A Novel Unity-based Realizer for the Realization of Conversational Behavior on Embodied Conversational Agents

IZIDOR MLAKAR[1], ZDRAVKO KAČIČ[1], MATEJ BORKO[2], MATEJ ROJC[1]
[1]Faculty of Electrical Engineering and Computer Science, University of Maribor
[2]A1 Slovenija
SLOVENIA
izidor.mlakar@um.si, kacic@um.si, Matej.Borko@A1.si, matej.rojc@um.si

Abstract: - Embodied conversational agents are virtual entities that tend to imitate as many features of face-face dialogs as possible. In order to achieve this goal, the ability to reproduce synchronized verbal and co-verbal signals coupled into conversational behavior becomes essential. Further, signals such as social cues, attitude (emotions), personality, eye-blinks, and spontaneous head movement are equally important. Modern 3D environments and 3D modeling tools, such as: Maya, Daz3D, Blender, Panda3D and Unity have opened up a completely new possibilities to design virtual entities, which appear almost (if no completely) like real-life persons. However, the modern 3D technology is not designed to handle highly dynamic and interchangeable contexts such as human interaction. Therefore, mostly animations are prepared in advance and support limited diversity as well as limited capacity to adapt to a new set of parameters. In this paper EVA realizer engine, which is a part of proprietary behavior realization components of EVA Framework, is presented. The represented engine is based on Unity game engine. EVA realizer exploits benefits of modern game engines as well as extend them with requirements of co-verbal realizers, by providing interpreter and manager for dynamic and in real-time generated animation. The animation is created and modeled by proprietary external co-verbal behavior generator component.

Key-Words: - embodied conversational agents, co-verbal realizers, animation, virtual reality, mixed reality, multimodal interaction

1 Introduction

One of the key challenges in the modern human-machine interaction (HMI) design, is generation of more natural multimodal output. In the last few years we can observe that conversation is becoming a key model in human machine interaction [1]. Thus, conversational agents (CA) are increasingly important in everyday scenarios. Namely, Apple and Microsoft, Amazon, Google, Facebook etc. have adapted their own variations of CAs. The CAs range from chat-bots and 2D, carton-like implementations of talking heads [2,3,4,5], to fully articulated embodied conversational agents (ECA's) performing interaction in various concepts [6,7,8].

Embodied Conversational Agents (ECAs) represent nowadays most natural human-machine interfaces. Further, studies in the field of face-to-face conversations show that the most natural way to implement interaction is through synchronized verbal and co-verbal signals (gestures and expressions) [9]. The co-verbal behavior represents a major source of discourse cohesion. It regulates communicative relationships and may support or even replace corresponding verbal counterparts [10]. Thus, the co-verbal behavior effectively retains semantics of the information, and gives a certain degree of clarity in the discourse. It clarifies the collocutor's communicative goal and reflects psycho-social nature of given information through social and psychological responses, attitudes, and personality. In this way, the ECAs have the capacity to resemble and foster the natural way of reacting and interacting. However, the natural multimodal interaction entails multiple behavior variations that are correlated in a dynamic and highly unpredictable settings [11]. Furthermore, natural conversational behavior incorporates various social and interpersonal signals in order

to 'color' the final outcome. Thus, the virtual entity must look and behave like human. It must also have the capacity to dynamically adapt itself to social and other situational contexts [1, 12]. As a result, the design of human-like ECA represents a complex and a difficult task.

Game engines, such as: Unity 3D[1], Panda 3D[2], Irrlicht Engine[3], Ogre3D[4], are becoming a must-have tool for developing 3D and Virtual Reality environments also in the field of embodied conversational agents. In combination with various 3D modeling tools (e.g. Maya, Daz3D, Blender), the production of highly realistic humans is much easier and affordable. However, if some virtual character tends to imitate a real human, it must also incorporate body motion and behavior that are synchronized with various contexts and aspects of everyday life. In case of embodied conversational agents, the *intent planner* and *behavior generator* usually serve for producing conversational behavior that is composed of speech acts, communicative intents, and non-verbal signals (gestures and expressions), and synchronized into expressive reaction. The task of the *behaviour realizer* is then to realize the behaviour on some targeted virtual entity [13]. The game engines and 3D modelling tools provide a perfect environment for design and deployment of realistic virtual entities and their highly realistic animation. However, they in general fail to satisfy several parameters of believability of conversational behaviour, such as diversity and multimodal planning, situational awareness, synthesis of verbal content, synchronization, etc. Thus, the integration of *behaviour planners* and *behaviour generators* (e.g. behaviour specification tools) is only natural. In this way both sets of tools covers the weak points of each other.

