MIST: An Interactive Storytelling System with Variable Character Behavior

Richard Paul, Darryl Charles, Michael McNeill, and David McSherry

School of Computing and Information Engineering
University of Ulster, Coleraine BT52 1SA, Northern Ireland, UK
paul-r@email.ulster.ac.uk,
{dk.charles,mdj.mcneill,dmg.mcsherry}@ulster.ac.uk

Abstract. Despite advances in game technology, most stories constructed by game designers remain inherently linear in nature, and player actions often have limited impact on the central story. In interactive storytelling approaches, an important challenge is the creation of stable yet dynamic environments to allow the emergence of unscripted stories involving both human-controlled characters and autonomous non-player characters (NPCs). In this paper, we present an architectural design for creating open-ended, interactive storytelling systems in which story structure emerges in real time and in response to player actions, thus providing a greater variety of game experiences than more scripted approaches. We present a partial implementation of the approach in a virtual environment populated by multiple NPCs that exhibit stable but interesting autonomous behavior. Finally, we present experimental results that demonstrate the scalability of the approach and variability of NPC behavior that it produces.

Keywords: multiplayer games, interactive storytelling, virtual worlds, autonomous agents.

1 Introduction

During the Artificial Intelligence Summit at the 2010 Game Developers Conference it was suggested that there are three different categories of games: rollercoaster, experimentation, and challenge [1]. Each game category provides players with opportunities to experience stories. Traditional challenge games such as Nintendo's *Tetris* typically contain minimal, designer crafted stories, while recent rollercoaster games such as Activision's *Call of Duty: Modern Warfare 2* and Sony's *Heavy Rain* illustrate the importance of story in modern game design. However, the stories used in rollercoaster games often provide only an illusion of choice and freedom. Typically they are based on linear progression, and player interaction has limited impact on the main story. First person and third person action-orientated genres naturally belong to the rollercoaster category and often use movie-like cut scenes to progress their stories, again in a linear fashion. Experimentation games such as Rockstar's *Grand Theft Auto IV* and Ubisoft's *Assassin's Creed II* often have more open game play, referred to as *sandbox* play.

R. Aylett et al. (Eds.): ICIDS 2010, LNCS 6432, pp. 4–15, 2010.

Players often have plenty of opportunities to make choices in experimentation category games but the choices that they make tend to have limited effect on the game's central story. Typically, branched or layered techniques are used to provide structure to non-linear storytelling [2]. Branched storytelling is a variation on linear approaches whereby players are given choices that affect the direction of the narrative at certain points in the game play, and may enable different endings to be reached. Recent role-playing games (RPGs) tend to use a layered storytelling approach that comprises a central linear or branching story line alongside a range of side stories. Layered stories provide players with a sense of empowerment over the direction of the story. However, the side stories used in this approach often have little or no relevance to the main story, and typically require a large amount of authoring.

In Massively Multiplayer Online Role Play Games (MMORPGs), it is common for many players to experience exactly the same story; for example, a villain is slain by one group of players during a quest, and then revived for a new instance of the quest so that another group can slay the villain again. While there are obvious practical reasons for designing games in this way, the result of allowing resources to be respawned is that player actions may have little influence on the main story [3].

Various approaches to maintaining plot coherence have been proposed by interactive storytelling researchers. In *Opiate* [4], for example, players interact with a three dimensional (3D) virtual world, talking with other characters and interacting with objects. The system uses case-based reasoning (CBR) to select the most appropriate arrangement of plot elements for a given story instance, which players must then follow in a linear fashion. *Haunt 2* [5] is a 3D interactive storytelling game in which heuristics are used to guide user interactions within a plot that is abstractly authored in advance. NPCs sometimes carry out plans to satisfy their own desires, while at other times they are assigned tasks by a *story director* in order to progress the plot.

