

1

Introduction

Let us play a little thought game: Get a pen and paper. Choose any game you know, and think about the elements required to make it work. Write down a list of these elements. Be as specific or indiscriminate as you want. Once you are finished, choose another game and think about it. Try to find items in the list of the first game that correspond to the second game and mark them. If there are features in the second game that the first one does not have, add them to the list. Repeat this procedure for two or three more games. Next, take the five most common items in your list and compare them to the following list. For each corresponding item you get one point.

The key elements of a game are as follows:

- Players who are willing to participate in the game
- Rules that define the limits of the game
- Goals that the players try to achieve during the game
- Opponents or opposing forces that prevent the player from achieving the goals
- A representation of the game in the real world.

How many points did you score?

J. Huizinga's definition for play from his classical work *Homo Ludens*, the playful human, captures most of the features:

[Play] is 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, and mirth and relaxation follow. (Huizinga 1955, p. 132)

A dictionary definition elaborates 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 2005). Crawford



Figure 1.1 Components, relationships, and aspects of a game.

(1984, Chapter 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, p. 24), who sees a game as ‘an interactive structure of endogenous meaning that requires players to struggle toward a goal’, and Salen and Zimmerman (2004, p. 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’.

The five components we have listed seem to be present in every game, and the relationships between them form three aspects of a game, which are illustrated in Figure 1.1 (Smed and Hakonen 2003, 2005b):

- (i) *Challenge*: Rules define the game and, consequently, the goal of the game. When players decide to participate in the game, they agree to follow the rules. The goal motivates the players and drives the game forwards, because achieving a goal in the game gives the players enjoyment.
- (ii) *Conflict*: The opponent (which can include unpredictable humans and random processes) obstructs the players from achieving the goal. Because the players do not have a comprehensive knowledge of the opponent, they cannot precisely determine the opponent’s effect on the game.
- (iii) *Play*: The rules are abstract but they correspond to real-world objects. This representation concretizes the game for the players.

The challenge aspect alone is not enough for a definition of a game, 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. throw of dice or shuffling of the deck of cards). The main feature of the opponent is that it is indeterministic for 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 be played, the game needs to be concretized into a representation. This representation can be a cardboard and plastic pieces as well as three-dimensional graphics rendered on a computer screen. The players themselves can be the representation, such 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 a hand as possible when the cards are laid on the table, 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 these formal features, the game play also includes subjective elements such as 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, and we turn our focus to a subset of games, namely, computer games.

1.1 Anatomy of Computer Games

Computer games are a subset of games. To be more precise, let us define a 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 the computer. For example, a game of chess can be played on the screen or on a real-world board, regardless of whether the opponent is a computer program. Also, location-based games (see Chapter 8) further obscure the traditional role of a computer game by incorporating real-world objects into the game world.

In effect, a computer program in a game can act in three roles:

- (i) coordinating the game process (e.g. realizing a participant's move in a chess game according to the rules),
- (ii) illustrating the situation (e.g. displaying the chessboard and pieces on screen), and
- (iii) participating as a fellow player.

This role division closely resembles the *Model–View–Controller* (MVC) architectural pattern for computer programs. MVC was originally developed within the Smalltalk community

