On Repairing Generated Behaviors for Graphical Characters

Andrea Corradini[1(✉)] and Manish Mehta[2]

[1] Design School Kolding, 6000 Kolding, Denmark
andrea@dskd.dk
[2] Accenture Technology Labs, San Jose, CA, USA
manish.mehta@accenture.com

Abstract. In this paper, we continue our work on the creation of artificial intelligence (AI) behaviors for graphical interactive characters by novice users. We refer to novice users as any persons who do not have any particular skills, training and experience in both programming and design. The focus of this paper is on the analysis of the needs and requirements that are necessary to identify and repair problems related to the rendering of the behaviors after the authoring stage. We also briefly sketch our strategy for the behavior repair in the behavior authoring system that we are developing.

Keywords: Non-playing characters · AI behavior repair and generation

1 Introduction

In a previous work [1], we addressed the problem of authoring AI behaviors of graphical interactive characters by novice users. This question has been explored also by researchers who have addressed the merit in developing tools for novice users to carry out part of product development and programming activities. We continue on that work by investigating the problem of repairing the behaviors authored by the same kind of users. At this point, it should be noted that we broadly use the term novice users to define users who do not have any skills, training and experience in both programming and design.

The repair of user-authored AI behaviors is an important step in many computing applications populated by graphical agents. Over long game sessions, a character's behavioral repertoire may result in repetitive behaviors that harm the agent's believability. An appropriate repair of the behavior set makes it possible for these characters to autonomously exhibit their author-specified personalities in new and unforeseen circumstances. Proper behaviors also ensure a better player experience. When authored behaviors fail, the player may get annoyed and bored, especially if the problems in the behaviors show up repeatedly. Moreover, each player enjoys different strategies to fight against (in the case of real time strategy games), or varying styles of storytelling (in the case of interactive dramas), different types of story development, different kinds of character behaviors and interactions, etc. Modifying the behaviors according to player's preferences provides a better interactive experience. It is also possible, for authored behaviors not to achieve the games objectives adequately, especially in realistic,

C. Stephanidis (Ed.): HCII 2016 Posters, Part I, CCIS 617, pp. 104–109, 2016.
DOI: 10.1007/978-3-319-40548-3_17

scaled-up domains or applications. These objectives could range from entertainment to education, training, etc. When these objectives are not met on a per-use basis, the AI behaviors should be accordingly modified. Many researchers and game developers hold that game AI is entertaining mainly when it is difficult to defeat [2]. Expert gamers in strategy games can quickly get bored of playing against a built-in AI because they can easily find and exploit flaws in the AI. We contend that modifying the behaviors in response to newer player's tactics would allow for a more challenging AI.

We organize the rest of this paper as follows: Sect. 2 reports on related works and background information. Section 3 briefly sketches the approach we chose to develop a behavior repair layer. Eventually, Sect. 4 summarizes our conclusions.

2 Related Work

The agent's behavior set can be seen as a reactive plan that dictates what the character should do under various conditions. From this perspective, behavior repair becomes a problem of reactive-plan revision. One approach to plan revision is to apply classical planning techniques to re-plan upon encountering failures. Such techniques are still ill suited to the unique requirements of a game domain. They typically assume that the agent is the sole source of change, actions are deterministic and their effects well defined, that actions are sequential and take unit time, and that the world is fully observable. In an interactive, real-time game domain, all these assumptions are violated. Actions are non-deterministic and their effects are often difficult to quantify. Game domains are typically not fully observable either. Some recent work in planning has focused on relaxing these assumptions. Conditional planners such as conditional non-linear planner [3] and sensory graph plan [4] support sensing actions so that during execution, changing environmental influences can be ascertained and the appropriate conditional branch of the plan is chosen based on the sensor values. Unfortunately, as the number of sensing actions and conditional branches increase, the size of the plan will grow exponentially. These techniques are mostly suited to deterministic domains with occasional exogenous or non-deterministic effects, rather than to continuously changing interactive domains. One of the approaches to planning that deals best with exogenous events and non-determinism is based on Markov decision processes (MDP). These approaches, however, require a large number of iterations to converge and only do so when certain conditions are met. In complex game domains, these techniques are intractable for MDP learning algorithms require a prohibitively large amount of time (polynomial to the number of states of the problem [5]). Upon adding game state information, the status level and internal states of various characters, the state space quickly grows untenable. Further, these approaches generalize poorly. An interactive player can significantly change the game world; the learned static policy must be re-trained to accommodate such changes.

In the AI planning community, previous work has investigated techniques for combining deliberative and reactive planning [6, 7]. Unfortunately, to varying degrees, they all make classical planning assumptions and are thus not applicable to real-time interactive games. Furthermore, these approaches treat reactive plans as black boxes;

planning sequences of black-boxed reactive plans, but not modifying the internals of the reactive plans themselves.

In transformational planning, the goal is to re-write the behavior set using a set of plan transformations in order to improve an existing reactive plan [8, 9]. This approach, although promising, is of limited usefulness in game domains because it requires a detailed casual model of the world. In games domains, there is neither the time for extended projective reasoning nor an accurate projection can be performed due to the interactive and stochastic nature of game domains.

