF AUSTRALIA (2003). “Game Industry Fact Sheet.” <http://www.gdaa.asn.au/about/gdaaindustryfactsheetoct2003.pdf> [2004-02-27]

GRAETZ, J. M. (1981). “The Origin of Spacewar.” *Creative Computing* August: 56-67. <http://www.wheels.org/spacewar/creative/SpacewarOrigin.html> [2004-10-14]

HUIZINGA, JOHAN (1955). *Homo Ludens: A Study of the Play-Element in Culture*. Boston, MA: The Beacon Press.

INTERNATIONAL GAME DEVELOPERS ASSOCIATION (2003). “IGDA Curriculum Framework: The Study of Games and Game Development.” <http://www.igda.org/academia/IGDA_Curriculum_Framework_Feb03.pdf> [2003-08-05]

INTERNATIONAL GAME DEVELOPERS ASSOCIATION (2004). “Foundations of Interactive Storytelling.” <http://www.igda.org/writing/InteractiveStorytelling.htm> [2004-09-17]

KAUKORANTA, TIMO, JOUNI SMED & HARRI HAKONEN (2003). “Understanding Pattern Recognition Methods.” *AI Game Programming Wisdom 2*. Ed. Steve Rabin. Hingham, MA: Charles River Media. 579-589.

KRASNER, GLENN E. & STEPHEN T. POPE (1988). “A Cookbook for Using the Model-View-Controller User Interface Paradigm in Smalltalk-80.” *Journal of Object-Oriented Programming* 1.3: 26-49.

LAIRD, JOHN E. & MICHAEL VAN LENT (2001). "Human-Level AI's Killer Application: Interactive Computer Games." *AI Magazine* 22.2: 15-25.

LINDLEY, CRAIG A. & MIRJAM ELADHARI (2002). "Causal Normalization: A Methodology for Coherent Story Logic Design in Computer Role-Playing Games." *Proceedings of the Third International Conference on Computers and Games*. Eds. Jonathan Schaeffer, Martin Müller & Yngvi Björnsson. (Lecture Notes in Computer Science, 2883). Berlin: Springer. 292-307

LIONHEAD STUDIOS (2001). *Black & White*. Electronic Arts.

MATEAS, MICHAEL & ANDREW STERN (2002). *Architecture, Authorial Idioms and Early Observations of the Interactive Drama Façade*. (Technical Report CMU-CS-02-198). Pittsburgh, PA: School of Computer Science, Carnegie Mellon University.

MAXIS (2000). *The Sims*. Redwood City, CA: Electronic Arts.

PEINADO, FEDERICO & PABLO GERVÁS (2004). "Transferring Game Mastering Laws to Interactive Digital Storytelling." *Technologies for Interactive Digital Storytelling and Entertainment*. Eds. Stefan Göbel et al. (Lecture Notes in Computer Science, 3105). Berlin: Springer. 48-54

PROPP, VLADIMIR (1968). *Morphology of the Folktale*. Austin, TX: University of Texas Press.

RELIC ENTERTAINMENT (1999). *Homeworld*. Bellevue, WA: Sierra Studios.

ROTOBEE (2004). *Singles: Flirt Up Your Life*. Deep Silver.

SALEN, KATIE & ERIC ZIMMERMAN (2004). *Rules of Play: Game Design Fundamentals*. Cambridge, MA: MIT Press.

SIIRA, ANTTI (2004). *Automatic Commentators*. Master's thesis. Turku: Department of Information Technology, University of Turku.

SMED, JOUNI, TIMO KAUKORANTA & HARRI HAKONEN (2002). "Aspects of Networking in Multiplayer Computer Games." *The Electronic Library* 20.2: 87-97.

SMED, JOUNI & HARRI HAKONEN (2003). "Towards a Definition of a Computer Game." (Technical Report, 553). Turku: Turku Centre for Computer Science.

SMED, JOUNI, TIMO KAUKORANTA & HARRI HAKONEN (2003). "AIsHockey – A Platform for Studying Synthetic Players." *Proceedings of the 2nd International Conference* *on Application and Developments of Computer Games*. Ed. Loo Wai Sing, Wan Hak Man & Wong Wai. Hong Kong. 183-188.

SPIERLING, ULRIKE (2002). “Digital Storytelling.” *Computers & Graphics* 26.1: 1-2.

SPIERLING, ULRIKE ET AL (2002). “Setting the Scene: Playing Digital Director in Interactive Storytelling and Creation.” *Computers & Graphics* 26.1: 31-44.

SULLIVAN BLUTH (1989). *Dragon's Lair*. ReadySoft.

VALVE SOFTWARE (1998). *Half-Life*. Sierra Studios.