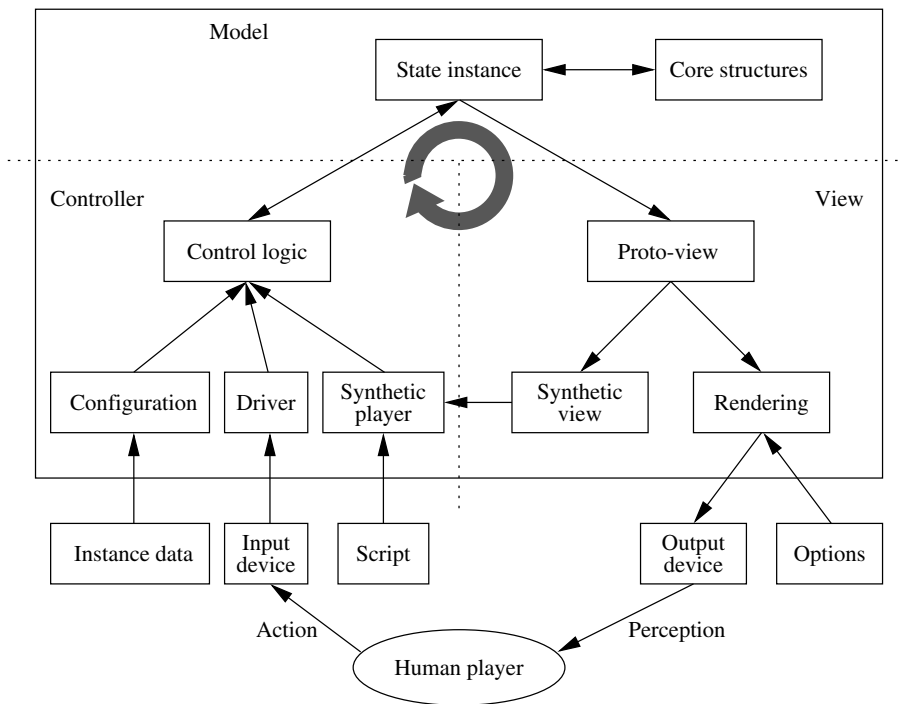


Figure 1.2 Model, View, and Controller in a computer game.

(Krasner and Pope 1988) and later on it has been adopted as a basis for object-oriented programming in general (Gamma *et al.* 1995). The basic idea 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 1.2 illustrates the MVC components and the data flow in a computer game.

The Model part includes software components that are responsible for the coordination 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. It remains unchanged while the state instance is created and configured for each game process. The core structures need not 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) and the instance data can provide the additional structures required for a game of poker (e.g. ranking of the hands, staking, and resolving ties).

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 coordinates rather than an image of the pieces on a board). Although rendering is often identified with visualization, it may as well include audification and other forms of sensory feedback. 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 keeps up 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 participates in the data flow by perceiving information from the output devices and performing actions through the input devices. Although the illustration in Figure 1.2 includes only one player, naturally there can be multiple players participating the data flow, each with their own output and input devices. Moreover, the computer game can be distributed among several nodes rather than reside inside a single node. Conceptually, this is not a problem since the components in the MVC can as well be thought to be distributed (i.e. the data flows run through the network rather than inside a single computer). In practice, however, the networked computer games provide their own challenges (see Section 1.3).

1.2 Synthetic Players

A synthetic player is a computer-generated actor in the game. It can be an opponent, a non-player character (NPC) that participates limitedly (like a supporting actor), or a *deus ex machina*, which can control natural forces or godly powers and thus intervene or generate the game events.

Because everything in a computer game revolves around the human player, the game world is anthropocentric. Regardless of the underlying method for decision-making (see Chapter 6), 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. As we can see in Figure 1.2, the data flow of the human player and the synthetic player resemble each other, which allows us to project human-like features to the synthetic player.

We can 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 to cope with uncertainties (see Chapter 7). Ideally, the synthetic players should be in a similar situation as their human counterparts, but of course a computer program is no match for human ingenuity. This is why synthetic players rarely display real autonomy but appear to behave purposefully (e.g. in *Grand Theft Auto III* pedestrians walk around without any real destination).

The more open (i.e. the less restrictive) the game world is, the more complex the synthetic players are. This trade-off between the Model and the Controller software components 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 a permitted area, path finding (see Chapter 5) for a synthetic player becomes simpler.

Let us take a look at two external features that a casual player is most likely to notice first in a synthetic player: humanness and stance. They are also relevant to the design of the synthetic player by providing a framework for the game developers and programmers.

1.2.1 Humanness

The success of networked multi-player games can be, at least in part, explained with the fact that the human players provide something that the synthetic ones still lack. This missing factor is the human traits and characteristics – 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* and *Halo* 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*, 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 a television show (Charles *et al.* 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* and *Singles*, where the synthetic players seem to act (more or less) with a purpose and the human player's influence is, at best, only indirect.

There are also computer games that do not have human players at all. In the 1980s *Core War* demonstrated that programming synthetic players to compete with each other can be an interesting game by itself (Dewdney 1984). Since then some games have tried to use this approach, but, by and large, artificial intelligence (AI) programming games have only been the by-products of 'proper' games. For example, *Age of Empires II* includes a possibility to create scripts for computer players, which allows to organize games where programmers compete on who creates 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.