In this paper we present an EVA framework for rapid design and development of embodied conversational agents capable of realizing conversational behaviour. We follow the modular concept of separating *behaviour specification* and *realization* into two independent processes, while EVAScript acts as interlink between the two [14]. EVA *behaviour generator* specifies the behaviour, while EVA *realizer* animates it [15]. This paper presents our latest effort towards adaptation of a modern widely used Unity 3D game engine, and its integration into novel EVA Realizer. In contrast to previously used Panda 3D engine, by using Unity we can achieve far greater realism of the environment and virtual characters, as well as establish far more controllable and more importantly, truly event oriented virtual environment for representation of more natural machine-generated responses. However, in terms of procedural animation and integration with EVA concepts, the native mechanism of animation and control in Unity were less compatible as Panda 3D, thus significant effort in the adaptation of the animation engine's was required, thus a new realizer was designed.

2 Related works

The conversational characters are becoming increasingly more realistic and human like. However, human eye is able to detect every detail, and even a small discrepancy in outlook or movement may appear too unnatural and too synthetic. Thus, the specification and animation of conversational behaviour are equally important. In the field of conversational agents, the animation part and handling of the virtual environment is usually left to the realizer component. The most spread are various BML realizers. Among them, Unity realizer has been chosen as the preferred animation engine.

For instance, Virtual Human Toolkit (VHTK) [16] uses SmartBody [17] as the underlying realization engine and animation engine. Greta [18] supports the realization of expressive conversational behaviour based on MPEG4 BAP-FAP layers. For the animation, the platform facilitates Ogre[5] game engine. There is also an ongoing effort to deploy Greta over

[1] https://unity3d.com/

[2] https://www.panda3d.org/

[3] http://irrlicht.sourceforge.net/

[4] http://www.ogre3d.org/

[5] Ogre3D: http://www.ogre3d.org/

Unity game engine[6]. Elckerlyc [19] is another BML realizer for generating multimodal verbal and nonverbal behaviour for Virtual Humans (VHs). It is a model-based platform for the specification and animation of synchronized multi-modal responsive animated agents. EMBR [20] is also a BML realizer that offers a high degree of animation control via the EMBRScript language as interlink between behaviour generator and the animation engine. EMBR uses Panda 3D[2] as the underlying animation engine. ASAP realizer-Unity3D Bridge [21] represents a BML realizer facilitating Unity game engine as the animation and rendering engine. The system combines the benefits of a modern game engine and a modern BML realizer.

In EVA Realizer, we tried to fuse benefits of several modern game engines with behaviour generation engines, and the EVA Behaviour generator [15] in particular. In this way, the EVA realizer is capable of animating high-quality conversational behaviour consisted of gestures, facial expressions, lip-sync, gaze and head movement, and posture. Further, the realizer facilitates procedural animation delivered via EVAScript acts, synthesis of subconscious behaviour in form of spontaneous and expressive eye and head movement, as well as targeted animation in form of LookAt, PointAt and Follow commands. The first version of the realizer presented in [22] had been implemented in Panda 3D environment[2], while the novel EVA realizer, presented in this paper, is developed in Unity 3D[1]. Both realizers support the realization of the for-mentioned conversational concepts.

