Proceedings of the Eighteenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment
(AIIDE 2022)

Never a Dull Moment: Believable Dynamic Character Beat Generation between Game World Events

Kyle Mitchell, Camden Pettijohn, Joshua McCoy
University of California, Davis
1 Shields Ave
Davis, CA 95616
kdmitch@ucdavis.edu, cpettijohn@ucdavis.edu, jamccoy@ucdavis.edu

Abstract

This work presents a heterogeneous system using ABL, CiF, and a virtual game world to demonstrate how moments between game world events can be made more believable. We describe the individual components used in our system and detail how each component contributes to the alignment of physical and mental contexts of NPC agents, and thus preserves believability across game events. Finally, we provide a brief account of the demo experience showcasing our system's capabilities.

Introduction

Despite the admirable work from many researchers in formalizing and realizing believable agent-driven characters in virtual worlds (Loyall 1997), (Mateas 1999), (Riedl and Stern 2006), (Bogdanovych, Trescak, and Simoff 2015), crafting such agents remains a difficult task. This difficulty stems from the fact that believability is derived from many layers of context, physical and mental, that should be meaningfully integrated into the behavior or reasoning of such agents. Moreover, each layer should be aligned with respect to each other, and such alignment should be preserved not only during moments of high player attention, but also during the moments between events. We can generate dynamic character beats during these interstitial moments to achieve this preservation.

This work presents a heterogeneous system using A Behavior Language (ABL) (Mateas and Stern 2002), a C# version of Comme il Faut (CiF) (McCoy et al. 2014), and a game world constructed with the Unity engine to demonstrate how such beats can be generated. We describe the individual components used in our system and detail how each component contributes to the alignment of physical and mental contexts of NPC agents, and thus preserves believability across game events. Finally, we provide a brief account of the short playable experience showcasing our system's capabilities.

Technical Description

The architecture of our system, shown in Fig. 1, consists of two programs that we will describe here: a Java proxy server that runs the ABL agent; and the game application itself, which manages CiF and all other game logic.

The ABL agent and Java proxy server. ABL is a language best suited for crafting virtual agents with strong real-time reaction capability–a desirable quality for this work. In addition to reactivity, however, we want our agents to plan courses of action and change plans over arbitrary periods of time–a task that is doable, but daunting, if using ABL alone. We argue that CiF offers some mitigation to this problem, which we discuss later in this work.

The GameAgent, our ABL agent, manages all NPCs in the game world. We pursue a "hivemind" approach (Mateas and Stern 2004) whereby the GameAgent runs a daemon (Weber et al. 2010) that waits for new NPCs that the player encounters. It then spawns a root node from which that particular character is managed more specifically. Thus, we maintain one behavior library from which multiple characters can draw and avoid incurring additional authorial burden. Specific character behaviors include: producing contextual dialogue; combat specific actions; and status-oriented animation. The output of all character behaviors is aligned with respect to the most recent copy of the character's CiF state, enclosed within the character's blackboard memory system. Any behavior producing an effect in the game world is delivered as an agent action to the game application via the proxy server.

Applications built with Unity are not natively compatible with the ABL runtime. Given this constraint, we wrote a custom Java program that serves a dual purpose: to run the GameAgent, and manage TCP/IP traffic between itself and the game application. This proxy server transforms actions produced by the GameAgent into data structures that can be interpreted by the game application, and it also transforms messages from the game application into working memory elements (WMEs) that are interpretable by the GameAgent. Outgoing data structures contain information like what action should be taken, which character should perform the action, and which character should be acted upon, if any.

CiF and the game world. While ABL can be described as goal-driven, CiF can be described as goal-setting (McCoy 2012). CiF allows us to implement and modify long-term plans for NPCs in a way that is less complex than doing the same solely with ABL. For instance, using our system,

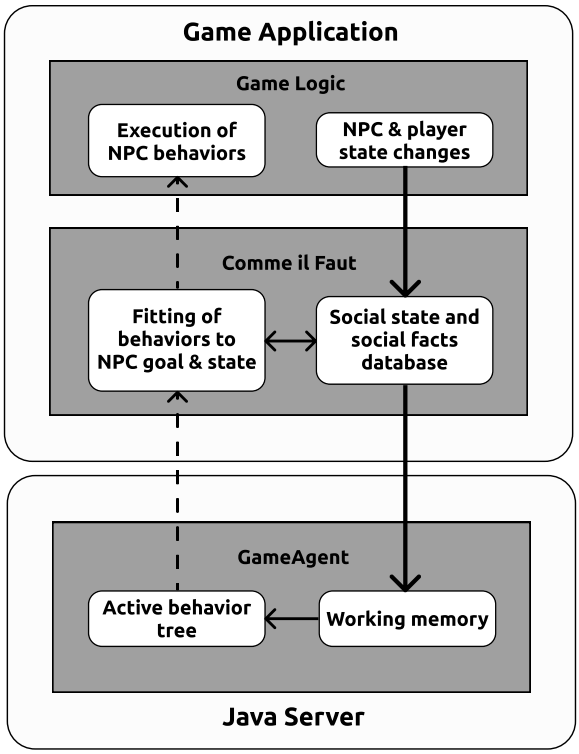

Figure 1: System architecture composed of three main entities: the ABL agent (GameAgent), CiF, and game logic. Solid arrows indicate the flow of state, while dashed arrows indicate the flow of decision-making.

the GameAgent may direct a NPC to produce a piece of dialogue when the player moves close to them. This behavior can then be modified by CiF so that it fits the character's long-term goals and current social state. We argue that, while CiF may not reduce authorial burden outright, writing rules in CiF to maintain a long-term, contextually-aware plan, coupled with small ABL behaviors chained together to accomplish the execution of the plan, is an easier task than trying to use ABL alone. By using both systems in appropriate ways, we can continually respect short-term context over longer periods of time without the incursion of undue authorial burden that may stem from the inherent complexity of ABL.

