

Chapter 2

Believability Through Psychosocial Behaviour: Creating Bots That Are More Engaging and Entertaining

Christine Bailey, Jiaming You, Gavan Acton, Adam Rankin
and Michael Katchabaw

Abstract Creating believable characters or bots in a modern video game is an incredibly challenging and complex task. The requirements for achieving believability are steep, yet the improvements in player engagement, immersion, and overall enjoyment and satisfaction with their experience make this a worthy problem in desperate need of further examination. In this chapter, we examine how the implementation of psychosocial behaviour in bots can lead to believability, and how this translates into new and exciting possibilities for advancing the state-of-the-art in this area.

2.1 Introduction

The field of Artificial Intelligence in games is a broad, yet demanding area of study. In [62], artificial intelligence is defined as being concerned with thought processes, reasoning, and behaviour as it applies to human performance as well as ideal intelligence, or rationality. Artificial Intelligence in games differs from traditional Artificial Intelligence in that it is optimized not for the simulation of human performance

C. Bailey · J. You · G. Acton · A. Rankin · M. Katchabaw (✉)
Department of Computer Science, The University of Western Ontario,
London, ON N6K 4K8, Canada
e-mail: katchab@csd.uwo.ca

J. You
e-mail: jyou23@csd.uwo.ca

G. Acton
e-mail: gacton@csd.uwo.ca

A. Rankin
e-mail: arankin@csd.uwo.ca

C. Bailey
e-mail: cdbailey@csd.uwo.ca

or rationality, but for entertainment and increased immersion in the game world [58, 73].

One of the more active areas of research in game artificial intelligence in both industry and academia is the creation of more believable characters or bots [11, 24–26, 30, 31, 37, 54, 59]. This is only natural, as players of modern video games increasingly expect Artificial Intelligence that is dynamic, is able to react to unexpected events, and behaves believably [17, 68–70]. This is particularly the case in character and story-driven genres, such as role-playing games and action-adventure games, but is also true of open world games and others in which individual characters or populations of characters are important to the overall player experience.

The state-of-the-art in the creation of believable decision making processes for bots has reached out beyond computer science using models from psychology (personality, emotions, appraisal theory) and sociology (social appraisal variables, relationships and role theory), as these elements have been recognized as foundations for believable behaviour [40]. Formalizing and encoding psychosocial elements for use in games has proven to be challenging, yet rewarding in terms of the believable bot behaviour that can be achieved through their use, and the richer and more engaging experiences that can be delivered to players as a result [4, 80].

This chapter provides a thorough and detailed treatment of this topic, delving into both theoretical and practical aspects of providing psychosocial bot behaviour in modern video games. The remainder of this chapter is sectioned as follows.

Background: Introducing the reader to this area, providing motivation for the use of psychosocial elements in defining more believable behaviour, and discussing the current state-of-the-art.

Introduction to Psychosocial Behaviour: A brief overview of relevant aspects of psychology and sociology and how these elements impact decision making processes and behaviour.

A Framework for Psychosocial Behaviour: An examination of how to formally model and represent various psychosocial elements for use in video games.

Implementation and Results: With the framework in mind, this section discusses practical aspects of implementing the framework and delivering psychosocial behaviour in a game.

Performance Optimization: Given that games are real-time and interactive by nature, the use of psychosocial behaviour raises performance considerations that must be dealt with [56].

Concluding Remarks: A summary of what has been covered, as well as a discussion of open problems and opportunities for further studies in this area.

2.2 Background

The literature in this area is rich with work taking various approaches to exploring aspects of psychosocial behaviour for believable bots. While this work provides many valuable lessons of importance to this area, the majority of this work has limitations

or lacks necessary elements for realistic and believable psychosocial behaviour. This section provides an overview of this background work.

Bot behaviour is often handled using finite state machines to represent state of mind [11, 24], or scripting to hard-code behaviour in each possible game situation [8]. Since the main goal for game artificial intelligence is to support the designers in providing a compelling game experience, supporting interactivity, player choice, and replayability [58], these commonly used approaches are too limiting to provide truly believable behaviour.

One of the factors in creating a compelling experience is suspending the player's disbelief. The Autonomy Interaction Presence (AIP) Cube model (as cited by [11]) states that the requirements for suspension of disbelief are: Autonomy of the bots, Interaction between the bots and the player, and the bots' Presence in the game world. The work in [11] also implicates personality, emotion, self-motivation, the illusion of life (which includes goals, plans, reactions to the environment, human-like restrictions, and many other behaviours which create the appearance of life), change, and social relationships as being important to character believability. This supports the commonly referenced definition of believability proposed by Loyall [40].

Some relevant readings in the area of affective computing and emotion synthesis include Funge's definition of emotion as the sum of past events [24], the discussion in [11] of character emotion in the context of games, and Picard's discussion of shifting emotions [52], and Picard in [52] and the overview in [11] of the field of affective computing. Personality in agents is defined as the agent's pattern of behaviours or interactions with its environment [37]. Crawford [15] outlines one possible personality model for bots in games that include intrinsic variables (i.e. integrity, virtue, intelligence, and so on), mood variables, relationship variables (beliefs about another's intrinsic variables), and the readiness to change the previous two variables. Isbister discusses the social psychology research and discusses the traits of agreeableness and dominance and how they can be used to form many different personalities [35].

Reputation systems—such as those in *Fable* [3], *Thief: Deadly Shadows* [34], and *Ultima Online* [29]—refer to systems that typically manage bot opinions of the player [3, 45], which are formed immediately and globally among all bots upon certain player actions [3, 10, 45, 29]. Reputation systems typically do not maintain individual opinions [3, 45], nor opinions about other bots, though they may maintain group opinions [3, 29, 34]. The work in [30] is an exception to this, however, providing a more flexible and realistic method of modeling reputation.

Some social science concepts of interest in video game research include an agent's roles [31, 35], cultures and subcultures [35], norms, values, worldview [31], and goals [35]. Some papers have attempted to address these [31, 35]. The area of social agents, or Socially Intelligent Agents (SIAs) [11, 55] would appear to have much relevance to this research, however many of these “social” agents do not exhibit realistic social behaviour [13, 31, 32, 35, 77]. Many “social” agents implement only communicative behaviour that is used in a multi-agent problem-solving context to reduce resource usage and increase efficiency. Some relevant research in this area includes Tomlinson and Blumberg's remembered interaction histories between agents [72], Prendinger

et al.’s change of attitudes and familiarity assessment between agents [55], Cole’s comparison of opinion flow in a multi-agent system to flocking behaviour [14], and Guye-Vuilleme’s high level architecture for social agents [31].

Finally, a particularly interesting approach to problems of providing unique and immersive experiences lies in emergent gameplay [51, 66, 78]. Until recently, emergent gameplay is known to have been used in bot behaviour only in relatively simple situations and behaviours [38, 51, 66, 69, 78], usually to deal with emergent properties of the game world and the objects contained within it. Emergence, however, had not been used to implement complex psychological states or social relationships. The foundations for the approach to believable bot behaviour discussed in this chapter come from our own previous work in [4], which first attempted to apply emergence to the creation of realistic psychosocial behaviour. This work was extended in [80] to provide better and more complete psychosocial modeling, and again in [56] to integrate more rigorous proactive planning elements to complement the more reactive behaviour present in our earlier work.

2.3 Introduction to Psychosocial Behaviour

The requirements for believability for bots in modern video games tend to be quite steep [40]. They include elements such as personality, emotion, self-motivation, social relationships, consistency, the ability to change, and the ability to maintain an “illusion of life”, through having goals, reacting and responding to external stimuli, and so on [40]. While crafting a game artificial intelligence capable of achieving all of this is a daunting task indeed, one can see that at its core, a complete and integrated psychosocial model is necessary, operating in a dynamic fashion [4].

In this section, we examine the needs of bots from both psychological and sociological perspectives, and briefly discuss key relevant works in these areas.

2.3.1 *Psychological Foundations*

Elements of a bot’s psychology that form the basis for its behaviour can include a model of emotion, a personality model, needs, and even values and a worldview [7]. Since there are different theoretical models of emotions and personality [23], and different game worlds may require different needs, values, worldview and cultural constructs, it is necessary to have a flexible approach to modeling psychological traits. For brevity, we will restrict our discussion here to personality and emotion, as these tend to be the most commonly used and referenced psychosocial elements in the literature [4].

Numerous models for personality have been developed over the years. One is the Myers-Briggs Type Indicator [44], which is based on four pairs of traits that are considered complementary and distinct that measure: how a person relates to

others (either by Extraversion or Introversion), how a person takes in information (either by Sensing or Intuition), how a person makes decisions (either by Thinking or Feeling), and how a person orders their life (either by Judging or Perceiving). Another popular model that has been advanced by many researchers is known as the Five Factor Model (FFM), the Big Five, or OCEAN, which assesses personality in five independent dimensions: Openness, Conscientiousness, Extraversion, Agreeableness and Neuroticism [74]. This work has origins that can be traced back to a sixteen factor model created by Cattell [12]. The PEN model [22], on the other hand, is comprised of just three personality dimensions, and is based on psychophysiology: Psychoticism, Extraversion, and Neuroticism. A slightly different perspective is offered in Reiss' model of basic desires [57], where personality is defined primarily by a set of tendencies and motivators that inspire or lead to action: power, curiosity, independence, status, social contact, vengeance, honour, idealism, physical exercise, romance, family, order, eating, acceptance, tranquillity and saving. Other work in this area has examined linkages amongst these and other models [1, 42], and while connections and correlations exist, there are still fundamental differences between these models, and no single complete over-arching model.

Similarly, several models of emotion have been formulated and studied. One of the more popular models was formulated by Ekman [18], defining six basic emotions: anger, disgust, fear, joy, sadness, and surprise. Another popular model is the OCC model, consisting of twenty-two emotions [47]. Scaled down versions of this model also exist [46]. Numerous other models exist, including Smith and Ellsworth's Emotion Appraisal Model [65], Mehrabian's PAD Emotional State Model [43], and models put forward by Tomkins [71], Plutchik [53], and Parrott [50]. Again, while there is overlap between these models, these models have many differences and were defined with different purposes in mind, such as facial expression, relation to adaptive biological processes, action readiness, and so on.

When applying these various psychosocial models to the creation of bots for games, researchers have followed one of two paths. In the first case, researchers have selected one of the models that they believe best suits their needs (or one each of personality and emotion), in the hopes that this will be sufficient and that their bots will not suffer from missing what is offered by the other models. This was done in work such as [5, 6, 19, 20, 75, 81] and our own previous work in [4]. In the second case, researchers have instead constructed their own custom models, borrowing aspects from the common standard models in an ad hoc fashion, as none of these models on their own meet their needs. This was done in work such as [27, 33, 48, 59, 61, 64]. Unfortunately, to date, there has been no work towards integrating the various models together or enabling their interoperability within agents or game characters, despite the potential benefits of leveraging the respective advantages of these well-studied and time-tested models all at once. An approach to doing this, however, is presented in Sect. 2.4.

2.3.2 *Sociological Foundations*

Humans are social beings, and our social relationships, culture, values, and norms all play an important part in defining who we are. As a result, believable bots must be socially situated, and able to reason in social terms [16, 28, 31, 41, 60]. Bots armed with social knowledge will have greater believability as they interact with players and each other [31, 60]. Psychology alone does not describe important aspects of human behaviour including social constraints, moral standards and routine behaviours [31], and so some notion of sociology is also required in bots. Unfortunately, social bots are particularly difficult to create as they must reproduce behaviours that are complex, reflecting a number of intricacies of human behaviour [31]. This has led to a number of creative approaches attempting to create believable social bots.

