Dragons, Bats & Evil Knights: A Three-Layer Design Approach to Character Based Creative Play

Joanna Bryson and Kristinn R. Thórisson

Artificial Intelligence Laboratory, MIT
545 Technology Square, Cambridge MA 02139, USA
joanna@ai.mit.edu

LEGO Digital
Billund, Denmark
kris@media.mit.edu

Abstract. Creative play requires a fertile but well-defined design space. This paper describes a design process for creating three-dimensional virtual reality play spaces that allow the development and exploration of social interactions and relationships. The process was developed as part of a commercial research effort to create an interactive virtual reality entertainment system that allows children to engage in creative and constructive play within an established action/adventure framework. The effort centres on designing AI characters for a *constructive narrative*. We claim that a behaviour-based architecture is an ideal starting point for developing agents for such a process, but that its full realization requires additional architectural structures and methodological support for the design process. In this paper, we describe a character architecture called Spark of Life (SoL). We also propose a three-layer design process for producing fertile and æsthetic constructive narratives. Finally, we describe our experience in implementing these ideals in an industrial setting.

Keywords: Design Team Methodology; Constructive Narrative; Personality and Action Selection; Character Architectures; Behaviour-Based AI.

1 Introduction

"Like good improvisational theatre, cyberspace presents the opportunity for the audience to create its own characters and worlds, to write its own plots and stories, and to essentially become the directors, producers, and actors within their own imaginary worlds." (Pearce, 1997, pp. 345) For this potential to be realized, there must be tools for the creative design of personalities. One modern view of creativity is that of a Darwinian process, involving novel recombination of existing design elements (Simonton, 1997; Boden, 1987). Part of what makes creativity challenging in artificial intelligence is that a single element of creative play often takes on multiple roles in such recombination. For example, a banana may be used as a telephone, bridging two or more representational threads simultaneously. This is a problem not only for those planning AI representations, but also for those employed in designing props for encouraging creative play. The challenge of designing for creative play lies in providing a rich and interesting design space without limiting the creative potential of participants' experience in that space.

One solution is to allow the players to become constructors of their own experience. This idea has been explored to a much greater extent spatially than socially. Examples of primarily spatial games include Internet MUDs, games such as SimCity (Joffe and Wright, 1989), and constructive toys, such as blocks or LEGO. Socially constructive toys also exist, though in fewer variants. For example, role-playing games such as Purple Moon's 'girl games' (Cassell and Jenkins, 1998), give the player the objective to find a place for the main character in an established society. The game Creatures (Grand et al., 1997) allows a player to evolve a society. The NICE project (Roussos et al., 1997) allows children to construct virtual gardens in a multi-user virtual world where social interaction with other "gardeners" is part of the world. However, none of these allow the player to freely create characters and narratives of complex personal interactions. This is partly because the right technology has not found its way into virtual worlds, and partly because personality creation has not been explored in a sufficiently thorough manner to allow developers to make the necessary off-the-shelf building blocks.

In this paper we describe a character-based approach to constructive narratives, and a design process for constructing a play environment to support character-based creative play. Designing such a system can in itself be a highly creative act, but it is particularly challenging to do so in a way that allows users ample opportunity to express their own creativity. The design process is separated into three levels:

1	A high, artistic design level for creating story and characters.
2	A middle, behaviour-based design level for creating personality in character agents.
3	A low, VR design level for basic capabilities and appearances.

In this paper, we primarily focus on the middle layer, which is the domain of the AI technologist. This is the layer to which we, as AI developers, can make the most contribution. In our experience, the AI engineer must not only master his or her own technical problems, but also serve as an important facilitator for the overall project. The AI component connects the visions of the character design team to the reality of the virtual (or possibly mechanical) world. The AI engineers are consequently in the difficult position of needing to understand and communicate the constraints of each end of the design to the other, as well as understanding and communicating their own technical requirements and capabilities.

We begin by describing Spark of Life (SoL), the architecture for virtual reality characters we developed over the course of a VR entertainment project at LEGO Digital. SoL combines two already established AI architectures, Ymir (Thórisson, 1996, 1999) and Edmund (Bryson and McGonigle, 1998; Bryson, 1999). We then describe the design process outlined above in more detail. Finally, we illustrate the methodology with an example drawn from the LEGO project, including lessons learned.

2 The Spark of Life (SoL) Character Architecture

2.1 Character Architecture and Constructive Narrative

Much research into agents for entertainment concentrates on the problem of combining the concept of a script with the notion of autonomous, reactive characters (Hayes-Roth and van Gent, 1997; Lester and Stone, 1997; André et al., 1998). Our constructive narrative approach eliminates this problem by changing the top level creative design from a *script* to a *cast of characters*. This simplifies the task of the player by removing the need for character addition, substitution, alteration, or removal. It has the penalty of removing a substantial element of narrative structure: a sequential order of events. However, this problem has already been addressed by the creators of role-playing and adventure games. Their solution is that plot, if desired, can be advanced by knowledgeable characters, found objects, and revealed locations. Structure is produced through the use of geographic space as well as character personalities. Personality traits such as loyalty, contentment or agoraphobia can be used to maintain order despite a large cast of autonomous character, by tying particular characters to particular locations. Developing such characters requires an agent architecture powerful enough to support this complexity. It also requires sufficient modularity to allow reasonably quick construction of behaviour patterns. We claim that SoL has these qualities.

Most virtual reality agent architectures are fundamentally behaviour-based, and at least partially reactive (see Sengers, 1998, for a recent review and critique). This is because the reactive, behaviour-based AI revolution of the late 1980s (Kortenkamp et al., 1998) was primarily the triumph of a design approach. Behaviour-based AI is simpler to design than a monolithic intelligence system because it allows the decomposition of intelligent behaviour into easy-to-program modules, with more localised control structures. Specifying that the intelligence should also be reactive removes the complex problems of learning and constructive planning from the agent. In spite of limiting the potential complexity of the agent's capabilities, the behaviour-based approach has been more successful in achieving interesting, believable characters than any fully human-specified or fully machine-learned approach simply because it empowers the human designer.

The limitations of completely reactive systems have been widely recognised, and are addressed in numerous architectures (see for example Hexmoor et al., 1997). Some authors have proposed that the community has moved beyond both constructive and reactive planning to a new dominant paradigm, situated planning (Levison, 1996; Kortenkamp et al., 1998). Situated planning architectures generally include reactive behaviours, pre-stored plans, elements of learning, and possibly constrained forms of on-line planning. Two of the most popular architectures of this paradigm are PRS (Georgeff and Lansky, 1987) and 3T (Bonasso et al., 1997). Both of these have at their centre a scripting language for allowing the specification of sequential and hierarchical behaviour structures. These structures provide additional information (in the form of internal state) for action selection in situations that might be perceptually identical. This allows the situated planner greater behavioral flexibility than the fully reactive planner, which is dependent on current sensing to select its next action. The script structures also allow for the combination of simple behaviour elements into larger modular forms, again simplifying the design task.

