Thalamus: Closing the Mind-Body Loop in Interactive Embodied Characters

Tiago Ribeiro, Marco Vala, and Ana Paiva

INESC-ID and Instituto Superior Técnico, Technical University of Lisbon
Av. Professor Cavaco Silva, 2744-016 Porto Salvo, Portugal
tiago.ribeiro@gaips.inesc-id.pt,
{marco.vala,ana.paiva}@inesc-id.pt

Abstract. We present the Thalamus framework, which is based on SAIBA and extends it by adding a perceptual loop. This perceptual loop enables embodied characters to perform continuous interaction. The framework was tested in a case study involving a NAO and an EMYS robots. After showing that our extension works, we point out some issues that were encountered during the development of the case study. We also suggest that the definition of a formal Perception Modelling Language (PML) based on the SAIBA framework can enable SAIBA-compliant embodied characters to perform continuous interaction, while still performing synchronized multimodal behavior based on BML.

Keywords: SAIBA, BML, Continuous Interaction, PML.

1 Introduction

We all imagine a future with robots living around us, behaving and interacting with humans and between themselves. But although fun to imagine, scientists have actually been struggling to create these interactive characters that can operate autonomously. Nevertheless, there have been great efforts in the community, and current research on interactive embodied characters (IEC) has been taking steps towards a unified form of behavior expression. However, for continuous and autonomous interaction, we need the character to be able to react to its environment, and trigger behaviors on that environment.

In this paper we present the Thalamus framework, which abstractly closes the loop between mind and body of an IEC. We close the loop by adding the ability to receive perceptions from the character's embodiment and send them up to the mind, in order to allow for continuous interaction, while maintaining an expressive system based on BML [9, 6].

Therefore, the two main contributions of this framework are: a) support for any kind of embodiment, virtual or robotic; 2) acting as an abstract interface to the character's sensors.

Y. Nakano et al. (Eds.): IVA 2012, LNAI 7502, pp. 189–195, 2012.

2 Related Work

Our work builds on the SAIBA framework [3, 9], shown in Figure 1. SAIBA is a representational framework for unified multimodal behavior generation. One of the cores of SAIBA is the Behavior Modelling Language (BML) [6].

Fig. 1. The three stages of behavior generation in the SAIBA framework and the two mediating languages FML and BML. [9, 6]

Several BML realizers have been developed throughout the community. Greta [4] is a virtual ECA that follows the three-level architecture of SAIBA along with BML. [5] has extended Greta's architecture by adding the ability to communicate with a NAO robot instead of the virtual character. They separate the Behavior Realizer in two sub-layers: Keyframe Generator which is common for both agents, and Animation Generator, which is specific to the embodiment.

Kipp et al. have also proposed that the Realization phase of the SAIBA framework should be separated into Realization Planning, and Presentation [2]. In their architecture, BML serves as input to the Realization Planning, just like on Greta's Keyframe Generator layer. However, the Realization Planner produces EMBRScript [1], which is an executable animation script that is sent into the Presentation module, which controls a 3D character

Smartbody [8] can also be used to control any virtual humanoid character. BML is given as input to a Behavior & Schedule Manager, which produces and runs the plan.

Elckerlyc [10] was developed as a more flexible BML realizer. A first stage parses the BML blocks and schedules them. The scheduler builds a plan that acts like a peg board. Whenever a behavior is scheduled, each sync point is solved and placed in a slot of the peg board so that sync points that should be executed at the same time are placed together. This allows for continuous interaction, because it is possible to change the plan after it has been scheduled, by modifying the placement of the syncpoints in the pegs.

We find that the current state of the art leads to being able to control characters independently of their embodiment, and to be able to continuously interact with them.

3 Thalamus Framework

The Thalamus Framework is a cross-media body interface for multiple simultaneous artificial embodied characters. It supports and is largely based on the architecture of BML. Having a framework that can act as an abstract interface to the character's sensors is especially important when dealing with robotic

characters, and follows on the proposal by [12] of having a tight feedback loop between the embodiment and the behavior planner.

