Behavior Primitives for End-User NPC Behavior Creation

Daniel J. Rea,[1] Takeo Igarashi,[2] and James E. Young[1]
[1]The University of Manitoba, [2]The University of Tokyo

Abstract: The creation of interactive character behaviors is a difficult task that is generally relegated to scripting and programming: techniques not always accessible to game designers who want to author such behaviors. We propose a new user-centered method for shifting the authoring process away from programming paradigms based on the idea of user-centered *behavior primitives*; we show how common interactive behaviors can be described in terms of easy-to-understand primitives such as a goal visibility (hidden, or peeking around a corner) or relative position (stay behind the character), and that this is sufficient to enable the generation of convincing results.

1 Introduction

Creating quality interactive characters for video games is a difficult task that requires a large amount of time and skill. One goal in improving this creation process has been to develop new tools that reduce the effort required to design and create quality game content [1], [2], [3]; this saves time, enables authors to focus more on artistic content and less on technical details, and makes content creation more accessible to a broader game developer audience. In this paper we present a new tool for aiding the creation of computer-controlled interactive characters.

Creation of story driven games that take place in dynamic worlds are particularly challenging due to the diverse range of autonomous interactive characters, often called Non-Player Characters, or NPCs. In addition to character appearances (3D models, etc.) and animations, it is often desirable for these characters to have a high level of believable interactivity with the environment and player; for example, how a thief character notices an approaching player and hides, how a guard protects a valuable treasure chest from the player (Figure 1), or the complex interactive combat tactics employed by an enemy attacking the player. It is precisely the creation of these sorts of *interactive* character behaviors that we target in our work.

Researchers have already made great contributions for end-user authoring of interactive behaviors. Methods have generally relied on visual programming [5], and scripting [3], which lower the barrier-to-entry but require people to define behaviors in a programming-like framing with exact events and conditionals. Others have used programming by demonstration for interactive characters [6], but have so far only been successful for simple behaviors.

We propose to enable game developers to define behaviors using a set of easy-to-understand parameters that we call *behavior primitives*. For example, a designer might tell a thief character to try and stay out of the player's line of sight as well as to stay close to, and behind, the player character. This would result in an NPC that moves closer while using paths that minimize the NPC's visibility.

Our approach is to explore the development of a minimalistic set of *behavior primitives* that can encapsulate a broad range of behaviors, and to explore how these primitives can be used to generate convincing results. In the long run, we envision creating an interactive sand-box tool where designers can configure behavior primitives to author desired behaviors without low level programing, for example by using demonstration.

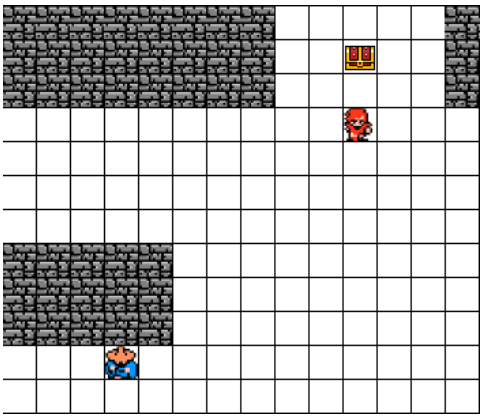

Figure 1: A guard (red) protecting treasure from a thief (blue)

2 Related Work

Some researchers simplify game design by providing high-level programming tools, for example, offering scripting languages for common complicated tasks, or using sentences to represent game logic [3]. Others enable behaviors to be created in a state-machine style structure by sequentially linking small predefined behavior components that are activated by user-defined if-else style triggers [5]. We propose to avoid such programming concepts which may not be accessible to designers without technical background.

One common method that avoids programming paradigms is programming by demonstration, where designers have been able to author complex 3D models by sketching simpler 2D representations [1], create convincing static animations through mouse-based performance demonstration [2], or even act out a desired interactive behavior style [6]. Current methods for authoring *interactive* character behaviors are only starting to emerge, and are currently

limited to simplistic behaviors and do not consider the environment [6]. We continue in this direction and aim to enable designers to create complex interactive behaviors without explicit programming.

Another non-programming approach to creating interactive behaviors is multi-agent systems and swarms, where agent-local goals are set and convincing complex group behaviors emerge (e.g., see [4]). Our proposed approach is similar in that we enable designers to configure parameters from which the desired result emerges, but with important differences: we focus on an emergent behavior for a single entity interacting with an environment and human-controlled player character, we aim for a small set of user-centered primitives that match how designers think about behaviors, and we want to enable the creation of a broad range of behaviors.

3 Behavior Primitives

We define *behavior primitives* to be clear, easy-to-understand characteristics of a target behavior that a designer could configure. For example, a designer could specify how close an NPC should be to the player, or how hidden they should be from the player. Behaviors are expressed by a combination of behavior primitives.

