An ASAP Realizer-Unity3D Bridge for Virtual and Mixed Reality Applications

Jan Kolkmeier ✉, Merijn Bruijnes, Dennis Reidsma, and Dirk Heylen

Human Media Interaction, University of Twente, The Netherlands
`[j.kolkmeier; m.bruijnes; d.reidsma; d.k.j.heylen]@utwente.nl`

Abstract. Modern game engines such as Unity make prototyping and developing experiences for virtual and mixed reality contexts increasingly accessible and efficient, and their value has long been recognized by the scientific community as well. However, these game engines do not have the capabilities to control virtual embodied characters, situated in such environments, with the same expressiveness, flexibility, and generalizability, as offered by modern BML realizers that generate synchronized multimodal behavior from Behavior Markup Language (BML). We implemented a Unity embodiment bridge to the Articulated Social Agents Platform (ASAP) to combine the benefits of a modern game engine and a modern BML realizer. The challenges and solutions we report can help others integrate other game engines with BML realizers, and we end with a glimpse at future challenges and features of our implementation.

Keywords: BML, SAIBA, Virtual Agents, Unity, Virtual Reality, Mixed Reality

1 Introduction

Natural multimodal behavior for an embodied conversational agent requires producing, at the right moment, utterances combined with appropriate non-verbal behaviors and the capability to 'color' these behaviors to adapt to the social situation [1]. In a SAIBA system [3], an *Intent Planner* produces intents composed of speech acts and communicative functions that are translated into expressive multimodal behavior by a *Behavior Planner*. In the Articulated Social Agent Platform (ASAP [4]), *embodiment* modules are tasked with the timely realization of planned behavior. This modularization allows control of both virtual and robotic embodied agents, as well as control of agents with embodiments situated in multiple environments, such as a robotic platform with a virtual head on a screen [8]. Furthermore, ASAP specializes in on-the-fly adaptation of the timing of the realized behavior, for mutually coordinated fluent interaction with an interlocutor [7].

In the following we present our work towards an integration of the ASAP platform and the popular Unity[1] game engine and editor through a *Unity embodiment*[2]. Although integrations with other 3D engines already exist, the Unity game engine is an attractive

[1] Developed by Unity Technologies: `https://unity3d.com`
[2] All software discussed in this work is publicly available at this repository: `https://github.com/hmi-utwente/AsapUnityBridge`

J. Beskow et al. (Eds.): IVA 2017, LNAI 10498, pp. 227-230, 2017.
DOI 10.1007/978-3-319-67401-8_27

addition to this list. It is becoming a popular tool for creating 3D and Virtual Reality environments also in scientific circles and among students, making collaboration and reproduction easier. What is more, Unity is currently one of the first engines offering API support for consumer and scientific grade MR and VR products, thus reducing the effort needed to incorporate these new technologies into existing systems. It further supports importing existing assets in various formats and has a wide range of assets – including virtual characters – available through the asset store, allowing for fast prototyping of virtual environments.

While Unity also provides a powerful animation engine, it does not aim to satisfy the same requirements that a BML realization engine does, such as multimodal behavior planning and synchronization and lip-sync. BML realizers provide an important missing set of functions not present in Unity or standard game engines. The Virtual Human Toolkit (VHTK) [2] already comes with a Unity integration of another popular BML realizer: SmartBody [9]. While this is a good option, it is always beneficial to have more choices, as there are strengths and weaknesses to different BML realizers. As mentioned above, one of the strengths of ASAP lies in its flexibility when looking to control agents with heterogeneous embodiments. This is useful, for example, in mixed reality settings, where agents and their modalities are situated in both the virtual and physical world.

2 Related work

ASAP provides a collection of software modules for building social robots and virtual humans [10]. SAIBA makes a distinction between the communicative intent and behavior descriptions of the Embodied Conversational Agent (ECA) with XML based interfaces, Functional Markup Language (FML) and Behavior Markup Language (BML) respectively [3]. ASAP is a SAIBA compliant, OS independent behavior realizer, enhanced with two features that allow for fast and fluent virtual human behavior: a close bi-directional coordination between input processing and output generation, and incremental processing of both input and output. The default virtual embodiment in ASAP is for the Armandia ECA. It is implemented as a Java OpenGL application that is controlled by directly binding to the ASAP platform code. Implementation of new embodiments is described in [8]. ASAP has been used not only for virtual humans, but also for controlling social robots, such the Zeno R25[3] in the EASEL project [6].

The Greta platform realizer supports controlling both virtual and robotics agents through MPEG4 BAP-FAP *players* [5]. The default player for virtual agents uses the Ogre engine[4]. There is mention of an experimental Greta branch[5] with Unity integration using thrift[6].

The VHTK [2] uses SmartBody [9] as BML realization engine. To our knowledge, there is no work on controlling heterogeneous or robotic agent platforms with Smart-

[3] hansonrobotics.com/robot/zeno, former Robokind

[4] ogre3d.org

[5] https://trac.telecom-paristech.fr/trac/project/greta/wiki/GretaUnity

[6] https://thrift.apache.org

Body. Written in C++, it can be compiled and thus included natively in various platforms (including Unity).