1.2.2 Stance

The computer-controlled player can have different stances (or attitudes) towards the human player. Traditionally, the 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 keep 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 reconnaissance 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 be concrete operations like in *Neverwinter Nights* or *Star Wars: Battlefront* where the computer-controlled teammates 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, synthetic players can be used to carry on 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.

1.3 Multi-playing

What keeps us interested is – surprise. Humans are extremely creative at this, whereas a synthetic player can be lacking in humaneness. One easy way to limit the resources dedicated to the development of synthetic players is to make the computer game a multi-player game.

The first real-time multi-player games usually limited the number of players to two, because the players had to share the same computer by dividing either the screen (e.g. *Pitstop II*) or the playtime among the participating players (e.g. *Formula One Grand Prix*). Also, the first networked real-time games connected two players over a modem (e.g. *Falcon A.T.*). Although text-based networked multi-player games started out in the early 1980s with Multi-user dungeons (MUDs) (Bartle 1990), real-time multi-player games (e.g. *Quake*) became common in the 1990s as local area networks (LANs) and the Internet became more widespread. These two development lines were connected when online game sites (e.g. *Ultima Online*) started to provide real-time multi-player games for a large number of players sharing the same game world.

On the technical level, networking in multi-player computer games depends on achieving a balance between the consistency and responsiveness of a distributed game world (see Chapter 9). The problems are due to the inherent technical limitations (see Chapter 8). As the number of simultaneous players increases, scalability of the chosen network architecture becomes critical. Although related research work on interactive real-time networking has been done in military simulations and networked virtual environments (Smed *et al.* 2002, 2003), cheating prevention is a unique problem for computer games (see Chapter 10).

Nowadays, commercially available computer games are expected to offer a multi-player option, and, at the same time, online game sites are expected to support an ever increasing number of users. Similarly, the new game console releases rely heavily on the appeal of online gaming, and a whole new branch of mobile entertainment has emerged with intention to develop distributed multi-player games for wireless applications.

The possibility of having multiple players enriches the game experience – and complicates the software design process – because of the interaction between the players, both synthetic and human. Moreover, the players do not have to be opponents but they can cooperate. Although more common in single-player computer games, it is possible to include a story-like plot in a multi-player game, where the players are cooperatively solving the story (e.g. *No One Lives Forever 2* and *Neverwinter Nights*). Let us next look at storytelling from a broader perspective.

1.4 Games and Storytelling

Storytelling is about not 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’ storytelling progresses linearly, a game must provide an illusion of free will (Costikyan 2002). According to Aylett and Louchart (2003), computer games differ from other forms of storytelling in that the story time and real time are highly contingent, whereas in traditional forms of storytelling (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 storytelling. Bringsjord (2001) lists four challenges to interactive storytelling: First, a plot and three-dimensional characters are not enough to produce a high-quality narrative but there has to be a theme (e.g. betrayal, self-deception, love, or revenge) behind them. Second, there should exist some element to make the story stay 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 on the basis of its own beliefs.

Anthropocentrism is reflected not only 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 while proceeding with the game. The goals of the game can get blurry, and synthetic players or events should 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 storytelling 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 – like in *Dragon’s Lair*, where, at each stage, the players can choose from several alternative actions of which all but one leads 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. For instance, in *Max Payne* or *Diablo II*, the plot lines of the previous chapter are concluded at the transition point, and many new plot alternatives are introduced. Still, many games neither include a storyline nor impose a sequence of events. Granted that some of them can be tedious (e.g. *Frontier: Elite II*, in which the universe is vast and devoid of action whereas in the original *Elite* the goal remains clearer) – but so are many games that include a story.

Research on storytelling computer systems is mainly motivated by the theories of V. Propp (1968), because they help to reduce the task of storytelling to a pattern recognition problem; for example, see Fairclough and Cunningham (2002); Lindley and Eladhari (2002); Peinado and Gervás (2004). This pattern recognition approach can even be applied hierarchically to different abstraction levels. Spierling *et al.* (2002) decompose the storytelling system into four parts: story engine, scene action engine, character conversation engine, and actor avatar engine. These engines either rely on pre-defined data or act autonomously, and the higher level sets the outline for the level below. For example, on the basis of the current situation the story engine recognizes an adaptable story pattern and inputs instructions

for the scene action engine to carry out. In addition to these implementation-oriented approaches, other methodological approaches to interactive storytelling have been suggested in the fields of narratology and ludology, but we omit a detailed discussion of them here.