GADIN [6] is a text-based interactive system that automatically generates narrative while focusing on dilemma situations between characters. If choices made by the user make the story goal improbable, then the system randomly selects a new story goal that does not involve any further player actions. *Mimesis* [7] provides intelligent narrative control over a 3D world space. A narrative planner generates a plan to satisfy a set of story goals, and player actions that threaten the plan's integrity are either incorporated into the plan or prevented from succeeding by the activation of pre-authored events (e.g., a gun jamming when the player tries to kill a central character). In *I-Storytelling* [8], the user interacts with the system either by influencing NPCs through voice recognition or by hiding objects. Plot coherence is ensured by allowing only NPC actions related to the ongoing story.

The goal that motivates the work presented in this paper is to develop an approach to interactive storytelling in which non-player characters (NPCs) behave autonomously and (multiple) player choices can be supported without compromising the coherence of the plot. In Section 2, we present an architectural design for building dynamic and stable game worlds to support an approach to interactive storytelling in which story structure emerges in real time and in response to player actions. In Section 3, we present a partial implementation of the proposed architecture in a virtual environment populated by multiple NPCs that exhibit stable but interesting autonomous behavior. In Section 4, we present the results of experiments to investigate the scalability of the approach and variability of NPC behavior that it produces. Our conclusions are presented in Section 5.

2 Interactive Storytelling in MIST

In this section we describe our approach to the design of MIST (Multiplayer Interactive StoryTelling), a system for interactive storytelling in a dynamic virtual world where NPCs can perform tasks autonomously to satisfy their internal motivations as well as interacting with each other in various ways. An important goal in our approach to interactive storytelling is to provide human controlled characters with greater freedom in the interactions that they choose to perform in the world than would be possible in a strictly plot based approach, particularly with many players logged on to the system.

System Design. As shown in Fig. 1, our approach is based on a two-tiered architecture. The lower tier is a virtual world consisting of locations and game objects and populated by players and autonomous NPCs. However, fully autonomous behavior is unlikely to provide sufficient dramatic interest for the user and may result in stories with limited narrative potential. A second tier therefore consists of a Drama Manager which is responsible for ensuring the enactment of stories that it generates by assigning story-related tasks to NPCs. The Drama Manager injects narrative into the game, re-planning in response to world state changes, and generally keeping the game moving towards a valid conclusion. As in *Opiate* [4] and *Haunt 2* [5], stories are represented at an abstract level, enabling them to be applied in different ways depending on the world state. Our approach also allows stories to be personalized by incorporating past player actions (e.g., making friends or enemies). The abstraction of story events helps to provide variability in the plans generated by the system.

Drama Manager. The Drama Manager uses an artificial intelligence (AI) planner to generate story plans in response to world state changes. It also uses its (complete) knowledge of the world state to assign tasks to NPCs to assist in story progression. Initially the Drama Manager has a hierarchical network of story elements, which can be pieced together in different ways to form a story. Each story element has a set of preconditions that determine whether or not it is valid for a particular game context. The current state of the game world is passed to the Drama Manager periodically by the game engine. The Drama Manager then attempts to build a story that fits the current world state using its AI planner and network of story elements (e.g., by checking if the preconditions of each story element are satisfied). In the event that no story is valid for the current world state, the Drama Manager assigns special tasks to characters with the aim of reaching a new state in which a valid story can be created.

Virtual World. The virtual world contains characters, locations (e.g., lake, forest, shop) and game world objects (e.g., matches, fires). NPCs are instances of professions such as thief, hunter, and woodcutter from which they inherit different behavior profiles. The virtual world is controlled by a game engine that is responsible for displaying the world, updating game world objects, and managing NPC states.

Non-Player Characters. NPCs operate under a Belief-Desire-Intention (BDI) framework [9]. One reason for this choice is to promote variability in NPC behavior. An NPC acquires its knowledge about the world from sensors. Internal sensors

provide the NPC with knowledge about its possessions, affiliations to other NPCs, current desire, and current location. External sensors provide additional knowledge related to game world objects (e.g., knowledge about items for sale in a shop) and the locations of other NPCs. Knowledge is represented as a list of facts (e.g., *Bob has matches*, *Sam is at the forest*). Facts have different lifetimes depending on the types of knowledge they represent, after which they are removed from the NPC's knowledge base. As shown in Fig. 2, each NPC also has its own AI planner.