To sum up, in this paper we represent novel behaviour realization engine with Unity-based EVA realizer. Architecturally, both realizers are similar in design. The major advantages of the Unity-based implementation over the Panda 3D– based implementation are the following:

- Integrated animation scheduling with frame by frame control and manipulation of all concurrent and planned animations. Thus, this represents truly supporting dynamic and event oriented environment. Panda 3D based realizer implemented stop and interrupt procedures to handle new events, while the scheduling was implanted externally and quite rudimentary in form of FIFO ques holding the pre-processed animation sequences to be realized.
- Inverse kinematics (IK) for implementation of targeted animation (e.g. *LookAtObject* and *FollowObject* behaviour). Panda 3D based realizer partially supported IK via external libs and implanted *LookAt* and *Follow* behaviours via transformation of 3D position to the rotations of neck joint and arm joint chain (online estimation).
- Specification of customized interpolation curves for more natural smoothing of the animation.
- Integrated scene and animation editor. It has the capacity to visually set-up environment parameters as well as to edit agent's posture and gestures in form of EVA Templates. Nevertheless, 3D modelling tools, such as: Maya 3D, Blender and Daz 3D can also be used.
- Deployment of ECAs to various mobile and stationary platforms. Thus, Unity brings improved support for various communicative context in various environments. Panda 3D based solution only partially supported this option via web clients.

3 The EVA Framework Architecture

The general concepts of EVA framework architecture are in line with related research in the field. Thus, the principal of SAIBA architecture is followed [23]. The architecture of EVA framework is outlined in Fig. 1.

[6] Greta in Unity: https://trac.telecom-paristech.fr/trac/project/greta/wiki/GretaUnity

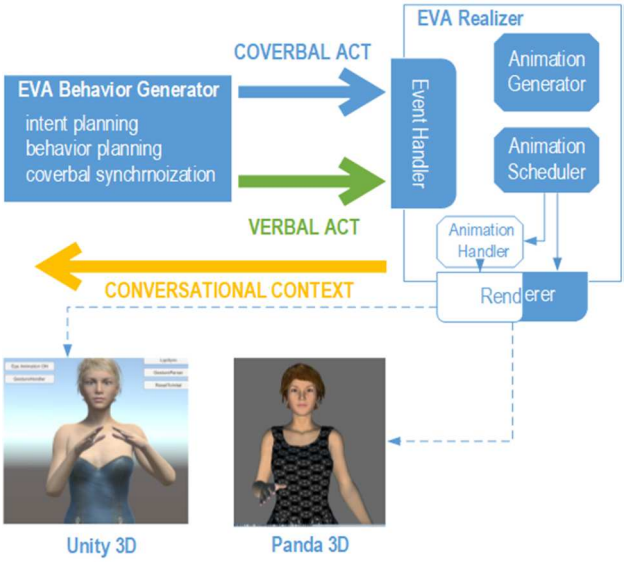

Figure 1: An architecture of the EVA framework.

As outlined in Fig. 1, the external engine is used for generating conversational behavior in form of synchronized co-verbal (e.g. gestures and expressions) and verbal acts (e.g. speech). In EVA architecture the engine is named *EVA behavior generator* [15]. This is actually an omni-comprehensive TTS engine, which generates the synchronized conversational behavior in form of conversational events. Each event consists of non-verbal events described in EVA Script, and aligned with the synthesized verbal counterparts. The end component of the presented architecture, which actually realizes the behavior and represents it to the user, is EVA realizer. When EVA realizer receives a conversational event, it transforms it into its physical representation. This is achieved by controlling the embodiment of the virtual character (its movement controllers). After completion, the realizer sends feedback in form of a conversational context.

As part of EVA framework we have designed and deployed two EVA realizers, one is based on Panda 3D game engine, and the novel one implemented in Unity 3D environment. Regardless of the type, the main components of both EVA realizers are: *Event Handler*, *Animation Generator*, *Animation Scheduler* and *Animation Handler*. *Event Handler* component intercepts and handles conversational events. It parses the stream of events, and checks for the type of the event and its priority. The *Animation Generator* then transforms the conversational behavior into three separate animation streams: lip-sync, facial expression, and gestures/gaze/postures. It actually transforms EVAScript descriptions into sequences of configurations of agent's embodiment implemented over designated time intervals. Furthermore, the *Animation Generator* also deploys final temporal and spatial constraints, which are adjusted to agents articulated body and prevent unnatural movements.