There are two sets of problems associated with using the established AI "complex agent" architectures. One is getting the correct level of control for scripted personalities or behaviours. PRS and its derivative architectures do not seem truly reactive enough to support the abrupt and frequent changes in context possible in a play scenario. The "reactive" elements of PRS is constrained to switching in new, complete plans in exceptional circumstances—for example if a fire alarm has sounded (Bryson and Stein, 2001). When working with children, a more consistent sort of responsiveness is required in order to respond to unexpected assistance or interruption by the child. These events are more likely require movement within the same script than restarting or changing scripts. On the other hand, more purely reactive architectures such as production systems or Soar (Newell, 1990) make scripting coherent behaviour very difficult.

The other set of problems is associated with the technical difficulties of controlling a real-time multi-modal VR system. Very few AI architectures support the millisecond precision and modality coordination necessary for believable, engaging real-time interactions. These concerns are critical for all VR, but are particularly apparent when dealing with dialogue and gesture (Thórisson, 1998).

2.2 Spark of Life

Our solution for these problems was to develop a new architecture, Spark of Life (SoL). SoL merges features from two prior architectures developed by the authors, Ymir (Thórisson, 1996, 1999) and Edmund (Bryson and McGonigle, 1998; Bryson, 2000), which each address one of the sets of constraints outlined in the previous section.

Ymir is a highly modular, hybrid architecture which combines features from classical and behaviour-based AI, and provides a system that can simulate in great detail the psychosocial dialogue skills of humans. Real-time, face-to-face dialogue encompasses a broad range of perceptual, cognitive and action requirements. Ymir addresses these phenomena, including natural language and multi-modal input and output (facial expression, gesture, speech, body language), load-balanced handling of time (from short reactive behaviours like fixation control to the execution of several seconds of multimodal actions), and employs a modular approach that enables the creation of complex, human-like behaviour.

Edmund is also a hybrid behaviour-based architecture, which emphasises the integration of semi-autonomous behaviours which govern sensing, acting and learning. This integration is done by a specialised module for situated or reactive planning. Edmund at its associated methodology, Behaviour Oriented Design (BOD), allow designers to iteratively develop both the perception/action modules that control *how* behaviour is expressed and the plan scripts that determine *when* it is expressed. Edmund allows designers to specify complex, even contradictory motivational structure for a creature, the way those motivations are met, and the circumstances under which they are expressed.

The main strength of the behaviour-based paradigm is is modularity. Because Edmund and Ymir were both behaviour-based and have complementary strengths, we were able to combine them into single, powerful architecture, SoL. SoL encompasses the following capabilities: multi-modal perception and action, real-time speech input and output, memory, and planning. SoL's modularity combined with robust, simple control makes it ideal for constructive play by allowing for easy additions and modifications.

The rest of this section details the attributes of SoL, as they derive from Edmund and Ymir. We particularly emphasise aspects that relate to the design methodology described in the remainder of the paper. We also describe an iterative methodology for designing the modules and plans that make up a particular character.

2.3 Edmund's Contributions to SoL

Behaviour-based AI simplifies design and increases responsiveness in complex intelligent agents by decomposing intelligence into specialised modules or *behaviours*. Behaviours autonomously control a particular sort of action, such as walking or laughing, providing not only motor competence, but whatever sensing and perception is necessary for their appropriate expression. Although this decomposition simplifies the design of individual actions, it increases the complexity of coordinating behaviour in a coherent manner. This problem is known as behaviour arbitration, a special case of the problem of action selection.

Edmund utilises Parallel-rooted, Ordered, Slip-stack Hierarchical (POSH) action selection. POSH action selection allows for persistent, rational-appearing behaviour to be generated through the appropriate attention to a task, while at the same time allowing for the specification of triggers, drives and higher-level goals which can distract or interrupt an agent, allowing it to remain broadly reactive. POSH action selection is specified through scripts representing reactive plans. The most primitive elements of these scripts are action and sensing interfaces to the agent's behaviours. Edmund's scripting language provides for three levels of control structure above these primitives. First is the *action pattern*, a simple sequence which runs uninterrupted except in the case of radical failure or severe high-level attention disruption (described below). The second is a *competence*. A competence consists of a prioritised set of elements, whose behaviour tends to converge towards the highest priority element, the goal. When a competence is active, it selects the highest priority element which is currently capable of being executed. If that element is the goal the competence finishes successfully, if no elements fire the competence fails. Their elements may consist of either behaviour primitives, action patterns or other competences.

The highest level control of an Edmund script is a special form of competence called a *drive collection*. The elements of a drive collection provide the activation or motivation for some coherent section of the script. A drive collection's elements, while prioritised like a competence, may also be scheduled. Thus if a high priority element has fired recently, it may be inhibited in order to allow lower priority elements to execute. Drives also maintain overall behaviour coherence by being persistent. As described above, competences and action patterns both terminate routinely. Further, if a competence selects an element which is itself a competence, the parent competence is replaced by its child in the action scheduling. This feature is called a *slip-stack hierarchy*, and allows for indefinite behaviour chaining or looping. A drive always remembers the root of its behaviour hierarchy. If a long chain does terminate, the drive returns its attention to the root starting point.

For an example of a competence, think of a character that needs to drink out of a particular sort of chalice. A high-level competence for this problem might look roughly like this:

1	(holding chalice) (touching held-object mouth)	*goal*
2	(holding chalice)	drink-from-cup
3	(see chalice)	grasp-attended-object
4	(desire chalice)	find-attended-object

Conditions are on the left, goals on the right. For each concept, the highest priority element for which its preconditions indicate it can fire, will fire. For most competences, the highest priority element (1) recognises when the goal condition has been met: in this case when the chalice is at the lips of the character. If the goal has not been met but the character is holding a chalice, this competence will simply direct the character to perform a primitive drinking motion by selecting 2. If the character is not yet holding the chalice, but is within line-of-sight of it, it will attempt to grasp the object, as directed by 3. In this case, "grasp-attended-object" would almost certainly be another competence, since the agent may have to move itself to within reach of the object, which may in itself be a complicated maneuver. Control attention for the drive would temporarily switch to the competence specified by 3, but when that competence completes, the parent drive of the drinking competence will return attention to this plan.

