Received: 25 April 2019 | Accepted: 29 April 2019
DOI: 10.1002/cav.1891

SPECIAL ISSUE PAPER

WILEY

Yet another low-level agent handler

Fabrizio Nunnari[1] | Alexis Heloir[2]

[1]German Research Center for Artificial Intelligence (DFKI), Saarbrücken, Germany
[2]LAMIH UMR CNRS 8201 Université de Valencienne, Le Mont Houy, France

Correspondence
Fabrizio Nunnari, German Research Center for Artificial Intelligence (DFKI) Saarbrücken, Germany.
Email: fabrizio.nunnari@dfki.de

Funding information
Federal Ministry of Education and Research (BMBF), Grant/Award Number: 16SV7768, Intera-KT project

Abstract
YALLAH is a framework for the creation of real-time interactive virtual humans. Its production pipeline supports the continuous, parallel development of both the character and the software, and allows users for the deployment of a new character in a few hours of work. YALLAH is based on freely available software, mostly open-source, and its modular software architecture provides a framework for the seamless integration of new features. Finally, thanks to transpilation, the whole framework is conceived to accommodate multiple game engines.

KEYWORDS
agent framework, character creation pipeline, virtual human

1 | INTRODUCTION

Recent advances in virtual reality and augmented reality are widening the application spectrum of virtual characters. The more users gets more accustomed to intelligent conversation agents, the more they will expect their presence in modern interactive applications.

The integration of virtual avatars into a consumer-grade application or an academic experiment setup raises many challenges and require a concentration of many diverse and specific skills which are often out of the reach of a medium size development team or an academic laboratory.

On the artistic side, creating a virtual human from scratch, through the use of a 3D modeler, is a costly and time-consuming job, involving multiple steps such as modeling, rigging, weighting, and authoring blend-shapes. In research contexts, artists are rarely members of the core team; rather, they are hired temporarily as freelance collaborators and leave the team once the appearance and the animation capabilities of the model have been validated.

Only after the team can focus on integrating the character into the interactive environment (which most of the time is an off-the-shelf game engine) and implementing the motion controllers and other control code required by the specific application context. This widely diffused production practice presents three major drawbacks, for both the single production team and for the whole research community.

First, once the 3D character is produced, the team might lack the skills needed to update or improve the character. However, during the development and maintenance of an interactive virtual agent, the 3D model (intended as a whole with its geometry, skeletal structure, blend shapes, and rendering materials) must evolve as the software does. For example, consider extending the blend shapes controlling the visemes of the character in order to support additional languages.

Second, adopting a commercial game engine binds the research team to a commercial product and to its ever-changing licensing policies. If licensing prices become unsustainable, the research team must invest many resources in order to migrate to a competing platform.

Third, different projects and research teams end up developing motion controllers on different game engines and on characters with different skeletal topologies and blend shapes. The result being a constellation of incompatible implementations which force new researches to a considerable implementation effort for reproducing state of the art virtual characters and use them as base line for improvements.

In this paper, we present a framework for the generation of real-time interactive characters which addresses all of the above-mentioned problems. YALLAH (Yet Another Low-Level Agent Handler) was born with the following intents: i) provide development teams with a platform to quickly deploy a fully functional 3D real-time interactive character in few hours of work, ii) provide a full production pipeline, from the creation of the avatar to its deployment, in which the character model can be updated as much as the software, iii) rely as much as possible on open-source and freely available software components, in order to keep independence from a specific rendering engine, and iv) provide the research community with a unified software architecture where different research teams can contribute to the integration of different motion controllers, yet independent from each other, into a single shared architecture.

Many different products exist which can be used as building blocks to deploy a fully functional animated interactive character, but all of those blocks have been created in insulation, and their integration implies a significant development effort. YALLAH cover this need by gluing together a set of open-source free software, thus enabling non artistically skilled user to refine and maintain the character model.

The framework is published as an open-source software* and includes already a set of motion controllers (text-to-speech, eye gaze, eye blinking, animation playback, facial expression control, locomotion) that the community can improve and extend.

YALLAH is currently implemented on the top of Unity, but its software architecture disentangles the implementation of motion controllers from any specific commercial platform through the use of cross-platform transpilation.

2 | RELATED WORK

This section presents an overview of existing solutions aiming at facilitating the integration of real time animated conversational agents into existing applications. We not only focus on academic work, but also on commercial middleware and assets creation tools.

2.1 | Virtual human animation frameworks

A well established platform is the Virtual Human Toolkit (VHT),[1] which gathers an extensive collection of modular tools aimed at creating and controlling the behavior of embodied conversational agents (ECAs). The entire collection of tools is powerful and sophisticated and has been used in many user studies and research projects. However, the configuration and deployment of a setup based on the VHT might be challenging and require a team of seasoned developers. Setting up and using VHT can thus be difficult for small research teams without background in Computer Science. Our solution aims at providing a cleanly packaged and as-consistent-as-possible application which has the potential to be conveniently embedded into an existing application and/or deployed on multiple platforms.

Greta is a realtime platform to control socio-emotional virtual characters.[2,3] It has been used in many European-scale and national-scale projects as the central element of avatar-based interactive user studies. Greta implements the SAIBA architecture[4]: the behaviors of an avatar in Greta are described into FML and BML markup languages. A part of the Greta platform is publicly available to download†, while more advanced functions are yet to be publicly released. In the public repository, Greta currently only permits the integration and control of a single avatar to be selected among the two characters made freely available. Importing a novel character in the Greta platform requires specific animation and modeling skills.