The animation sequences generated by the *Animation Generator* are on the other hand engine specific. Thus, they are specifically designed and optimized for Panda 3D realizer, and for Unity 3D realizer. As a result, the Unity 3D realizer implements another component called *Animation Handler*, which handles frame-by-frame operations. In contrast, the Panda 3D based-one, handles frame-by-frame operations internally via its native renderer. Finally, the *Animation Scheduler* generates an execution plan for the generated animation sequences, and deploys them to the *Render* engine (or *Animation Handler*) accordingly. The *Render* renders the animation into the user interface.

After the realization of each animation sequence is completed, the *Event Handler* signals its status (conversational context) to the *Behavior Generator* and *Dialog Handlers*. After full realization of the behavior que, the *Event Handler* triggers generation of inactive behavior. When some conversational event is received, while another event is still being processed and executed, the *Animation Handler* stores it into the que handled by the *Animation Scheduler* (if the event is regular). And when some event should override current behavior, the *Event Handler* triggers the replacement of the que with the new behavior. The animation step currently being rendered is completed and fully realized. Steps after rendering, however, will get updated according to the new behavior. Finally, if some event has to interrupt currently displayed behavior, the *Event Handler* triggers the interrupt procedure. This procedure is also engine specific.

In Panda 3D based realizer, the frame –by-frame operations are handled internally by the

renderer. Thus, the *Animation Scheduler* only feeds renderer with the targeted transformation (e.g. end pose), interpolation and time interval for animation sequence as a whole. Frame-by-frame calculations are then implemented automatically and internally by the renderer. This means that each animation sequence can be controlled only on the 'step level'. Once the step had been fed to the renderer, an abrupt stop is the only possibility to immediately change/stop the animation. For a smooth continuation, therefore, the step fed to the realizer has to play out to the end. On the other hand, in Unity based realizer we are able to handle frame-by-frame operations outside the render. The *Animation Handler* calculates the transformation for each frame separately, and then feeds the frame-to-frame transformations to the renderer. This means that the animation can be modified at any stage, even, during the execution of a step. Thus, for a smooth transition the scheduler does not have to wait and adjust its temporal scheduling. It just has to adjust its frame-by-frame schedule and replace it with new configurations. It can actually instantiate changes instantly as they occur. It can also insert new behavior in between configurations, etc. As a result, the virtual character becomes more responsive and can react to contextual, system and environmental changes instantly. The agent also 'remembers' what it was gesturing prior to the excited state. It can continue with the realization of that behavior after an excited state dissipates. In the next chapter we will address the novel Unity based realizer and its implementation in more detail.

4 A novel Unity based realizer

In order to integrate and fully facilitate all aspects of conversational behavior supported by other components in EVA framework and Panda 3D based Realizer in particular, we follow the overall approach presented in [14]. Firstly, we need support for the body-part based animation generated online, and supporting behavior expressed in form of gestures and gaze, facial expressions and lip-sync. This requires an implementation of a proper articulated agent and its embodiment to be exposed in Unity. For the design and the implementation of the agent, the Unity directly support most of the standard 3D animation formats from *obj, collada*, and *fbx* to directly import Maya scene files (*.mb*). The implementation of EVA Realizer's architecture in Unity (Fig. 1) is outlined in Fig. 2.