We also follow the trend of dividing SAIBA's Behavior Realization level in two sub-layers, as can be seen in Figure 2. The first one is the Behavior Scheduling, which actually keeps in line with [2, 5]. The second sub-layer of our Realization level is the Body Execution.

Fig. 2. Our subdivision of SAIBA's Behavior Realization layer into the Behavior Scheduling and Body Execution sub-layers

3.1 Structure

A Character in Thalamus is composed of a mind interface and a body interface, as can be seen in Figure 3. BML blocks are sent via the mind into the character, and the character sends them for scheduling to the plan. The scheduling process solves the sync points of each behavior in order to create *ActionEvents* for it in the *Eventline*. When an *ActionEvent* is activated, it launches the execution of the corresponding behavior. This behavior will call the corresponding actions in the *BodyInterface*, with the correct parameters.

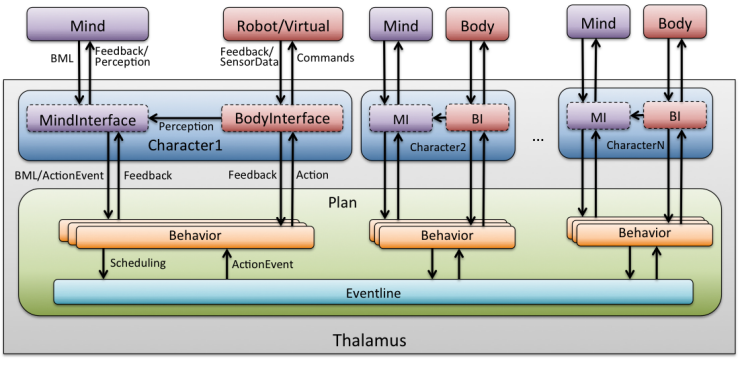

Fig. 3. The Thalamus Framework architecture

3.2 Behavior Scheduling

The scheduling process is inspired by Elckerlyc's peg-board mechanism [10], which we call *Eventline* in our architecture. The *Eventline* contains slots that relate *Events* to *ActionEvents*. An *Event* can be an absolute time instance, like *time=1*, a *SyncPoint* from BML, or any external event that is sent to the plan and that does not originate in BML.

It is important to emphasize the fact that an event is not sent when the plan executes the behavior, but only when the character acknowledges that it

has actually started. This is very important with robots, as they usually have some delay between the request for executing an action, and actually starting to execute it.

Our *Eventline* makes it possible for continuous interaction, as the behaviors are not hard-constrained on a timeline, just like in Elckerlyk [10]. Conflicts and overlaps are managed by the usual BML mechanisms.

3.3 Body Interface

There may be several different *Characters*, each with its own *BodyInterfaces*, specific to different embodiments. All *BodyInterfaces* follow the same interface, so they can implement the same set of routines, thus allowing the behaviors to call them regardless of the embodiment they represent.

However, if a behavior tries to call a routine that has not been implemented in a specific *BodyInterface*, it will report back to the plan and mind that it failed.

Body Events. Besides implementing the necessary set of routines for executing BML behaviors, the BodyInterface also supports receiving events. These events can be BML events, non BML events, or sensory events.

The BML events are used for the *BodyInterface* to notify the plan about the executed behaviors. These events are, for example, *SpeechStart*, *SpeechEnd*, *FaceLexemeStart*, *FaceLexemeEnd*, etc.. There is also a *SyncPoint* event for notifying about a specific syncpoint, which is useful for the <*Sync..*> tags in BML *Speech* nodes. This *SyncPoint* event can also be used to send specific non BML events to the *EventLine*.

The *BodyInterface* also supports perception events, originated by the embodiment's sensors. These are sent by the *BodyInterface* to the *Character* in the form of a *Perception* structure which is represented in Figure 4. The perception has an *Id*, which is a unique identifier for each perception, and a *Type*.

The currently supported perceptions were defined for our scenario. They can be of type *SoundLocated*, *SensorTouched*, or *VisionObjectDetected*. Each parameter is composed of a *Name* and a *Value*. The *Name* is a *String*, while the value can currently be an *Integer*, a *Float*, a *Boolean* or a *String*. This list can furtherly be extended.