The competence is a basic sort of reactive plan. Reactive plans are structures which allow an agent to determine what to do next simply by taking into account the internal context which is tracked by the plan and the external context which is provided by perception. Many different action sequences can be generated by single reactive plan responding to different contexts. A competence can be opportunistic — if the agent is handed a chalice while it had been seeking it, it will immediately drink from it. The agent will skip the grasping step, because it's plan will recognise the new context. Similarly, if the grasp fails for some reason, or another character takes the chalice, then steps 3 or even 4 can be repeated as needed. The competence will fail if none of the actions can trigger. This allows for a measure of self-control — the desire for the chalice not only marks an internal deictic reference so that the general primitive "find-attended-object" can operate, it might also monitor the character's frustration level. If the character cannot find or grasp the chalice over time, the desire predicate can fail, allowing an alternate competence to operate.

For an example of a slip-stack hierarchy, consider that a bat may have two typical behaviours which can never be expressed at the same time: *orbiting* which is flying around the outside of a tower, and *attacking* which means swooping towards a person inside a room. Because of the slip-stack mechanism, it is possible for the *orbiting* competence to contain an element that specifies that *if* the bat is called by another character *then* it switches to the *attack* mode, while at the same time the *attack* competence may have a precondition that results in the bat returning to *orbit*.

Because no stack is maintained, this chain may be looped indefinitely or abandoned when a competence fails, but there will be no break or delay in the bat's behaviour. The return to the root facilitates coherent as well as reactive behaviour. Each element of a drive collection is persistent. If a competence terminates, the drive element that was attending to it switches attention to its original, ancestral root. If the environmental context has not changed much, then attention will again pass to the parent of that competence. The parent is then free to examine the environment and determine whether to execute the competence again, or whether one of its own higher priority elements can now fire, and its own plan progress. In the chalice example, this would be the case if the competence for grasping-attended-object completed. The drinking competence, once attention had returned to it, would then be able to sense whether it was holding the chalice, in which case it would proceed to drink, or whether it was not, in which case it would attempt the grasp again.

Experiments comparing Edmund's POSH approach to action selection with other reactive and behaviour-based architectures show Edmund's advantages over purely reactive approaches (Bryson, 1999). Experiments have also demonstrated its ability to control real-time, multi-modal sensing and action on a mobile robot (Bryson and McGonigle, 1998). However, Edmund had not previously been applied to a problem of the complexity of a virtual reality character capable of full-scale, multi-modal natural language dialog. Such complexity and specialisation requires additional architectural support beyond Edmund's design.

2.4 Ymir's Contribution to SoL

Getting a synthetic character to engage in real-time dialogue with a human requires coordination of complex sensory processing and multi-modal action generation on a broad range of time scales, from gaze control (100-200 msec) to telling stories (minutes, hours). Ymir is a general-purpose architecture for orchestrating perceptual, decision, cognition and action processes for just such a purpose. Thórisson (1996) presents an implementation of Ymir capable of knowledge-based dialogue skills, world knowledge, and mechanisms for managing the sensory systems and motor control of multiple modes in situated dialogue. In SoL, Ymir provides the architectural framework, and supplies many general routines for dialogue processing, including a symbolic, frame-based mechanism for natural language understanding.

The segmentation of perception and action in Ymir is finer than in most behaviour-based systems (including Edmund), providing separate module types for decision, perception, knowledge and action. Context-dependent perception-action loops as short as 100 msec are supported, and Ymir's mechanisms provide a framework for short-term planning from 100 msec up to roughly 2 seconds, with special focus on motor control in communicative action. Edmund's more general-purpose, and longer-term planning structure is therefore a strong complement to these functionalities.

Ymir's action generation is compositional: Movement morphology is composed of a collection of movement elements, each consisting again of a collection of smaller elements, etc., effectively instantiating a tree where the leafs represent the movement of single motors or muscles for a given duration of time. These segments, and their composition, is chosen at run-time based on a number of factors (e.g. the current state of the agent's body) according to compositional rules. Given a rich Motor Lexicon (see below), the result of this system is that the exact form of an agent's behaviour never happens in quite the same way twice. A fine-grain control of an agent's behaviour in the time domain, combined with the compositional nature of its actions, make it highly responsive to external events. To take a hypothetical (but realistic) example, a user points at a location in space (t=0 msec). The the agent recognises the gesture first as a communicative gesture (t=200 msec), and then as a deictic gesture (t=300 msec), and starts planning a gaze and/or head movement in the direction pointed (t=350 msec). If the user retracts the gesture and looks back at the agent (t=500 msec) just as the agent starts to execute the gaze action, and the agent recognises this (t=600 msec), the plan may be cancelled within 100 msec of the recognition of the gesture retraction (t=800 msec). Alternatively, if the human doesn't retract the hand in this example, the agent starts to look in the direction pointed less than half a second after the deictic gesture started. Mimicking human perceptual and motor dialogue behaviour at this fine level of granularity makes for very life-like, responsive agents.

Ymir has been tested in an interactive, conversational system (Thórisson, 1996; Cassell and Thórisson, 1999). Results show the behaviour patterns generated by Ymir in real-time interaction with people to be comparable to task-oriented dialogue between humans, and users rate its performance to be very believable (Thórisson, 1998, 1996). Detailed explanation of the motor planning and control in Ymir has already been published (Thórisson, 1997). Thórisson (2001) explains some of the perceptual mechanisms in the first prototype of Ymir, such as real-time intonation analysis and vision.

The six main types of elements in Ymir are:

1. Perception: A set of Unimodal Perceptors, and Multi-Modal Integrators.
2. Decision: A set of Deciders, overt and covert.
3. Action: A set of Action nodes (also called 'behaviours' in prior publications; we use 'action' to avoid confusing these with the larger, more involved Behaviours in Edmund) and Action Morphologies - the lowest-level motor programs in the system.
4. Intermodule communication: A set of Blackboards.
5. Knowledge: A Dialogue Knowledgebase module, and a set of Topic Knowledgebase modules.
6. Organisation: Four semi-independent process collections: three perception-decision layers, containing Unimodal Perceptors and Multi-Modal Integrators, and an Action Scheduler that operates on Action nodes.

The three perception-decision layers group together perception and decision modules with common time dependencies, in order to load-balance process execution at run-time and sort decisions into high, medium and low priority for execution. Multimodal sensory data can flow in to all layers, though not all data is relevant everywhere. Ymir can accommodate any number of modules — action, perception, decision, planning, and knowledge — and they can be added incrementally, since all control and information flow happens via message passing through Blackboards. Blackboards thus allow communication between the modules, both within and between process collections. The Edmund planning mechanism was fitted into Ymir's architecture using the basic modules of Ymir, greatly extending the power of the original implementation without introducing new modules or changing the way they communicate.