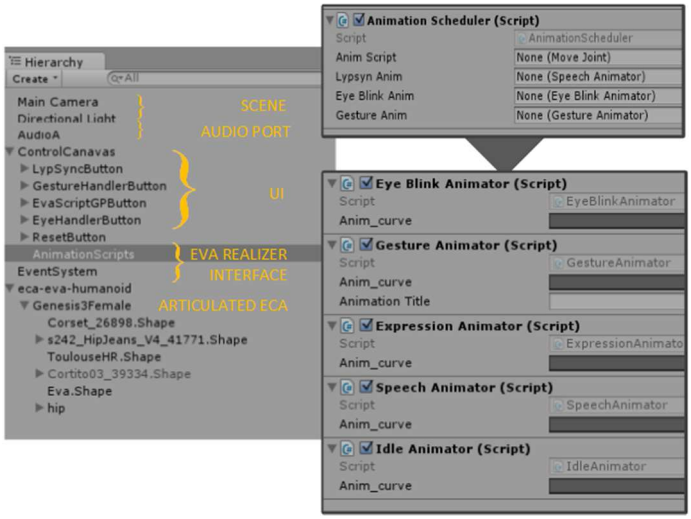

Figure 2: Implementation of the EVA Realizer architecture in Unity

In order to adapt to the EVA Architecture, and implement the required functionalities of an expressive ECA in Unity, as defined through EVA Framework, we decided to use C# as programming language. Fig. 2 outlines those components that are implemented as scene objects, and can interact with the scene and all objects in it directly. These are either actual animation objects (such as: lights, objects in environment, cameras, interpolation curves, and articulated virtual characters, etc.), or C# scripts extending the native *MonoBehaviour* type. *MonoBehaviour* is the base class of all scripts that can be attached to 'game' objects, and manipulate objects in the scene. Firstly, the *Event Handler* is represented via *EventSystem* object, which extends the engines' native event handling framework by incorporating EVA conversational events and publish-subscribe (unsubscribe) mechanism for handling of particular animation streams. Thus, the *EventSystem* object represents the link between EVA *Behavior Generator* with the virtual 3D scene, and all exposed objects within it

(including the ECA). The *EventSystem* also posts requests to the *Animation Scheduler*. It is implemented as a bridge between the context of the realizer and all outside contexts. As such, it is actually the only component that controls the scheduler and exposes its selected particulates to the outside world. The *Animation Scheduler* is then represented by the *AnimationScheduler* object. The *AnimationScheduler* is used in order to interpret EVA conversational events, and to transform them into responses represented by the articulated agent. These responses are animations, and sequences of configurations of the movement controllers of agent's embodiment (e.g. joints and blend shapes).

The *AnimationScheduler* realizes these events in form of three independent animation streams. One stream is generated for speech, one for facial expressions, and one for head and arm gestures. Each stream handles one group (region) in conversational motion. In the context of EVA events these groups are defined via <speech>, <fgesture> and <bgesture> XML tags. The *AnimationScheduler* animates these streams via three independent objects, all of *AnimationHandler* type. Each *AnimationHandler* represents the link between the rendering engine and the objects in the scene, including the articulated embodiment. The speech is animated via *Speech Animator*, the facial expressions are then animated via *Expression Animator,* while gestures, including posture and head movement (with gaze), are handled via *Gesture Animator*. We have also added two additional *Animator handlers*, which handle subconscious/idle behavior. These are *Eye Blink Animator* (for spontaneous expressive eye-blinks), and *Idle Animator* (for non-verbal movements during idle time). In addition to rendering, each *AnimationHandler* is interconnected with a proper *Animation Generator*, capable to parse EVAScript descriptions into configurations of embodiment, and/or automatically generating configurations for behavior, such as: eye-blinks, subconscious head movement, Follow/LookAt object, etc.

4.1 Event-Oriented Model of Realizer

The communication between various processes of the *Realizer* is implanted via event-oriented publish/subscribe model. The *Realizer* has the information about the virtual character (or characters) that is being controlled, such as: the composition of the skeleton and the face, and the initial/rest configurations of the embodiment. The configuration also stores all temporal and spatial constraints of the articulated body. These are loaded into *Realizer*'s environment upon initialization. Upon initialization, the *Animation Scheduler* registers itself into the event system along with available animation handlers.

When the *Realizer* intercepts a conversational event, it firstly checks its type and priority. Afterwards, the realizer publishes it to the *Animation Scheduler.* After publishing the event, realizer instantiates a new event listener. Through it, the realizer subscribes to events generated by the system. In this way it gains information regarding the internal status of the available conversational context and the state of its components. When the *Animation Scheduler* receives the conversational event, it initiates internal interpreter in order to segment the behavior into the three before mentioned animation streams. The interpreter transforms the EVA script behavior into a body-part segmented schedule of parallel/consecutively executed behavior in form of animation streams. At the same time the *Scheduler* smoothly stop any idle behavior, destroy its handlers, and move to the rest pose.