The list of supported or required parameters depends on the type of the perception. Taking a *SoundLocated* perception as example, it can have parameters

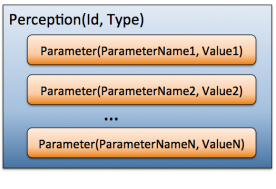

Fig. 4. The Perception structure. Each perception contains a Name, a unique Id, and a set of Parameters.

"Angle":float, and *"Intensity":float*, so that the character may know where the sound came from, and how loud it was. A *SensorTouch* perception can have parameters *"SensorName":string* and *"State":bool*, so that it may know which sensor was touched, and if it was actually touched or released (touched would mean a *True "State"*, while released would mean a *False "State"*).

Mind Events. The perceptions that are generated by the environment are sent through the *Character* into the mind. This way the mind can take appropriate action.

The mind may be a deliberative mind with intent and behavior planning, or even a simple reactive mind which can just immediately react to the events it receives from the body [11]. The only requirement on the mind is that it must also implement an interface we call *MindInterface*, which is capable of maintaining bidirectional communication with the *Character*.

Currently, the *MindInterface* supports a) receiving *Perceptions* and behavior execution *Feedback*, b) requesting the execution of pre-loaded behaviors (stored in .bml files) by their Id, c) sending BML code to the *Character*, and d) request the execution of an *ActionEvent* by the *Character*. This last feature is very interesting to support continuous interaction and interruption of behavior execution, as it makes it possible for the mind to interfere on the scheduled plan.

4 Case study: The Path of NAO

To test our framework, we created a scenario[1] in which two completely different robots interact with each other by running a set of BML scripts, while also interacting with the environment through their sensors. The robots used are a NAO robot[2] and an EMYS robot [7].

The EMYS robot is a robotic head, that can speak, gaze and perform facial animations. It currently has only a sound location sensor that is accomplished by a Microsoft Xbox Kinect™. It will perform *Speech* and *Face-Lexeme* behaviors. The perception its mind will react to is *SoundLocated*.

The NAO robot is a humanoid robot that can walk and perform body animations. It also has lots of sensors that can be used to interact with the environment. It will perform *Locomotion*, *Speech* and *Posture-Pose-Lexeme* behaviors. The perceptions its mind will react to are *VisionObjectDetected* and *SensorTouched*.

Beause we have not implemented a Behavior Planner that could generate BML for this scenario, we have previously written it as BML blocks which are preloaded and scheduled into the plan. Each of these characters has a very simple reactive mind, which, on reception of each perception, send an *ActionEvent* back through its *Character* and into the plan. This ActionEvent can interrupt the current behaviors, and eventually trigger new ones.

[1] The full scenario is shown in the video that accompanies this paper.
[2] `www.aldebaran-robotics.com`

4.1 Discussion

We found our framework to be capable of executing the aforementioned scenario. However, there were some issues that we noted and are worth mentioning.

On the NAO robot's side, we found some complications in having accurate and responsive control both over the robot's actions and sensors. Sometimes the robot raises an internal event stating that the animation has started when in fact it hasn't. That triggers a *start SyncPoint* in the plan, which in turn, triggers a speech that should start synchronized with the animation, but that actually starts before it.

It is clear that robotic animation systems may have these kind of flaws. We thus suggest robotic control system developers to work more closely with users and high-level developers, by looking at this kind of needs.

Taking another example, EMYS' control system was developed by us, and therefore, matches our needs in a higher level. The result is accurate control over it's behavior: when we play or stop and animation, it responds immediately, and sends accurate events back to the character's body interface.

As to using sensors to trigger BML behaviors, we sometimes encountered false positives, both due to noise in the sensor's circuitry and also due to misintepretation of perceptual data. On sound location sensors, for example, echoes might introduce noise in the readings. NAO's head touch sensor however, actually suffers from noise, and frequently triggers events without having been touched.

We therefore had to filter the perceptions in the mind level, before reacting to them.