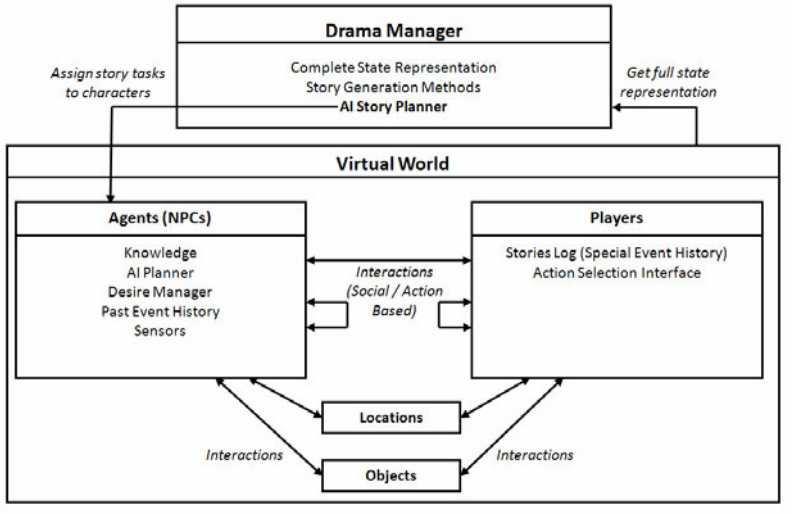


Fig. 1. System Architecture

Character Desires. Each NPC has the following desire types in order of decreasing priority: get warm, drink, eat, rest, make money, increase safety, socialize, reproduce and increase respect. The priorities are loosely based on Maslow's [10] hierarchy and ensure that an NPC can only have one active desire at a given time (i.e., the desire with the highest priority). Desires are realized as NPC attributes (e.g., heat, thirst, hunger), and each attribute has a current value and a threshold for activation of the desire. As the game progresses, the value of each attribute is modified according to a *decay rate* that depends on the attribute and results in a desire being triggered when the value falls below the threshold. Initially, attribute values and thresholds are set randomly to increase variability in behavior. Also, an attribute's decay rate can be increased (or decreased) to trigger NPC desires more (or less) frequently, for example to prevent NPCs from being stuck in a particular state for too long. Each NPC uses different methods to satisfy its desire to make money depending on its profession (e.g., a thief knows how to steal money or items). As we show in Section 4, this helps to provide variation in NPC actions and movements.

Character States. Each NPC is always in one of three states: *exploring*, *relaxing*, or *executing*. We now discuss these states in relation to the AI planner that an NPC uses

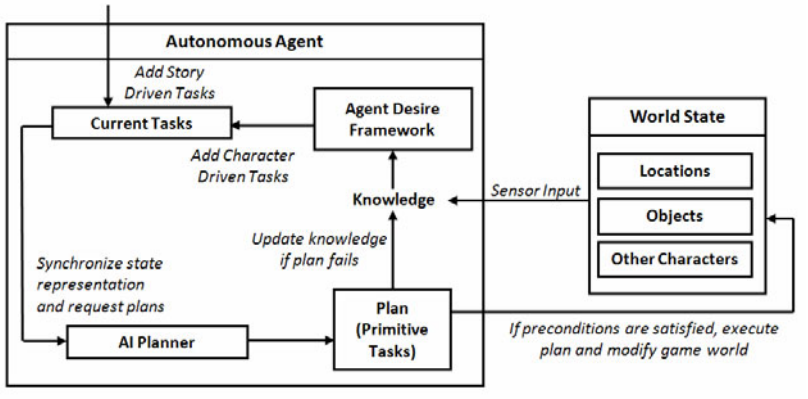


Fig. 2. NPC architecture

to generate a plan to satisfy its current desire (if any). Plans are created based on the NPC's knowledge of the world. A plan consists of one or more plan steps. Each plan step is a primitive task that the NPC attempts to carry out (e.g., *walk to campsite*). An NPC generates a new plan to achieve its current desire when one of the following events occurs:

- A desire is triggered by an attribute falling below a threshold value
- An NPC already has a desire and a new fact is added to its knowledge base, or an existing fact is removed
- A plan step fails because of a change of state that is unknown to the NPC

For example, an NPC's plan might be to buy matches from a shop in order to light a fire (e.g., to satisfy a desire to get warm). The NPC knows from a previous visit to the shop that it sells matches. When it reaches the shop, however, the NPC discovers that the shop is sold out of matches and its knowledge is updated accordingly. This triggers a re-planning process, based on the NPC's updated knowledge, which may result in a new plan being generated. While executing a plan, an NPC is in the *executing* state. If an NPC has a desire but does not have sufficient knowledge to generate a plan to satisfy its desire, it adopts the *exploring* state to gather more facts about the world, which should eventually lead to a plan being generated. An NPC that does not have any current desire is in the *relaxing* state. Eventually a desire will be triggered by changes in the NPC's attributes, which are continuously updated by the game engine according to the *decay rate* for each attribute.

Knowledge Sharing. Socializing (or conversation) between characters is managed by the game engine. Every six seconds there is an 80% chance that a character will share a fact not already known by another nearby character and a 20% chance that the two characters will say goodbye and depart. This basic mechanism helps to ensure variability in character behavior as well as enabling characters to share knowledge.

3 Implementation

In this section we describe our implementation of the two main components of the architecture, namely the AI planner used by each NPC (and also by the drama manager) and the virtual world.

3.1 The AI Planner

NPCs in the virtual world use hierarchical task network (HTN) planning [11-12] to create plans to satisfy their desires. In HTN planning, the planner is provided with a task to be performed (e.g., *make money*, *have drink*) and a set of manually authored *methods* for decomposing tasks into subtasks. Initially the main task is decomposed into subtasks according to the methods available for the task. A method can be applied only if its *preconditions* are satisfied. Each subtask is decomposed, if necessary, into a further set of subtasks. This process continues until only *primitive* tasks that can be executed without further decomposition remain. On successful completion of the decomposition process, the list of primitive tasks that have been generated provides a possible plan for performing the main task.

In our approach, all possible plans are generated by the planner and ranked in order of decreasing plan cost. If there is more than one possible plan, the planner returns one that minimizes the total cost of all the primitive tasks in the plan. For example, the cost of walking from one location to another is proportional to the distance between the two locations. Fig. 3 shows a simplified HTN for the task *have drink* that an NPC in the virtual world needs to perform because it is thirsty. Tasks shown with dashed boundaries (e.g., *go to well*) are primitive tasks that can be used as plan steps without further decomposition. Other tasks (e.g., *purify water*) need to be further decomposed. Initially, the items that the NPC has in its possession include a rope, a bucket, a bottle, matches, and sticks.

In this example, the planner has two methods for decomposing the *have drink* task into subtasks. The preconditions of the first method (shown as bullet points in Fig. 3) are *has rope* and *has bucket*, both of which are satisfied. So one possible plan for having a drink is: (1) go to well, (2) fill bucket, (3) drink water. The second method decomposes *have drink* into the subtasks *get some lake water*, *purify water*, and *drink water*. The planner has one method for *get some lake water* and one for *purify water*. As the preconditions of both methods (e.g., *has bottle*) are satisfied, another possible plan for having a drink is: (1) go to lake, (2) fill bottle, (3) go to campsite, (4) make a fire, (5) boil water, (6) cool water, (7) drink water. As the well is close to the NPC's location, the cost of the first plan is less than that of the second plan, so the planner returns the first plan to the NPC.

The HTN planner used in our work is written in Prolog, and the system is implemented with many instances of the planner (i.e., one for each NPC) running in the background and executing on separate threads. The Prolog code is linked with the game engine through DLL calls that allow strings to be passed as input and output.