An important social element within bots is social relationship. While they may be simply defined as a link between two people, social relationships are clearly identified with additional information including social variables, emotions, and roles. Isbister notes that relationships are in part defined by two primary social variables being agreeableness and dominance [35]. Dominance is tied to the concept of social power, while agreeableness helps define the type of relationship as friendly, unfriendly, and neutral [35]. Our own earlier work in [4] implemented Isbister's social variables through a rule-based emergent social system that allowed for the varied interaction among its bots based on their social variables, emotional state and personality factors.

Social relationships have also been bolstered with quantitative social emotions in order to give feeling to the bots' relationships. The OCC cognitive models [47] have been used to define relationships with variables such as the degree to which a bot likes or dislikes another, the degree to which a bot is familiar with another, and the degree to which a bot has formed a cognitive unit with the other [60]. Several systems have implemented this approach to relationships. Some use this existing information in the causal attribution of blame for an event [28], as well as defining when another is friend or foe. These social emotions may update using a rule-based approach during the appraisal of the event. If a bot is perceived to have caused a harmful event, for example, a rule will convert emotional reaction such as fear to an increased dislike for the bot [27].

Another view of society and social relationships is through the concept of roles. Based on role theory, a role defines the relationship of a person to another person, group or even object [9]. Roles are often thought of as a concept used in theatre, where an actor plays an assigned part [9]. In reality, a role is more intuitively defined, with an individual assuming potentially multiple roles, depending on social context, relative role importance, and a variety of other factors. A role does more than simply formalize a relationship; it also brings together desires (goals), intentions, and beliefs [49]. Applying this to the creation of believable bots, roles provide a formalized description of what is expected from a bot in various social contexts [31, 49]. This provides significant advantages for designers as roles can be reused, and are intuitively understood by non-experts [31, 49]. For these reasons, we ultimately adopt a role-based approach in this chapter for social modeling, as discussed in Sect. 2.4.

2.4 A Framework for Psychosocial Behaviour

The fundamental design of a psychosocial bot behaviour framework is based on the core ideas presented earlier in this chapter. To provide dynamic and unscripted behaviour, this framework relies on various elements of emergence. Recall that emergence focuses on a bottom-up view of the game world, where simple component-level behaviours interact to form complex system-level behaviour. This is done by making every object in the world self-centred, goal-directed, and responsible for its own needs and responses. Emergent systems do not focus on algorithmic behaviours, but rather very simple stimulus responses at the component level [78]. It is important to note, however, some structure to cognitive processes is required for a bot; otherwise, pure emergence can lead to bots that appear to be too reactive or impulsive, with no long-term goals or planning [80].

To extend the ideas behind emergent systems into a psychosocial context, simple psychosocial objects must be defined that can react to psychosocial stimuli and other psychosocial objects, and be able to maintain their own attributes, needs, and responses to the dynamic game environment. Such psychosocial objects and stimuli can include emotions, personality traits, roles, bots, groups, or static objects of import to bots (such as possessions). In this way, simple component-level psychosocial behaviours will be defined that can interact in complex and interesting ways.

With this in mind, we first discuss the modeling of bots from a psychosocial perspective that is necessary to enable the mechanisms for emergence presented. Following this, we then describe the logic and mechanisms required to support emergent psychosocial behaviour.

2.4.1 Psychosocial Modeling

As discussed above, the work of Loyall [40] identified several psychological and sociological elements required to support believable behaviour in bots. Chief amongst these elements were personality, emotion, and some form of social network connecting bots with one another in various social contexts [40]. Consequently, for brevity, we will focus our discussion in this section on these elements, but our approach to modeling is general and applicable to other psychosocial elements as well [80].

2.4.1.1 Modeling Personality and Emotion

Selecting individual existing approaches to psychosocial modeling, as discussed in Sect. 2.3.1, is problematic as there is no single universal approach that captures all characteristics and possibilities to be modeled. As a result, a flexible approach to multi-model integration was adopted in our earlier work in [80], and provides

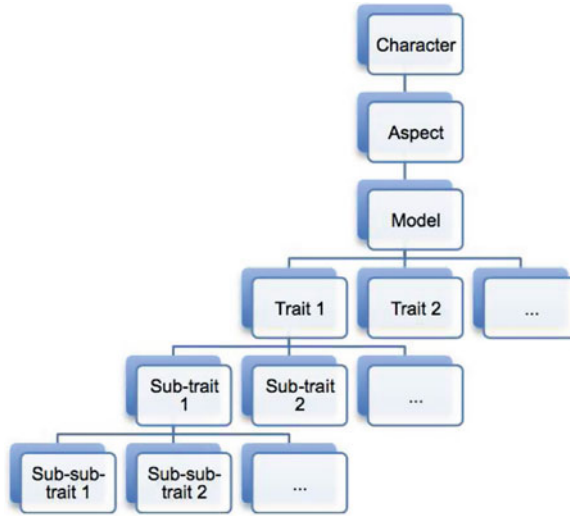


Fig. 2.1 Template for vertical scaling in a model

a method of accessing traits across model boundaries through both vertical and horizontal scaling of models.

The process of vertical scaling in psychosocial models refers to a hierarchical decomposition of the model into a collection of various traits, sub-traits, sub-sub-traits, and so on, typically in a tree or graph like fashion. Higher levels of the trait hierarchy would represent more abstract traits, while lower levels of the hierarchy would contain more concrete or more detailed traits. These hierarchies of traits have been suggested in the literature, but do not appear directly in most of the core models [50, 63, 67].

A template for vertical scaling in a model is shown in Fig. 2.1. At the highest level of the hierarchy is the character itself. Below that would be the aspect of the character being modeled below, such as personality or emotion. Below the aspect would be the model in question, and below that would be the traits, sub-traits, and additional refinements necessary to fully and completely describe the model. Doing so allows designers and developers to explore depth, subtle nuances, and detailed elements of a particular character with respect to a given aspect and model. (As we will demonstrate shortly, this vertical scaling also enables horizontal scaling, and the integration of multiple models with overlapping or otherwise similar features.)

For example, consider the emotion model of Parrott [50] with primary, secondary, and tertiary emotions. A partial set of traits for a particular bot, Alice, using only Parrott’s emotion model could be visualized in Fig. 2.2.

Naturally, to define a character and drive believable behaviour from them, we need to assign values to the particular traits. (For instance, if a character is feeling anger, we must specify how angry that particular character is.) Even on this seemingly simple point, the literature in the area is fairly divided, with some researchers favouring

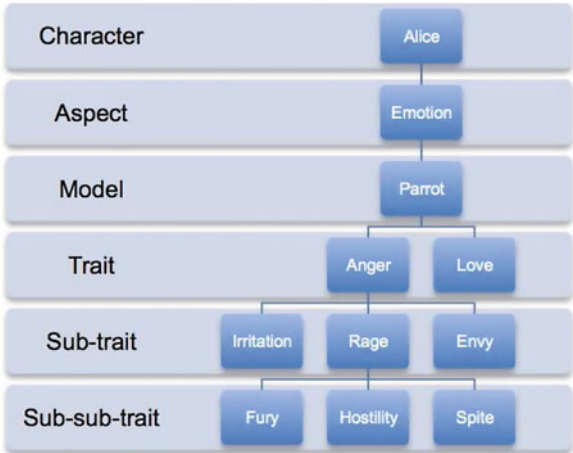


Fig. 2.2 Partial trait model for bot alice

discrete values (such as on or off, or present or not present), and others favouring a continuous scale (such as between 0.0 and 1.0 or -1.0 and 1.0). For our purposes, we have chosen to use a continuous scale between -1.0 (strongly negative) and 1.0 (strongly positive) for primitive trait values, as this would accommodate the majority of other approaches with the least difficulty. (For example, a discrete present or not present trait could be modeled by values of 1 and 0 respectively.) Some models like Reiss’ [57] require compound traits with multiple values (such as a set point or target and a current value, in this case), but these approaches can typically be supported using a small set of primitive trait values [80].

To maintain consistency within the model, weights can be assigned to the linkages between a trait and its sub-traits to determine the contribution of each sub-trait towards the parent trait and the distribution from the parent trait to each sub-trait. With these weights in place, adjustments to lower-level traits made manually during production by game designers or developers, or at run-time by the game itself, can then propagate to higher-level traits, and vice-versa.

For example, consider the personality scenario in Fig. 2.3, showing the FFM trait of extraversion from [74] decomposed into just two of the possible sub-traits identified in [63]. In this example, the Outgoing trait accounts for 70 % of the extraversion trait value, while Independent accounts for 30 %. The character in question is very independent (0.9) and quite outgoing (0.8), so we would expect that the character would also be highly extraverted. Given these contributions and the sub-traits in Fig. 2.3, the value for extraversion can be computed as $(0.7 \times 0.8) + (0.3 \times 0.9) = 0.83$.

With this approach to vertically scaling psychosocial models, we can flexibly provide and automatically manage a great deal of depth and detail on the various aspects of non player characters. Through this scaling to sub-traits and sub-sub-traits, and the weightings applied in the process, we can now turn our attention



Fig. 2.3 Applying weights to traits and sub-traits

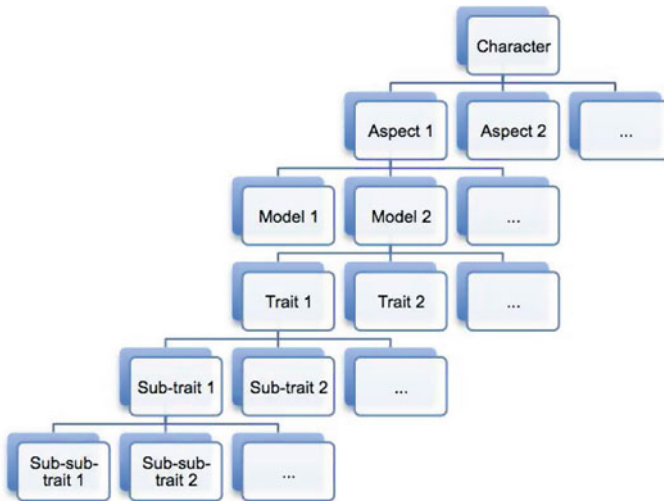


Fig. 2.4 Template for horizontal scaling across models

to horizontal scaling. While vertical scaling of a psychosocial model adds depth, horizontal scaling adds breadth, in this case integrating various models together and providing interoperability. Depending on designer and developer requirements, horizontal scaling can integrate entire models or only certain traits and sub-traits from models. Furthermore, additional traits from outside traditional psychosocial models can also be integrated, and other changes can be made to tailor and tune the resulting model to a great degree.

Conceptually, this can be as simple as the template for horizontal scaling shown in Fig. 2.4, demonstrating the integration of multiple aspects and multiple models for each aspect. When integrating multiple aspects, such as personality and emotion, with only one model per aspect, this is a relatively simple process as the aspects themselves are usually independent and do not contain conflicting traits or other potential issues.

For example, we can perform a relatively simple integration of personality and emotion using the FFM model [74] for modeling personality and Ekman's model [18] for representing emotion. A partial set of traits for a particular character, Bob, integrating these aspects and models could be visualized in Fig. 2.5.

Fig. 2.5 Partial trait model for bot Bob

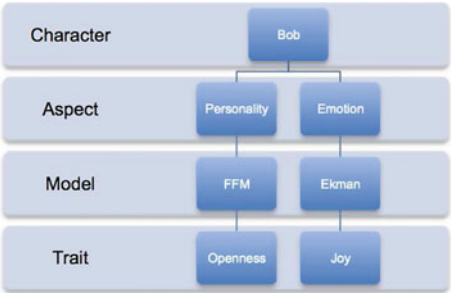
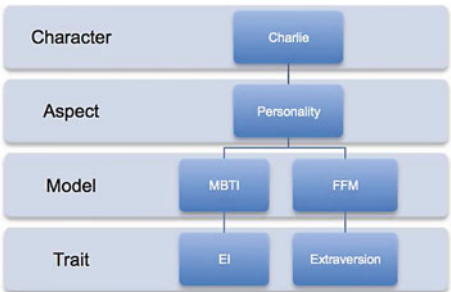


Fig. 2.6 Partial trait model for bot Charlie



The situation gets significantly more complicated, however, when combining multiple models under a single aspect because of the potential overlap and conflict in the models. Consider, for example, the partial set of traits for a character Charlie, whose personality we want to model to leverage elements of both MBTI [44] and FFM [74], as shown in Fig. 2.6. This combination of models can be useful, for example, when character behaviour is guided through FFM traits, and career and professional orientation are determined through MBTI [36, 44].