The main problem with the often-used top-down approach is that the program generating the story 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 twist in the plot, or has there been too much action and the players would 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 storyteller or dominant dungeon master, the plot could get 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 alone with a grave responsibility: They must provide a sense of purpose in the chaotic world.

1.5 Other Game Design Considerations

Although defining what makes a game enjoyable is subjective, we can list some features that alluring computer games seem to have. Of course, our list is far from complete and open to debate, but we want to raise certain issues that are interesting in their own right but which – unfortunately – fall out of the scope of this book.

- *Customization*: A good game has an intuitive interface that is easy to learn. Because players have their own preferences, they should be allowed to customize the user interface to their own liking. For example, the interface should adapt dynamically to the needs of a player so that in critical situations the player has more detailed control. If a player can personalize her avatar (e.g. customize the characteristics to correspond to her real-world persona), it can increase the immersion of the game.
- *Tutorial*: The first problem a player faces is learning how the game works, which includes both the user interface and the game world. Tutorials are a convenient method for teaching the game mechanics to the player, where the player can learn the game by playing it in an easier and possibly assisted mode.
- *Profiles*: To keep the game challenging as the player progresses, it should support different difficulty levels that provide new challenges. Typically, this feature is implemented by increasing certain attributes of the enemies: their number, their accuracy, and their speed. The profile can also include the player's preferences of the type of game (e.g. whether it should focus on action or adventure).
- *Modifications*: Games gather communities around them, and members of the community start providing new modifications (or 'mods') and add-ons to the original game. A modification can be just a graphical enhancement (e.g. new textures) or it

can enlarge the game world (e.g. new levels). Also, the original game developers themselves can provide extension packs, which usually include new levels, playing characters, and objects, and perhaps some improvement of the user interface.

- *Replaying*: Once is not enough. We take pictures and videotape our lives. The same also applies to games. Traditionally, many games provide the option to take screen captures, but replays are also an important feature. Replaying can be extended to cover the whole game, and the recordings allow the players to relive and memorize the highlights of the game, and to share them with friends and the whole game community.

It is important to recognize beforehand what software development mechanisms are published to the players and with what interfaces. The game developers typically implement special software for creating content for the game. These editing tools are a valuable surplus to the final product. If the game community can create new variations of the original game, longevity of the game increases. Furthermore, the inclusion of the developing tools is an inexpensive way – since they are already implemented – to enrich the contents of the final product.

Let us turn the discussion around and ask what factors are responsible for making a computer game bad. It can be summed in one word: *limitation*. Of course, to some extent limitation is necessary – we are, after all, dealing with limited resources. Moreover, the rules of the game are all about limitation, although their main function is to impose the goals for the game. The art of making games is to balance the means and limitations so that this equilibrium engrosses the human player. How do limitations manifest themselves in the program code? The answer is the lack of parameters: The more the things are hard-coded, the lesser the possibilities to add and support new features. Rather than shutting out possibilities, a good game – like any good computer program! – should be open and modifiable for both the developer and the player.

1.6 Outline of the Book

The intention of our book is to look at the algorithmic and networking problems present in commercial computer games from the perspective of a computer scientist. As the title implies, this book is divided into two parts: algorithms and networking. This emphasis on topic selection leaves out components of Figure 1.2 that are connected to the human-in-the-loop. Most noticeably, we omit all topics concerning graphics and human interaction – which is not to say that they are in any way less important or less interesting than the current selection of topics. Also, game design as well as ludological aspects of computer games fall out of the scope of this book.

The topics of the book are based on the usual problems that game developers encounter in game programming. We review the theoretical background of each problem and review the existing methods for solving them. The given algorithmic solutions are given not in any specific programming language but in pseudo-code format, which can be easily rewritten in any programming language and – more importantly – which emphasizes the algorithmic idea behind the solution. The algorithmic notation used is described in detail in Appendix A.

We have also included examples from real-world computer games to clarify different uses for the theoretical methods. In addition, each chapter is followed by a set of exercises, which go over the main points of the chapter and extend the topics by introducing new perspectives.