2.5 Real-Time Planning in Ymir and SoL

Typically, control of and Ymir agent is non-deterministic. That is, the results of events are not guaranteed since they take into account unpredictable local and global delays and system load at all levels of processing. Perception modules produce intermediate and final data reports, posted to blackboards over time, which contain results such as direction of gaze (of user), the agent's own position in space, user's gestures, facial expressions, prosody, and words spoken. An example of an intermediate perceptual result is the classification of an arm movement as communicative. Knowledge processes piece these perceptual data reports together to come up with a meaning of user's behaviour. The status of the knowledge processing is intermittently reported to the blackboards as well, providing the rest of the system with information about the state of this more complex — and therefore more time consuming — processing.

The perceptual and knowledge data reports are read by decision modules, which decide what to do from moment to moment based on the perceptual state, as well as internal processing states and interpretation of the user's speech, prosody, body language and gesture. Decisions can be either overt or covert. Covert decisions concern strategically turning sub-systems on and off, and initiating complex, goal-initiated internal computation such as "parse-speech-input-now" and "look-for-deictic-referent-in-gaze-stream". Overt decisions become Motor Act Requests, which may or may not result in outward behaviour of the agent. The expression of overt decisions is determined by the Action Scheduler.

The Action Scheduler receives Motor Action Requests, e.g. "nod" or "smile", from deciders and from Edmund's Competences. Motor Action Requests are "intentions" to perform specific acts; the Acts can be executed in many ways by the Action Scheduler by selecting from alternatives stored in a Motor Lexicon. The Motor Lexicon is constructed as an AND-OR tree. The tree combines the idea of stored postures with the idea of hierarchical storage of increasingly smaller, and more detailed units. This method for representing motor control leads to a database where functional (e.g. "show-happiness") and morphological (e.g. "smile") definitions of motion co-exist in the same space, with no need for distinct division lines between the two classes. This is powerful because it allows a designer of decisions to access any behaviour node with a simple reference (e.g. "blink-once", or "smile"), and allows for very rapid construction of other modules, once the Motor Lexicon has been created. The structure of the lexicon also allows it to be extended iteratively. Extensions to the Motor Lexicon will enrich the expression of established Acts without altering the representations in the Deciders or reactive plans.

Once an Act has been turned into motor events by a lookup to the Lexicon, the Action Scheduler executes the events in increments of 100 ms. In the current implementation a graphics system receives output from the Action Scheduler and controls addressable motors with relative positions for a number of control points that link to the character's anatomy, in our case creatures with up to 10 degree-of-freedom movement. For example, to move the left wing upwards, the motor controlling the wing motion will be moved to position 0 (all the way up) in a given time (performance duration). The full command to the animation system may look something like this: (Motor:LeftWing; Position: 0; Time: 500msec). When executed, the Action Scheduler creates 5 slices from this motor event, each taking 100 msec to perform.

2.6 Modularity and Constructive Narrative Characters

The explicit separation of decisions (via decision modules) and perception (via perception modules) has several advantages, the primary ones being simplicity of construction and easy re-use of functionality. For example, perceptual data already produced by a collection of perception modules may be re-used for a new decision, given that it is relevant. Any Action already created can be triggered (requested) by a newly created decision module under different circumstances, e.g. a "nod" action may be triggered when the agent complies with a request, but also when the agent acknowledges that its name has been uttered by the user. This makes the architecture particularly well-suited to construction, since decision modules are really just a link between perception and action; as long as an agent or creature has a sufficient perception and action repertoire, a child constructing the mind of an agent can do so by hooking together the perceptions and actions via decider modules, using simple rules.

Another feature particularly relevant to playful worlds is a mind's control from the outside: Events in the world (represented both symbolically or metrically) can have direct influence on the creatures' minds. A creature can be given virtual ESP by connecting outside events directly to its decision processes. Outside events can also be connected directly to the knowledge structure, leaving the decision mechanism untouched. A creature with such virtual ESP could tell children about things that are happening in a remote part of the world as they speak. The same feature allows easy remote control of creatures, either at the lowest motor level or at a more abstract level, using buttons labelled with "smile", "show anger", "bid-farewell", etc. A separate representation of motor behaviour patterns allows the addition of knowledge and decision mechanisms that access motor patterns already created, at a high level, allowing "plugging" and "unplugging" of knowledge, such that a player can give a creature the ability to tell stories by simply plugging in the story telling module.

We believe that the combination of Ymir and Edmund is ideal because it addresses a broad range of behavioral phenomena - psychosocial dialogue skills, 3D navigation, planning, natural language, vision, and more - all of which are prime candidates for free play. It is extremely well-suited to construction because of its modular nature.

Integrating Edmund and Ymir into SoL was very easy because both architectures are behaviour based and implement forms of reactive planning. Although Edmund supports coherent behaviour, reactive plans expect the unexpected in both the environment and in the agent's own performance. Such considerations are necessary in Edmund's original domain of robotics, because of the unreliability of mechanical actuation. Consequently, the non-determinism of Ymir's plan execution posed no problems for Edmund's POSH plan structure. Similarly, Ymir's plan representation structure was more than adequate to represent the fundamental POSH components.

The biggest difference between the two architectures is knowledge representation. Ymir, like many architectures including PRS and Soar, represents explicit knowledge in a single, general purpose knowledge base. Edmund and the BOD methodology keep all knowledge within the behaviour metaphor, using specialised representations, and associating it with relevant perception and behaviour. Since SoL's VR timing skills hinge on Ymir's Action Scheduler and Motor Lexicon, SoL follows Ymir's representational framework, including knowledge representation. However, modularity at least in documentation is still recommended for ease in scaling and maintenance. This is also facilitated in Ymir, where the knowledge base is divided into a variety of Knowledge Areas.

2.7 Implementing Characters in SoL

The SoL architecture provides the framework for the middle layer of our proposed three-layer design approach. AI facilitates the creation of a socially engaging world; however such a world also requires careful overall creative design, and a rich visual and behavioral structure. Because SoL is both behaviour based and has POSH action selection, it is an excellent platform for practising Behaviour Oriented Design. BOD breaks the AI design process into two phases: an initial specification phase and a cyclic development phase of implementation and testing.