In the example in Fig. 2.6, both models have traits to represent the concepts of extraversion and introversion. (There are also other sources of overlap and potential conflict between these models, as discussed in [42], but the concepts of extraversion and introversion are the most highly correlated, pose the most obvious problem, and are the easiest to discuss here.) If these overlapping traits are not synchronized with one another, character behaviour could become erratic and inconsistent, thereby destroying any player immersion or suspension of disbelief [11, 39], which is highly undesirable in the game. Without any linkage between traits across models, synchronization is unfortunately difficult and error-prone, as changes and updates made manually by designers and developers must occur multiple times, and those that occur at run-time during the game must occur to all affected traits in tandem. (Doing so is complicated even further when the connections and correlations between traits across models are less obvious and better hidden within the models.)

To solve the above problem and ensure model consistency and believable behaviour when scaling horizontally to accommodate additional psychosocial models, we

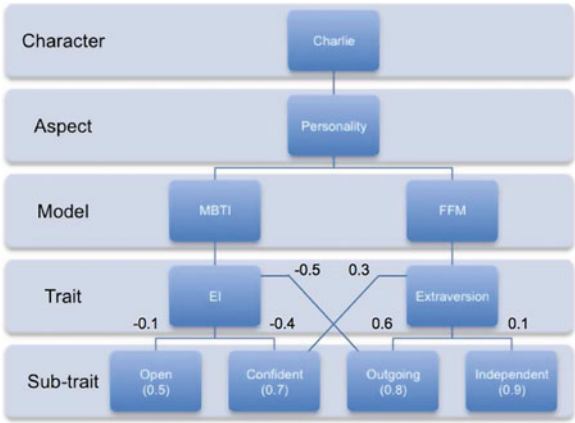


Fig. 2.7 Expanded partial trait model for bot Charlie

can take advantage of the vertical scaling discussed in the previous section. Overlap and conflicts occur between models because of elements in common somewhere in the trait hierarchies of the models. Otherwise, the models would be independent, and there would be no problems or difficulties in the first place.

By hierarchically decomposing model traits into sub-traits, sub-sub-traits, and so on, we can uncover the common elements between the models. Instead of duplicating the common elements in each trait hierarchy, we instead use only a single element with separate links into the higher levels of the corresponding trait hierarchies. By doing things in this fashion, we can take advantage of the common elements to link the various models together and synchronize them with one another.

For example, recall the partial set of traits for the character Charlie given in Fig. 2.6. While there is a strong correlation between the respective extraversion and introversion elements between the MBTI and FFM models used in this case [42], it is not perfect, suggesting that we cannot simply use a common extraversion trait to replace the corresponding traits in each model. Consequently, we can decompose these traits into their respective sub-traits, and potentially lower-level traits beyond that point as necessary. In the process of doing so, we can also assign weights to the linkages between the higher-level and lower-level traits in the hierarchies to indicate relative contribution and distribution, as we did in the previous section. When we do this, we end up with the expanded partial trait model shown in Fig. 2.7. (Note that we are following the same simplified decomposition as in Fig. 2.3 for the extraversion trait of FFM, and that decompositions and weightings were made for illustrative purposes in this example, and do not necessary reflect how scaling should actually be done.) In this example, we can make note of several things:

- Not every sub-trait needs to be linked to both trait hierarchies. This is only logical, since different models focus on different things. In this example, the sub-trait Open

is only associated with EI of the MBTI model, and the sub-trait Independent is only associated with Extraversion in FFM.

- The weights assigned to links between the traits and sub-traits differ between the two models. Again, this makes sense since different models emphasize and consider traits differently.
- The weights assigned can be negative, as was done with sub-traits linked to the EI trait of the MBTI model. In this case, doing so makes sense, since strongly negative values of EI indicate a highly extraverted individual and strongly positive values indicate a highly introverted individual.

With trait hierarchies linked in this fashion, the models can remain synchronized and consistent. Additional models and traits can be added for further horizontal scaling, building additional linkages as necessary.

When performing horizontal and vertical scaling, it is important to base decisions in hierarchical decomposition, linkages, and weight assignment on scientific study, such as the work in [42, 63, 67], and the literature in this area. Even a rigorous approach, however, will not be able to handle all integration scenarios, as all the background research to do so does not exist, or has yet to be completed. In such cases, we must use our best judgment given the research results that are available, but leverage the flexibility and openness in our approach to allow easy changes, revisions, and additions in the future.

2.4.1.2 Modeling Social Information

As discussed in Sect. 2.3.2, we adopt a role-based approach to social modeling in this chapter. As noted in [31], there is little to no standardization in social models in the literature, and so we initially developed our own model for roles in [2].

Roles, in essence, are used to clearly define the relationship of a bot to its environment, including the player, other bots, and other things in the world around it. As such, they are used to contain the social knowledge used by the bot in its decision making processes. This includes desires, beliefs, and intentions as in other work [31, 49] as well as role personalization through reference to emotional state and personality [2]. This personalization enables prioritization of roles based on social context and bot psychology. Unlike other approaches, multiple roles can be attached to an individual, even conflicting roles, as such conflicts can be resolved using elements of personalization and prioritization. Consequently, roles are dynamic and can be added, removed, changed, disabled, and re-enabled over time on a bot-by-bot basis.

Figure 2.8 presents the key components of a role. Each of these components, as well as other meta-information required to support a role, is discussed below in further detail.

Name: The name of the role. Simply, it is a label used to refer to a role, and is usually human-friendly to allow designers to more readily make decisions about its attachment to and usage by bots in the game. For example, one role could be “Mother”.

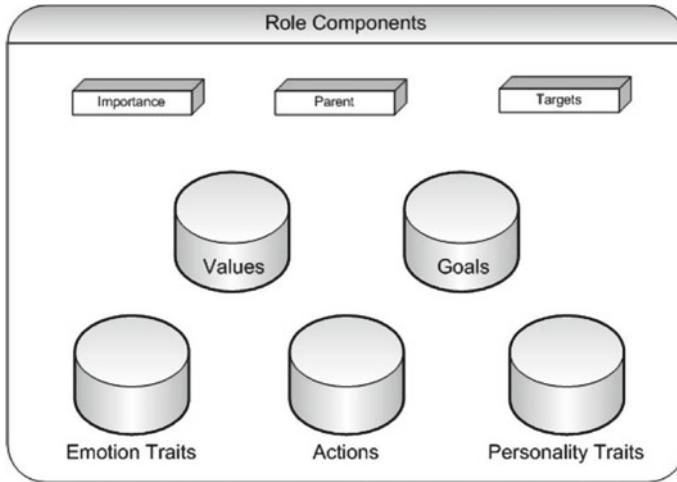


Fig. 2.8 Components of a role

Bot: The bot to whom this role is attached. This is assigned when the role is added to the bot and allows the role to tap into and access other bot-specific information and internal state. Continuing the above example, any female bot with children could have the “Mother” role attached to it, and this element of the role refers back to the appropriate “Children” bots.

Target: Who or what are the targets of the role. Typically this will be another bot, a group, or even the bot itself. The target can be any other object within the world. Targets are assigned to the bot during the role’s instantiation. Continuing the above example, other bots designated as children of a female bot would be the targets in that bot’s “Mother” role.

Goals: Goals are used to capture desires associated with a role. In essence, a role’s goals are world states achievable through actions that can be taken by the bot. Typically, although not always, goals are related to the targets of the role either directly or indirectly. Continuing the above example, the mother in the “Mother” role could have a goal stating that the targets (her children) are in a happy emotional state.

Values: Each role can also be composed of values or beliefs about the world that produce emotional reactions to events and generate information relating to the consequence of actions. Values, norms, or a world view make up a bot’s belief system and play an important part in many decision making processes including appraisal and planning. Continuing the above example, the mother in the “Mother” role could have a belief that she should not physically strike her children in disciplining them.

Emotion Traits: A role’s emotions define how the bot feels towards the role’s targets. These emotions play a critical role in defining how a bot will interact with the target, and also contribute towards the bot’s general mood, which govern how it behaves in general. Continuing the above example, the mother in the “Mother” role could feel love for her children, and happiness to be with them. Not only does this impact

her actions with her children, but her happiness will carry over into her mood and interactions with other people. Likewise, the removal of her children will decrease her happiness, and their absence could have a noticeable impact on her mood and interactions with others.

Personality Traits: A role may need access to the attached bot's personality, as these traits could impact decision making processes involving the role. Furthermore, similar to emotions above, there may be alterations of personality traits imposed by the presence of a role that might not otherwise occur naturally in the bot. Continuing the above example, a mother in the "Mother" role could have a heightened sense of responsibility and a more caring demeanour. (Of course, she might not, which could lead to interesting consequences.)

Actions: The actions defined within the role are role specific actions used during the planning process to create a plan to meet the goals of the role. Actions will use the role's emotions and personality traits to modify their utility during the planning process. Continuing the above example, a mother in the "Mother" role could have access to actions to feed and clothe her children, which are likely not present in many of the other roles in the game.

Importance: The importance of the role to the bot defines, at a high level, how critical the role is to the agent. This importance impacts goal and action selection of the bot, ultimately determining its behaviour. It also influences how events associated with the role are committed to emotional memory, and how the emotions of the role impact general mood. Continuing the above example, the mother in the "Mother" role could also be an employee in an "Employee" role and have to balance the needs of her job against the needs of her children. The importance of each role determines the balance. If her "Mother" role is more important, she will sacrifice aspects of her job in support of her children. Likewise, the happiness derived from her children will outweigh negative feelings that come from a hard day of work. Role importance is dynamic, and can be influenced by emotion, personality, social context, and other factors.

Parent: Roles can be hierarchical in nature, with child roles inheriting aspects of parent roles. This allows for a more modular, reusable role structure. Continuing the above example, the "Mother" role could have a "Parent" role as its parent. Social knowledge and information common to both a "Mother" and a "Father" role could be encapsulated in this "Parent" role, with only specific differences appearing in the particular child roles.

2.4.2 The Mechanics of Psychosocial Behaviour

With models for psychological and sociological elements defined in the previous section, we can now turn our attention to using these models to create believable psychosocial behaviour in bots. In this section, we present two mechanisms that work in tandem: a stimulus-response system to produce reactive actions, and a

goal-oriented utility-based planning system to produce proactive actions. These systems are derived from our earlier work in this area [2, 4, 56, 80].

2.4.2.1 A Stimulus-Response System

As mentioned above, one form of action that is required as part of believable bot behaviour is the reactive action, or simply reaction. In this case, the bot is responding to a stimulus in the environment and carrying out an action with minimal or no formal planning involved in the process. To support this, a stimulus-response system can be used. We created one using principles of emergence in our earlier work in this area [4] and then extended this approach in [80].

Before delving into the specifics of the stimulus-response system, it is necessary to understand how emergent systems operate. Emergent systems use simple component-level behaviours that can interact to form complex system-level behaviour [66, 78]. These component-level behaviours operate as stimulus-responses, in that every entity in the world monitors its own perceptions of stimuli and are responsible for responding to those stimuli [78]. Such systems have been used in games before [66, 76] for handling interactions between physical objects and physiological needs, but not for complex psychosocial modeling, as discussed in this section.

