Hybrid Control for Embodied Agents Applications

Jan Miksatko and Michael Kipp

DFKI, Embodied Agents Research Group, Saarbrücken, Germany
{jan.miksatko,michael.kipp}@dfki.de

Abstract. Embodied agents can be a powerful interface for natural human-computer interaction. While graphical realism is steadily increasing, the complexity of believable behavior is still hard to create and maintain. We propose a hybrid and modular approach to modeling the agent's control, combining state charts and rule processing. This allows us to choose the most appropriate method for each of the various behavioral processes, e.g. state charts for deliberative processes and rules for reactive behaviors. Our long-term goal is to architect a framework where the overall control is split into modules and submodules employing appropriate control methods, such as state-based or rule-based technology, so that complex yet maintainable behavior can be modeled.

1 Introduction

Embodied agents (or anthropomorphic agents or avatars) can be a powerful user interface. The characters communicate with the human user using verbal and nonverbal channels (such as gaze, gestures or facial expressions), they efficiently handle turn-taking and can express their emotions and personality [1]. Virtual characters have already been applied to education, computer games, training environments, sign language communication, interactive drama, and therapy. With the rise of the 3D internet and social online platforms and games they may become a universally present device in the near future.

Virtual character applications must control all aspects of agent behavior, from low-level reactions to higher level reasoning. Low-level behaviors include gesture, gaze behavior, distance regulation or avoiding obstacles. Higher-level reasoning includes path planning, dialogue management or managing emotions. This modeling of the agent's mind has been approached in different fields like virtual characters, multi-agent systems, robotics and cognitive sciences. The approaches differ in how far they integrate a theory (cognitive architectures) or remain completely generic (multi-agent systems). Highly generic approaches leave the whole development work to developers and are unsuitable for non-experts, and theory-driven approaches may necessitate cumbersome workarounds for situations not covered in the theory. Since in our previous work we have created applications with heterogeneous control paradigms – using plan-based, rule-based and state-based approaches – we are now aiming at creating a unified framework where various technologies can be used in a complementary yet integrated fashion to allow intuitive and maintainable authoring of complex, interactive behavior. The modules we suggest for implementing different aspects of control are: extended state charts (XSC), rule-based modules and a hybrid module that combines the two former ones. Since our approach is highly motivated by pragmatic concerns, we plan to iterate through a number of interactive applications that will serve as a testbed for our framework. The first of these applications is an e-learning system since pedagogical scenarios have been shown to benefit greatly from the presence of virtual characters [2,3]. ITeach, a virtual vocabulary trainer, which we will introduce in this paper. While ITeach has specific requirements (e.g. managing linguistically rich representations and a pedagogical module) we derive a first sketch of a general architecture from it.

2 Related Work

In robotics, the dichotomy between reactive and deliberative behavior has become much more obvious early on. While Brooks' subsumption architecture [4] made a radical shift toward reactive behaviors, Gat [5] suggested a three-layered architecture to unify reactive and deliberative tasks. The three layers are the *controller* (reactive behaviors), the *deliberator* (planning), and the *sequencer* (managing the realization of actions). These architectural entities find their counterparts in our approach, however we do not restrict the developer in how many modules (e.g. multiple reactive modules) to implement and in which programming paradigm (rules, finite state, scripting).

To clarify what we mean by "different paradigms" we will briefly review our previous work in this respect. The CrossTalk [6] system was a multi-party interaction between a human user and three virtual characters. Part of the interaction was modelled with finite state machines, another part was natively plan-driven, but both parts ran seamlessly in a final plan-based system. It was an early attempt to integrate plan-based processing with FSMs. COHIBIT [7] is an edutainment exhibit for theme parks in an ambient intelligence environment. Visitors interact with two virtual characters whose dialogue is controlled by a large hierarchical finite state machine (HFSM). ERIC is an affective embodied agent for realtime commentary on a horse race [8], based on parallel rule engines for reasoning about dynamically changing events, generating natural language and emotional behavior. In IGaze [9], a semi-immersive human-avatar interaction system of an interview scenario, it was shown that finite state machines can nicely model reactive gaze behavior.