In this work, a 3D game built with Unity serves as a testbed virtual world. Given that the smallest common unit over which any ABL agent reasons is a WME, we construct a one-to-one C# WME class. Each NPC has access to their own WME, which contains information like a unique identifier and their current CiF state. Similar to the Java proxy server, the game application manages the TCP/IP connection between itself and the server. At frequent intervals throughout the game, WMEs for each active NPC and the player are generated, transformed, and sent to the Java proxy server. In addition, the game application listens to incoming messages from the Java proxy server that may contain actions that should be executed in the game world.

System in Play: Oops! All Bards Demo

As a piece of companion software to this submission, we designed a short game demo with sufficient complexity so as to highlight the capabilities of our system. We envision Oops! All Bards (O!AB) as a 3D role-playing game (RPG) in which the player can inhabit the persona of a bard living in a medieval fantasy world currently under siege by a maleficent entity. To illustrate how NPCs act and react in consistent ways across scenarios, the demo features Quinton: a semi-retired, grizzled bard who manages entertainment at a tavern called The Thirsty Whale. Quinton is a NPC who can be recruited to the player's party and will aid the player in the demo.

The demo features two scenarios: exploration and combat, which are staple gameplay loops of classic and modern computer RPGs. The player is first introduced to the exploratory scenario, wherein they can engage in dialogue with NPCs and learn more about the world of the game. After recruiting Quinton, the player begins the combat scenario. In combat, we highlight how all the components described in our system work together. Certain behaviors of the GameAgent are gated by CiF state values. If the affinity between Quinton and the player is high enough, behaviors will be selected and modified such that Quinton will act in more helpful ways. Quinton may also become critically injured, which applies a subsequent CiF status "Requires Assistance." For any NPC in combat with this status, the GameAgent will direct the character to reactively call out for help. Following the combat scenario, Quinton's "Requires Assistance" status will either have been removed because the player helped him in some way, or will transform via CiF into a "Left Hanging" status–the result of the player ignoring Quinton's plea. In the resolution of the demo, Quinton will approach the player differently in both physical demeanor and dialogue depending on the presence of the "Left Hanging" status.

Takeaways

In this work, we have presented a system consisting of an ABL agent, a virtual game world, and a C# version of CiF. We have described two main architectural components: (1) the Java proxy server that runs the ABL agent, transforms data sent by the game application into interpretable WMEs, and sends the output of ABL behaviors as actions to be executed by the game; and (2) the game itself, which handles all gameplay logic, packages game and CiF state into interpretable WMEs, and executes actions based on information provided by the Java proxy server. Our system allows agents to react appropriately in real-time given physical context via ABL, as with Quinton's calls for help during combat. It also allows agents to form longer-term plans that cohere with their mental context via CiF, as with Quinton protecting the player during combat, and approaching the player in different ways depending on how the player treated them. Combined, this system offers a way to generate dynamic NPC beats that help to preserve believability across gameplay events in a way that continually respects the NPC's physical and mental context.

References

Bogdanovych, A.; Trescak, T.; and Simoff, S. 2015. Formalising Believability and Building Believable Virtual Agents. In Chalup, S. K.; Blair, A. D.; and Randall, M., eds., *Artificial Life and Computational Intelligence*, 142–156. Cham: Springer International Publishing. ISBN 978-3-319-14803-8.

Loyall, A. B. 1997. *Believable Agents: Building Interactive Personalities*. Ph.D. thesis, Carnegie Mellon University.

Mateas, M. 1999. *An Oz-Centric Review of Interactive Drama and Believable Agents*, 297–328. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-540-48317-5.

Mateas, M.; and Stern, A. 2002. A Behavior Language for Story-Based Believable Agents. *Intelligent Systems, IEEE*, 17: 39– 47.

Mateas, M.; and Stern, A. 2004. *A Behavior Language: Joint Action and Behavioral Idioms*, 135–161. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-662-08373-4.

McCoy, J.; Treanor, M.; Samuel, B.; Reed, A. A.; Mateas, M.; and Wardrip-Fruin, N. 2014. Social Story Worlds With Comme il Faut. *IEEE Transactions on Computational Intelligence and AI in Games*, 6(2): 97–112. Conference Name: IEEE Transactions on Computational Intelligence and AI in Games.

McCoy, J. A. 2012. *All the World's a Stage: A Playable Model of Social Interaction Inspired by Dramaturgical Analysis*. Ph.D. thesis, University of California, Santa Cruz.

Riedl, M. O.; and Stern, A. 2006. Believable Agents and Intelligent Story Adaptation for Interactive Storytelling. In Göbel, S.; Malkewitz, R.; and Iurgel, I., eds., *Technologies for Interactive Digital Storytelling and Entertainment*, 1–12. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-540-49935-0.

Weber, B. G.; Mawhorter, P.; Mateas, M.; and Jhala, A. 2010. Reactive planning idioms for multi-scale game AI. In *Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games*, 115–122.