Central to our approach is the generalized stimulus-response system shown in Fig. 2.9. This system operates by having objects create and listen for stimuli. By defining a general channel that allows only certain objects to interact with each other through a stimulus that is created and responded to, we have an easy to use mechanism for defining emergent interactions.

The object or event that begins the interaction by creating a stimulus is referred to as the Actor. We will refer to the object that detects and responds to the stimulus as the Reactor (which may then become an Actor through its response). The stimulus acts as a message that is broadcast from the Actor and is received by the Reactors in the area. Reactors listen for messages with particular stimulus types, since each object will react to different stimuli. The stimulus messages can also hold data about the

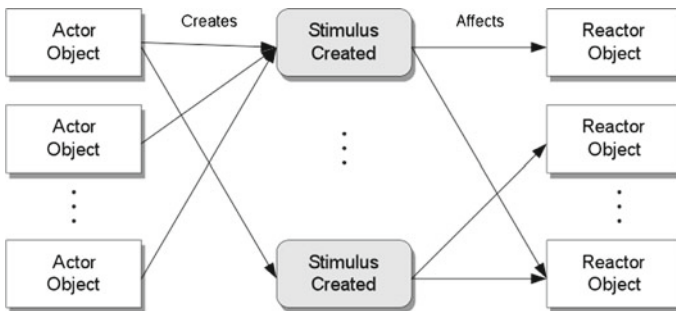


Fig. 2.9 A generalized stimulus-response system

attributes of the associated stimulus. These attributes can be mapped onto different behaviours; it is important to note that Reactors do not have to share the same response to a given stimulus.

For a bot to react in a believable fashion, they should have the ability to react to the actions of other bots, seeing bots or objects (i.e. seeing an enemy upon turning a street corner), as well as other events or occurrences in the world, such as natural disasters or accidents. All of these antecedents could be considered the Actors that cause some sort of stimuli. Since these events are to be affected and be reacted to in a psychosocial context, the stimuli created by these events would be psychosocial in nature, such as emotional stimuli (i.e. a witnessed attack may cause an emotional stimulus of fear) or a stimulus that causes a perceived change in situation (i.e. the same witnessed attack may cause the bot to feel the situation has become hostile). Psychosocial stimuli are created by Actors as messages and are broadcast to the social objects in the area, as done in Fig. 2.9. These messages hold data about the psychosocial stimulus, such as stimulus source, type, magnitude, propagation, and fall off.

Stimulus source refers to the origin of the stimulus. Stimulus type refers to various emotional and social stimuli, such as happiness, sadness, anger, hostile environment, and groupthink. Magnitude of the stimulus would likewise describe the magnitude of the psychosocial effect of the stimulus (i.e. was this a highly emotional event?). Propagation would describe whether the psychosocial stimulus only affects the people directly involved in the event (such as two people greeting each other), if it affects the people witnessing it at a distance as well (such as two people attacking each other). Other types of social stimuli can also be modeled using these stimulus messages, such as gossip and the spread of information. Social stimuli over long distances (such as gossip through a telephone) could be sent as a direct message from one person to another.

It is useful to note that a social fall off radius could serve as a way of simulating social phenomena. By defining the psychosocial stimulus to propagate by radius and defining a fall off rate, an entire group of people can be affected by the behaviour of a few. This may assist in modeling social behaviour such as riots or group-induced panics. While this may not be an accurate model, it can simulate reactive behaviour that appears to be life-like.

The Actors can be anything that broadcasts psychosocial stimuli, including bots, objects, and events. The psychosocial stimuli are broadcast to all other social objects in the nearby area. The social objects that are receiving these broadcasts are the interaction Reactors as described above. These Reactor objects can be any social object that can react, including bots or other social beings, such as groups. Note that non-responsive social objects (such as possessions) will not react to psychosocial stimuli in any way and therefore it would not be desirable to include these objects as Reactors. It may sometimes be desirable to have more general emergent gameplay, by having a stimulus-response system that processes both physical stimuli, and psychosocial stimuli simultaneously. In these cases non-socially-reactive objects may be treated as Reactors that are only listening for physical, not psychosocial stimuli.

Unlike some of the Reactors described above, social objects, in general, react to, or at least detect, psychosocial stimuli of all types. People tend to react to all emotions, any social situation or stressors they find themselves in, and participate in (or at least detect and make a decision in response to) social phenomena that are occurring around them. In cases where these reactions do not take place, it is because the person (or social object) has gone through some decision making process to decide how (or how not) to react. This process will require the detection and processing of the stimulus, regardless of its type. Therefore, social objects listen for all types of psychosocial stimuli. However, social objects will typically ignore events that are not relevant to them (i.e. a discussion between two strangers). Therefore social objects can listen for psychosocial stimuli with particular relevance types; some may only be relevant to individual social objects or groups of social objects, while others may be relevant to all.

By defining a psychosocial mechanism in the way described in this section, bots and other social objects can react to psychosocial stimuli in realistic as well as unexpected and novel ways to their environment. This can result in chain reactions as reactions to psychosocial stimuli create behaviours or events that can in turn cause more psychosocial stimuli that can be reacted to. This would achieve the goal of having emergent social behaviour in bots. By allowing the game player to participate in this mechanism through their actions in the game world, this enables meaningful player decision making and emergent gameplay.

Given the potential of this approach, it is also important to recognize its limitations. Bots equipped only with this stimulus-response system can only react to what is going on around them. They would come across as overly reactive or impulsive, with no long-term goals or planning [80]. In other words, they would not be very believable for very long. When something of import is going on, the bots could behave quite well, but in the absence of significant stimuli, the bots would be aimless and directionless. A solution to this problem comes in the form of proactive action, as discussed in the next section.

2.4.2.2 A Goal-Oriented Utility-Based Planning System

Given the limitations of a purely reactive stimulus-response system, as discussed above, a new approach was taken to introduce goal-oriented planning [80] to bot decision making processes. Doing so allows bots to be more proactive in their actions, carrying out actions to move themselves towards accomplishing their goals. This approach was then extended to support utility-driven psychosocial behaviour and appraisal theory [2, 56].

At the core of this system is the decision making process illustrated in Fig. 2.10. While more structured than the stimulus-response system of Sect. 2.4.2.1, this process is still an emergent one, with complex results arising from relatively simple interactions that occur during the process. The various elements of this process are discussed below in further detail. It is important to note that this system can either work on its



Fig. 2.10 A goal-oriented utility-based planning system

own, or together with the stimulus-response system of Sect. 2.4.2.1 to have reactions handled through that mechanism instead of through planning.

Event: A notification of activity in the game world, whether it is from other bots, the world itself, or simply the passage of time. When the planning system is connected to the stimulus-response system of Sect. 2.4.2.1, these events are Actors which could lead to the generation of stimuli upon appraisal.

Appraisal: The event is examined to see if it is of interest to the bot, and if so, does the event or its consequences have sufficient importance to the bot to warrant further consideration. Deadlines for action/reaction may also be set, depending on the event. When the planning system is connected to the stimulus-response system of Sect. 2.4.2.1, appraisal flags events requiring reaction to create stimuli for the stimulus-response system after coping has been completed.

Coping: The bot reflects on the event and updates its internal psychosocial and physiological state based on the event. This adjusts the bot's mood, emotional memory, and so on. When the planning system is connected to the stimulus-response system of Sect. 2.4.2.1, any events flagged by appraisal to create stimuli are sent to the stimulus-response system. The planning system continues to operate in parallel as goals and plans may also need to be adjusted in response to this event, in addition to any reactive actions generated by the stimulus-response system.

Goal Selection: Given the current state of the bot, its surroundings, its social context, and recent events, the bot determines the goals that it should be trying to meet. Goals might be immediate, short-term, medium-term, and long-term. Goals are largely selected from roles attached to the bot according to current relevant role importance, as discussed in Sect. 2.4.1.2 but could be derived from other sources as well.

Planning/Action Selection: Considering the goals in place, a plan is created to achieve the goals with the desired results and maximum utility, with acceptable side effects and consequences. If a plan already exists and is in process, it might be revised to reflect changes in goals, bot state, and so on. With the plan in mind, appropriate actions are selected to have the plan implemented. Note that if events indicated a reaction is required, as described above, the current plan may be temporarily bypassed or abandoned to carry out alternate actions. (This may be the case, for example, to ensure self-preservation.)

Output Events: Based on the actions selected, new events are generated and propagated into the game accordingly.

With these systems designed and in place, we can now examine how we can implement and provision these systems for use in creating believable bots.

2.5 Implementation and Results

Proof of concept of the framework of psychosocial models and systems presented in Sect. 2.4 has been completed through the creation of a series of prototype implementations, each of which capable of supporting believable psychosocial characters of varying degrees of complexity. Such characters would be suitable for story-driven role-playing or action-adventure games, or other types of games as discussed in Sect. 2.1. In this section, we discuss the creation of these prototypes and briefly highlight results obtained through their use.

2.5.1 First Generation Prototype

The first generation prototype was created as part of the work described in [4]. It focused primarily on the creation of the emergent stimulus-response system discussed in Sect. 2.4.2.1 based on a simple psychosocial model that was a precursor to the modeling approaches described in Sect. 2.4.

Microsoft Visual Studio was used to develop the prototype on the Microsoft Windows platform. To promote portability of the code, the prototype was developed in C++ using OpenGL to render a simple graphical representation of social objects, the social ties between them, and their moods. A limited character state was implemented including a categorical representation of emotion, a simple personality

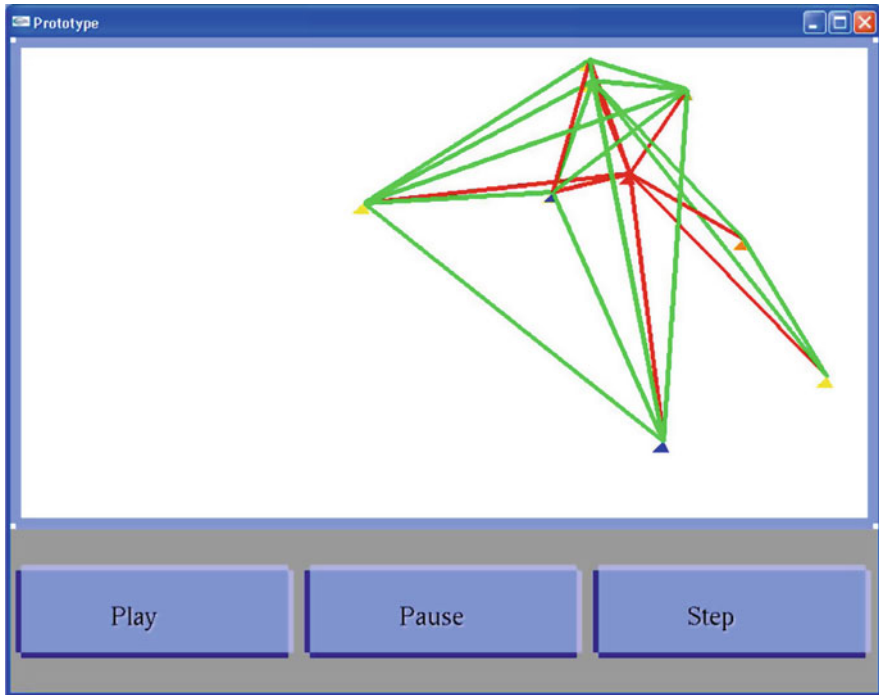


Fig. 2.11 Screenshot from first generation prototype

model, symmetric social ties, and a small behaviour/stimulus set. These elements provided sufficient proof of concept for initial validation and testing.

A screenshot from the first generation prototype system is given in Fig. 2.11. The system provided feedback on bots in the system and their relationships with one another through a set of colour-coded nodes and edges between them.