The Scenemaker authoring tool facilitates the development of embodied agents systems [10]. In Scenemaker, the control flow of the interaction is separated from content, such as speech utterances, gestures, camera motion etc. The content is organized into indexed scenes that are controlled by a scene flow, a hierarchical finite state machine, similar to state charts [11]. Parts of our approach (see Sec. 4.1) can be seen as a direct continuation and extension of the Scenemaker.

3 Use Case: The ITeach Application

Our control framework is motivated from an example e-learning application, ITeach, where an interactive vocabulary trainer presents *flash cards* with vocabulary using speech, gesture and images (Figure 1). The user interacts face-to-face with the life-size agent.

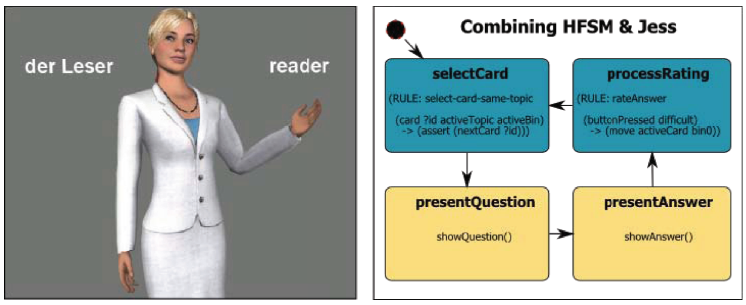
der Leser
reader
Combining HFSM & Jess
selectCard
(RULE: select-card-same-topic
(card ?id activeTopic activeBin)
-> (assert (nextCard ?id)))
processRating
(RULE: rateAnswer
(buttonPressed difficult)
-> (move activeCard bin0))
presentQuestion
showQuestion()
presentAnswer
showAnswer()

Fig. 1. The ITeach agent "Gloria" (left) and a simplified hybrid module (right)

The flash card learning system is a question/answer game where a card with an expression in a foreign language is presented by the agent and the user answers with an equivalent in his/her native language (or vice versa). The user then rates the answer with respect to how well the word was known. A simple yet effective pedagogical model now moves the cards between multiple *bins* depending on this rating and regulates the time interval between repetitions.

The ontological vocabulary representation connects linguistic information (e.g. grammar, categorical relations) with meaning. Such knowledge allows the agent to generate gestures from semantic components of the word and to answer additional user questions about, e.g., the past tense or synonyms. The agent's believability is increased by employing gaze strategies [9] and regulating the agent-user distance based on the user's location. The pedagogical module monitors the user's progress and gives feedback using verbal comments, facial expressions and gestures. The agent is equipped with an emotion model that allows to generate surprise, joy, disappointment etc. depending on user performance and the agent's personality. Furthermore, we are planning to increase the setup's instrumentation by giving the user pen and tablet for writing the words. Pen and tablet are being tracked for automatic recognition of user actions (start/finish writing) and steering the agent's gaze (looking at tablet).

In this scenario, we identified the following requirements for the control task:

- **Dialogue Management:** How to specify the overall user-agent interaction, e.g. QA dialogues, taking the past interaction history into account.
- **Interactivity:** The "regular" interaction can be interrupted at any time, (e.g. user questions), must be met by an appropriate reaction and then return to the previously interrupted sequence.

- **Domain Intelligence:** The agent has background knowledge about the topic, maintains knowledge about the world and is aware of the past actions. For instance, the agent knows that the currently presented word, an apple, is a fruit and can enumerate not yet presented fruits with similar shape.
- **Reactive Behavior:** Behaviors, that respond immediately to changes in the environment (e.g. user's position), make the user-avatar interaction a tightly coupled feedback loop and ultimately increase believability.
- **Varied and non-repetitive behavior:** The agent's behavior should be varied and non-repetitive using composition and randomization.
- **Multimodal Input Processing:** Continuously monitor several input channels such as user position, button events or pen/tablet writing, fusing the input information and reacting to it.
- **Multimodal Output Generation:** Based on the current state of the system, the agent's emotional state and user's input, generate character's gestures, facial expression, speech, camera position, gaze plus graphics and text relevant to the presented word.
- **Visual Authoring Tool:** In order to make the development accessible to non-experts (i.e. pedagogical advisors, professional authors), the authoring tool should have an intuitive visual interface.