3.2 The Virtual World

Fig. 4 shows a 2D realization of the virtual world. The main window is where game objects in the environment are rendered and NPCs can be seen performing actions to

complete their tasks. As each task is completed, an appropriate message is shown near the NPC. A map below the main window shows the current view relative to the rest of the world. Symbols above NPCs indicate the states they are in (e.g., relaxing or exploring). Information about the currently selected game object (if any) is displayed in the pane to the right of the main window. For a selected NPC, it shows the NPC's current knowledge, attribute values, and current plan (if any). Below the main window to the right of the map is the event history for a selected NPC or a global event history if no NPC is selected.

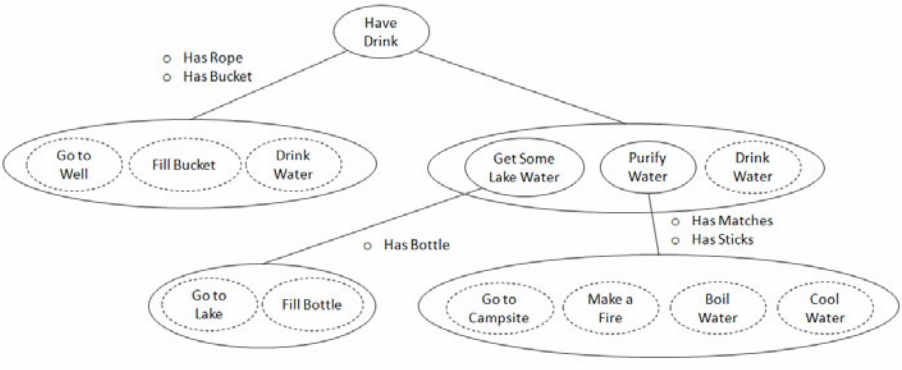


Fig. 3. Possible decompositions of an example task in an HTN

4 Empirical Study

In this section, we present the results of an empirical study in which we investigate NPC behavior in MIST under various conditions. Our experiments are based on simulations parameterized by the world size, the number of NPCs, and their assigned professions (e.g., woodcutter, berry picker), and the numbers of items and locations in the world. At the start of a simulation, items are distributed randomly throughout the world, either on the ground or in an NPC's possession. Some items can be spawned in the world through actions such as a woodcutter cutting wood or a berry picker picking berries. To satisfy their desire to make money, NPCs often collect these items and sell them to shops, thus creating a supply chain. Other items, such as matches and bottles, are regenerated by restocking shops at regular intervals.

We monitor population behavior in terms of NPC states (i.e., relaxing, exploring, or executing). NPCs relax if they do not have a current desire for anything, they explore if they have a desire but cannot generate a plan with their current knowledge, and they execute plans to satisfy their current desires once they have succeeded in generating a suitable plan.

Experiment 1. First we examine the proportions of NPCs in the relaxing, exploring, and executing states for 3 population sizes (100, 200, and 300) in a virtual world with numbers of items and locations fixed at 100. The world is square with a fixed size in the experiment of 10^8 pixels. The results shown in Fig. 5 are averages over 5

simulation runs. With 200 characters, the *relaxing* state is most prominent as the simulation converges to a stable pattern. However, it does not particularly dominate the other two states. With fewer NPCs (100), the relaxing state tends to dominate as the simulation settles. This indicates a simulation in which NPCs are able to carry out plans to satisfy their desires. For the largest population size (300), the distribution of NPC states tends to be more even, with *exploring* beginning to emerge as the dominant state. Although the distributions of NPC states are different for the 3 population sizes, NPC behavior can be seen to quickly stabilize in all three simulations.

Fig. 4. Virtual world in MIST

Experiment 2. In this experiment, we investigate population behavior in world sizes of 2×10^8 and 4×10^8 while keeping the other world parameters fixed (i.e., 200 NPCs, 100 items, and 100 locations). The results are shown in Fig. 6, again with results averaged over 5 simulation runs. It can be seen that the *exploring* state becomes more dominant as world size increases. This is to be expected as there is more space to cover between locations in the larger world, so that facts used to generate plans may no longer be true on arrival at the final destination. However, population behavior stabilizes quickly for both world sizes.