A series of experiments was conducted with this system. While it was found that bots would have believable reactions to one another in the system [4], the limitations of a purely reactive stimulus-response system were evident and it was clear that a more advanced system was necessary [80].

2.5.2 Second Generation Prototype

The second generation prototype was created as part of the work described in [80]. Building upon the first generation prototype system discussed in Sect. 2.5.1, this prototype was aimed at improving upon its predecessor in several ways. From a modeling perspective, it supported the multi-model integration discussed in Sect. 2.4.1.1 as well as a simple role-based social model that was a precursor to the model discussed in

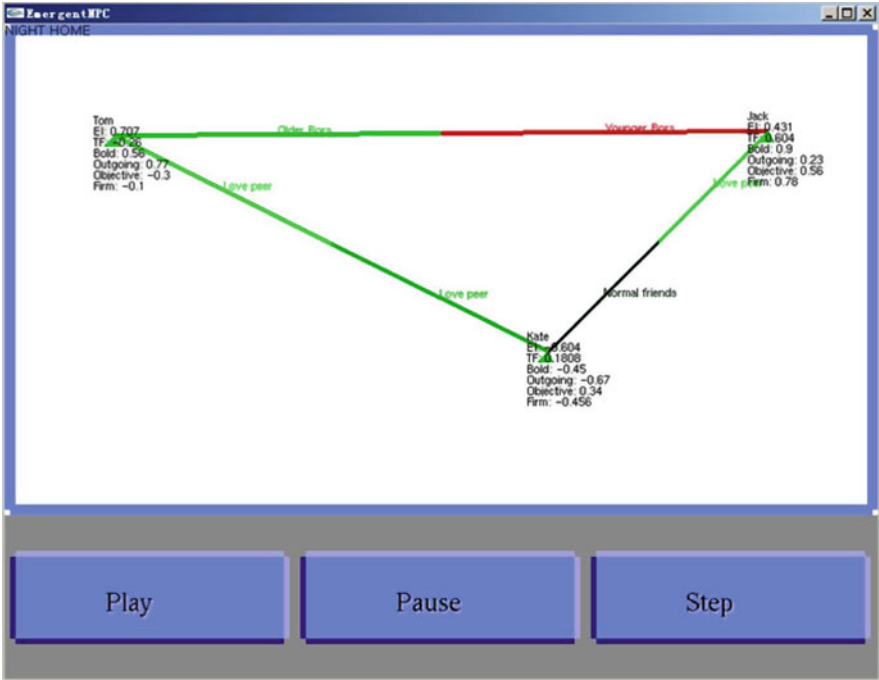


Fig. 2.12 Screenshot from second generation prototype

Sect. 2.4.1.2. This prototype also supported goal-oriented planning that was the first step towards what would become the goal-oriented utility-driven planning system from Sect. 2.4.2.2.

Like the first generation prototype, the second generation prototype system was developed for the Microsoft Windows platform, written in C++ using Microsoft Visual Studio, with OpenGL used for interface development. This prototype system also introduced a collection of management tools to help designers develop and work with psychosocial models. These tools were created for the same platform, using C# as a language instead, because of its rapid prototyping capabilities. Screenshots of these aspects of this prototype are provided in Figs. 2.12 and 2.13 respectively. Note that the interface in Fig. 2.12 has been augmented to provide additional feedback and data in comparison to its predecessor, while its overall functionality remained quite similar.

A series of experiments was also conducted with this system. Results were, in fact, improved over the previous system [80], and the system benefited from the more flexible approach to psychosocial modeling and the addition of planning. Bots were more flexible and more expressive in terms of their emotion and personality, and the use of goals and planning enabled bots to act more thoughtfully and intentionally, instead of in a simple, reactionary fashion. It was found, however, that the system

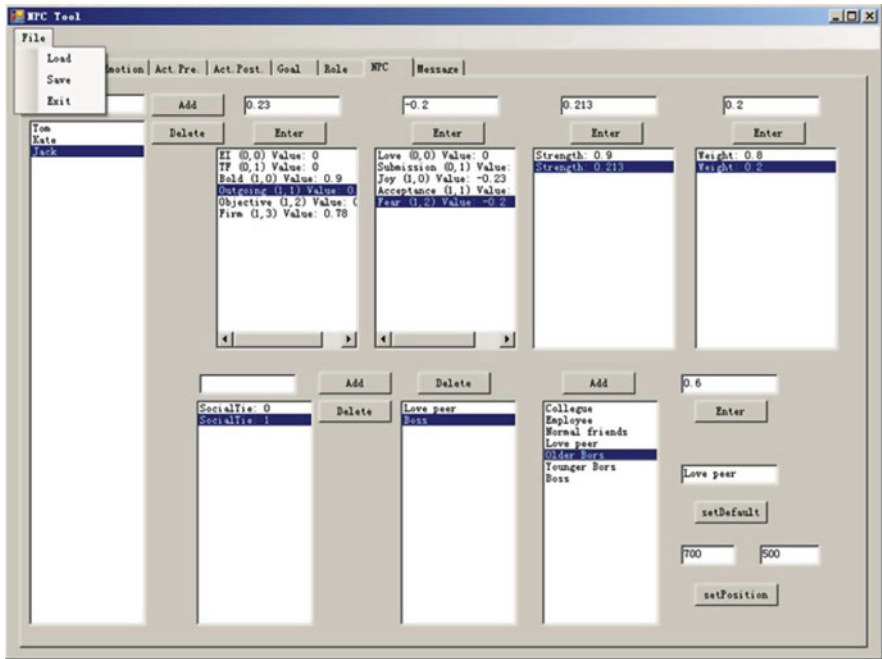


Fig. 2.13 Screenshot from second generation management tools

could be improved through an improved role model and a more robust planning system, which led to the creation of a third generation prototype.

2.5.3 Third Generation Prototype

The third generation prototype was created as part of the work described in [2, 56]. Improving on its predecessors, this prototype now supported a richer and more complete model of social roles, as discussed in Sect. 2.4.1.2, as well as the goal-oriented utility-driven planning system outlined in Sect. 2.4.2.2. This allows bots to make decisions and select courses of actions based on how well those actions satisfy their goals, taking into consideration the positive and negative consequences of doing so. (For example, an action might be very useful to satisfying a goal, but have very negative side-effects, making that action less desirable than one with less utility but no negative consequences.)

This prototype system was once again developed for the Microsoft Windows platform using C++ as part of Microsoft Visual Studio. This generation of the prototype system employed a logging system to report on the status of bots during testing

instead of a graphical interface, because there was now such a volume of information that the graphical approach grew too cluttered and difficult to use.

A series of experiments was also conducted using this prototype, to great success. By this stage in development, the prototype could re-enact segments of Shakespeare's *Romeo and Juliet* quite believably. Bots playing the characters from the play were given personalities and emotional states based on an analysis of the original play, and a social structure was put in place reflecting the familial struggles taking place. With these models in place, various scenarios were played out, producing similar results to the original play [2].

Consider, for example, one of the most pivotal moments in the play, Act 3, Scene 1, as the outcome of this scene ultimately leads to Romeo's exile and the suicide of the lovers. In the scene, Romeo and Mercutio are socializing when the sworn enemy Capulet cousin Tybalt enters. Mercutio and Tybalt fight, and Romeo is faced with a dilemma: should he fight the sworn enemy of his family or save his lover's cousin? In an attempt to save both lives, Romeo separates the two by putting himself between them. It is then that Tybalt slays Mercutio. Seeing the death of his best friend, Romeo is overcome with rage, and having lost all care of consequence, slays Tybalt and sets in motion the tragic events in latter parts of the play.

With appropriate psychosocial models in place to reflect the characters in the scene [2], we set the bots in motion using the prototype system and observed the unscripted results as recorded by the logs. The beginning of the scene can be viewed below in an excerpt of the log file's events. (Note that the actual logs are much more detailed, showing planning processes, utility calculations, consequence calculations, and so on at each step, but these details are only shown where necessary below for brevity.)

```
1) Event: Agent: Mercutio Action: talk Target: Romeo
   Utility: 0.89375
2) Event: Agent: Romeo Action: talk Target: Mercutio
   Utility: 0.7
** Enter Tybalt **
3) Event: Agent: Tybalt Action: Threaten_Yell Target:
   Mercutio Utility: 0.85125
```

As expected, Romeo and Mercutio converse until Tybalt enters the scene and threatens Mercutio, as the two are enemies according to their social roles and dislike for each other. During the appraisal here, Romeo adds an enemy role towards Tybalt in response to Tybalt's threat to his friend.

```
4) Event: Agent: Mercutio Action: Threaten_Yell
   Target: Tybalt Utility: 0.849304
5) Event: Agent: Romeo Action: talk Target: Mercutio
   Utility: 0.6915
6) Event: Agent: Tybalt Action: Threaten_Yell Target:
   Mercutio Utility: 0.850875
```

In the second log file excerpt, Mercutio angered by Tybalt's threat returns the favour that in turn causes Romeo to adjust his view towards Mercutio. Nonetheless,

Romeo continues to talk with Mercutio and Tybalt continues his verbal assault on Mercutio.

```

7) Event: Agent: Mercutio Action: Attack_Sword Target:
   Tybalt Utility: 0.731439
8) Event: Agent: Romeo Action: Defend_Separate Target:
   Tybalt Utility: 0.64784
9) Event: Agent: Tybalt Action: Attack_Sword Target:
   Mercutio Utility: 0.729146
***** Mercutio Has Died *****

```

Looking at event seven, we see that Mercutio has loosed his sword in response to Tybalt's threats. (As an interesting note, we had to lower Mercutio's sword skills in this scene. Otherwise, he was just as likely to kill Tybalt instead, which would lead to a very different, although still realistic, retelling of the scene.) We now have a pivotal moment for Romeo whose internal struggle is apparent with his planning process. Despite the tense situation, attacking either Mercutio or Tybalt is not an option. As we see below for the utility calculations for a strike against Tybalt, the consequence utility is zero, meaning that this action has very negative consequences. While Romeo views Tybalt as an enemy for his threats against his friend, he cannot bring himself to fight because of the consequences and his love for Juliet. So, even though there is equal utility to fighting and separating the two combatants, the poor consequences for fighting cause Romeo to opt for separating the combatants instead, as there are the consequences are better (positive) for doing so.

```

Action Selection: Attack_Hand Targets: Tybalt
                  Utility 0.64784
State: 0.64784
Consequence: 0

```

So, Romeo attempts to separate the two combatants, as this is the best option available. Tybalt, responding to Mercutio's threat, attacks and kills Mercutio. The result here is extreme anger in Romeo. Examining his state after the appraisal of Tybalt's killing blow, we find rage in Romeo. Below shows his logged anger emotion both before and after Mercutio's death.

```

Emotion: anger Value: 0.0387244 (Before Event 9)
Emotion: anger Value: 1 (After Event 9)

```

Being completely enraged, Romeo's planning reflects this state. His care of consequence drops to zero given the extreme emotions of the situation. (Using consequence values during planning is controlled by a care of consequence variable. In highly charged emotional states, people may care less about the consequences of their actions, and so this variable is tuned in bots according to certain emotional states. This allows them to care about the consequences of their actions or not, depending on their state, as Romeo does in this example.) As a result, attacking Tybalt becomes an option, because he is ignoring the negatively appraised consequences of such an action.

```

* Care of Consequence: 0
Target Watch Triggered: Tybalt
Action Selection: Attack_Hand Targets: Tybalt
                  Utility 0.74575

State: 0.62575
106
Consequence: 0.12
Target Watch Triggered: Tybalt
Action Selection: Attack_Sword Targets: Tybalt
                  Utility 1.02075

State: 0.75075
Consequence: 0.27
10) Event: Agent: Romeo Action: Attack_Sword Target:
    Tybalt Utility: 0.75075
11) Event: Agent: Tybalt Action: Attack_Sword Target:
    Romeo Utility: 0.689941
***** Tybalt Has Died *****

```

Therefore, Romeo in his fury slays Tybalt, thus ending the dispute and the scene. By examining this scene, Romeo's actions appear to be believable. Not only did we see restraint in his initial actions, action utilities, care of consequence calculations, and appraisal of events, but we also saw the raw power that emotion can have on the planning process. This resulted in the believable, unscripted re-enactment of Act 3 Scene 1 of Shakespeare's *Romeo and Juliet*.

