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. Here we describe an architecture of these characters, relate architectural modularity to the realization of construction and play, and propose a three-layer design process for producing fertile and æsthetic constructive narratives. We also discuss 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. A 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 a 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), gave 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 take the necessary building blocks off-the-shelf.

In this paper we describe a character-based approach to constructive narratives, and a design process for constructing a play environment to support 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 level, highly artistic design level for creating story and characters,
2. a middle, behaviour-based design level for creating personality in the agents, and
3. a low level for design of basic behaviours.

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 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 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 SoL: A Character Architecture for 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 online 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.

The work described below uses a new architecture called Spark of Life (SoL), which merges features from two prior architectures developed by the authors, Ymir (Thórisson, 1996, 1999) and Edmund (Bryson and McGonigle, 1998; Bryson, 2000).

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 multimodal 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 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's primary contribution to SoL is planning: it provides Parallel-rooted, Ordered, Slip-stack Hierarchical (POSH) action selection. POSH action selection allows for persistent, rational-appearing behaviour 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.

By combining the strengths of both architectures within the modular architectural approach of behaviour-based AI, SoL encompasses the following capabilities: multimodal 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 some of the contributions of Edmund and Ymir to SoL, with particular emphasis on aspects that relate to our 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.

2.1 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.

Action selection under Edmund is handled by reactive plans specified in a scripting language. The most primitive elements of these scripts are action and sensing interfaces to the agent's behaviours. Edmund's scripting language allows for three levels of control structure above these primitives. First is the *action pattern*, a simple sequence which run 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 possible level 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 a 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. If a long chain does terminate, the drive remembers the root of the behaviour, and returns to this starting point.

For an example of a competence, think of a character that needs to drink out of a particular 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. The highest priority element for which its preconditions indicate it can fire, will fire. The highest priority element is to recognise 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. If the character is not yet holding the chalice, but is within line-of-sight of it, it will attempt to grasp the object. 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.

Notice that because the competence is set up to be a reactive plan, it can be opportunistic — if the agent is handed a chalice while it had been seeking it, it will immediately drink from it, it can skip the grasping step. Similarly, if the grasp fails for some reason, then that step can be repeated. Notice that this 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 may 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). Edmund has also been compared and integrated with related approaches such as PRS (Georgeff and Lansky, 1987), though not in a rigorous experimental setting (Bryson and Stein, tion). 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.

2.2 Ymir's Contribution to SoL

Edmund's planning structure works well for high-level design, but is too cumbersome for micro-managing details of behavioral implementation. It also does not by default contain customised knowledge-based behaviours for dialogue skills, world knowledge, nor mechanisms for orchestrating the sensory systems and motor control of multiple modes. Real-time human to virtual-reality character interaction and dialogue requires coordination of complex sensory processing and multimodal action control on a broad range of time scales, from gaze control to telling stories. Fortunately, Edmund, as a behaviour-based architecture, is designed to exploit expertise from other, autonomous or semi-autonomous sources.

In stand-alone implementations of Edmund, fine behavioral details are handled by the platform, either by a robotic platform (Bryson and McGonigle, 1998), where physics combined with well-designed behaviours provide continuity, or by an abstract artificial life platform with discrete time-steps (e.g. Bryson, 1999), where there are no fluencies below a certain level of detail. The complexity of dialogue in VR characters is the domain. Providing fluid behaviour in VR, where physics cannot be relied on, and managing the special complexity of natural language and multi-modal dialogue are the specialities of Ymir. Consequently, in SoL, Ymir provides the behaviour architecture framework as well as many specialised routines for dialogue processing.

Ymir has been thoroughly tested in an interactive, conversational system (Thórisson, 1996). Results show the behaviour patterns generated by Ymir in real-time interaction with actual people to be comparable to interaction between two human beings engaged in task-oriented, and its performance by users is considered very life-like and believable (Thórisson, 1998, 1996).

The six main types of elements in Ymir are:

1. Perception: A set of Unimodal Perceptors, and Multimodal Integrators.
2. Decision: A set of Deciders, overt and covert.
3. Action: A set of Behaviours and Behaviour Morphologies (low-level motor programs).
4. Interprocess communication: A set of Blackboards.
5. Knowledge: A Dialogue Knowledge Base, and a set of Topic Knowledge Bases.
6. Organisation: Four semi-independent process collections: three perception-decision layers, and an Action Scheduler.

The three perception-decision layers group perception and decision objects together into time-dependent groups, which are used to load-balance process execution and prioritise decisions into high, medium and low priority for execution. Multimodal sensory data can flow in to all layers (but not all data is relevant everywhere). Ymir can accommodate any number of modules — behaviour, perception, decision planning, and knowledge, and these 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 process collections and between. Typically, events in the model are non-deterministic. That is, the results of events are not guaranteed since they take into account unpredictable delays and system load at all levels of processing in a distributed fashion. The Action Scheduler uses a Motor Behaviour Lexicon, consisting of hierarchically defined action elements, to carry out motor-level actions in small increments based on decisions from either Decider modules or from the Edmund competences. In the current implementation a graphics system receives output from the Action Scheduler and controls addressable 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.