Experiment 3. Fig. 7 shows the results of a third experiment in which we vary the number of locations in a world of size 10^8 from 25 to 125 while keeping the other world parameters fixed (i.e., 200 NPCs, 100 items). Again the results are averaged over 5 simulation runs. When the number of locations in the world is low relative to

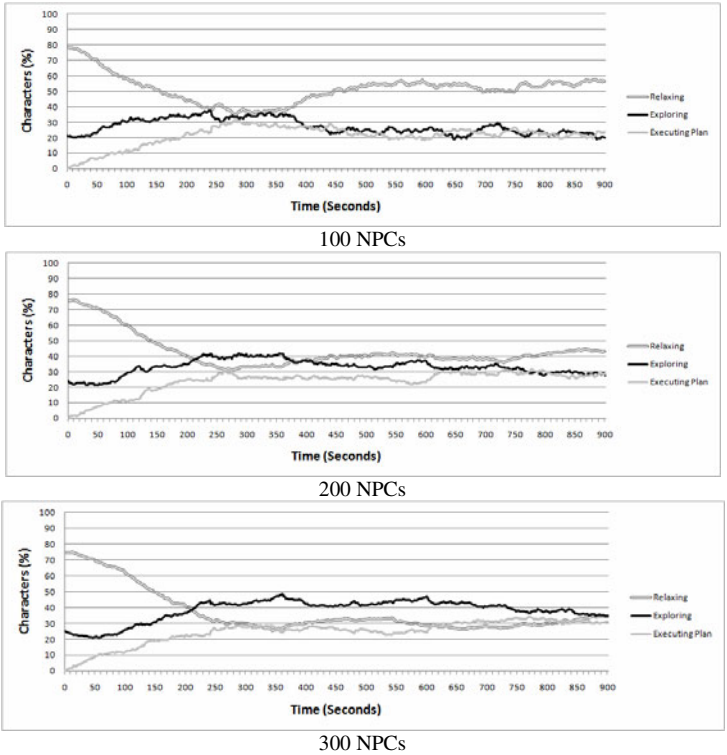


Fig. 5. Percentages of NPCs in relaxing, exploring, and executing states in virtual worlds with 100, 200, and 300 NPCs

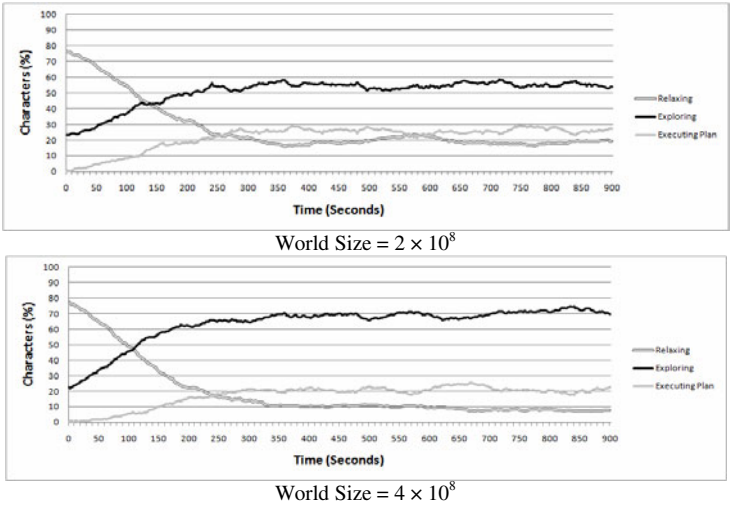
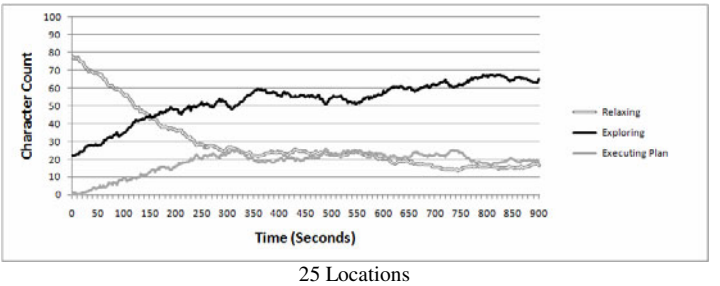
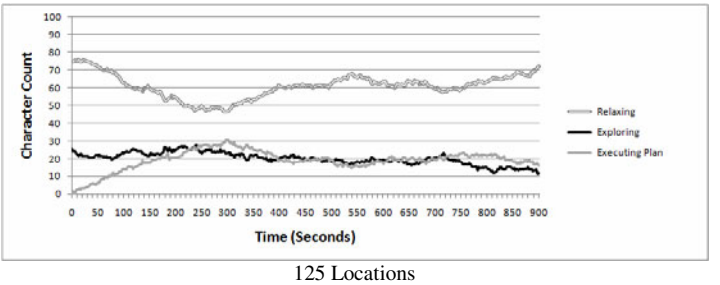