For each animation stream, the handler initiates a dedicated animation handler, based on the type of behavior. After the animation handling processes are initialized, the scheduler subscribes to the messaging systems of the handlers, and stores them into the active handler list. When new conversational event is intercepted during execution of current one, the scheduler in this way always has the ability to append the arrived behavior, or even to smoothly/abruptly override the existing behavior schedule of the targeted animator.

During the execution of the schedule, the *Animation Scheduler* feeds the *Animation Handler* with animation segment that is supposed to be animated (e.g. one configuration

of the targeted part of the embodiment), and waits for a 'sequence complete' message. After such message is received, it continues with the *Que*. After the *Que* is emptied, the *Animation Scheduler* signals that animation stream has been completed, and will destroy the animator objects, while releasing the reserved resources. After all animation segments are completed, the *Animation Scheduler* will signal the end of the conversational event. As a result, the *Event handler* will, if no more co-verbal events arrive, trigger the manifestation of the idle behavior.

5 Discussion and future work

In this paper we have presented a novel EVA Realizer based on Unity 3D game engine. The implementation described in this paper allows us to fully facilitate all the capacities of EVA behavior generator, and to re-use existing conversational agents and existing behavior templates. Namely, EVA framework enables synthesis of natural conversational behavior by synchronizing verbal and non-verbal behavior in form of conversational events. Each such event is *Realizer* agnostic. Thus, it can be realized via any engine that can interpret the message. With the presented approach, we have proven this hypothesis as well as gained another powerful conversational behavior realization engine.

The presented Unity based EVA Realizer is highly modular and event oriented. Unity environment seems ideal basis for rapid development of articulated agents. Namely, it does not require artists or even developers especially trained in animation. E.g. Virtual charters Eva and Adam, outlined in Fig. 3, were generated in Daz3D[7] using the default Genesis3 male and female setups. They were then imported into Unity environment via Autodesk's FBX format. Conversational acts (gestures) displayed by agents, are generated in the form of EVA conversational events by using the EVA Behaviour generator. These events are then broadcasted to both realizers (Unity, and Panda based), without any further contextual knowledge regarding the engine, or the virtual character. As outlined in Fig. 3, the realizers are able to represent the specified co-verbal behaviour regardless of the selected engine, or the male or female character.

Another benefit of the novel realizer implementation are also needed time and effort that are required for the design and integration of some new agent. Namely, Unity features a diverse *Asset Store*, from which various models, objects and even algorithms may be integrated into one's virtual environment. Furthermore, Unity can directly import industry-grade animation formats (COLADA, FMB, MB). Therefore, developers can now rapidly integrate other assets directly from various 3D modelling environments (e.g. DaZ). Previously we modelled in Maya[8], or had to use Maya (or Blender) as a bridge between *Realizer* and *Modelling* environments.

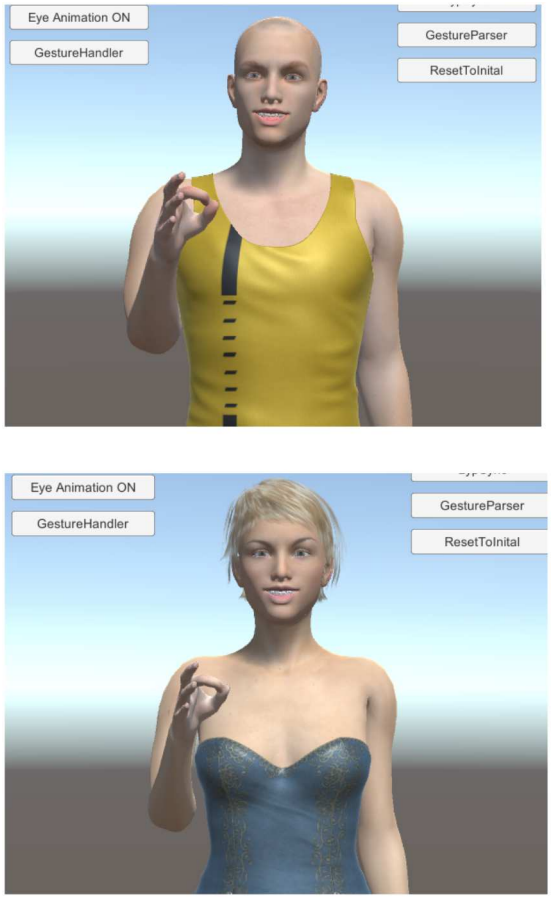

Figure 3: Unity based realizer when performing same behavior on Adam and Eva embodied conversational agents