The solution we present in this paper is very similar to VAIF, recently introduced by Gris et al.[5]: an "extensively documented system that attempts to […] provide inexperienced researchers the tools and means to develop their own agents in a centralized, lightweight platform that provides all these features through a simple interface within the Unity game engine." The VAIF framework offers a developer-friendly solution which seems to be easy to set up and integrate into any Unity-based project. It embeds a dialog editor/manager as well as an event scheduler. The Text-to-speech synthesis is provided by the Windows 10 speech API only and the framework is tightly linked to the Unity game engine. The VAIF

*https://github.com/yallah-team/YALLAH | Apr 25th, 2019
†https://github.com/gretaproject/greta | Apr 25th, 2019

framework can be downloaded and used under a permissive Open-Source MIT licence[‡]. It features two rigged characters capable of facial-animation together with a set of motion clips. The documentation mostly consists of a series of online video tutorials. Integrating a different character in the framework requires specific modeling and animation skills.

The YALLAH framework capitalizes on a 10+ years history of collaboration with researchers in psychology and psycholinguistics.[6] The solution employed back then was comparable to the ECA platforms mentioned above[2,7–9] and offered a BML[10] interface to the experimenters to interactively control the avatar. This raised a number of issues, the most significant drawbacks being a cumbersome deployment process, a non-trivial compilation and packaging pipeline, and the impossibility for experimenters to interactively edit the scene layout or the assets. These issues hindered the collaboration flow between our multidisciplinary team and impacted the experimental setup eventually used in our studies.

2.2 | Character creation

Character creation is an artistic work, traditionally accomplished through the use of a 3D modeling editor. The creation of high quality humans requires a team of skilled 3D artists, which often do not fit into the budget of independent developers and research projects.

There exists tools capable of speeding up and easing character authoring. They all share the same authoring metaphor which relies on a set of sliders, each one controlling the deformation of a specific body part.

Among commercial applications following this paradigm, one can mention: iClone[§], Fuse[¶], and Daz3D[#] . All of them exposes a user-friendly GUI but are closed-source and subject to commercial licensing. They all are stand-alone applications, meaning that transferring the generated characters into another 3D editor relies on some 3D exchange format (e.g., OBJ, FBX, or Collada), which might lead to data-losses.

One of the most popular free and open-source product for the generation of virtual humans is MakeHuman[‖]. It is entirely written in Python, allowing for the development of custom plugins, and offers a dedicated open format to reliably transfer the generated characters into Blender.

MB-Lab[**] (previously known as ManuelBastionLab) is a project initiated by the early author of MakeHuman and now supported as open-source free project by a community of 3D artists. MB-Lab is distributed as a plugin for Blender, hence no data exchange is needed.

Both MakeHuman and MB-Lab offer the advantages of exposing a scripting interface to programmatically control the generation of the virtual character. This can be useful in research contexts where the aspect of the generated character is subject of investigation and needs to be procedurally controlled, as in.[11]

Another popular product for the creation of virtual characters is UMA[††]. It enables for the creation and customization of characters within the Unity editor. As such, the characters are immediately available in the run-time environment, without the need to undergo an asset exchange procedure. Since Unity doesn't have an off-line scripting interface, it is impossible to perform scripted customization, and the created characters are bound to Unity only.

3 | PRODUCTION PIPELINE

3.1 | Abstract pipeline

Figure 1 shows an overview of a typical production pipeline, from the crafting of the avatar, to its deployment in a real-time environment.

The upper part shows an abstract view of the 4 stages composing the pipeline. It start with the creation of the virtual character (Stage 1), which is generally accomplished by a specialized artist and might take several days of work. The crafted character (possibly exported into some different editor) very likely needs to undergo some modifications in order to be adapted to the specific requirements of a game engine (Stage 2). Simplifying the skeleton, adding bones needed

[‡]https://github.com/isguser/VAIF | Apr 25th, 2019
[§]https://www.reallusion.com/iclone/ | Apr 25th, 2019
[¶]https://www.adobe.com/products/fuse.html | Apr 25th, 2019
[#] https://www.daz3d.com/ | Apr 25th, 2019
[‖]http://www.makehumancommunity.org/ | Apr 25th, 2019
[**]https://github.com/animate1978/MB-Lab | Apr 25th, 2019
[††]https://github.com/umasteeringgroup/UMA | Apr 25th, 2019