The initial decomposition is a set of steps. Executing them correctly is not critical, since the main iterative development strategy includes correcting assumptions from this stage of the process. Nevertheless, good work at this stage greatly facilitates the rest of the process. The steps of initial decomposition are the following:

1. Specify at a high level what the agent is intended to do.
2. Describe likely activities in terms of sequences of actions. These sequences are the basis of the initial reactive plans.
3. Identify and prioritise goals or drives that the agent may need to attend to. This describes the initial roots for the POSH action selection hierarchy.
4. Identify an initial list of sensory and action primitives from the list constructed in step 2. These are the initial Acts.
5. Select a minimal set of basic behaviours required for expressing and testing the plans behaviour. This forms of basis for the initial version of the Motor Lexicon.

The remainder of the development process is not linear. It consists of the following elements, applied repeatedly as appropriate:

– coding behaviours in the Motor Lexicon,
– coding reactive plans,
– testing and debugging this code, and
– revising and elaborating (scaling) the specifications made in the initial phase.

Heuristics for revising specifications, particularly for deciding whether intelligence should be coded as part of a behaviour or as part of a reactive plan, have been described elsewhere (Bryson, 2000). One of the main attributes of BOD is that it allows for modular development, so that a particular plan element might initially be stubbed as a simple primitive, but then later in the development process be replaced by a more complex element, with no change to old reactive plan scripts. In fact, old scripts can be reused for testing as changes to established behaviours are made. Similarly, replacing a plan with a behaviour, if necessary, can be achieved with a minimum of disruption.

3 Designing Character-Based Creative Play: The Three-Layer Approach

As mentioned in the introduction, creative play can be viewed as consisting principally of novel recombination of established elements. In fact, the evolutionary utility of play is considered to lie in enabling an individual to acquire and rehearse complex behaviours, as well as to learn appropriate situations in which to express them (Bekoff and Byers, 1998; Byrne and Russon, 1998).

In our view, it would be a mistake to attempt to design agents which were themselves expected to develop playful skills over time, in a self-sufficient a way. Even children, who are highly-honed learning machines, take years to acquire such behaviours to any degree of entertaining proficiency. Consequently, even if such a task were within the ability of science and technology, in the relatively pragmatic and demanding field of entertainment, an artificial system is best instilled from the beginning with as much knowledge as its designers can impart. This has been referred to as the engineering approach to artificial intelligence development (Ziemke, 1998), and follows from our work on Edmund and Ymir.

Similarly, AI developers should not necessarily be expected to be sufficiently skilled artists that they can create the plots and characters needed for a fully engaging interactive play experience. AI attracts (and perhaps requires) developers with a hubristic belief in their own ability to replicate the thinking skills of others. However, good artists devote years of attention, and often their formal education, to perceiving and constructing the things that make a situation interesting, æsthetic and fun. Our design process places the AI developer as an intermediary between the artistic and the engineering aspects of the project. This is level 2 of our design process model, specified in the table in Section 1. The AI developer is in the best situation to understand both requirements and restrictions of the overall project, and therefore has considerable responsibility for communication as well as developing solutions.

The AI expert is responsible for taking a set of motivations, goals, knowledge, personality quirks and skills, and creating an agent that will behave coherently according to these. In a rich virtual environment designed for free, creative play an autonomous character should be able to prioritise its goals and display its intentions. It should exhibit both persistence and resolution while at the same time being aware and opportunistic. In short, it should have a recognisable personality. Developing the initial set of character attributes, however, is not necessarily solely the task of the agent expert. It *is* necessarily the task of one or more creative artists. The artist's responsibility is to provide well formed and interesting characters, skills and situations, to design potential plots and plot twists. This is level 1 of our design process model. In this, as in most industrial design, it will be best if the artists work in a team with the agent developers, who can help the artists understand the limits of the agent's behavioral and expressive capabilities.

The agent developers are themselves constrained by the particular platform on which the artificial agent is to be implemented. In robotics these constraints come from the robot's hardware; in virtual worlds they come from the graphics environment in which the agent will be embodied. Creating this platform is level 3 of our design process. It is the responsibility of the AI developer to provide requirements for, and understand the constraints of, the underlying platform. Again, the character personality developer may or may not be the correct person to develop the agent's behavioral platform, depending on whether the platform in this context also provides the basic behaviours, or behaviour primitives, for the agents.

It is our view that the motor control of an autonomous character belongs to the realm of AI, but where precisely the "brain" meets the "body" can get blurry, especially in a virtual world. For example, it may make sense to put collision detection or motor smoothing into the "world" (i.e. the graphics environment itself), either for more efficient performance of the system or for cleaner implementation and easier debugging. In nature, vertebrates have dedicated systems for providing such smoothing in their hindbrain (Carlson, 2000), as well as being able to rely on physics for smoothness and consistency. In a simulated world the division between an agent's own perception and the world itself may not be well defined. Implementations in level 3 can become a point of contention because on either side of the fence between graphics and AI, very different skill sets have been developed, and people working on each side may prefer very different solutions to the problems at hand.

Grossly, the levels of our design process model correspond to the different sides of SoL. The interface between levels 1 and 2 leads to specifications of personalities and drives, and the interface between levels 2 and 3 lead to the implementation of the behaviours. But as is emphasised under BOD, the design process has to happen iteratively. Many forms of technical constraint might only be recognised after development has begun. Further, as the system develops, it can provide considerable creative inspiration to the designers. Even more importantly, early users, particularly those coming from outside the project, will discover both shortcomings and unforeseen creative potential in the system. All of these sources of information should lead to periods of redesign and renegotiation between the various levels of the project. Further, personality may be demonstrated in subtle motions best provided in the behavioral level, or complex behaviour may require or suggest changes to the plans and drives. Thus all three levels of the design process must be available for cyclic development and reanalysis. The AI programmers working primarily at level 2 cannot be abandoned to try to satisfy potentially impossible constraints coming from isolated processes on either side of the project.

4 Case Study: Creating Characters for an Adventure Narrative

The design process described above was developed as part of a research effort at LEGO to create an interactive virtual reality entertainment package that allows children to engage in creative and constructive play within an established action/adventure framework. The project illustrates the design principles above, and gives indication of the efforts and difficulties involved. We will refer to the AI portion of this large-scale, multifaceted research effort as the "castle character project". This effort included a detailed, relatively large virtual world with a castle situated on rolling hills, surrounded by a mountain range. A full moon hangs in the sky; the sun just under the horizon. Users can enter the world either through a desktop, or as fully embodied virtual (humanoid) LEGO characters with full body tracking and immersive glasses with displays.