Meta-reasoning is the process of monitoring and controlling reasoning. The meta-level control of computational activities deals with the question of when to stop a given computational process. The monitoring part is responsible of figuring out what can fail through the process of introspection also called introspective reasoning. Various approaches to meta-reasoning and more specifically to the task of introspective reasoning have been proposed [10–13, 16, 17].

Another body of related work is in the area of adaptive AI [14]. Dynamic scripting is a technique based on reinforcement learning that is able to generate scripts by drawing subsets of rules from a pre-authored large collection of rules [14]. If the rules are properly authored, different subsets of them provide meaningful and different behaviors. The technique is based on learning which subsets work better under different circumstances. The adaptive AI approach only learns conditions for the applicability of different rules. It assumes that the initial authored rules are perfect and it is unable to identify any problems with the internals of these rules. The Adaptive Behavior Language (ABL) provides an approach towards adaptive AI where the adaptive programming primitives are directly built into the ABL language [15]. The approach supports partial programming, a paradigm in which a programmer needs to specify only the details of behavior known at code-writing time, leaving the run-time system to learn the rest. The approach, though promising, requires novice authors to learn a programming language to allow the characters to be adaptive. Moreover, the adaptation of characters authored in the language to changing circumstances requires many examples.

3 Our Approach

3.1 Identifying and Repairing Failures: General Issues

We assume that a novice author has created an initial set of behaviors. Creating behaviors for graphical characters requires an enormous amount of work to make them properly react to their environment. Despite such efforts, it is however not possible to envision all the possible circumstances that a character would encounter in the game, ultimately leading to authored behaviors that do not reflect an agent's intended personality or do not fit the given interaction context.

There are several dimensions to the task of identifying and repairing failures in user-generated AI behaviors, which make it an especially hard and interesting problem. In general, these are the main questions that arise:

- Which metrics can be used to indicate that the authored AI behaviors need to be improved/modified?

- When does the detection of behavior repair occur?
- How is the feedback on whether the behaviors are performed correctly i.e. on whether they match the specified metrics provided?
- How can feedback be provided?
- When is the actual behavior repair carried out?
- Who has the responsibility for repairing the behaviors?

Performance Metrics. Different game genres present various performance requirements for a repair approach situated within them. For instance, god games usually require the game AI to solve resource allocation problems and long-term strategy problems, while interactive drama require the game AI to adapt the story according to the player interactions in a way that it is more appealing to the player (thus, the latter requires user modeling and story planning). Adventures, interactive dramas and other genres with embodied characters require believable behaviors. These requirements affect the kind of repair approach and results on a set of different metrics. A possible metric is believability when the goal of the repairing task is to have characters behave in accordance with their personality. Another metrics is AI performance when the goal of the repair approach is to provide a more challenging opponent to the player. Other common metrics are defined by player experience and adaptation to different levels of challenge. In both situations, the repair approach has to continuously scale up based on the game difficulty level to the point that a human player is challenged, but not completely overpowered. These metrics are not mutually exclusive. Any AI based repair approach that provides for a stronger AI could possibly provide for a better player experience [2]. Similarly, a believable performance by the characters could result in a more pleasurable experience for the player. In our approach, we use player experience as the performance metric.

Performance Measurement Stage. The performance of the resulting AI behaviors can be measured at different times in the game episode. Roughly speaking, there are two main points when this can occur: either, during the performance of the AI behavior sets in the game episode itself or after the completion of the game episode. In our approach, we detect the need for adaptation after the completion of each interaction episode. Once the player finishes the interaction with the avatar, s/he is asked to provide feedback in order to figure out whether any repair is needed or not.

Feedback Agent. The feedback about whenever the behavior performance does not match one or more of the desired criteria can by provided by the game itself, by the designer of the system, by the player interacting with the system, or a combination thereof. In our approach, we use the player as the feedback agent.

Type of Feedback. The feedback may simply inform the system whether the solution it produced succeeded or failed. In some cases, the feedback consist of a complete trace of the execution of the solution produced by the system, localizing the steps, which need modification. In others, the designer himself provides another preferred solution along with a complete trace of how this solution could be produced. Alternatively, the feedback only specifies the alternative preferred solution. In our current work, the

feedback provided by the player is in the form of several numerical ratings about the performance of the avatar.

Behavior Repair. The behavior repair can be carried out during the performance of the AI behaviors or after the completion of problem solving task. We carry out the repair after the completion of the player interaction with the avatar.

Responsibility of Modification. The responsibility of repair can occur either without any designer intervention i.e. lie solely with the underlying AI system (fully autonomous repair) or could be shared between the system and the designers (i.e. novice users) carrying out a mixed initiative repair. In our work, we choose a mixed responsibility shared between the AI system and the designer.

3.2 Overview of Our Repair Approach

The core idea of this work is the addition of a repair layer to the underlying AI system. This layer would allow to identify and to repair the failures in the behavior sets for them to better match the changing context and personality.