Fig. 6. Percentages of characters in relaxing, exploring, and executing states in virtual worlds of sizes 2×10^8 and 4×10^8



25 Locations



125 Locations

Fig. 7. Percentages of NPCs in relaxing, exploring, and executing states in virtual worlds with 25 and 125 locations

the size of the world, as in the first configuration, most NPCs in the population tend to be in the *exploring* state. However, when the number of locations increases, the pattern of population behavior changes to one in which most NPCs are in the *relaxing* state. Again this is not a surprising result as NPCs can solve their planning problems more easily when there are more locations to choose from.

Experiment 4. In a final experiment, we examine the distribution of the number of state changes that NPCs make, as a percentage of the overall population, in a 20 minute simulation with 200 NPCs, 200 items, and 25 locations. The world size in this experiment is 25×10^6 and results are averaged over 5 runs. The observed distribution of state changes among NPCs in each profession (e.g., woodcutter, fisherman) is shown in Fig. 8. The results provide clear evidence of the variability in NPC behavior; there is some variation between professions and in each profession there is a close-to-normal distribution of the state change count across the full population. Table 1 shows the state change statistics obtained by repeating the experiment with equal numbers of NPCs and items ranging from 50 to 250, the same world size (25×10^6), and the same number of locations (25). It can be seen from the results that the distribution of state changes is consistent for all these configurations of the virtual world.

Discussion. Our experimental results show the robustness of our simulation for large world sizes populated with many autonomous NPCs. The results also highlight some of the effects of resource levels (i.e., locations / items) available to NPCs on their behavior in large virtual worlds. Too few resources can lead to situations where

NPCs are continually trying to achieve goals. For example, if the world has too few locations for its size, then by the time an NPC has satisfied one desire, another desire may be activated, so that it seldom gets a chance to enter the *relaxing* state. In our storytelling system, we aim to reduce the time spent by NPCs in non-relaxing states (i.e., *exploring*, *executing*) to a reasonable level to avoid the world becoming too chaotic for stories to be effectively conveyed. On the other hand, if the virtual world has too many resources, this may have the effect of reducing the variability of NPC behavior within the population. For example, if a shop always has an item in stock, an NPC that plans to buy the item from the shop will never be forced to generate an alternative plan (except for reasons unrelated to the item's availability).

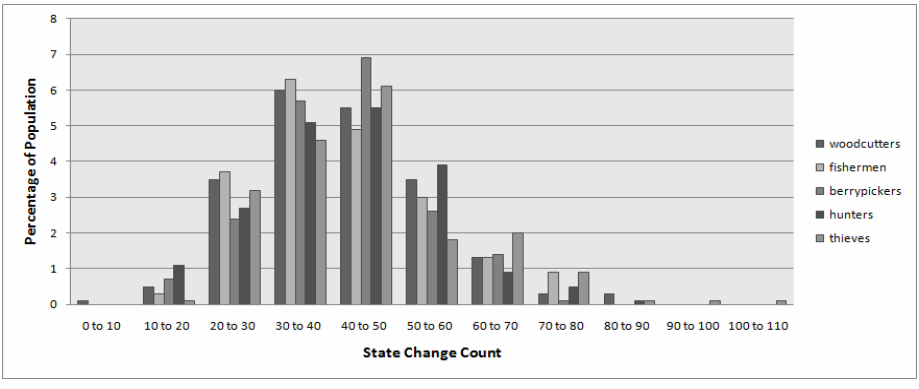


Fig. 8. Distribution of state changes in a population of NPCs

Table 1. Statistics of state changes for virtual worlds with equal numbers of NPCs and items