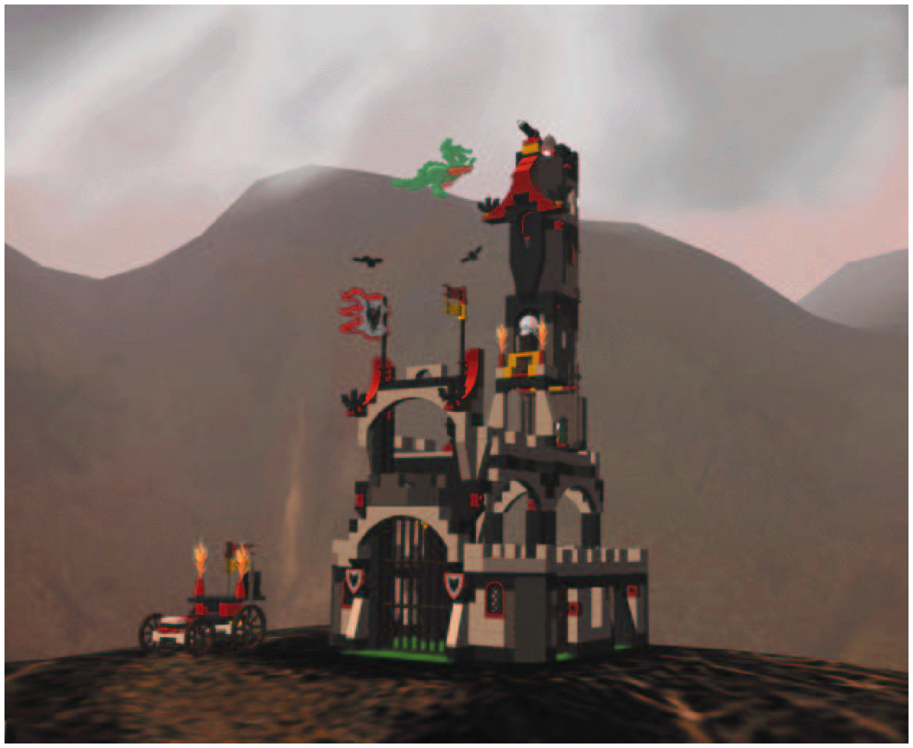

Fig. 1. Still from "the castle project," a real-time interactive virtual reality environment. Image ©1998 The LEGO Group

4.1 High Level Design

In the case of the castle character project, much of the character content was predetermined, as it was a virtual version of an active product. The general appearance of the characters, an outline of their personalities, as well as their world, had been developed as a part of the marketing, but no stories had been created. The domain was a magic castle, inhabited by an evil knight and various magical entities. Much of the larger VR research effort was dedicated to ensuring that simply exploring the space would be intrinsically rewarding, but it was the introduction of moving characters that made the virtual experience become alive and magical. For example, there is a SoL character named Puff. Puff is a talking, flying green LEGO dragon. Puff can discuss the castle, or be encouraged to demonstrate his flying ability.

The first step toward creating an interesting narrative for a set of characters is to understand the constraints of the task and the system. One set of constraints comes from the character's environment, e.g. the size and features of open spaces: The castle world, though complex and interesting, is not very large relative to the size of the characters, so this constrains the characters motions inside the castle. This can be compensated by setting the most gross motion (such as large-character flying and sword fights) to the space surrounding the castle. Another set of constraints are those dependent on the expected users of the system. Because expected users were young, naïve to virtual worlds and, perhaps most importantly, only exposed to the system for a few minutes total, we considered it essential to make the characters interesting whether or not the user deliberately attempted to interact with them. The solution was to make the characters interact with each other as well. They were also designed to react to the visitor in their domain in a way that encouraged exploration, but not to be too forceful or too intrusive on the user's experience. To maintain interest, the characters should act and interact in such a way that they generate continuous change. There should be no steady state that the system of characters can reach if the user is being passive.

The constraints of the virtual environment and the pre-existing product meant that most of this change had to take the form of arrivals and departures, as well as a few gross gestures. This effect was achieved by designing characters with various incompatible goals. For example, a witch could frequently fly around the castle in a quest for intruders. When she found the intruder she would do little other than land nearby, slightly approach the stranger and cackle. However, her presence might attract other characters, some of whom might in turn repulse her (she was designed to fear flying bats). Having characters that are attracted by some situations, yet repulsed by either crowds or other characters, can help maintain the amount of free space needed for character motion. In addition, it limits the number of simultaneous interactions, and therefore the amount of confusion. This allows the designers to quickly focus the interest for the short-term visitor.

Notice that stateless "reactive" social behaviours such as flocking (e.g Reynolds, 1987; Matarić, 1992) will not be sufficient — the characters here are doing more than being repulsed, attracted and avoiding obstacles. They are displaying personalities. A visitor can learn individual character's traits, and then manipulate these deliberately. Exploring the personality space of the characters in the world becomes part of the puzzle, and part of the fun.

4.2 Encoding Personality

As described in BOD above, after creating a rough description of the desired world, the next task is to develop a first-cut description of the reactive plans which will encode each character's personality. Starting from the descriptions of the characters set by the marketing department of the product, and keeping in mind the constraints determined in evaluating the task, each character was described in terms of three to five goals or drives. Further, the behaviour associated with achievement of these goals was visually described. This work was done by a team of in-house artists and external creative consultants, with the AI team participating both creatively and as technically informed resources.

Once the personality of the characters has been sketched, the next steps were as follows:

– Prioritising goals or gross behaviours and determining their necessary preconditions. For example, the witch described above has a goal of patrolling the castle from the air. This has a fairly high priority, but the motivation should be reduced by the performance of the act, so that in general she circles the castle only three times. She has a priority of landing in a room in which she has seen an intruder, once she no longer desires to fly. She also avoids bats.

- Determining necessary behaviour primitives and behaviour states. For example, the witch has to remember if she saw an intruder on her patrol. A bat might approach an intruder closer and closer over successive swoops. A state within the bat's swooping behaviour enables it to keep track of its current level of "boldness," which in turn determines its trajectory. Some characters can be made into friends by playing with them. These would have to remember how friendly they feel towards a particular person. Seeing the user, avoiding the walls of the castle, flying and landing are behaviour primitives required by all of these agents.
- Developing and testing the behaviour libraries and the scripts.

The architectural and methodological support we developed for this level has already been discussed, in Section 3.

4.3 Developing Perception and Action Primitives

In developing behaviour libraries, the task of the personality designer connects to the task of environment's architects. For the castle character project, some of the potential difficulties of this relationship were overlooked, and caused some of the greatest difficulties of the AI effort.

There are several possible approaches for building the basic movement primitives. One straightforward approach would be for the character developers to program the behaviours from scratch using models prepared by the graphic artists. There is a general problem for this approach: As mentioned earlier, AI programmers are not necessarily artists or students of natural motion. Animals have evolved complex motion behaviours, constrained by physical forces and structures not normally modelled on an artifact, particularly one designed to run in real time, so difficult to take into account. Animals are also constrained by habits of behaviour, whether general to a species or specific to an individual. Even if æsthetic motion primitives are achieved by an AI programmer, the process of programming them is likely to have been very time-consuming.