Numerous other scenarios have been enacted using this system, also to good results. For brevity, these results have been omitted here. The reader is urged to consult [2] for further examples and details.

2.5.4 Fourth Generation Prototype

At present, we are currently working on the fourth generation prototype system. Functionally, the system will have the same psychosocial framework and capabilities as its predecessor. This prototype, however, is being developed using the Unreal Development Kit (UDK) [21]. In doing so, we can tap into a fully-functional 3D environment, and develop a system that can be readily deployed in modern video games.

In addition to the psychosocial framework, we are developing an environment in UDK, a house, for experimenting with various social scenarios. A screenshot of this environment is given in Fig. 2.14.



Fig. 2.14 Screenshot from fourth generation prototype (in development)

2.6 Performance Optimization

While we have made great strides towards the creation of more believable bots for modern video games, computationally, these bots are orders of magnitude more complex than traditional approaches to bot Artificial Intelligence. Consequently these bots introduce the potential for serious performance problems, especially when there are a great number of them inhabiting a game world [4]. This problem is only exacerbated by the computational needs of other game sub-systems, which together put an even larger strain on often limited and over-taxed system resources. As a result, if we are to truly take advantage of believable bots in modern video games, we must take into consideration issues of performance and scaling with Artificial Intelligence as it has been done with other aspects of games, such as graphics [56]. While others have examined similar problems in the past [79], this has not been done for bots with psychosocial behaviour, which presents unique and interesting challenges and opportunities.

This chapter examines a scalable approach to believable bots that utilizes several techniques to improve in-game performance, derived from our work in [56]. These include dynamic importance calculation, capability scaling or reduction, and pre-emptive, priority-based bot scheduling and dispatching. Through utilizing these techniques, we can ensure that scarce computational resources are allocated to bots where they are most needed, and that bots are using the decision-making processes best suited to their current state and importance to the game, while still maintaining believability.

2.6.1 Approach to Performance Optimization

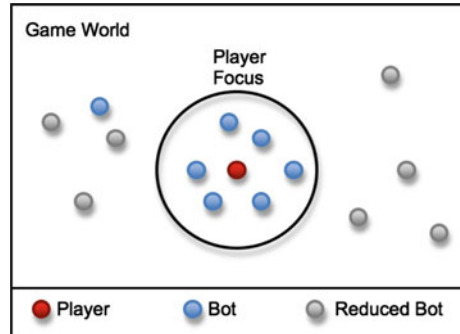
We begin by recognizing that not every bot in a game has equal importance to the player and the game at any point in time. Some bots are the focus of attention, are engaged in significant activities, or are otherwise critical to what is unfolding in the game. Others are of less importance, and their activities will largely, if not completely, go unnoticed to the player. As the player moves through the game world and plays the game, the importance of the various bots changes, with some becoming more important and others less so.

Furthermore, we note that bots that are not visible to the player, or otherwise not important to the player, do not need the same richness, detail, and fidelity in their behaviour as those that are the focus of attention, or are otherwise important, to maintain believability. In those cases, a game can safely do less with those bots, in many cases significantly so, with no perceptible difference to the player's experience. For example, suppose we have two bots in a game that are hungry; one, Alice, is in the same room as the player, while the other, Bob, is in a different locale far across the game world, outside of the player's ability to sense. To maintain believability, Alice must recognize her hunger, formulate a plan to satisfy her hunger considering her current psychosocial and physiological state and surroundings, and then execute this plan. After all, the player is in the same room and is able to see and hear everything she does; the slightest out of place action could sacrifice believability and adversely affect the player's experience. On the other hand, Bob would not need to do anything except simply continue to live to maintain believability. In the player's absence, it is likely reasonable to believe that he could provide himself with the necessities of life, without actually needing to do anything about it. In fact, the game would not even need to track Bob's hunger level to maintain the same level of believability. (That said, Bob would still need to progress in his life in the absence of the player, as it is likewise unreasonable to believe that he would do absolutely nothing at all when the player is not around. How this should be done to maintain believability is discussed below.)

With this in mind, we can safely scale or reduce the capabilities of a bot according to their current visibility and importance to the player while still maintaining believability across all bots in the game, as shown in Fig. 2.15. In doing so, we can achieve tremendous performance savings in bots with reduced capabilities as they require significantly less processing and this processing is far less complicated than bots operating at full capacity. This, in turn, allows the game to support a much larger number of bots without requiring additional resources to be dedicated to Artificial Intelligence processing.

To support this approach, each bot in a game is a separately schedulable and executable entity, enabling computational resources to be allocated by a scheduling and dispatching subsystem on a bot-by-bot basis according to their importance. Furthermore, each bot has access to a range of decision making and processing algorithms, allowing their capabilities to be scaled or reduced based on their importance. Lastly, bot importance is calculated and updated regularly based on a variety of factors to

Fig. 2.15 Bots in the game world



ensure that scheduling, dispatching, and capability adjustment are all carried out using the best available information. Each of these activities is discussed in further detail in the sections that follow below.

2.6.1.1 Scheduling and Dispatching of Bots

To allocate computational resources to bots in a game, a scheduling and dispatching subsystem is added to the game. The goal of this subsystem is rather simple—choosing the most appropriate characters to run at each game tick or frame of game execution.

Producing an optimal schedule for each tick is itself a computationally expensive task, and so we take a simpler, more heuristic approach using a system of priorities assigned to each bot in the game. (Priorities are derived from the perceived importance of the bots, as discussed earlier in this section, and below in further detail.) With this in mind, the most appropriate bots to execute in any tick are simply those with the highest priorities. Resource starvation is averted in bots of low importance through an aging mechanism built into the calculation of priorities.

Each game tick, the x bots with highest priorities are selected to run, where x is a positive integer configurable and tuneable at run-time, based on a number of factors including the number of available processor cores, the workload being generated by other game subsystems, and so on. Each of the x selected bots is allocated one or more update cycles, depending on their relative importance. Each update cycle allows a bot to execute for a period of time. This execution can be pre-empted, and does not necessarily allow the bot to complete the task on which it was working. If the task is not completed when the bot's update cycle ends, the bot is paused and its priority is adjusted to reflect the work in progress.

Depending on the number of bots in the game, the priority system could be realized as either a single list sorted by priority, or a series of priority queues. While the mechanics of each approach is different, they can both deliver the same pre-emptive, priority-driven scheduling and dispatching of bots in a game.

2.6.1.2 Capability Scaling or Reduction

As discussed earlier, not every bot in a game needs the highest levels of richness, detail, and fidelity in their behaviour in order to be perceived by the player as believable. Indeed, some bots could function believably with greatly reduced capabilities, depending on the state of the player, the game, and the various bots within it.

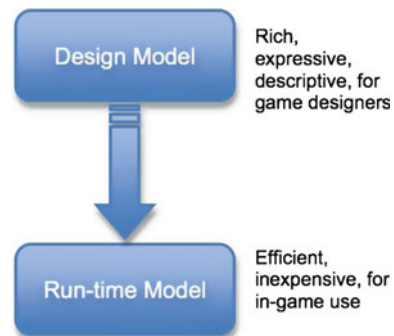
When the goal-oriented utility-driven planning system from Sect. 2.4.2.2 is used, scaling a bot can be accomplished in several ways. Generally, these methods can be grouped into either data reductions or processing reductions.

The purpose of data reductions is to limit the amount of information that is used in the various stages of the decision making process used by the bot. When done properly, this can greatly collapse decision spaces into smaller problems, making them far more efficient and less costly to work with. With care, these reductions can be carried out with little to no perceivable change in bot believability. Data reductions can themselves take many forms.

One such form is a model or state reduction. Recall from Sect. 2.4.1 that every bot has a psychosocial model associated with them, defining their personalities, emotions, relationships, roles, values, and so on. A fully populated model could have a bot's state defined by dozens of traits, all of which need to be maintained, and all of which could affect decisions made by the bot. While rich, expressive, and useful during design to fully define a bot, this can be very expensive computationally to use at run-time, as these factors must be consulted during appraisal, coping, goal selection, and planning/action selection. A careful reduction of this design model to a few key traits can produce a run-time model that is orders of magnitude more efficient, and far less costly to use within a game, as depicted in Fig. 2.16.

Another form of data reduction is event reduction. In this case, the number of events used to trigger decision making or provide information during decision making is limited, by reducing the types of events of interest, the importance of the various events or their consequences, or the frequency of their reporting to the bot. This results in either fewer events reaching the bots, fewer events moving past the appraisal stage (as the events are now deemed irrelevant, unimportant or inconsequential), or simpler

Fig. 2.16 Bot model reduction



decisions throughout the various stages of processing with fewer variables and factors to consider. All of these alternatives have the potential to improve performance substantially.

Processing reductions, on the other hand, generally involve altering a bot's decision making processes by changing algorithms or omitting aspects or complete stages of decision making to improve performance. Again, if done with care, these performance improvements can be achieved without sacrificing believability. Possible processing reductions include the following.

Use of Defaults: To accelerate the various stages of decision making, default strategies can be used instead of analyzing and developing them dynamically on demand. For coping, events could have default impacts on bot state, instead of determining this impact from the current state and other factors. For goal selection, bots could be assigned default goals to meet instead of developing them from scratch. For planning and action selection, default plans could be provided for each possible goal, complete with prescribed actions so that they do not need to be developed at run-time. While the defaults used will not do as well at reflecting the current state of bots or the game, the performance savings can be substantial even if this approach is used only for out of focus bots, or those that are unimportant.

Use of Randomization: Much like through the use of defaults, randomization can be used to replace various aspects of behaviour, although doing so likely makes sense only with unimportant bots in the game. It is likely also wise to constrain randomness to ensure that bots are still acting believably. This can be accomplished in a variety of ways, such as constraining randomness to choose from a pre-defined set of defaults, or by permitting some level of initial processing to develop sensible options that are chosen from randomly, instead of using a more expensive algorithm to make a more optimal choice.

Streamlining of Coping: Integrating the impact of events into a bot's internal state can be expensive, especially with a complex psychosocial model in use. To improve performance we can perform coping only in response to a limited set of critical events when bots are outside of the focus of the player. While this will result in bots whose moods and emotions change only in extreme circumstances, the player will be largely unaware of this.

Disabling of Goal Changing: Whenever a bot changes goals, any existing plan must be discarded and a new plan must be formulated, which can be quite expensive. To improve performance, bots can be prevented from modifying their goals while a plan is executing to avoid re-planning, unless very exceptional circumstances arise. While this will result in bots sticking with plans when they should likely be changed, this should not have too large an impact on their believability, provided that they are outside the focus of the player.

Increased Reliance on Reaction: When the stimulus-response system from Sect. 2.4.2.1 is used with the planning system, we can take advantage of this system to reduce load induced by the planning system in certain circumstances. For example, in a heated moment with a considerable amount of activity going on, such as a battle, the stimulus-response system can take over full control of a bot, while the

planning system is temporarily disabled. This will reduce processing load while still maintaining believability, as the stimulus-response system is sufficient to provide realistic bot reactions.

