A Platform for Building Mobile Virtual Humans

Andrew W. Feng[⋆1], Anton Leuski[⋆⋆1], Stacy Marsella[⋆⋆⋆2], Dan Casas[†1], Sin-Hwa Kang[‡1], and Ari Shapiro[§1]

[1] USC Institute for Creative Technologies
[2] Northeastern University

Abstract. We describe an authoring framework for developing virtual humans on mobile applications. The framework abstracts many elements needed for virtual human generation and interaction, such as the rapid development of nonverbal behavior, lip syncing to speech, dialogue management, access to speech transcription services, and access to mobile sensors such as the microphone, gyroscope and location components.

Keywords: system, mobile, virtual human, chat

1 Motivation

Virtual Humans (VH) have been shown to be effective elements of training simulations, interactive entertainment and other virtual experiences. Non-embodied, audio-based, or text-only agents have been well explored in the research community. A 3D embodied virtual human significantly expands the amount of information that can be communicated to a user over an audio-only or text-based interaction. Virtual humans can potentially display non-verbal behavior similar to humans in face-to-face interactions. However, developing a 3D virtual human can be complicated due to the need to model both visual and behavioral elements of human-to-virtual human interaction. For example, a conversational virtual human might need to keep track of the dialogue turn, exhibit both speaking and listening behavior, and be able to express emotions non-verbally.

Complex virtual humans are typically found in museums, specialized training installations, and similar settings. In many cases, a virtual human resides in a particular place, and is often only available under formal or brief encounters. Thus encounters with virtual humans are limited in place, interaction and time. In addition, constructing a virtual human is a complex, multi-disciplinary effort. Thus research into pervasive aspects of virtual humans have been limited by the few venues in which they can appear, and the complexity of their construction.

⋆ feng@ict.usc.edu
⋆⋆ leuski@ict.usc.edu
⋆⋆⋆ marsella@neu.edu
† dan.casas@gmail.com
‡ kang@ict.usc.edu
§ shapiro@ict.usc.edu

Studies of virtual humans and their impact on real humans are restricted to expensive, location-specific, and domain-specific interactions.

Mobile platforms such as smartphones and tablets are a pervasive technology, and are now capable of running the software and hardware components that are necessary for a convincing, interactive Virtual Human. Potentially, a virtual human running on a mobile platform changes three significant relationships with real humans: (1) a mobile virtual human can be accessible to a mobile user at any time, (2) a mobile virtual human can be accessible to a mobile user at any location, and (3) the group of real users who could potentially interact with a mobile virtual human is broadened to all people with a smartphone. Thus, rich, long-term interactions with a broad range of types of people would now be possible on mobile devices, through a broad set of domains.

While it is possible to create a virtual human application on a mobile device, the effort to do so is very large and requires experts from many disciplines. By providing a mechanism for non-experts to assemble various functionality related to virtual humans and mobile platforms, we extend the domain of virtual human applications to a larger audience than would ordinarily be possible. Rather than requiring several experts from artificial intelligence, graphics, and programming to assemble a single application, we anticipate that a mobile application author could construct a virtual human application without any external assistance. Thus researchers from the social and behavioral sciences will be able to access and manipulate virtual human technology on mobile devices.

In this paper, we further describe an authoring framework for developing virtual humans on mobile applications. The framework abstracts many elements needed for virtual human generation and interaction, such as the rapid development of nonverbal behavior, lip syncing to speech, dialogue management, access to speech transcription services, and access to mobile sensors such as the microphone, gyroscope and location components. We describe our experience in building virtual humans on mobile devices, and how we have attempted to abstract key features into a singular platform that allows the rapid production of similar mobile apps.

Our platform differs from desktop-based virtual human systems in that: 1) it leverages commonly used mobile capabilities, such as the location, gyroscope and microphone sensors, 2) it runs on mobile platforms, such as Android, and does not require a separate mobile application to be built, and 3) it utilizes an Application Programming Interface (API) that allows the author to generate a virtual human application through the use of run-time scripts (using Python [17]), that allows access to mobile capability as well as to virtual human capability.

2 Related Work

Many mobile applications have been developed that use 3D characters, and some mobile applications have been developed as embodied virtual humans. However, there have been very few mobile platforms specifically designed for virtual human research. A number of virtual human systems designed for desktop use include

Greta [15], Elckerlyc [21], EMBR [6] and SmartBody [18] which all use the Behavioral Markup Language [8]. In addition, there are a number of other systems that use different behavioral specifications, such as BEAT [3] and Maxine [1]. A system that integrates many components for virtual human development can be found in [5]. Our framework differs in that we seek rapid authoring for a specific set of capability for mobile devices.

There are numerous studies that use mobile platforms. A description of virtual humans on personal digital assistants is found in [4]. A study of animated characters on a handheld device explored different modalities of agents, such as text, static image, or animated character [2]. More recently, [16] investigates the use of a 3D facial avatar for chat applications. Closest to our work, is the description of the Elckerlyc BML realizer for mobile systems [7], which allows the embodiement of a 2D character on an Android system using the behavioral capabilities of Elckerlyc. Our platform differs in that (1) it uses 3D, not 2D, characters, which allow for a greater potential for nonverbal communication, and (2) it is capable of rendering high-fidelity (photorealistic) virtual humans, (3) our platform provides a set of interfaces to the sensors and to dialogue management, and (4) finally, in contrast to custom systems like [11], we focus on unifying the overall tool set as a platform for building different virtual humans.