[7] https://www.daz3d.com/

[8] https://www.autodesk.com/products/maya/overview

Furthermore, the modular approach now allows for quickly developing a prototype of an idea. Namely, Unity already incorporates several powerful and state-of-the-art 3D modeling tools, such as: drag & drop editing, online shades, animation and similar features. These allow developers to dive right into developing functionality, or to test visualizations of new conversational concepts, without any actual changes of the engine itself. The editor's GUI, featured in Unity, is also very powerful and intuitive. It allows for pausing the execution, and manipulating the scene at any time as well as progress gameplay frame by frame. It also has powerful asset management and attributes inspection. Finally, the event oriented concept designed as part of EVA Realizer, allows the virtual agents to properly react to any given conversational context instantly as it appears. This enables us to integrate proprietary conversational agents into highly dynamic situations, where agents are not only expressive but also pro-active. Such agents are not only presenters or listeners, but can also provide reactive feedback and even take over the conversation, just as a real humans would. Thus, the capacity to react to event instantly and to manipulate animation at frame-level is of great importance.

The major downside of the realizer, however, is the fact that in contrast to Panda3D, Unity3D is proprietary, closed source game engine. However, non-commercial variations are available. Furthermore, when performance does not satisfy growing requirements of a project migration to another, the engine will require some extra work. Namely, Unity3D uses very unique approach for doing things and some of things may be less compatible with other game engines. As a result, we will also continue to support the Panda3D realizer.

The ability to express one-self his attitudes, feeling and emotions, plays a central role in defining ECA's personality, its emotional state, and can make such an agent truly active participant in a conversation. However, in order to make him to be perceived even more natural, the agent must be able to respond to situational triggers smoothly, and almost instantly as they are perceived. Thus, the presented framework seems not only exciting, but also an important step towards generating more natural and human-like companions and machine generated responses. However, some challenges, such as generation of EVAScript behavior templates, variety and plausibility of idle behavior etc. still remains. Further, the use of lightning and shaders and materials, and exploitation of the scene is quite rudimentary. In the near future we will work on the idea to exploit various physical objects in the scene for even more natural conversation. Through the use of *Unity Animation editor*, its *Inverse Kinematics*, and a powerful *PhsyX* engine (with IK), these tasks are viable extensions in the future.

Acknowledgments:
This work is partially funded by the European Regional Development Fund and the Ministry of Education, Science and Sport of Slovenia; project SAIAL

This work is partially funded by the European Regional Development Fund and Republic of Slovenia; project IQHOME.

References:

[1] Luger, E., & Sellen, A. (2016, May). Like having a really bad PA: the gulf between user expectation and experience of conversational agents. In Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (pp. 5286-5297). ACM.

[2] Ochs, M., Pelachaud, C., & Mckeown, G. (2017). A User Perception--Based Approach to Create Smiling Embodied Conversational Agents. ACM Transactions on Interactive Intelligent Systems (TiiS), 7(1), 4.

[3] Fabian, R., & Alexandru-Nicolae, M. (2009). Natural language processing implementation on Romanian ChatBot. In WSEAS International Conference. Proceedings. Mathematics and Computers in Science and Engineering (No. 5). WSEAS.

[4] Malcangi, M. (2009). Soft-computing methods for text-to-speech driven avatars. In Proceedings of the 11th WSEAS international conference on Mathematical methods and computational techniques in electrical engineering (pp. 288-292). World Scientific and Engineering Academy and Society (WSEAS).