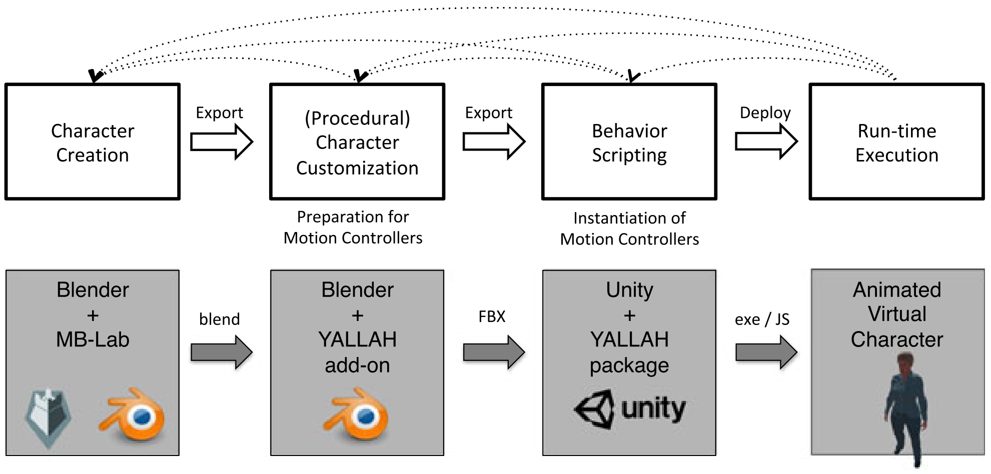


FIGURE 1 Top: overview of a typical production pipeline for the production of interactive virtual humans. Bottom: the tools selected in YALLAH for the implementation of such pipeline

for run-time control, limiting the maximum number of bones influencing some vertices, adjusting the resolution or the format of some textures, and more, are only few of the problems that are generally encountered in real-world productions.

Once the character has been customized, it can be exported to the game-engine (Stage 3). Here, another import configuration phase will take place before the character will be decorated with a set of procedural motion controllers. Together with the basic motion capabilities, the character will be enhanced with some behaviour logic, normally very dependant from the specific final application. Finally, the character is deployed by the game engine into the target platform, and ready to run (Stage 4).

It is important to notice that during any of those stages, it is not uncommon that the developers or the artists need to roll back to any of the previous stages in order to fix unforeseen flaws (such as, co-penetrations of the mesh during the animation, unsatisfying animation quality, unexpected rendering artifacts in real-time, or judgments of the artistic director, to name a few).

3.2 | Implemented pipeline

YALLAH is an implementation of the above-described pipeline achieved through a careful selection of the tools and the development of a significant amount of scripts implementing all the technical details needed to process the whole character and make it ready for deploy.

For YALLAH, we selected the free and open-source Blender[‡‡] 3D editor for both character creation and editing.

The creation of the character is accomplished with MB-Lab (previously known as Manuel Bastioni LAB). MB-Lab is a plugin of Blender enabling the generation of human-like (as well as anime and fantasy characters) in few clicks. The customization of the characters works through a set of slider controlling the degree of deformation of a specific body part via blend shapes (For example, torso width, neck length, eye size are only a few among the more than 200 controllers available).

Once the character is created, it can be customized within the same Blender environment, eliminating the burdens often introduced when transferring a 3D model between editors using intermediate 3D assets exchange formats.

Though YALLAH has been designed for the ground up to use different rendering engines, YALLAH currently only supports the Unity Engine. Unity has got much popularity in the last years, also within the academic realm, for its usability and its permissive license for research purposes. Unity integrates by default an importer for Blender files. One of the major advantages of using Unity is the availability of a deploy procedure for all the major modern computing platforms (Desktop, Mobile, Consoles, and WebGL).

In YALLAH, agents behaviors and core features are implemented in a domain-specific language, which is transpiled into the language of the target rendering engine during the development of behavior scripts. This aspect is described in more detail in a later section.

[‡‡]https://www.blender.org/ | Apr 25th, 2019

3.3 | Practical use-case

The following paragraphs describe the steps leading to the deployment of a newly authored character. All the steps are illustrated in an accompanying video[§§].

First of all, using MB-Lab in Blender, in few clicks the user generates a new character and can customize its proportions. Afterwards, the user *finalizes* the character. This procedure freezes the proportion of the character. Proportion data can be save for later re-editing.

Following, the user will add clothes (as additional geometries) and animations. Here, manual editing is needed to remove the vertices of the mesh that are possibly copenetrating with the clothes. Finally, with a single click, the user applies the main `YALLAH setup` script provided by the YALLAH plugin for Blender. The script programmatically edits the skeleton and the materials of the character in order to prepare it for usage in a game engine.

At this point the character is ready to be saved and exported to Unity. In Unity, the user will instantiate the character and add a set of components to the `mesh` and to the `avatar` game objects. The list of available components include some basic scripts to setup the character materials for run-time rendering and a list of (optional) motion controllers (e.g., eye-blink, text-to-speech, eye gaze). A later section reports a detailed description of the above-mentioned edits and components.

Once all the desired components have been added, the character is ready to be used for the domain-specific purposes of the application.

Given the availability of pre-made clothes and animations, the full creation pipeline, from zero to the availability of the character, takes an average Blender user no more than two hours, the technically most advanced skill being the ability to edit the mesh in order to remove some vertices.

4 | DESCRIPTION OF THE SOFTWARE

4.1 | Structure of the features

YALLAH enables the control of virtual characters in interactive environments. Some features are actually Motion Controllers, driving some aspect of the real-time motion of the character, while other features fulfill technical requirements.

Each feature is implemented by two pieces of code: the first is a (set of) Blender script(s), editing the character as needed by the game engine, the second is the code ran by the game engine. Some of the features require only one of the two sides.

On the Blender side, all the scripts that programmatically edit the character are divided in sub-modules with a well-defined function as entry point. A main `YALLAH setup` script sequentially invokes all the entry points. All the modules are independent from each other, thus facilitating the insertion of new features or removal of existing ones. Inter-dependencies are not (yet) supported.