Automatic Achievement of Goals: As mentioned above, planning and action selection can be computationally expensive tasks. When a bot is outside the focus of the player, these tasks can be avoided entirely by simply allowing the character's goals to be achieved automatically. After all, if a goal is achievable by the bot, and the player is not present to witness the goal actually being achieved, planning and action selection and execution are not required for the player to believe what has happened. It is important to ensure, however, that the goal is likely to be achieved by the bot, and that the time required to meet the goal is properly taken into account; otherwise believability may be inadvertently sacrificed.

Disabling of All Processing: If a bot is relatively unimportant and is someone with whom the player has had no prior personal contact or knowledge thereof, it is possible to disable all, or nearly all, decision making in the bot. After all, the player would have little to no expectation of the bot's current mood, goals, or actions and so it is believable for the bot to be in their initial state when first encountered by the player. The player has no way of knowing that the bot was largely inactive up until when they first entered the focus of the player. Of course, this assumes a degree of randomness in initial bot settings, as a population of identical bots in this fashion would be unrealistic.

This strategy might also be applicable to important bots or bots that have been previously encountered, provided that they are out of the player's focus and will remain out of their focus until the next break in the game, such as a cut-scene, level transition, and so on. If important events are recorded, their effects on the bot, as well as the bot's goals, plans, and actions can all be simulated during the break, so that they are up-to-date when the player next encounters them. (This technique may also need to be applied to bots with no prior contact, as discussed above, as sufficiently important events should impact those bots and move them from their initial states accordingly. This, of course, depends on the nature of the events and the social networks that the bots are a part of, as these would also influence the flow of information to and from the bots in question.)

When we combine the various forms of data and processing reductions together, we have great flexibility in the amount of capability scaling or reduction available to a game. If taken too far, this can eventually impact the believability of the game, but if done with care in an intelligent fashion, we can achieve tremendous performance savings with little to no effect on the believability perceived by the player.

2.6.1.3 Character Importance and Priority Calculation

The process of scheduling and dispatching, as well as the process of capability scaling or reduction both use a measure of a bot's importance as a factor that ultimately affects both resource allocation and performance. Capability scaling or reduction

uses a measure of importance directly, adjusting the capabilities of a bot accordingly. Scheduling and dispatching use importance in the form of a priority in determining which bots are run in each game tick. Below, we examine how each measure is computed.

The importance of a bot is determined by a collection of factors, with one calculating the importance, i , of a character as:

$$i = (\alpha f + \beta d + \gamma r + \delta c)/4$$

where α , β , γ , and δ are weights between 0 and 1.0 to tune and balance the equation. The factor f is a measure of player focus on the bot, which takes into consideration the distance between the player and the bot, whether the bot is within range of the player's senses, and the strength of relationships between the player and the bot. The factor d is a designer-imposed measure of importance of the bot, usually with respect to the story of the game. The factor r is a measure of importance defined by the currently active roles of the bot in question, as some roles in the game are inherently more important than others. Lastly, the factor c is a measure of importance that comes from bot interactions. If a given bot is involved with other, more important bots, their own importance might require a boost to put the bots on more equal footing. (For example, if the two are fighting each other, an inherently more important bot could enjoy an unfair advantage over less important bots because the seemingly more important bot has more capabilities and better access to computational resources.) The factors f , d , r , and c all range between 0 and 1.0 and so after scaling, the importance of a bot also lies between 0 and 1.0.

With this in mind, the priority, pri , of a non player bot can be computed as follows:

$$pri = \varepsilon i + \zeta s - \eta rc + \theta p$$

where ε , ζ , η , and θ are also weights between 0 and 1.0 to tune and balance the equation. (Carefully setting of these weights can also result in various scheduling policies, such as fair, least-slack-time, and so on [56].) The factor i is the importance of the bot as defined above. The factor s is a starvation factor that increases at a certain rate for each game tick that the bot is not run; by doing this, even an unimportant bot's priority will eventually exceed the most important bot, allowing the unimportant bot to run and avoid starvation. The factor rc is a run counter used to ensure that a single bot is not over-allocated update cycles despite its importance. Lastly, the factor p is a progress measure that approaches 1.0 as the bot approaches completion of its task at hand, to allow scheduling to clear out near-complete tasks from bots. All of factors i , s , rc , and p are normalized to between 0 and 1.0.

If desired, we can add a fifth factor to priority calculations to reflect the amount of capability reduction being applied to a particular bot. Doing so may be reasonable since a bot with its capabilities reduced by data or processing reductions will require fewer computational resources and therefore can cope with its schedule being reduced as well. Ordinarily, this would be accomplished using the importance factor i , as a low importance would trigger both capability and schedule reduction simultaneously.

If importance and capability reduction were not so closely linked, a separate factor indicating reduction would then be necessary. (This can occur, for example, when there is a very large number of non player bots needing to be managed; in such a case, even the capabilities of fairly important bots would need reduction despite their importance in order to maintain game performance at an acceptable level.)

2.6.2 Implementation and Results with Performance Optimization

As a proof of concept, a scheduler and dispatcher module was added to the third generation prototype discussed in Sect. 2.5.3 to allocate computational resources to bots. This was based on a simple serial sort and search algorithm to determine the next bots to run based on priorities as described in the previous section. Capability scaling or reduction was implemented with multiple levels of reduction. A bot running with full capabilities uses the complete goal-based utility-driven planning system described in Sect. 2.4.2.2. The first level of reduction uses rudimentary partial planning in which planning is carried out over several update cycles, with actions selected from partial plans in earlier update cycles while the current cycle continues to refine the plan. The second level of reduction uses full appraisal, coping, and goal selection capabilities, but then uses default plans and actions associated with goals selected, instead of carrying out a full planning/action selection stage. Finally, the third level of reduction uses randomization to select a goal and select a plan and actions capable of achieving this goal. While this is not the most realistic of approaches, it can still be appropriate for non critical characters in the game.

To assess the operation and performance of this approach, detailed logs were collected during a series of experiments. All experimentation was executed on an Intel Core 2 Duo system with a clock speed of 2.0Ghz and 4.0GB of RAM. The 64-bit variant of Windows Vista was the operating system. This configuration provided more than enough power for the experimentation we conducted.

The prototype system was configured to use the psychosocial model described in Sect. 2.4.1, with bots having access to 4 roles, 5 goals, and 8 actions during processing. While a typical game would have more possibilities open to its characters, this configuration on its own was sufficient to demonstrate the effectiveness of the system. Time in the system was simulated so that 4 bots could run each game tick, there were 30 milliseconds between game ticks, and each action consumed one tick for execution. While actions would ordinarily take longer and have varied lengths in reality, this accelerated experiments and simplified analyses, as it was easier to confirm that factors such as importance were being properly handled by the system. Lastly, for simplicity and balancing, all weights used in calculating importance and priority were set to 1.0, except during starvation experiments. It is possible that better (or worse) results could be obtained through the fine-tuning of these weights.

2.6.2.1 Initial Experiments

Prior to more rigorous experimentation, initial testing was conducted to assess the basic operation of our prototype system. From this, we were able to verify:

- Equal fixed importance and priority resulted in an even distribution of resources to bots and equal opportunity for execution.
- Increased importance and priority translated into an increase in resource allocations to bots and a corresponding increase in execution time.
- Starvation of bots with low importance was effectively prevented by our approach to scheduling, and would be a serious issue if these measures were disabled or not provided in the first place.
- Bots with reduced capabilities required fewer resources to execute than bots with full capabilities intact.
- The prototype system could handle several bots of varying importance well, adjusting scheduling and capabilities accordingly without difficulty.

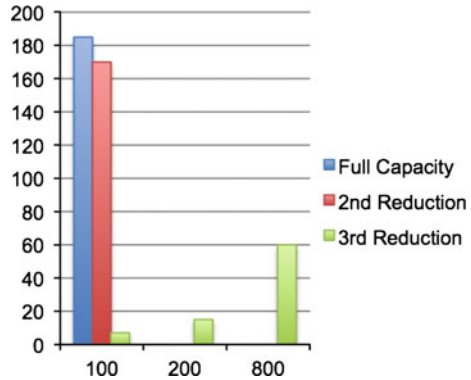
While these tests verified the correct operation of the prototype system, we still needed to assess the improved performance and scalability enabled by our approach. This is accomplished through experimentation outlined in the next section.

2.6.2.2 Stress Testing

To assess performance and scaling improvements, we used the prototype system to manage hundreds of characters simultaneously. In these experiments, we executed three scenarios with 100, 200, and 800 bots respectively. The bots used psychosocial models as described in Sect. 2.6.2, with traits initialized randomly. Non-violent actions and interactions were used to ensure that the bot population was not inadvertently thinned during experimentation. No human player was involved in these experiments, as this could introduce unwanted variations and anomalies in the results. Each scenario was itself run three times, once with all bots at full capability, once with all bots at the second level of reduction (as described in the previous section), and once with all bots at the third level of reduction. (In these experiments, locality was not used as a factor, to ensure that all bots stayed at a fixed level of reduction.)

Results from this experimentation are shown in Fig. 2.17, measuring the time to completion of 200 game ticks in seconds. With 100 bots executing, performance under full capacity suffered greatly. There was a slight improvement under the second reduction, but performance was still unacceptable. With the third reduction, performance improvements were substantial. As the number of bots increased, only the third reduction bots were able to complete. While their time to completion increased in a roughly linear fashion, performance was still improved dramatically through this reduction. (The linear scaling of third reduction bots is due to their simplicity.

Fig. 2.17 Bot stress testing



Bots with more capabilities, especially those that are more socially active and conscientious enough to factor others into their decision making processes, would not enjoy this linear scaling.)

It is important to note that while full capacity and slightly reduced bots had performance issues, the system would never be expected to support this many at a time. Through dynamic adjustments to capabilities, only a few would run at these capability levels at a time, depending on the game, with the others reduced further. This experiment was to solely demonstrate performance improvements through our approach.

From these results, we can see great improvements in the performance delivered by our approach to scalable believable non player bots. We are able to deliver a collection of bots that can adapt to various computational requirements through proper scheduling and capability adjustment.

2.7 Concluding Remarks

This chapter has examined the creation of believable bots for video games through the use of psychosocial behaviour. By using bots with personality, emotion, social awareness and other psychosocial traits, we have shown how to create bots capable of creating engaging and immersive gameplay. Prototype implementations have proven our concepts and demonstrated some of the possibilities achievable through their use. Finally, we have discussed and tested various techniques for performance optimization to ensure that believable bots will meet the real-time requirements and resource constraints of modern video games.