However, in this kind of framework, it would be useful to be able to filter out false perceptions before they reach our character. After having solved the sensor-imperfection related problems, the scenario ran correctly, except for ocasional delays on NAO's motion response.

5 Conclusions and Future Work

We have developed the Thalamus framework, which is based on the SAIBA framework, but extends it in order to support a perceptional loop for virtual or robotic embodied characters. The perceptional loop can interact with the BML behavior loop, in order to provide continuous interaction based on the SAIBA framework. In order to process the perceptual data in our framework, we have abstracted the data from the sensors into a generic perceptual structure. This perceptual structure was shown to be adequate for the perceptual loop, however it currently lacks formal specification.

We conclude by suggesting the definition of a formal Perception Modelling Language (PML) which meets our extended SAIBA architecture and can interact with BML. Our results show that a widespread specification of PML may provide current embodied characters with generalized continuous interaction capabilities, thus closing the interactive loop.

Acknowledgements. This work was supported by the EU FP7 ICT-215554 project LIREC (LIving with Robots and intEractive Companions). This research was supported by EU 7th Framework Program (FP7/2007-2013) under grant agreement no. 215554, FCT (INESC-ID multiannual funding) through the PIDDAC Program funds.

References

[1] Heloir, A., Kipp, M.: EMBR: A realtime animation engine for interactive embodied agents. In: 2009 3rd International Conference on Affective Computing and Intelligent Interaction and Workshops, vol. 1, pp. 1–2 (September 2009), http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5349524

[2] Kipp, M., Heloir, A., Schr, M.: Realizing Multimodal Behavior: Closing the gap between behavior planning and embodied agent presentation. Framework (2010)

[3] Kopp, S., Krenn, B., Marsella, S., Marshall, A.N.: Towards a Common Framework for Multimodal Generation: The Behavior Markup Language. Information Sciences (2006)

[4] Mancini, M., Niewiadomski, R., Bevacqua, E., Pelachaud, C.: Greta: a SAIBA compliant ECA system. Language (2008)

[5] Niewiadomski, R., Obaid, M., Bevacqua, E., Looser, J., Le, Q.A., Pelachaud, C.: Cross-media agent platform 1(212), 11–20 (2011)

[6] Reidsma, D., Welbergen, H.V.: BML 1.0 Standard, http://www.mindmakers.org/projects/bml-1-0/wiki/Wiki

[7] Ribeiro, T., Paiva, A.: The Illusion of Robotic Life Principles and Practices of Animation for Robots. In: HRI 2012, vol. 1937 (2012)

[8] Thiebaux, M., Rey, M., Marshall, A.N., Marsella, S., Kallmann, M.: SmartBody: Behavior Realization for Embodied Conversational Agents. Information Sciences (Aamas), 12–16 (2008)

[9] Vilhjálmsson, H., Cantelmo, N., Cassell, J., Chafai, N.E., Kipp, M., Kopp, S., Mancini, M., Marsella, S., Marshall, A.N., Pelachaud, C., Ruttkay, Z., Thórisson, K.R., van Welbergen, H., van der Werf, R.J.: The Behavior Markup Language: Recent Developments and Challenges. In: Pelachaud, C., Martin, J.-C., André, E., Chollet, G., Karpouzis, K., Pelé, D. (eds.) IVA 2007. LNCS (LNAI), vol. 4722, pp. 99–111. Springer, Heidelberg (2007)

[10] Welbergen, H., Reidsma, D., Ruttkay, Z.M., Zwiers, J.: Elckerlyc. Journal on Multimodal User Interfaces 3(4), 271–284 (2010), http://www.springerlink.com/index/10.1007/s12193-010-0051-3

[11] Wooldridge, M.: An Introduction to MultiAgent Systems. John Wiley and Sons (2002)

[12] Zwiers, J., van Welbergen, H., Reidsma, D.: Continuous Interaction within the SAIBA Framework. In: Vilhjálmsson, H.H., Kopp, S., Marsella, S., Thórisson, K.R. (eds.) IVA 2011. LNCS, vol. 6895, pp. 324–330. Springer, Heidelberg (2011)