One of the key problems in repairing the set of behaviors is to define a metric to measure the success of both the authored behavior sets and the overall interaction. Our strategy is broken down into three main parts. At first, we associated behavioral constraints to authored behaviors in order to provide a metric to measure the success of their rendering. Further, we collected players' feedback about their interaction. Failures are then detected by comparing successful execution traces to unsuccessful ones. Eventually, the repair step involves looking at execution traces (notably, where failures do not occur) to suggest modifications about the failed/wrong behaviors. This is much similar to the work in [10], which focuses on using a library of pre-defined patterns of erroneous interactions among reasoning steps to recognize failures in the story understanding task. Differently from that, we do not require the construction of pre-defined patterns of erroneous interactions and a library of plans to repair the faulty behaviors. This frees the system designer from having to think beforehand of all the possible situations. We were also inspired by the works on model-based diagnosis by [13, 17] where a model of the system is used to localize the errors and to construct alternative traces, which could lead to the desired solution. Our repair approach does however not require a detailed model of the system. Instead, it provides a unique way of detecting the failures through use of knowledge available in successful and unsuccessful traces.

4 Conclusions

We addressed the problem of repairing authored behaviors. After discussing common issues, we introduced several dimensions to take into account when dealing with the task of identifying and repairing failures in user-generated AI behaviors.

The approach consists of three parts (a) constraints that are associated with behaviors that provides a way to measure the success of the behavior sets, (b) player feedback where players provide feedback at the end of their interaction, and (c) failure

detection and repair approach through comparison of successful and unsuccessful execution traces [18]. The repair then involves looking at execution traces (where these failures do not exist) to suggest modifications on the failed behaviors to the author.

We evaluated our approach with thirty authors who created a behavior set. To provide the necessary data for the AI repair system, we collected data from sixty-five players who interacted with animated avatars. When asked to compare the original behaviors with the repaired ones, all subjects found these latter smoother and more believable.

References

1. Mehta, M., Corradini, A.: Second mind: a system for authoring behaviors in virtual worlds. In: Proceedings of the Interaction, 16th International Conference on HCI, pp. 345–350 (2015)
2. Buro, M.: RTS games as test-bed for real-time ai research. In: Proceedings of the 7th Joint Conference on Information Science (JCIS 2003), pp. 481–484 (2003)
3. Peot, M., Smith, D.: Conditional nonlinear planning. In: Proceedings of the 1st International Conference on AI Planning Systems (1992)
4. Weld, D.S., Anderson, C.R., Smith, D.: Extending graphplan to handle uncertainty and sensing actions. In: Proceedings of AAAI (1998)
5. Kearns, M., Singh, S.: Near-optimal reinforcement learning in polynomial time. Mach. Learn. **49**(2–3), 209–232 (2002)
6. Gat, E.: Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots. In: Proceedings of the AAAI Conference (1992)
7. Bonasso, P., Firby, J., Gat, E., Kortenkamp, D., Miller, D., Slack, M.: Experiences with an architecture for intelligent reactive agents. J. Exp. Theoret. AI **9**, 237–256 (1997)
8. McDermott, D.: Transformational planing of reactive behavior. Technical report, Yale University (1992). Research Report YALEU/DCS/RR-941
9. Beetz, M.: Structured reactive controllers a computational model of everyday activity. In: Proceedings of the 3rd International Conference on Autonomous Agents (2000)
10. Cox, M.T.: Metacognition in computation: a selected research review. Artif. Intell. **169**(2), 104–141 (2005)
11. Cox, M.T., Ram, A.: Introspective multi-strategy learning: on the construction of learning strategies. Technical report, Georgia Institute of Technology (1996)
12. Anderson, M.L., Oates, T.: A review of recent research in metareasoning and metalearning. AI Mag. **28**, 7–16 (2007)
13. Stroulia, E., Goel, A.K.: Functional representation and reasoning in reactive systems. J. Appl. Intell. **9**, 101–124 (1995)
14. Spronck, P., Ponsen, M., Sprinkhuizen-Kuyper, I., Postma, E.: Adaptive game AI with dynamic scripting. Mach. Learn. **63**(3), 217–248 (2006)
15. Simpkins, C., Bhat, S., Isbell Jr, C., Mateas, M.: Towards adaptive programming: integrating reinforcement learning into a programming language. SIGPLAN Not. **43**, 603–614 (2008)
16. Cazenave, T.: Metarules to improve tactical go knowledge. Inf. Comput. Sci. **154**(3–4), 173–188 (2003)
17. Ulam, P., Jones, J., Goel, A.K.: Combining model-based meta-reasoning and reinforcement learning for adapting game-playing agents. In: Proceedings of AIIDE (2008)
18. Mehta, M., Ontanon, S., Ram, A.: Using meta-reasoning to improve the performance of case-based planning. In: Proceedings of the 8th International Conference on Case-Based Reasoning, pp. 210–224 (2009)