In Unity, all the scripts related to a feature are contained in a dedicated folder under the `/YALLAH/Scripts/` asset path. For each feature, the user must add just one script to either the mesh or the avatar of the virtual character. This operation can not be automatized because of the absence in Unity of an off-line scripting mechanism. However, the whole operation can be performed by an expert user in less than 1 hour, including the time needed to re-load the Blender asset when adjusting the importing parameters.

4.2 | List of features

Table 1 summarizes the list of features currently implemented in YALLAH, specifying the presence of a relative script in the Blender and in the Unity sides. The Type can be either *technical requirement* or *motion controller*. Each feature exposes a *public API*, i.e., a list of methods for controlling the feature from a high-level perspective. A brief description of each feature follows.

Skeleton Fixing This is a technical feature which adds a bone at the top of the head of the character. Such extra node is needed to have, at run-time, a reference about the position of the upper part of the head. It is used by an external script to automatize the camera movement and perform a proper framing of the face, regardless of its absolute size.

Material Fixing This is a technical feature. On the Blender side, a script re-creates the materials used by the MB-Lab character, which are originally fine-tuned for off-line rendering. Since not all of the information about transparency, color,

[§§]https://youtu.be/7pYVY5TGGvg | Apr 25th, 2019

TABLE 1 List of YALLAH features

Functionality	Type	Blender	Unity
Skeleton fixing	tech	yes	no
Material fixing	tech	yes	yes
Eye blink	mc	no	yes
Text-to-speech	mc	yes	yes
Facial Expressions	mc	yes	yes
Eye Gaze	mc	yes	yes
Animation playback	mc	no	yes
Locomotion	mc	no	yes

and texturing are transferred when exporting to Unity, on the Unity side the user has to add the script `Fix Realtime Materials` to the character's mesh, which programmatically fixes the materials at game start.

Eye Blinking When added to the Unity character's mesh, this script makes the character blink at irregular intervals, as measured on real humans.[12] The blinking, implemented through a simple state machine, is performed by modulating the weight of two dedicated blend shapes for the lids.

Facial Expressions This is a script to attach to the character mesh in Unity. The script exposes methods to retrieve the list available expressions, to a activate an expression, and to reset to the default expression. All the transitions are performed via smooth animations at a user-definable speed. Technically, the Unity component exposes all the blend shapes with the prefix `fe_`: a list that is prepared by the Blender counterpart.

Animation Playback All the animations which are present in the Blender `Action Library` are exposed in Unity and can be controller via a component to be added to the avatar game object. The component offers methods to list the available animations, to playback, and to stop execution. There is an explicit support for the *ambient* animation, which is automatically paused when another animation is triggered. The main advantage for the user is that the Unity component manages the animations using a "dummy" `Animation Controller` which is dynamically re-structured via code. It means that the list of animations can be changed by simply editing a list of names, without the need to manually edit the state machine of the animator.

Text-to-speech The text-to-speech is implemented using an external MaryTTS[13] server. The user must add the `MaryTTSController` script as component of the character mesh, and then configure the server address and the voice to use. The controller exposes methods to say a sentence (taking free text as parameter), query the speaking status, and to stop talking. The controller sends the text to the MaryTTS server and receives back a `wav` file together with a table describing the sequence of phonemes to say, each one associated to a time stamp synchronized with the sound file. YALLAH contains a mapping between the phonemes to pronounce and a set of visemes (blendshapes modulating lips and mouth) which are generated during the setup on the Blender side. While playing back the wav file, a simple internal sequencer increase and decreases the weights of the visemes synchronized with the speech track.

Eye gaze The `EyeGazeController` allows the character to move the eyes and the neck in order to look a specific point in space. It must be attached to the Mesh but it needs also references to the neck bone and the "eye-bones". The MB-Lab character can rotate the eyes using a set of dedicated blendshapes. This is preferable than rotating the eyeballs through bones and rigid skinning because the blendshapes are tuned to move the skin around the orbits when the eyes rotate. However, the blender part of this functionality is creating two dummy bones at the center of both eyes. This bones are needed at run time to query the exact absolute position in space of the eye balls. This controller exposes method to watch to a specific point (saccade + fixation), activate the constant tracking of a target object, or stop any fixation and go back to neutral look-front position. All rotations are smoothly animated. Optionally, the eye gaze can involve the rotation of the neck, animated at a speed matching human average.[14]

Locomotion The script `Locomotion Controller`, applied to the avatar, uses the same animation controller used by the animation playback, but on a dedicated layer of higher priority. The only requirement, in Blender, is to provide three loopable animations: one to advance two steps forward, and two to rotate on place, left and right, of about 90 degrees per cycle, in two steps. The controller exposes the methods to ask the character to walk to a specific point or to stop walking. At the moment of writing, the implementation is limited to movements on an horizontal plane.

4.3 | JavaScript bridging

Deploying an application in a WebGL context displayed inside a browser is the most ubiquituous way to distribute a 3D product. The provider of the application doesn't have to provide explicit download links, and users do not have to explicitely install or update the application.

However, Unity offers a very limited mechanism to communicate between the 3D content and the JavaScript environment running in the containing page: the methods of a Game Object can be executed via a single `SendMessage` function which takes as parameters the name of the object to address, the name of the method to invoke (the method name is searched in all the components attached to the object), and a single argument. Worst of all, return values can not be read back in the JavaScript side.

To overcome these limitations, YALLAH features a JavaScript bridging which allows for the use of the whole YALLAH Public API from JavaScript code. The method is based on the use of intermediate native C functions[¶¶] acting as "bridge" between the C# and the JavaScript code.