Each requirement implies a different, best suited modeling technique. For instance, basic reactive behaviors (e.g. looking at user if user moves) are best implemented as a set of rules, whereas the overall flow of the interaction (welcome – learning – repetition – . . .) is best modeled as a state chart. Our framework allows to combine different techniques, encapsulated in modules.

4 System Framework

The ITeach system is a processing pipeline as depicted in Fig. 2. Inputs like user location, button events, hand/pen position are pre-processed by the Input Interpretation modules (e.g., translate to system coordinates) and passed through the Input Fusion Module (e.g., recognize writing begin/end events) to the relevant Control module(s). The Control Modules are divided into a deliberative and a reactive layer. The reactive layer contains gaze control, implemented as an extended state chart (XSC, cf. Sec. 4.1), and distance regulation, implemented using rules (Sec. 4.2). The deliberative layer contains the main control module, a hybrid between XSC and rules (Sec. 4.3), which models the dialogue between user and agent, manages the vocabulary ontology and the pedagogical model, and handles interrupt events such as *user asked a question* or *user pressed rating high button*. Another deliberative module is an OCC emotion model implemented in JESS [12]. It receives emotion eliciting conditions (e.g. user knows right word $\rightarrow$ good event) and outputs an emotion state (joy, disappointment, satisfaction etc.) that is used for behavior generation (smile, frown, body posture etc.). Both reactive and emotion modules can be easily reused in another application.

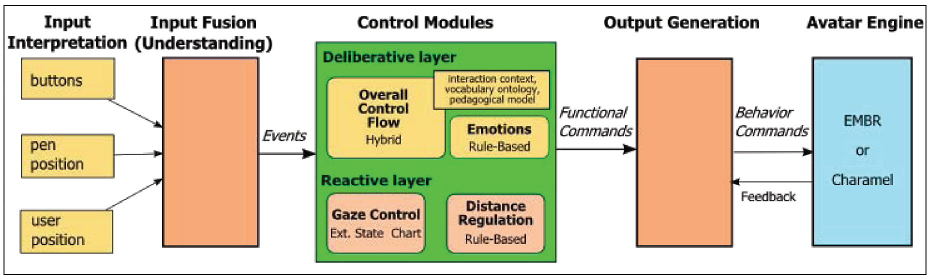

Fig. 2. Overall architecture of the ITeach system

The Output Generation module receives functional commands from the Control modules, such as *present a question*, *play a scene script* or *avert gaze* and translates them to low-level behavior commands for the Avatar engine. The *play a scene script* command executes a script [10] that resembles a movie script with dialogue utterances and stage directions for controlling, for instance, gestures or facial expressions. Scenes can be clustered into groups, where one is randomly chosen each time the group is called, to increase variety and avoid repetitive behavior. Note that the functional commands are independent from the Avatar engine so that different avatar engines can be connected. Currently, we use both a commercial engine from Charamel[1] and our own research prototype engine called EMBR (Embodied Agents Behavior Realizer) [13].

In general, the underlying framework of the ITeach application is a system that loosely integrates modules of different technologies. The modules run in parallel and communicate via messages. Each module can be implemented using a control technology described in the following sections and the framework transparently handles data conversions among them.

4.1 Extended State Chart (XSC) Module

Our *extended state charts* (XSC) extend traditional state charts [11] with transition types from the SceneMaker system[10]: probabilistic, timeout and interrupt transitions. Our extensions are fully embedded in the SCXML standard[2], an emerging W3C standard for describing Harel's state charts.