Another potential source of behaviour primitives explored on the castle character project were the efforts of a team of animators already working on the project. The idea was to segment animations into sets of behaviours suitable as exemplars of various behaviour primitives. A continuous variety of behaviour could be derived from combining and connecting fixed sets of canned behaviours. Unfortunately, animations also proved slow and difficult to develop. More importantly, the format the animations were produced in was determined to be incompatible with the primary real-time virtual reality environment. Real-time was an important part of the AI effort, and a critical feature of playful, creative spaces. The graphics rendering loop tends to be the critical element in the eye of the perceiver, since glitches (e.g. delays) in a character's thought process can be interpreted in many acceptable ways (hesitation, sluggishness, character flaws), whereas glitches in frame advancement are perceived as system failure. In the case of Puff, the SoL LEGO dragon, the real-time limitations for the behaviours were most obvious when trying to synchronise speech synthesis, which was run on a separate computer, to the graphical movements of the dragon's mouth. The animated approach was never-the-less used to account for the preoccupation of the evil knight who had possession of the castle: he is seen being engaged outside the castle in a sword fight with a good king. This arrangement was ultimately still unsatisfying, because without AI, the action was repetitive, and worse could not move to actively avoid wandering embedded user characters.

We also explored an intermediate solution: a purpose built animation tool for "quick and dirty" animation segments stored in an appropriate format for the main VR engine. This technique was used for creating some of the most life-like motion on the castle, a guard that responded to an approaching camera / observer by turning and facing it. The intelligence behind this character was purely reactive, and did not use SoL, but it did show the promise of this technique. Motion capture of humans participating as puppeteers was the final source of "intelligence" explored in the project. This could also have potentially served as a source of primitives for AI, but this alternative was not explored due to lack of time.

The approach used on Puff was to heavily exploit the Action Scheduler mechanisms derived from Ymir. Using this method combined with routines established in the project's VR library for controlling the dragon model's degrees of freedom, building a complete movement library took only two days for a single AI programmer. These motions are not as elegant as the hand-crafted animations, but they do provide complete, integrated control of the creature's body at all levels, from tiny finger and eye movements to body language and action. The Puff character integrated an array of technologies, including speech recognition and generation. Interactions with the dragon were constrained to eliciting explanations and short stories (e.g. "Tell me about the castle") so that the character need only recognise a limited set of queries and requests, facilitating the use of situated planning to give multi-modal responses such as gestures and actions as well as speech.

Another difficulty residing in the third design layer is the actual, technical connection of the agents' intelligence to their world. The most obvious solution to this problem is to consider the agents' minds and the graphics world as separate, asynchronous processes. This is essentially the approach we took for Puff. One problem we had to solve was getting data from the agent's sensory apparati - which are in the graphics domain - to its mind, which is running independently, in our case written in another programming language (LISP) and running on a separate computer. One of the technical solutions we used was CORBA. CORBA, being platform and language independent, solves a number of integration and organisational problems. The main problem with the CORBA solution is that over a 100 Mb Ethernet the loop time from a raw perception (collected via the graphics loop) to the mind and back (i.e. perception-action loop) takes too long to simulate realistically very fine-granularity responses such as being startled (e.g. by a scream behind the agent's back) or to provide fluid, rapid turn-taking, both essential characteristics of highly interactive play.

In developing character behaviours, the task of the personality designer connects to the task of environment's architects. Some of the issues of interfacing between a character's behaviours and the muscles of its body, as well as the character's senses and perceptual mechanisms can present large difficulties, unless the VR world and interface is built with the problem of supporting characters in mind. Nevertheless, this first practical effort of using the three-layer approach for constructive narrative enabled us to unify designers with very different skill sets, and to test the employment of an advanced AI architecture in a large virtual world.

5 Conclusions

In this paper, we have presented and described our experiences with a three-layer design process for developing an environment for constructive social play. We have also presented SoL, an architecture for complex characters capable of multi-modal real-time dialogue with humans, and some of our experiences from using these techniques on a large-scale industrial VR project at LEGO.

A constructive narrative is creative on several levels. In designing a creative experience, the goal is to provide both interesting media for expressing the content to be recombined, and tools that facilitate the recombination. If the media itself includes active creators, in our case agents that autonomously create situations and social dynamics, then the user has the opportunity to engage in truly complex constructive play. This kind of creative experience is currently only afforded to writers of drama, corporate managers, and public policy makers. However, creating an environment for such play takes considerable artistic and technical skill and planning.

We have described how a creative environment with constantly changing stories and adventures can be developed using artificial intelligence and design techniques that exploit and express the creativity of the designers. The intelligent agents in these environments are literally agents of creativity rather than being significant creators themselves: they embody the rules and knowledge both invented and learned by their designers. The design approach presented here can be used for designing creative environments and constructive narratives. The creative experience for the user consists of exploring the possibilities of the physical and social environment, and finding new, entertaining ways to exploit the spaces and personalities. Our design process focuses on the roles of the various team members in communicating and constructing an interesting reality, based around AI characters. The characters are implemented using behaviour-based techniques, for simplicity of design, combined with situated planning devices, to allow for complexity of characterisation and behaviour, and more traditional knowledge-based systems for natural language and dialogue abilities. In the future, we hope to develop more fully interactive characters. We would also like to develop more open narrative architectures that allow the users to design the characters themselves.

Acknowledgements

The team who created the project and the environment described were: Dent-de-Lion du Midi, Kristinn R. Thórisson, Christian Greuel, Svend Tang-Petersen, Victor Bonilla, Dan Mapes, Claude Aebersold, Inge E. Henriksen, Preben Kæseler, and Julian E. Gómez. Their work as artists, visionaries and engineers is gratefully acknowledged. The AI and dialogue work was directed by Kris Thórisson, and included, in addition to those mentioned above, Joanna Bryson, Mikkel Arentoft, Súsanna Thorvaldsdóttir and Christian

George. Claude Aebersold was responsible for the animations mentioned in Section 4.3, and Victor Bonilla for the guard character. Lieselotte van Leeuwen assisted with child psychology; Celia Pearce assisted with concept development; Jacob Buck and Michael Nielsen contributed to technical implementation. The castle project was developed using Silicon Graphics equipment and expertise. Thanks also to Lynn Andrea Stein, Will Lowe, the anonymous reviewers, and the lawyers who contributed to the final form of this paper. The authors gratefully acknowledge LEGO's generosity in supporting this work. LEGO is a registered trademark of The LEGO Group, Billund, Denmark. Kris Thórisson is now at Soliloquy, 255 Park Avenue South, New York, NY 10010.