The implementation exposes a replica of the whole Public API of YALLAH to the JS side: each of the functions is visible with fully typed parameters and return values. Further technical details are available in the YALLAH developers' documentation.

4.4 | Supporting multiple game engines through transpilation

It is desirable to maintain a research framework as much as possible independent from a commercial product, which in this case is Unity. Another main issue mining portability is that different game engines use their own programming language. To address this problem we adopted the approach proposed in.[15]

The solution we adopted consists on decoupling motion controllers into two components. First, a `CoreLogic` implements the core functionality of the controller, defining an API to control the functionality and using datatypes that are independent from the specific game engine. Second, an `Adapter`[16] interfaces the core component with the game engine. This allows for reusing the core component among different game engines.

To support portability, the core logic blocks are implemented using the Haxe cross-platform development framework[##].[17] Haxe is an object-oriented, optionally statically typed language widely adopted in the game industry. Haxe code be transpiled (i.e. trans-compiled) into many target languages, including C#, Java, Python, and C++.

As an example, in order to re-use a motion controller in the Unreal game engine, one must transpile the core logic of the controller in C++ (instead of C#) and rewrite only the code of the adapter, which would consist into a line-by-line manual conversion performed by a programmer.

At the moment of writing, three functionalities are written in Haxe: eye-blink, text-to-speech, and eye gaze.

4.5 | Implementation Details

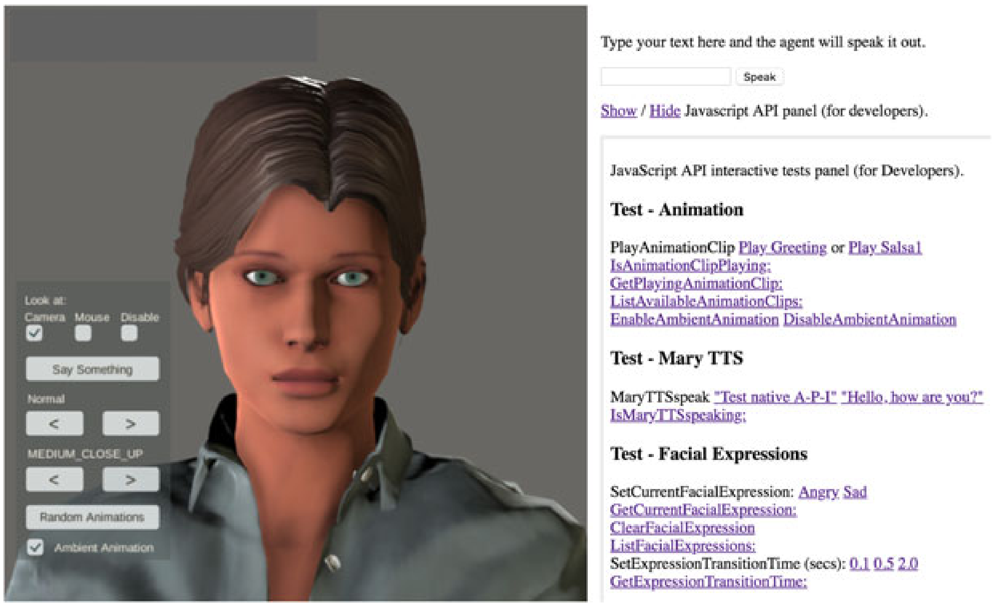
In its first release, YALLAH consists of about 1500 lines of Python code for the Blender addon, about 700 lines of haxe code for the implementation of the transpiled features, and about 2300 lines of C# Unity code for the motion controllers and other testing code.

5 | EXAMPLES

YALLAH has been already used to produce three demos and has been selected to steer the behavior of animated agents in four research projects.

The **Web Talking Head** (available online[‖‖]) demo is deployed as web page (Figure 2). On the left side the user sees the 3D rendering canvas showing the face of a woman. A small control panel allows to trigger some of the functionalities

[¶¶]https://docs.unity3d.com/Manual/webgl-interactingwithbrowserscripting.html | Apr 25th, 2019
[##]https://haxe.org/ | Apr 25th, 2019
[‖‖]http://decad.sb.dfki.de/ | Apr 25th, 2019



Type your text here and the agent will speak it out.

Speak

Show / Hide Javascript API panel (for developers).

JavaScript API interactive tests panel (for Developers).

Test - Animation

PlayAnimationClip Play Greeting or Play Salsa1
IsAnimationClipPlaying:
GetPlayingAnimationClip:
ListAvailableAnimationClips:
EnableAmbientAnimation DisableAmbientAnimation

Test - Mary TTS

MaryTTSspeak "Test native A-P-I" "Hello, how are you?"
IsMaryTTSspeaking:

Test - Facial Expressions

SetCurrentFacialExpression: Angry Sad
GetCurrentFacialExpression:
ClearFacialExpression
ListFacialExpressions:
SetExpressionTransitionTime (secs): 0.1 0.5 2.0
GetExpressionTransitionTime:

FIGURE 2 A screenshot of the *Web Talking Head* demo web page

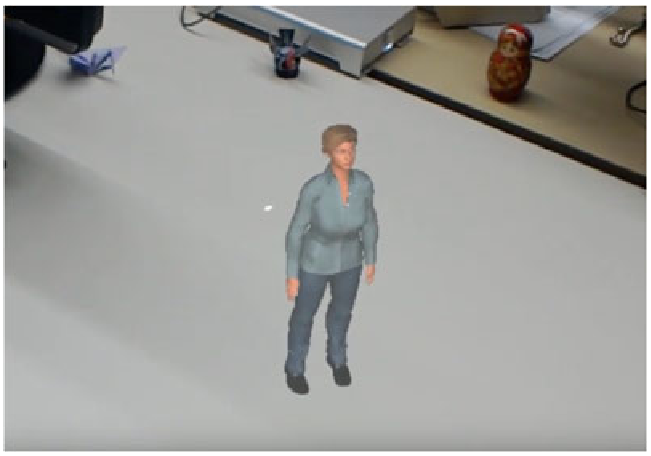


FIGURE 3 A screenshot of the *Yallens* demo running on the Hololens

presented before, plus a control on four levels of zoom on the character; following the terminology from cinematography: *medium shot* or *medium/full/extreme close-up*.

More interestingly, the right side of the page contains a set on clickable text, plus a text area as input for the text-to-speech, each invoking one the the methods on the Public API. This demo shows the capability of YALLAH to be fully controlled from JavaScript code, thus usable, for example, for the development of real-time online chatbots.

Yallens (Figure 3) is a demo showing an interactive character through the Microsoft Hololens***. The user can place a 20cm tall character on an horizontal surface (like a table) and, via voice commands, ask it to perform and animation (waving or dancing), look at the user, or walk to another point on the same surface.

It shows the capability of YALLAH to build characters for augmented reality devices.

The demo **Gang of Four** (Figure 4) runs in an HTC Vive††† virtual reality device. The user is immersed in a scene and surrounded by four different characters, all realized with the YALLAH pipeline. When the user approaches a character,

***https://www.microsoft.com/en-us/hololens | Apr 25th, 2019
†††https://www.vive.com/ | Apr 25th, 2019

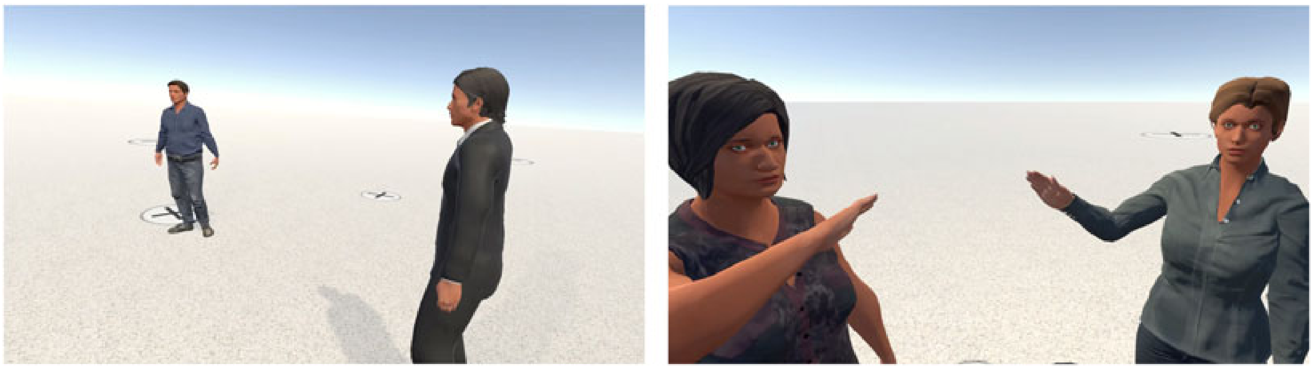


FIGURE 4 Screenshots from the *Gang of Four* demo running on HTC Vive

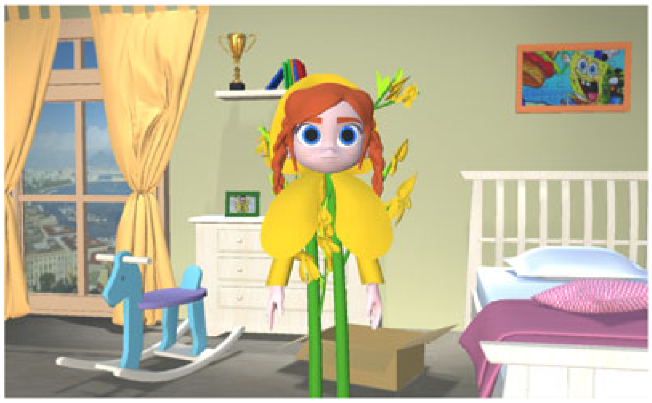


FIGURE 5 Screenshot from the project *BRAVO*

the character looks back at the user and greets him/her. The user can also instruct the characters to walk towards a mark point by selecting it.

The project **Intuitiv**[‡‡‡] explores the use of cooperative robots in the setting of a rehabilitation clinic for orthopaedics, psychosomatics, and neurology. These robots should either help the patients to find their destinations within the clinic area (walking aids), to transport their luggage to their rooms (autonomous robot platform) or to help the clinic staff (robot arm).

The interaction between humans (patients or staff) and the robots should be as intuitive as possible and the movement trajectories of the robots should also be easy to understand, so that humans can easily (and also intuitively) predict where or how the robot is moving next. A virtual avatar shown on a screen mounted to the robots are one way to facilitate the communication between humans and robots, as it allows to combine speech output with facial expressions and gestures while also giving the robot some kind of personality.