[5] Kuhnke, F., & Ostermann, J. (2017, July). Visual speech synthesis from 3D mesh sequences driven by combined speech features. In Multimedia and Expo (ICME), 2017 IEEE International Conference on (pp. 1075-1080). IEEE.

[6] Caridakis, G., & Karpouzis, K. (2004). Design and implementation of a greek sign language synthesis system. WSEAS Transactions on Systems, 3(10), 3108-3113.

[7] Rojc, M., Presker, M., Kačič, Z., & Mlakar, I. (2014). TTS-driven Expressive Embodied Conversation Agent EVA for UMB-SmartTV. International journal of computers and communications, 8, pp. 57-66.

[8] Tolins, J., Liu, K., Neff, M., Walker, M. A., & Tree, J. E. F. (2016). A Verbal and Gestural Corpus of Story Retellings to an Expressive Embodied Virtual Character. In *LREC*.

[9] Esposito, A., Esposito, A. M., & Vogel, C. (2015). Needs and challenges in human computer interaction for processing social emotional information. Pattern Recognition Letters, 66, 41-51.

[10] Kok, K. I., & Cienki, A. (2016). Cognitive Grammar and gesture: Points of convergence, advances and challenges. Cognitive Linguistics, 27(1), 67-100.

[11] Kopp, S., & Bergmann, K. (2017, April). Using cognitive models to understand multimodal processes: The case for speech and gesture production. In The Handbook of Multimodal-Multisensor Interfaces (pp. 239-276). Association for Computing Machinery and Morgan & Claypool.

[12] Pelachaud, C. (2015, May). Greta: an interactive expressive embodied conversational agent. In Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems (pp. 5-5). International Foundation for Autonomous Agents and Multiagent Systems.

[13] Neff, M. (2016). Hand Gesture Synthesis for Conversational Characters. Handbook of Human Motion, 1-12.

[14] Rojc, M., Mlakar, I., 2016. An Expressive Conversational-behavior Generation Model for Advanced Interaction Within Multimodal User Interfaces, (Computer Science, Technology and Applications). Nova Science Publishers, Inc., Corp., New York, 234.

[15] Rojc, M., Mlakar, I., & Kačič, Z. (2017). The TTS-driven affective embodied conversational agent EVA, based on a novel conversational-behavior generation algorithm. Engineering Applications of Artificial Intelligence, 57, 80-104.

[16] Gratch, J., Hartholt, A., Dehghani, M., & Marsella, S. (2013). Virtual humans: a new toolkit for cognitive science research. Applied Artificial Intelligence, 19, 215-233.

[17] Thiebaux, M., Marsella, S., Marshall, A. N., & Kallmann, M. (2008, May). Smartbody: Behavior realization for embodied conversational agents. In Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 1 (pp. 151-158). International Foundation for Autonomous Agents and Multiagent Systems.

[18] Pelachaud, C., 2015. Greta: an interactive expressive embodied conversational agent. In: Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, International Foundation for Autonomous Agents and Multiagent Systems, (pp. 5-5).

[19] Klaassen, R., Hendrix, J., Reidsma, D., & van Dijk, B. (2013). Elckerlyc Goes Mobile Enabling Natural Interaction in Mobile User Interfaces.

[20] Heloir, A., & Kipp, M. (2010). Real-time animation of interactive agents: Specification and realization. Applied Artificial Intelligence, 24(6), 510-529.

[21] Kolkmeier, J., Bruijnes, M., Reidsma, D., & Heylen, D. (2017, August). An asap realizer-unity3d bridge for virtual and mixed reality applications. In International Conference on Intelligent Virtual Agents (pp. 227-230). Springer, Cham.

[22] Mlakar, I., & Rojc, M. (2011). EVA: expressive multipart virtual agent performing gestures and emotions. International journal of mathematics and computers in simulation.

[23] Bédi, Branislav, et al. "Starting a Conversation with Strangers in Virtual Reykjavik: Explicit Announcement of Presence." Proceedings from the 3rd European Symposium on Multimodal Communication, Dublin, September 17-18, 2015. No. 105. Linköping University Electronic Press, 2016.