Through iterative and qualitative analyses, we focused on distilling a range of example behaviors into a minimal set of behavior primitives. As an example, a "sneak up to the player" behavior may encompass a thief drawing closer to the player while staying out of sight; this can be characterized as a combination of player-character distance and player-character visibility primitives. We repeated this process across 3 behaviors that are common in story based games; Table 1 shows the example behaviors and the related primitives we devised. We note that all of our primitives are spatial in nature, and some inherently include the environment in their definition (e.g. visibility).

Behavior	Primitives
Guard a point	*point of interest* (guarded point), *relative position based on point of interest*, *distance from player to point of interest*, *speed of NPC*
Sneak up to player	*relative position based on player's orientation*, *visibility*
Chase player	*distance from player to NPC*, *speed of NPC*

Table 1: Behaviors and their distilled behavior primitives

Below we outline how our behavior primitives generalize to other behaviors, giving a name to the new behavior, as well as describing the role the primitives play:

Follow – A behavior similar to chase but maintains a *relative position* slightly beside the player, with *NPC speed* matching player speed instead of exceeding it.

Bully – Try to harass the player by staying in their way. Uses *relative position based on point of interest* (a *point of interest* picked to be in the direction the player is heading).

Escape – This is the compliment of chase. Use a high *NPC speed*, minimize *visibility*, and have a large *distance from player to NPC*.

Guide – Lead the player to a certain location (*point of interest*). The guide wants to have a small *distance from player to point of interest*, but maintain a high *visibility*, so the player can always see the NPC and move towards it.

More behaviors could be imagined as well. With a larger set of sample behaviors to break down, we expect to create a general set of behavior primitives that will be able to create an exponential number of behaviors with respect to the number of primitives.

3.1 Generation of Behavior

When all primitives are considered together, they define an abstract target state that the NPC attempts to move towards. As each primitive has a different semantic meaning, the way movements are generated by each primitive is treated individually, and often conflict: for example, for visibility, the NPC moves towards a space that is closer to its desired visibility, but this may be further away from a target relative position. Currently, we combine all movements from each primitive evenly, but how the target is decided and how a user may author a behavior in this context is a main question of future work.

4 Limitations and Future Work

We have shown how a few simple behavior primitives can be used to describe and generate a broad range of interactive behaviors. Currently, our set of target behaviors was arbitrarily chosen, and moving forward we intend to interview professional game designers to develop a better representative sample.

We also investigated how the behavior primitives interact and combine to produce the final interactive behavior. When testing NPC movement generation with a player character, we manually defined the behavior primitive values. Through this, we noticed that the target state may have to change throughout interaction. In some behaviors the target is static, while in others it changes during the interaction. This makes the authoring process harder: how does a designer specify when and how these target primitives change?

These considerations will be used in prototyping and exploring user interfaces for creation of interactive behaviors based on our behavior primitives. Such an interface is non-trivial; for instance, Young *et al.* had numerous iterations of their behaviour-authoring interface with numerous user studies to evaluate them, each time noting different advantages and drawbacks [6]. As our fundamental approach is different (to our knowledge, no previous work has attempted to use such parameter based behavior authoring), we hope to be able to explore different options, while also integrating insights from our interviews with game designers into our interface prototypes.

5 Conclusion

We proposed a novel approach to author interactive behaviors for non-player characters in games by considering the designer's perspective. With an analytical approach, we created a set of *behavior primitives* that can describe our sample behaviors. We validated this by creating interesting interactive behaviors by hand. Next we will analyze many more behaviors from real game designers to investigate the robustness of our approach. While considering previous work and data from interviews, we will prototype an interactive behavior authoring interface for users that takes advantage of our behavior primitive approach.

References

[1] T. Igarashi, S. Matsuoka, and H. Tanaka, "Teddy," in Proc.- SIGGRAPH '99, pp. 409–416. ACM

[2] T. Igarashi, T. Moscovich, and J. F. Hughes, "Spatial keyframing for performance-driven animation," in Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation - SCA '05, 2005, no. July, p. 107.

[3] M. McNaughton, M. Cutumisu, D. Szafron et. al "ScriptEase : Generative Design Patterns for Computer Role-Playing Games," in Proc. Conference on Automated Software Engineering, 2004, pp. 386–387. IEEE.

[4] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," in Proc. SIGGRAPH '87, 1987, pp. 25–34. ACM.

[5] E. Y. Shen and B. Chen, "Toward gesture-based behavior authoring," in Proc. Computer Graphics International, pp. 59–65, 2005. IEEE.

[6] J. E. Young, T. Igarashi, E. Sharlin, D. Sakamoto, and J. Allen. "Design and Evaluation Techniques for Authoring Interactive and Stylistic Behaviors." In Proc. TiiS '12. ACM.