The YALLAH pipeline is used to realize the needed avatars, as it can run the avatar in a web browser on different devices and on different operating systems and also allows to flexibly test out on different visual appearances of the character.

The project **BRAVO**[§§§][18] aims at developing a system able to help young patients with ADHD (Attention Deficit Hyperactivity Disorder) to improve their health conditions. Height year old children will interact with fictional characters (See Figure 5) as "intermission" between therapeutic sessions.

The virtual characters used in BRAVO do not come from the MB-Lab Blender plugin (as suggested by the production pipeline proposed by YALLAH), but are completely hand crafted by an artist in 3D Studio Max. However, four functionalities (blinking, text-to-speech, facial expression, and animation player) are applied to the BRAVO characters, the only adaptation being the remapping of the phonemes generated by MaryTTS for the Italian syllabication.

[‡‡‡]https://www-cps.hb.dfki.de/research/projects/INTUITIV | Apr 25th, 2019
[§§§]http://progettobravo.it/en/ | Apr 25th, 2019

This was possible thanks to the modularity of the Motion Controllers developed for the YALLAH project, which are engineered for re-usability on different virtual characters with little re-parameterization.

The goal of project **HuManS**[¶¶¶] is to conceive and develop a sensorized T-shirt for the monitoring of worker's posture. The Blender plugin of YALLAH has been used to quickly generate a real-time avatar for the pre-visualization of the sensors output.

Finally, YALLAH will be used to conduct **Multimodal Dementia Tests with Embodied Conversational Agents in VR**[###] . Embodied conversational agents will be used for exploring multimodal approaches of active input in virtual reality (combination of speech and pen input[19,20]) to determine the cognitive status of the user.

6 | CONCLUSIONS

This paper introduced YALLAH, a new framework for the quick creation and deployment of real-time interactive virtual humans. In addition to productivity and ease of use, YALLAH was designed with the goals of being freely accessible, support programmatic character customization, and extensibility. YALLAH has already been already adopted by a number of research projects at different laboratories and already benefited from new motion controllers developed by external international research teams.

In the future, YALLAH will undergo for improvements of his existing motion controllers (e.g., eye-gaze involving torso, locomotion on non-flat surfaces) and the development of new features, like procedural finger *pointing* to support deictic communication. For 3D authors, we would like to insert the possibility to save the hidden vertices in order to speed up the re-application of clothes in case of re-creation of the character. A longer term challenge is the development of a coordination system to prevent the overlap between competing controllers (e.g., avoid blinking during saccades).

To our knowledge, YALLAH is the first ECA platform which leverages transpilation to establish a clear cut between the programming of the ECA's behavior and the peculiarities implied by the porting the ECA's control logic on an existing game engine. We believe this choices have to potential to make YALLAH a robust and future-proof ECA framework as well as an interesting pedagogical tool for students willing to focus on the design and the programming of ECAs.

ACKNOWLEDGEMENTS

The authors would like to thank Kiarash Tamaddon for his work on the development of the Web, HTC Vive, and Hololens demos. Our thank also to Timo Gühring for the text-to-speech and Haxe ports. Thanks to Ingmar Steiner for his help with MaryTTS. A special thank to Manuel Bastioni for starting the development of the MB-Lab. This research has been partly funded by the Federal Ministry of Education and Research (BMBF) under grant number 16SV7768, Intera-KT project[‖‖].

ORCID

Fabrizio Nunnari https://orcid.org/0000-0002-1596-4043

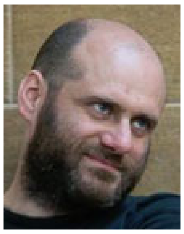
REFERENCES

1. Hartholt A, Traum D, Marsella SC, et al. All together now: introducing the virtual human toolkit. In: Intelligent virtual agents: 13th International Conference, IVA 2013, Edinburgh, UK, August 29–31, 2013. Proceedings. Berlin, Germany: Springer-Verlag Berlin Heidelberg; 2013.
2. Mancini M, Pelachaud C. Dynamic behavior qualifiers for conversational agents. In: Intelligent virtual agents: 7th International Conference, IVA 2007 Paris, France, September 17–19, 2007 Proceedings. Berlin, Germany: Springer-Verlag Berlin Heidelberg; 2007.
3. Mancini M, Niewiadomski R, Bevacqua E, Pelachaud C. Greta: a SAIBA compliant ECA system. Paper presented at: Troisiéme Workshop sur les Agents Conversationnels Animés; 2008 Nov 28; Paris, France.
4. Riviere J, Adam C, Pesty S, et al. Expressive multimodal conversational acts for SAIBA agents. In: Intelligent virtual agents: 10th International Conference, IVA 2011, Reykjavik, Iceland, September 15–17, 2011. Proceedings. Berlin, Germany: Springer-Verlag Berlin Heidelberg; 2011.