Bibliography

André, E., Rist, T., and Müller, J. (1998). Integrating reactive and scripted behaviors in a life-like presentation agent. In Sycara, K. P. and Wooldridge, M., editors, *Proceedings of the Second International Conference on Autonomous Agents*, pages 261–268. ACM Press.

Bekoff, M. and Byers, J. A., editors (1998). *Animal Play: Evolutionary, Comparative, and Ecological Perspectives*. Cambridge University Press.

Boden, M. (1987). *Artificial Intelligence and Natural Man*. Basic Books, New York, 2nd edition.

Bonasso, R. P., Firby, R. J., Gat, E., Kortenkamp, D., Miller, D. P., and Slack, M. G. (1997). Experiences with an architecture for intelligent, reactive agents. *Journal of Experimental & Theoretical Artificial Intelligence*, 9(2/3):237–256.

Bryson, J. (1999). Hierarchy and sequence vs. full parallelism in action selection. In Ballin, D., editor, *Intelligent Virtual Agents 2*, pages 113–125, Salford, UK.

Bryson, J. (2000). Making modularity work: Combining memory systems and intelligent processes in a dialog agent. In Sloman, A., editor, *AISB'00 Symposium on Designing a Functioning Mind*, pages 21–30.

Bryson, J. and McGonigle, B. (1998). Agent architecture as object oriented design. In Singh, M. P., Rao, A. S., and Wooldridge, M. J., editors, *The Fourth International Workshop on Agent Theories, Architectures, and Languages (ATAL97)*, pages 15–30, Providence, RI. Springer.

Bryson, J. and Stein, L. A. (2001). Architectures and idioms: Making progress in agent design. In Castelfranchi, C. and Lespérance, Y., editors, *The Seventh International Workshop on Agent Theories, Architectures, and Languages (ATAL2000)*. Springer. *in press*.

Byrne, R. W. and Russon, A. E. (1998). Learning by imitation: a hierarchical approach. *Brain and Behavioral Sciences*, 21(5):667–721.

Carlson, N. R. (2000). *Physiology of Behavior*. Allyn and Bacon, Boston.

Cassell, J. and Jenkins, H., editors (1998). *From Barbie to Mortal Kombat: Gender and Computer Games*. MIT Press, Cambridge, MA.

Cassell, J. and Thórisson, K. R. (1999). The power of a nod and a glance: Envelope vs. emotional feedback in animated conversational agents. *International Journal of Applied Artificial Intelligence*, 13(4/5):519–538.

Georgeff, M. P. and Lansky, A. L. (1987). Reactive reasoning and planning. In *Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87)*, pages 677–682, Seattle, WA.

Grand, S., Cliff, D., and Malhotra, A. (1997). Creatures: Artificial life autonomous software agents for home entertainment. In Johnson, W. L., editor, *Proceedings of the First International Conference on Autonomous Agents*, pages 22–29. ACM press.

Hayes-Roth, B. and van Gent, R. (1997). Story-making with improvisational puppets. In Johnson, W. L., editor, *Proceedings of the First International Conference on Autonomous Agents*, pages 1–7. ACM press.

Hexmoor, H., Horswill, I., and Kortenkamp, D. (1997). Special issue: Software architectures for hardware agents. *Journal of Experimental & Theoretical Artificial Intelligence*, 9(2/3):147–156.

Joffe, B. A. and Wright, W. (1989). Simcity: thematic mapping + city management simulation = an entertaining, interactive gaming tool. In *Proceedings of GIS/LIS*, volume 2, pages 591–600, Orlando, Florida.

Kortenkamp, D., Bonasso, R. P., and Murphy, R., editors (1998). *Artificial Intelligence and Mobile Robots: Case Studies of Successful Robot Systems*. MIT Press, Cambridge, MA.

Lester, J. C. and Stone, B. A. (1997). Increasing believability in animated pedagogical agents. In Johnson, W. L., editor, *Proceedings of the First International Conference on Autonomous Agents*, pages 16–21. ACM press.

Levison, L. (1996). *Connecting Planning and Action via Object-Specific Reasoning*. PhD thesis, University of Pennsylvania. School of Engineering and Applied Science.

Matarić, M. J. (1992). Designing emergent behaviors: From local interactions to collective intelligence. In *Proceedings of the Second International Conference on Simulation of Adaptive Behavior*, pages 432–441, Cambridge, MA. MIT Press.

Newell, A. (1990). *Unified Theories of Cognition*. Harvard University Press, Cambridge, Massachusetts.

Pearce, C. (1997). *The Interactive Book : A Guide to the Interactive Revolution*. Macmillan Technical Publishing, Indianapolis, IN.

Reynolds, C. W. (1987). Flocks, herds, and schools: A distributed behavioral model. *Computer Graphics*, 21(4):25–34.

Roussos, M., Johnson, A. E., Leigh, J., Vasilakis, C. A., Barnes, C. R., and Moher, T. G. (1997). NICE: Combining constructionism, narrative, and collaboration in a virtual learning environment. *Computer Graphics*, 31(3):62–63.

Sengers, P. (1998). Do the thing right: An architecture for action expression. In Sycara, K. P. and Wooldridge, M., editors, *Proceedings of the Second International Conference on Autonomous Agents*, pages 24–31. ACM Press.

Simonton, D. K. (1997). Creative productivity: A predictive and explanatory model of career trajectories and landmarks. *Psychological Review*, 104:60–89.

Thórisson, K. R. (1996). *Communicative Humanoids: A Computational Model of Psychosocial Dialogue Skills*. PhD thesis, MIT Media Laboratory.

Thórisson, K. R. (1997). Layered modular action control for communicative humanoids. In Thalmann, N. M., editor, *Computer Animation '97*, pages 134–143, Geneva. IEEE Press.

Thórisson, K. R. (1998). Real-time decision making in face to face communication. In Sycara, K. P. and Wooldridge, M., editors, *Proceedings of the Second International Conference on Autonomous Agents (Agents '98)*, pages 16–23, Minneapolis. ACM Press.

Thórisson, K. R. (1999). A mind model for multimodal communicative creatures & humanoids. *International Journal of Applied Artificial Intelligence*, 13(4/5):519–538.

Thórisson, K. R. (2001). Machine perception of real-time multimodal natural dialogue. In McKevitt, P., editor, *Language, vision and music*. John Benjamins, London. *in press*.

Ziemke, T. (1998). Adaptive behavior in autonomous agents. *Presence*, 7(6):564–587.