3 Implementation Summary

Several things are needed to integrate a BML realizer (ASAP) with a new embodiment (Unity). We follow the overall steps to extend ASAP with a new embodiment as described in [8]. First, ASAP needs the capability to control the Unity embodiment for animation of posture, gesture and facial expressions. This requires an implementation of the face and skeleton embodiment interface in ASAP, whereas in Unity the character needs to be configured to expose the right animation controls to be able to execute the requested behavior. Communication between the two processes (Unity and ASAP) is implemented with a generic middleware implementation, allowing for a distributed setup. ASAP needs to have access to information about the Unity character that is being controlled, such as the composition of the skeleton and the face. We refer to this as the *AgentSpec*. The AgentSpec is requested by ASAP upon initialization of the Unity embodiment loader. Now, when ASAP receives a BML to be realized, it can compute both skeleton and face data internally. Resulting bone transforms and morph targets are shared through *AgentState* messages with Unity at a desired frame rate.

To realize lip-sync, mouth shapes need to be defined in a *facebinding* using the facial controls that the Unity character exposes.

ASAP is capable of using Inverse Kinematics (IK) for pointing or directed gaze. However, to do this the locations of the relevant objects in the Unity environment need to be made available to ASAP. This is done by the implementation of the WorldObjectEnvironment interface.

As part of this work, we further provide a Unity editor extension to create forward kinematic animations that can be exported to BML keyframe animations that can be used in the ASAP gesture binding. These animations support procedural changes for multimodal synchronization using named sync points.

4 Conclusion and Future work

The integration we present makes it possible to drive characters in Unity with the ASAP BML realizer. The implementation allows re-use of most existing agent behaviors that were defined in BML in earlier work. This includes automatically generated BML through SAIBA-compliant agent architectures. Through the distributed implementation, it is possible to run an embodied conversational agent on a system with limited computing power such as a tablet. Additionally, mixed-modality agents, such as a robot with a virtual face on a screen, are supported by ASAP. The Unity integration with ASAP, combined with novel AR technology such as the HoloLens, offers the exciting possibility to place persistent virtual conversational agents throughout the persistent mixed reality world that the HoloLens offers.

The possibility to export BML from the Unity animation editor is a promising approach for quick and flexible creation of reusable animations and postures.

Some challenges remain, for instance, TTS engine output uses the default audio system, and is not streamed through a virtual audio source in Unity's 3D environment. Thus, the 3D localization of the sound of the agent's speech in virtual space is not possible. This is important for making the experience immersive in VR settings. An exciting next step is the integration of mo-cap systems to allow for agents mirroring posture or behaviors of the agent. Some of the challenges that we faced in the implementation of the Unity embodiment, such as animation re-targeting and the generation of BML in the Unity Animation editor contribute in making this possible in the future.

Acknowledgements
This work has been supported by the following projects: EU-projects ARIA-VALUSPA-H2020, No 645378; EASEL-FP7, No 611971; And the Dutch projects COMMIT/R3D3 and Project Avatar.

References

1. Bruijnes, M.: Believable suspect agents: response and interpersonal style selection for an artificial suspect. Ph.D. thesis, University of Twente, Enschede (October 2016), http://doc.utwente.nl/101371/, sIKS dissertation series no. 2016-39
2. Hartholt, A., Traum, D., Marsella, S.C., Shapiro, A., Stratou, G., Leuski, A., Morency, L.P., Gratch, J.: All Together Now: Introducing the Virtual Human Toolkit. In: 13th International Conference on Intelligent Virtual Agents. Edinburgh, UK (Aug 2013), http://ict.usc.edu/pubs/All%20Together%20Now.pdf
3. Kopp, S., Krenn, B., Marsella, S., Marshall, A.N., Pelachaud, C., Pirker, H., Thórisson, K.R., Vilhjálmsson, H.H.: Towards a common framework for multimodal generation: the behavior markup language. In: Proceedings of the 6th international conference on Intelligent Virtual Agents. pp. 205–217. Springer-Verlag, Berlin, Heidelberg (2006)
4. Kopp, S., van Welbergen, H., Yaghoubzadeh, R., Buschmeier, H.: An architecture for fluid real-time conversational agents: integrating incremental output generation and input processing. Journal on Multimodal User Interfaces 8(1), 97–108 (2014)
5. Niewiadomski, R., Bevacqua, E., Mancini, M., Pelachaud, C.: Greta: an interactive expressive eca system. In: Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 2. pp. 1399–1400. International Foundation for Autonomous Agents and Multiagent Systems (2009)
6. Reidsma, D., Charisi, V., Davison, D., Wijnen, F., van der Meij, J., Evers, V., Cameron, D., Fernando, S., Moore, R., Prescott, T., et al.: The EASEL project: Towards educational human-robot symbiotic interaction. In: Conference on Biomimetic and Biohybrid Systems. pp. 297–306. Springer (2016)
7. Reidsma, D., Van Welbergen, H.: Multimodal plan representation for adaptable bml scheduling. In: In: Autonomous Agents and Multi-Agent Systems. Citeseer (2013)
8. Reidsma, D., van Welbergen, H.: AsapRealizer in practice–a modular and extensible architecture for a bml realizer. Entertainment computing 4(3), 157–169 (2013)
9. Shapiro, A.: Building a character animation system. In: International Conference on Motion in Games. pp. 98–109. Springer (2011)
10. Van Welbergen, H., Yaghoubzadeh, R., Kopp, S.: AsapRealizer 2.0: the next steps in fluent behavior realization for ECAs. In: International Conference on Intelligent Virtual Agents. pp. 449–462. Springer (2014)