1.6.1 Algorithms

Part I of this book concentrates on typical algorithmic problems in computer games and presents solution methods. The chapters address the following questions:

- Chapter 2 – Random Numbers: How can we achieve indeterminism required by games using deterministic algorithms?
- Chapter 3 – Tournaments: How we can form a tournament to decide a ranking for a set of contestants?
- Chapter 4 – Game Trees: How can we build a synthetic player for perfect information games?
- Chapter 5 – Path Finding: How can we find a route in a (possibly continuous) game world?
- Chapter 6 – Decision-Making: How can we make a synthetic player act intelligently in the game world?
- Chapter 7 – Modelling Uncertainty: How can we model the uncertainties present in decision-making?

1.6.2 Networking

Part II turns the attention to networking. We aim at describing the ideas behind different approaches rather than get too entangled in the technical details. The chapters address the following questions:

- Chapter 8 – Communication Layers: What are the technical limitations behind networking?
- Chapter 9 – Compensating Resource Limitations: How can we cope with the inherent communication delays and divide the network resources among multiple players?
- Chapter 10 – Cheating Prevention: Can we guarantee a fair playing field for all players?

1.7 Summary

All games have a common basic structure comprising players, rules, goals, opponents, and representation. They form the challenge, play, and conflict aspects of a game, which are reflected, for instance, in the Model–View–Controller software architecture pattern. The computer can participate in the game as a synthetic player, which can act in the role of

an opponent or a teammate or have a neutral stance. For example, synthetic player must take the role of a storyteller, if we want to incorporate story-like features into the game. Multi-playing allows other human players to participate in the same game using networked computers.

Game programming has matured from its humble beginnings and nowadays it resembles any other software project. Widely accepted software construction practices have been adopted in game development, and, at the same time, off-the-shelf components (e.g. 3D engines and animation tools) have removed the burden to develop all software components in house. This maturity, however, does not mean that there is no room for artistic creativity and technical innovations. There must be channels for bringing out novel and possibly radically different games, and, like music and film industry, independent game publishing can act as a counterbalance to the mainstream.

Nevertheless computer games are driven by computer programs propelled by algorithms and networking. Let us see what they have in store for us.

Exercises

- 1-1** Take any simple computer game (e.g. *Pac-Man*) and discern what forms its challenge aspect (i.e. player, rules and goal), conflict aspect, and play aspect.
- 1-2** A crossword puzzle is not a game (or is it?). What can you do to make it more game-like?
- 1-3** Why do we need a proto-view component in the MVC decomposition?
- 1-4** What kind of special skills and knowledge should game programmers have when they are programming
 - (a) the Model part software components,
 - (b) the View part software components, or
 - (c) the Controller part software components?
- 1-5** Let us look at a first-person shooter (FPS) game (e.g. *Doom* or *Quake*). Discern the required software components by using the MVC. What kind of modelling does it require? What kind of View-specific considerations should be observed? How about the Controller part?
- 1-6** *Deus ex machina* (from Latin ‘god from the machine’) derives from ancient theater, where the effect of the god’s appearance in the sky, to solve a crisis by divine intervention, was achieved by means of a crane. If a synthetic player participates in the game as a *deus ex machina*, what kind of role will it have?
- 1-7** What does ‘anthropocentrism’ mean? Are there non-anthropocentric games?
- 1-8** *The Sims* includes an option of free will. By turning it off, the synthetic players do nothing unless the player explicitly issues a command. Otherwise, they show their own initiative and follow, for example, their urges and needs. How much free will

should a synthetic player have? Where would it serve the best (e.g. in choosing a path or choosing an action)?

- 1-9** Many games are variations of the same structure. Consider *Pac-Man* and *Snake*. Discern their common features and design a generic game that can be parameterized to be both the games.
- 1-10** Judging rules can be difficult – even for an objective computer program. In football (or soccer as some people call it), the official rules say that the referee can allow the play to continue ‘when the team against which an offence has been committed will benefit from such an advantage’ and penalize ‘the original offence if the anticipated advantage does not ensue at that time’ (Federation Internationale de Football Association 2003). How would you implement this rule? What difficulties are involved in it?