References

1. Acton, G.: Great ideas in personality—theory and research (2006). <http://www.personalityresearch.org>. Accessed Jan 2011

2. Acton, G.: Playing the role: towards an action selection architecture for believable behaviour in non player characters and interactive agents. Masters thesis, Department of Computer Science, The University of Western Ontario (2009)
3. Alt, G., King, K.: A dynamic reputation system based on event knowledge. In: *AI Game Programming Wisdom*. Charles River Media, Hingham (2002)
4. Bailey, C., Katchabaw, M.: An emergent framework for realistic psychosocial behaviour in non player characters. In: *Proceedings of FuturePlay 2008*, Toronto, Canada (2008)
5. Baille, P.: The synthesis of emotions in artificial intelligences. Ph.D. Dissertation, University of Southern Queensland (2002)
6. Bates, J., Loyall, A., Reilly, W.: *An Architecture for Action, Emotion, and Social Behaviour*. Lecture Notes in Computer Science, vol. 830. Springer, Heidelberg (1994)
7. Beaumont, L.: Life's guidance system: traveling your path through life (2008). <http://www.emotionalcompetency.com/guidance.htm>. Accessed Jan 2011
8. Berger, L.: Scripting: overview and code generation. In: *AI Game Programming Wisdom*. Charles River Media, Hingham (2002)
9. Biddle, B., Thomas, E.: *Role Theory: Concepts and Research*. Wiley, New York (1966)
10. Brockington, M.: Building a reputation system: hatred, forgiveness, and surrender in never-winter nights. In: *Massively Multiplayer Game Development*. Charles River Media, Hingham (2003)
11. Byl, P.B.D.: *Programming Believable Characters in Games*. Charles River Media, Hingham (2004)
12. Cattell, R.: *The Description and Measurement of Personality*. Harcourt, Brace, and World, New York (1946)
13. Cesta, A., Miceli, M., Rizzo, P.: Robustness of the social attitude and system performance. In: *Proceedings of the First International Conference on Multi-Agent Systems, Help Under Risky Conditions* (1996)
14. Cole, S.: Modeling opinion flow in humans using Boids algorithm and social network analysis (2006). http://gamasutra.com/features/20060928/cole_01.shtml. Accessed Jan 2011
15. Crawford, C.: *Chris Crawford on Interactive Storytelling*. New Riders, Indianapolis (2005)
16. de Freitas, J.S., Imbert, R., Queiroz, J.: Modeling Emotion-Influenced Social Behavior for Intelligent Virtual Agents. *Lecture Notes in Computer Science*, vol. 4827. Springer, Heidelberg (2007)
17. Dias, J., Paiva, A.: Feeling and Reasoning: A Computational Model for Emotional Characters. *Lecture Notes in Computer Science*, vol. 3808. Springer, Heidelberg (2005)
18. Ekman, P., Friesen, W., Ellsworth, P.: *Emotion in the Human Face: Guidelines for Research and an Integration of Findings*. Pergamon Press, New York (1972)
19. Elliott, C.: The affective reasoner: a process model of emotions in a multi-agent system. Ph.D. Dissertation, Northwestern University Institute for the Learning Sciences (1992)
20. El-Nasr, M.: Modeling emotion dynamics in intelligent agents. Masters thesis, American University in Cairo (1998)
21. Epic Games. *UDK—Unreal Development Kit* (2010). <http://www.udk.com>. Accessed Jan 2011
22. Eysenck, H.: Biological Dimensions of Personality. In: Pervin, L.A. (ed.) *Handbook of Personality: Theory and Research*. Guilford, New York (1990)
23. Funder, D.C.: *The Personality Puzzle*. W. W. Norton & Company, New York (2001)
24. Funge, J.D.: *Artificial Intelligence for Computer Games*. A K Peters, Natick (2004)
25. *Game Informer*. *Assassin's Creed*. June 2006, Issue 158. Vol XVI: No 6 (2006)
26. *Game Informer*. *Bully*. September 2006, Issue 161, Vol XVI: No 9 (2006)
27. Gratch, J., Marsella, S.: A domain-independent framework for modeling emotion. *Cogn. Syst. Res.* **5**(4), 269–306 (2004)
28. Gratch, J., Marsella, S.: Evaluating a computational model of emotion. *Auton. Agent. Multi-Agent Syst.* **11**(1), 23–43 (2005)
29. Grond, G., Hanson, B.: Reputation system FAQ (1998). <http://update.uo.com/repfaq>. Accessed Jan 2011

30. Gruenewoldt, L., Katchabaw, M., Danton, S.: Achieving Realistic Reactions in Modern Video Games. In *Worlds In Play*. Peter Lang Press, New York (2007)
31. Guye-Vuilleme, A., Thalmann, D.: A high-level architecture for believable social agents. *VR J.* **5**, 95–106 (2001)
32. Hogg, L.M.J., Jennings, N.R.: Socially intelligent reasoning for autonomous agents. *IEEE Trans. Syst. Man Cybern. Part A Syst. Hum.* **31**(5), 381–393 (2001)
33. Imbert, R., de Antonio, A.: An Emotional Architecture for Virtual Characters. *Lecture Notes in Computer Science*, vol. 3805. Springer, Heidelberg (2005)
34. Ion Storm. *Thief: Deadly Shadows*. Eidos Interactive (2004)
35. Isbister, K.: *Better Game Characters by Design*. Morgan Kaufmann, San Francisco (2006)
36. Keirse, D.: *Please Understand Me II: Temperament, Character, Intelligence*. Prometheus Nemesis, Del Mar, CA (1998)
37. Lawson, G.: Stop relying on cognitive science in game design—use social science (2003). http://www.gamasutra.com/php-bin/letter_display.php?letter_id=647. Accessed Jan 2011
38. LeBlanc, M.: Formal design tools: emergent complexity, emergent narrative. In: *Proceedings of the 2000 Game Developers Conference* (2000)
39. Livingstone, D.: Turing's test and believable AI in games. *Comput. Entertain. (CIE)* **4**(1), 6 (2006)
40. Loyall, A.: *Believable agents: building interactive personalities*. Ph.D. Dissertation, Stanford University (1997)
41. Mateas, M.: An Oz-centric Review of Interactive Drama and Believable Agents. *Lecture Notes in Computer Science*, vol. 1600. Springer, Heidelberg (1999)
42. McCrae, R., Costa, P.: Reinterpreting the Myers-Briggs type indicator from the perspective of the five-factor model of personality. *J. Pers.* **57**(1), 17–40 (1989)
43. Mehrabian, A.: Framework for a comprehensive description and measurement of emotional states. *Genet. Soc. Gen. Psychol. Monogr.* **121**(3), 339–361 (1995)
44. Myers, I.B., McCaulley, M., Quenk, N., Hammer, A.: *MBTI Manual: A Guide to the Development and Use of the Myers Briggs Type Indicator*, 3rd edn. Consulting Psychologists Press, Palo Alto (1998)
45. Mythic Entertainment. *Reputation, karma and fame* (2005). <http://www.uoherald.com/node/167>. Accessed Jan 2011
46. Ortony, A.: On Making Believable Emotional Agents Believable. Appeared in *Emotions in Humans and Artifacts*. MIT Press, Cambridge (2003)
47. Ortony, A., Clore, G., Collins, A.: *The Cognitive Structure of Emotions*. Cambridge University Press, Cambridge (1988)
48. Paanakker, F.: The Emotion Component: Giving Characters Emotions. Appeared in *AI Game Programming Wisdom*, vol. 4 (2008)
49. Panzarasa, P., Jennings, N., Norman, T.: Social mental shaping: modeling the impact of sociality on the mental states of autonomous agents. *Comput. Intell.* **17**(4), 738–782 (2001)
50. Parrott, W.: *Emotions in Social Psychology: Essential Readings*. Psychology Press, Philadelphia (2001)
51. Pfeifer, B.: Creating emergent gameplay with autonomous agents. In: *Proceedings of the Game AI Workshop at AAAI-04* (2004)
52. Picard, R.W.: *Affective Computing*. Media Laboratory, Perceptual Computing TR 321, MIT Media Lab (1995)
53. Plutchik, R.: The nature of emotions. *Nature* **89**(4), 344 (2001)
54. Prendinger, H., Ishizuka, M.: Social role awareness in animated agents. In: *International Conference on Autonomous Agents* (2001)
55. Prendinger, H., Ishizuka, M.: Evolving social relationships with animate characters. In: *Proceedings of the AISB-02 Symposium on Animating Expressive Characters for Social Interactions* (2002)
56. Rankin, A., Acton, G., Katchabaw, M.: A scalable approach to believable non player characters in modern video games. In: *Proceedings of GameOn 2010*, Leicester, United Kingdom (2010)

57. Reiss, S.: Multifaceted nature of intrinsic motivation: the theory of 16 basic desires. *Rev. Gen. Psychol.* **8**(3), 179–193 (2004)
58. Reynolds, B.: How AI enables designers (2004). http://gamasutra.com/php-bin/news_index.php?story=11577. Accessed Jan 2011
59. Rizzo, P., Veloso, M., Miceli, M., Cesta, A.: Personality-driven social behavior in believable agents. In: *Proceedings of the AAAI Fall Symposium on Socially Intelligent Agents* (1997)
60. Romano, D., Sheppard, G., Hall, J., Miller, A., Ma, A.: BASIC: a believable adaptable socially intelligent character for social presence. In: *Proceedings of the 8th Annual International Workshop on Presence*, London, United Kingdom (2005)
61. Rousseau, D., Hayes-Roth, B.: Personality in synthetic agents. Technical report, Knowledge Systems Laboratory, Department of Computer Science, Stanford University, Stanford, CA (1996)
62. Russell, S., Norvig, P.: *Artificial Intelligence: A Modern Approach*. Prentice Hall, Englewood Cliffs (2003)
63. Sinclair, P.: The “Big Five” factors personality model (2011). <http://www.odportal.com/personality/big-five.htm>. Accessed Jan 2011
64. Sloman, A.: Beyond shallow models of emotion. *Cognitive processing*. *Lengerich* **2**(1), 177–198 (2001) (Pabst Science Publishers)
65. Smith, C., Ellsworth, P.: Attitudes and social cognition. In: *Journal of Personality and Social Psychology*, American Psychologists Association, Washington, vol. 48(4) (1985)
66. Smith, H., Smith, R.: Practical techniques for implementing emergent gameplay: would the real emergent gameplay please stand up? In: *Proceedings of the 2004 Game Developers Conference* (2004)
67. Straker, D. Big five factors (2010). http://changingminds.org/explanations/preferences/big_five.htm. Accessed Jan 2011
68. Sweetser, P.: An emergent approach to game design, development and play. School of Information Technology and Electrical Engineering, The University of Queensland (2006)
69. Sweetser, P.: *Emergence in Games*. Charles River Media, Hingham (2008)
70. Sweetser, P., Johnson, D., Sweetser, J., Wiles, J.: creating engaging artificial characters for games. In: *Proceedings of the Second International Conference on Entertainment Computing*. Carnegie Mellon University, Pittsburgh, Pennsylvania (2003)
71. Tomkins, S.: Affect theory. In: Scherer, K.R., Ekman, P. (eds.) *Approaches to Emotion*. Erlbaum, Hillsdale (1984)
72. Tomlinson, B., Blumberg, B.: Using emotional memories to form synthetic social relationships (2002). <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.19.4475>. Accessed Jan 2011
73. Tozour, P.: The Evolution of Game AI. In *AI Game Programming Wisdom*. Charles River Media, Hingham (2002)
74. Tupes, E., Cristal, R.: Recurrent personality factors based on trait ratings. Technical report ASD-TR-61-97, Lackland Air Force Base, TX: Personnel Laboratory, Air Force Systems Command (1961)
75. Velasquez, J.: From affect programs to higher cognitive emotions: an emotion-based control approach. In: *Proceedings of the Workshop on Emotion-Based Agent Architectures*, Seattle, Washington (1999)
76. Woodcock, S.: Game AI: the state of the industry (2000). http://www.gamasutra.com/view/feature/3570/game_ai_the_state_of_the_industry.php. Accessed Jan 2011
77. Wooldridge, M.J.: The logical modeling of computational multiagent systems. Ph.D. thesis, UMIST, Manchester (1992)
78. Wooton, B.: Designing for Emergence. *AI Game Programming Wisdom*, vol. 3. Charles River Media, Hingham (2006)
79. Wright, I., Marshall, J.: Egocentric AI processing for computer entertainment: a real-time process manager for games. In: *Proceedings of the First International Conference on Intelligent Games and Simulation (GAME-ON 2000)*, London, United Kingdom (2000)

80. You J., Katchabaw, M.: A flexible multi-model approach to psychosocial integration in non player characters in modern video games. In: Proceedings of FuturePlay 2010, Vancouver, Canada (2010)
81. Zhou, C., Yu, X., Sun, J., Yan, X.: Affective Computation based NPC behaviours modeling. In: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology. IEEE Computer Society Washington, DC, USA (2006)

Believable Bots

Can Computers Play Like People?

Hingston, P. (Ed.)

2012, X, 318 p., Hardcover

ISBN: 978-3-642-32322-5