	No. of NPCs/Items				
	50	100	150	200	250
Average	45.0	43.5	49.0	39.9	42.6
Standard Deviation	13.6	14.6	14.9	13.4	14.4
Kurtosis	0.54	0.60	0.99	-0.003	-0.43
Skew	0.54	0.67	0.77	0.23	0.30

5 Conclusions

Most game stories are inherently linear in nature, often with plot coherence maintained by restricting player choices and re-spawning resources. The result is that player interactions may have little effect on the central story. In this paper, we presented an approach to interactive storytelling in which story elements change dynamically in response to run-time events in the game world. We also presented experimental results that demonstrate the scalability of the approach and its ability to support autonomous NPC behavior that is stable and consistent with respect to world

and population sizes. Multiple player characters will be introduced in future work, and we expect that their actions, like those of NPCs, will have direct consequences in the virtual world, thus enabling stories to be generated and plot coherence to be maintained with no need for re-spawning game objects or constraining player choices.

References

1. Kline, D.: Bringing Interactive Storytelling to Industry. In: AI Summit, Game Developers Conference (2010), http://aigamedev.com/open/coverage/gdc10-slides-highlights/
2. Thue, D., Bulitko, V., Spetch, M.: Player Modeling for Interactive Storytelling: A Practical Approach. In: Rabin, S. (ed.) AI Game Programming Wisdom, vol. 4, pp. 633–646. Charles River Media, Boston (2008)
3. Tychsen, A., Hitchens, M.: Ghost Worlds – Time and Consequence in MMORPGs. In: Göbel, S., Malkewitz, R., Iurgel, I. (eds.) TIDSE 2006. LNCS, vol. 4326, pp. 300–311. Springer, Heidelberg (2006)
4. Fairclough, C.R., Cunningham, P.: AI Structuralist Storytellin. Computer Games. Technical Report TCD-CS-2004-43. Trinity College Dublin (2004)
5. Magerko, B.: Story Representation and Interactive Drama. In: Proceedings of the 1st Artificial Intelligence for Interactive Digital Entertainment Conference, pp. 87–92. AAAI Press, Menlo Park (2005)
6. Barber, H., Kudenko, D.: Generation of Dilemma-Based Interactive Narratives with a Changeable Story Goal. In: Proceedings of the 2nd International Conference on Intelligent Technologies for Interactive Entertainment, pp. 1–10. Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, Brussels (2008)
7. Young, R.M., Riedl, M.: Towards an Architecture for Intelligent Control of Narrative in Interactive Virtual Worlds. In: Proceedings of the 8th International Conference on Intelligent User Interfaces, pp. 310–312. ACM, New York (2003)
8. Cavazza, M., Charles, F., Mead, S.J.: Character-Based Interactive Storytelling. IEEE Intelligent Systems 17, 17–24 (2002)
9. Johnson, D., Wiles, J.: Computer Games with Intelligence. In: Proceedings of the 10th IEEE International Conference on Fuzzy Systems, pp. 1355–1358. IEEE, Los Alamitos (2001)
10. Maslow, A.H.: A Theory of Human Motivation. Psychological Review 50, 370–396 (1943)
11. Lekavý, M., Návrat, P.: Expressivity of STRIPS-Like and HTN-Like Planning. In: Nguyen, N.T., Grzech, A., Howlett, R.J., Jain, L.C. (eds.) KES-AMSTA 2007. LNCS (LNAI), vol. 4496, pp. 121–130. Springer, Heidelberg (2007)
12. Nau, D., Cao, Y., Lotem, A., Muñoz-Avila, H.: SHOP: Simple Hierarchical Ordered Planner. In: 16th International Joint Conference on Artificial Intelligence, pp. 968–973. Morgan Kaufmann, San Francisco (1999)