Behaviour Requests are produced by the Decider modules and are "intentions" to perform specific acts; the acts can be executed in many ways by selecting from a set of Motor Behaviour Morphologies available for each act. Motor Behaviour Morphologies are chosen from a library of alternatives, constructed as an AND-OR tree. The tree uses the idea of stored postures, as well as the idea of hierarchical storage of increasingly smaller units. This method for representing motor control leads to a database where functional and morphological 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 once the Motor Behaviour Lexicon has been created. (Alternatively the Motor Behaviour Lexicon can be created after the fact to mirror the decisions that the creature has to turn into motor plans.) Competences trigger any Motor Behaviour Morphology and have the Action Scheduler take care of executing the resulting motion.

The Deciders are explicitly modelled, separating perception processes in the Perceptors and action morphology and motor control in the Action Scheduler. This has several advantages, the primary one being simplicity of construction. 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 — giving a creature virtual ESP simply involves 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 patters 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.

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. We believe that this combination is ideal because it addresses psychosocial dialogue skills, 3D navigation, planning, natural language, vision, and more — a broad range of behavioral phenomena. It is extremely well-suited to construction because of its modular nature.

2.3 SoL and Behaviour Oriented Design (BOD)

As mentioned earlier, 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 previous list of actions.
5. Identify the state necessary to enable the described primitives and drives. Cluster related state elements and their primitives into specifications for behaviours. This is the basis of the behaviour library.

6. Select a first behaviour to implement.

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

- coding behaviours,
- coding reactive plans,
- testing and debugging code, and
- revising 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 replaces 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 Agents for 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 expected to develop playful skills over time, in a self-sufficient a way. Children themselves 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 probably 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 the Ymir architectures.

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 seems to attract (or perhaps require) 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, occupying level 2 of our design process model, specified 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, occupying 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. Sometimes it makes sense to place smoothing and blending functions into the "world" (i.e. the graphics environment itself), either because of more efficient performance of the system or a cleaner implementation and easier debugging. In nature, vertebrates have dedicated special systems for providing such smoothing in their hindbrain, particularly the cerebellum (Carlson, 1994), as well as being able to rely on physics for smoothness and consistency. Similarly, in a simulated world the division between an agent's own perception and the world itself may not be well defined. This can become a point of contention because on either side of the fence, 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, these levels correspond to different sides of the BOD: 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 (c)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 more 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 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.

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

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 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 nevertheless 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 move-

ments 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.

In developing behaviour libraries, 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 creative environment with constantly changing stories and adventures can be developed by 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 called *constructive narratives*. The 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. We hope to eventually develop more fully interactive characters, and 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. 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. Joanna Bryson is funded by Lynn somehow.

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.

Barlow, H. (1994). What is computational goal of the neocortex? In Koch, C. and Davis, J. L., editors, *Large-Scale Neuronal Theories Of The Brain*, pages 1–22. mitpress.

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.

Brooks, R. A. (1991). Intelligence without representation. *Artificial Intelligence*, 47:139–159.

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*.

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. (in preperation). Architectures and idioms: Making progress in agent design. In *The Seventh International Workshop on Agent Theories, Architectures, and Languages (ATAL2000)*. to be presented July 2000.

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. (1994). *Physiology of Behavior*. Allyn and Bacon, Boston, 5 edition.

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

Clark, A. (1996). *Being There: Putting Brain, Body and World Together Again*. MIT Press, Cambridge, MA.

Coad, P., North, D., and Mayfield, M. (1997). *Object Models: Strategies, Patterns and Applications*. Prentice Hall, 2nd edition.

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.

Hallam, J. C. T. and Malcolm, C. A. (1994). Behaviour, perception, and action — the view from situated robotics. *Philosphical Transactions of the Royal Society of London*, 349:29–42.

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.

Horswill, I. D. (1993). *Specialization of Perceptual Processes*. PhD thesis, MIT, Department of EECS, Cambridge, MA.

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.

Matarić, M. J. (1997). Behavior-based control: Examples from navigation, learning, and group behavior. *Journal of Experimental & Theoretical Artificial Intelligence*, 9(2/3):323–336.

Nilsson, N. (1994). Teleo-reactive programs for agent control. *Journal of Artificial Intelligence Research*, 1:139–158.

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

Pöppel, E. (1994). Temporal mechanisms in perception. *International Review of Neurobiology*, 37:185–202.

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. (1998). Real-time decision making in multimodal face-to-face communication. In *Proceedings of the Second International Conference on Autonomous Agents*, pages 16–23.

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

von der Malsburg, C. (1995). Binding in models of perception and brain function. *Current Opinion in Neurobiology*, 5:520–526.

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