Behavior is represented by states and transitions. Actions are attached to either state or transition and executed as the graph is traversed. A state can be a superstate containing another state chart. The transitions may have an event and/or a condition attached. An incoming message triggers a transition with a satisfied condition, a matching event or with both satisfied condition and matching event. If no transition can be selected for the current state, transitions of the parent superstate will be checked. The state chart maintains a context memory containing variables scoped by the state hierarchy. These variables are

[1] http://www.charamel.de
[2] http://www.w3.org/TR/scxml/

used in the conditions and the actions. Inspired by the SceneMaker [10] we added three new transition types: *Timeout edges* model wait behavior. If mixed with standard edges, either an event arrives and/or a condition is satisfied until the timeout expires, or the timeout edge is taken. *Probabilistic edges* model random branching by randomly selecting an outgoing transition; they cannot be mixed with standard edges. *Interrupt edges* are attached to supernodes and handle interruptive events. The execution of the supernode is terminated if an event matching the interrupt edge arrives and a conditional constraint is satisfied.

In the ITeach application, the main deliberative module is modeled using an XSC. The visual nature of the XSC intuitively models the dialogue. Interrupt edges can handle unexpected user events for increased interactivity. Probabilistic edges are used to add variability. Additionally, the gaze module of ITeach is also modeled with an XSC according to [9]. Advantages of the XSC are that they are fast and easy to debug. The drawbacks are the need to model all possible variations as states, although this is alleviated with the help of superstates that wrap complex functionality and interruptive edges that are inherited by all substates.

4.2 Rule-Based Module

In a rule-based system the behavior is declaratively encoded in a knowledge base containing initial facts and condition-action rules. Our framework uses JESS (Java Expert System Shell) [14] as a rule processor. An incoming message inserted into the working memory as a fact and processed by JESS. Triggered rules can dispatch commands to the Output Processor. Rule-based systems are particularly useful for two task types. First, for encoding and using expert knowledge. The ITeach system uses JESS along with an ontology in Protégé [3] to reason about vocabulary (e.g. selecting the next word depending on the topic of the current word). Second, for behavior systems that are based on intensive if-then branching, it is easier to understand and maintain them if written in a rule-based language. This is the case for some kinds of low-level behavior such as Distance regulation module of the ITeach system. However, if the knowledge base becomes large rule-based technologies can be difficult to maintain and debug. For such cases it is better to use state charts (Section 4.1). However, in some cases one may want to use both state charts and rules in an integrated fashion (Section 4.3).

4.3 Hybrid Module

A hybrid module is an XSC module (Section 4.1) with rules written in JESS (see Fig. 1). The framework transparently updates the JESS knowledge base (KB) from/to the XSC context memory before and after executing a JESS action. The synchronization is designed in such a way that the JESS rules can reason about XSC variables and, in the other direction, the conditions and actions of the XSC can use variables that mirror JESS facts. The mapping is done as follows:

[3] http://protege.stanford.edu

XSC variable type	*JESS type*
primitive type (integer, float...)	ordered fact: `(varName value)`
boolean	simple fact: `(varName)`
struct	unordered fact with struct fields as slots: `(varName (field1 val1) (field2 val2) ..)`
set (struct type)	set of unordered facts: `(varName (field1 valA1) ..)` `(varName (field1 valB1) ..) ..`

This data synchronization preserves the state hierarchy of the XSC: the JESS facts are scoped in the same way as the XSC variables and the JESS facts are organized into modules corresponding to (i.e. named after) the (super)states of the XSC. In the ITeach system, various actions in the main deliberative module are implemented in JESS, for instance, card selection (reasoning based on card history and ontological representation of vocabulary) or user rating of cards (intensive if-then branching). The hybrid approach offers two advantages: (1) an appropriate technology for actions that involve reasoning or intensive if-then branching, and (2) runtime modifications of the actions since JESS is a scripted language.

One of the challenges is maintaining the knowledge of the hybrid model (partly represented as JESS facts, partly as XSC variables, partly synchronized) and keeping track of activated rules when a rule in state A actives rules in state B of the XSC. Making this more robust and intuitive (e.g. by visual feedback) is part of ongoing work.