[¶¶¶] https://dmd.it/humans/ | Apr. 25th, 2019
[###] http://iml.dfki.de/multimodal-dementia-tests-with-embodied-conversational-agents-in-vr/ | Apr. 29th, 2019
[‖‖] http://www.dfki.de/MedicalCPS/?page_id=725 | Apr. 25th, 2019

5. Gris I, Novick D. Virtual agent interaction framework (VAIF): a tool for rapid development of social agents. Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS); 2018 Jul 10–15; Stockholm, Sweden. Richland, SC International Foundation for Autonomous Agents and Multiagent Systems; 2018.
6. Staudte M, Crocker MW, Heloir A, Kipp M. The influence of speaker gaze on listener comprehension: contrasting visual versus intentional accounts. Cognition. 2014;133(1):317–328.
7. Thiebaux M, Marsella S, Marshall AN, Kallmann M. Smartbody: behavior realization for embodied conversational agents. Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS); 2008 May 12–16; Estoril, Portugal. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems; 2008.
8. van Welbergen H, Reidsma D, Kopp S. An incremental multimodal realizer for behavior co-articulation and coordination. In: Intelligent virtual agents: 12th International Conference, IVA 2012, Santa Cruz, CA, USA, September, 12–14, 2012. Proceedings. Berlin, Germany: Springer-Verlag Berlin Heidelberg; 2012.
9. Courgeon M. multimodal affective and reactive characters In: Lecture notes in artificial intelligence. Berlin, Germany: Springer-Verlag Berlin Heidelberg; 2011.
10. Kopp S, Krenn B, Marsella S, et al. Towards a common framework for multimodal generation: the behavior markup language. In: Intelligent virtual agents: 6th International Conference, IVA 2006, Marina Del Rey, CA; USA, August 21–23, 2006, Proceedings. Berlin, Germany: Springer-Verlag Berlin Heidelberg; 2006.
11. Nunnari F, Heloir A. Generation of virtual characters from personality traits. In: Intelligent virtual agents: 17th International Conference, IVA 2017, Stockholm, Sweden, August 27–30, 2017, Proceedings. Cham, Switzerland: Springer International Publishing; 2017.
12. Johnston PR, Rodriguez J, Lane KJ, Ousler G, Abelson MB. The interblink interval in normal and dry eye subjects. Clinical Ophthalmology. 2013;7:253–259.
13. Le Maguer S, Steiner I. Uprooting MaryTTS: agile processing and voicebuilding. In: Trouvain J, Steiner I, Möbius B, editors. Paper presented at: 28th Conference on Electronic Speech Signal Processing (ESSV); 2017; Saarbrücken, Germany. Dresden, Germany: TUD Press; 2017. p. 152–159.
14. Mitsutake T, Sakamoto M, Horikawa E. Effect of neck and trunk rotation speeds on cerebral cortex activity and standing postural stability: a functional near-infrared spectroscopy study. J Phys Ther Sci. 2015;27(9):2817–2819.
15. Nunnari F, Heloir A. Write-once, transpile-everywhere: re-using motion controllers of virtual humans across multiple game engines. In: Augmented reality, virtual reality, and computer graphics: 5th International Conference, AVR 2018, Otranto, Italy, June 24–27, 2018, Proceedings, Part I. Cham, Switzerland: Springer International Publishing; 2018.
16. Gamma E, Helm R, Johnson R, Vlissides J. Design patterns: elements of reusable object-oriented software. London, UK: Pearson Education; 1994.
17. Cannasse N. Using HaXe. In: The essential guide to open source flash development. Berkeley, CA: Apress; 2008. p. 227–244.
18. De Paolis LT, Paladini I, D'Errico G, et al. BRAVO: a gaming environment for the treatment of ADHD. In: Augmented reality, virtual reality, and computer graphics. Berlin, Germany: Springer-Verlag Berlin Heidelberg; 2019. In press.
19. Prange A, Sonntag D. Modeling cognitive status through automatic scoring of a digital version of the clock drawing test. Proceedings of the 27th Conference on User Modeling, Adaptation and Personalization (UMAP); 2019 Jun 9–12; Larnaca, Cyprus. New York, NY: ACM; 2019.
20. Sonntag D. Interactive cognitive assessment tools: a case study on digital pens for the clinical assessment of dementia. 2018. https://arxiv.org/abs/1810.04943

AUTHOR BIOGRAPHIES



Fabrizio Nunnari is a Senior Researcher at the German Research Center for Artificial Intelligence (DFKI) in Saabrücken, Germany. In 2001, he obtained a Master degree in computer science from the University of Torino, Italy. From the same university, in 2005, he received his PhD on the use of 3D data visualization for collaborative work. Between 2006 and 2012, he worked as researcher and lead developer at the Virtual Reality and Multimedia Park in Torino, Italy. Between 2013 and 2018, he has been part of the Sign Language Synthesis and Interaction group at the Multimodal Computing and Interaction Excellence Cluster in Saarbrücken. His current research interests are on the animation and generation of interactive virtual humans and on the synthetic animation of sign language avatars.

Alexis Heloir is an assistant professor at the University of Valenciennes – LAMIH Laboratory (UMR CNRS 8201). He obtained in 2004 a Master degree in computer science from the University of Lille I. He received his PhD from the University of Southern Brittany in 2008 on the specification and design of intelligible Signing Avatars. Between 2008 and 2012 has collaborated with the German Research Center for Artificial Intelligence (DFKI). Between 2012 and 2018 he was an independent group leader at the Multimodal Computing Excellence Cluster. His current research interests are the control and the animation of embodied conversational agents as well as the generation of intelligible sign language utterances using avatars.

How to cite this article: Nunnari F, Heloir A. Yet another low-level agent handler. *Comput Anim Virtual Worlds*. 2019;30:e1891. https://doi.org/10.1002/cav.1891