5 Conclusion and Future Work

Controlling interactive embodied agents is a challenging problem requiring dialogue management, interactivity, domain intelligence and reactive behavior. Different programming paradigms are appropriate for implementing low-level reactive behavior on the one hand and high-level reasoning on the other hand. We try to console these different requirements in a hybrid and modular architecture. This means splitting the control into independent modules, each implemented using a control technology appropriate for the given task: Extended state charts (XSC) allow intuitive and visual coding of low-level reactive behaviors such as gaze. Rule-based systems can elegantly process knowledge items and represent behaviors with intensive if-then branching, in our case reactive behaviors, or input fusion/output fission procedures. We propose a third hybrid module, an XSC with rule processing useful for implementing the deliberative module, i.e. modeling the overall flow of the application, including the user-agent dialogue, along with simple reasoning and knowledge management at certain states of the interaction. The architecture and application presented in this paper is the first step in a long-term iterative development and research towards an authoring framework for embodied agents. In future work, we want to extend our control approaches by a planning module, e.g. for generating dialogue. Additionally, we will design a visual interface, that simplifies the development and debugging, and enables non-experts users to author embodied agents applications.

Acknowledgements. This research has been carried out within the framework of the Excellence Cluster Multimodal Computing and Interaction (MMCI), sponsored by the German Research Foundation (DFG). We would also like to thank to Charamel GmbH for providing us with their Avatar engine.

References

1. Vinayagamoorthy, V., Gillies, M., Steed, A., Tanguy, E., Pan, X., Loscos, C., Slater, M.: Building Expression into Virtual Characters. In: Eurographics Conference State of the Art Report, Vienna (2006)
2. Lester, J.C., Converse, S.A., Kahler, S.E., Barlow, S.T., Stone, B.A., Bhogal, R.: The persona effect: Affective impact of animated pedagogical agents. In: Proceedings of CHI 1997 Human Factors in Computing Systems, pp. 359–366. ACM Press, New York (1997)
3. Bailenson, J., Yee, N., Blascovich, J., Beall, A., Lundblad, N., Jin, M.: The use of immersive virtual reality in the learning sciences: Digital transformations of teachers, students, and social context. The Journal of the Learning Sciences 17, 102–141 (2008)
4. Brooks, R.A.: Intelligence without representation. Artificial Intelligence (47), 139–159 (1991)
5. Gat, E.: Integrating reaction and planning in a heterogeneous asynchronous architecture for mobile robot navigation. In: Proceedings of the National Conference on Artificial Intelligence (AAAI), pp. 809–815 (1992)
6. Klesen, M., Kipp, M., Gebhard, P., Rist, T.: Staging exhibitions: Methods and tools for modeling narrative structure to produce interactive performances with virtual actors. Virtual Reality. Special Issue on Storytelling in Virtual Environments 7(1), 17–29 (2003)
7. Ndiaye, A., Gebhard, P., Kipp, M., Klesen, M., Schneider, M., Wahlster, W.: Ambient intelligence in edutainment: Tangible interaction with life-likeexhibit guides. In: Maybury, M., Stock, O., Wahlster, W. (eds.) INTETAIN 2005. LNCS (LNAI), vol. 3814, pp. 104–113. Springer, Heidelberg (2005)
8. Strauss, M., Kipp, M.: Eric: A generic rule-based framework for an affective embodied commentary agent. In: Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems (2008)
9. Kipp, M., Gebhard, P.: Igaze: Studying reactive gaze behavior in semi-immersive human-avatar interactions. In: Prendinger, H., Lester, J.C., Ishizuka, M. (eds.) IVA 2008. LNCS (LNAI), vol. 5208, pp. 191–199. Springer, Heidelberg (2008)
10. Gebhard, P., Kipp, M., Klesen, M., Rist, T.: Authoring scenes for adaptive, interactive performances. In: Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 725–732 (2003)
11. Harel, D.: Statecharts: A visual formalism for complex systems. Sci. Comput. Program. 8(3), 231–274 (1987)
12. Ortony, A., Clore, G.L., Collins, A.: The Cognitive Structure of Emotions. Cambridge University Press, Cambridge (1988)
13. Heloir, A., Kipp, M.: Embr - a realtime animation engine for interactive embodied agents. In: Proceedings of the 9th International Conference on Intelligent Virtual Agents, IVA 2009 (2009)
14. Friedman-Hill, E.J.: Jess, the java expert system shell, Livermore, CA. Distributed Computing Systems, Sandia National Laboratories (2000)