3 Experience Developing Mobile Virtual Humans

Our architecture is inspired from our past experience of developing virtual humans on mobile devices. We describe some of these experiences and demonstate how they inform our mobile virtual human architecture. Code examples for the following sections can be found at `http://smartbody.ict.usc.edu/mobilevirtualhumans/`.

Development environments for building mobile apps can require numerous tool sets and build environments and has a slow iteration speed. A mobile platform development typically requires a set of tools that is specialized and sometime unfamiliar to the mobile application developer.

In addition, the iteration time for a typical mobile application is very slow. When iteratively developing a mobile application, the app either has to be copied to a mobile device, or it has to run inside of a simulator, which can be slow, particularly when 3D graphics are used.

In this platform, we simplify such a process by providing an executable app that is configured through the use of a set of runtime-interpreted scripts (in the Python scripting language). By building the executable (the "vanilla app"), we eliminate the need to build the mobile application separately. We abstract the elements needed to configure that application into a set of APIs. By allowing the application to be configured by scripts, an application author can iterate over the changes by simply running the application directly on the device.

Game engines have a steep learning curve and are not designed for conversational virtual characters Modern game engines contain excellent tools for authoring 3D content. However, the use of a 3D game engine requires a learning curve to understand its architecture and design, as well as its build processes. In addition, game engines typically employ very generic animation capabilities; they allow for the playback, blending and overlay of animations, but typically do not provide fine-grain control over subtle human emotion and expression, such as gazing and gesturing.

We base our framework on an existing animation system and BML [8] realizer that employs a large number of conversational capabilities that the virtual human community has identified through research, [18], including automated lip syncing to speech [22]. The learning curve for our platform is not based on knowledge of a game engine API, but rather on a set of APIs targeted to the use of virtual human development: a Virtual Human API (Section 4) which controls characters and their behaviors, an Interface and Sensor API (Section 4) which controls widgets and sensors, a Rendering API (Section 4) which controls the appearance of the app, a Dialogue Management API (Section 4) which controls the dialogue turn, and a Communication API (Section 4) which controls the communication between the device and other systems.

Character configuration is time consuming and complicated. Regardless of the capabilities of the underlying engine or platform, an effective virtual character needs to be configurated properly to perform its functions related to communication and behavior. This requires a detailed face rig, a set of compatible gestures, and a means to lip sync to speech. Typical 3D characters that can be acquired through online marketplaces do not typically have complicated facial rigs, and there are very few standards for facial rigs, emotional expression or lip syncing.

In this platform, we provide a small set of characters of varying gender and age that are designed to be able to express varying emotions and nuance through a set of controls that allow arbitrary combinations of facial poses over time. In addition, our platform provides high-fidelity lip syncing automatically both from text-to-speech, as well as from recorded voice. Our platform also provides a set of male and female gestures that include a large set of deictic (pointing), metaphoric and beat gestures that are suitable for many conversational situations. Thus, the creation of a character that includes many conversational capabilities can be done with just a few lines of code.

Nonverbal behavior can be difficult to generate. In developing virtual human applications, one of the key lessons that we have learned is that hand crafting nonverbal behavior is time consuming and requires considerable knowledge of the particulars of when nonverbal behavior is exhibited, as well as an strong aesthetic sense for the physical manner of that behavior. At the same time, social psychology as well as the virtual human community has extensively

documented the powerful impact behaviors have on face-to-face interaction between humans as well as between humans and virtual humans. Leaving out these behaviors is not a viable option. Further, we have known going back to the pioneering work of film director Lev Kuleshov in the early 1900s on what is now studied as the Kuleshov effect that people will falsely infer attitudes and emotions in the absence of any behavioral signals. For these reasons, researchers and application developers have crafted a range of tools to help automate this process, including BEAT [3], NVBG [9] and Cerebella [13].

Over the process of using these tools in applications, developers have acquired considerable practical expertise. For example, we noticed early on that people tended to slightly nod their heads on initial noun phrases and verb phrases. Automating this behavior in virtual humans quite remarkably brings otherwise seemingly dead characters to life [9]. Later machine learning research [10] bore out this correlation in human data. Further, application developers often wanted very simple mechanisms that could specialize the performance of these tools on specific scenarios and characters, by, for example, triggering specific behaviors when the character spoke specific words or phrases. This functionality was not novel, all of the aforementioned tools had this capability. What developers wanted were simple mechanisms that did not require them to have expertise in a tool's internal workings. Based on both knowledge of the challenges of crafting nonverbal behavior, its importance as well as the practical expertise garnered over the years, we have chosen to incorporate a nonverbal behavior generator based on a few basic principles tailored to application developer use: parsing an utterance, breaking it down into syntactic and lexical elements that would allow an author to simply uses a text file that would provide a map between these elements and behaviors.

Voice-based interfaces are familiar to mobile device users. A speech-based interface is important for many mobile applications that require conversations with virtual humans. The Automatic Speech Recognition (ASR) module of a Virtual Human system, which converts the audio of the user's speech into machine readable text, has to process the audio in real time and be robust to different acoustic conditions, noise levels, and speaker accents. We leverage the Google Speech API to allow easy authoring of speech capture.

Easy sensor access is needed to leverage the unique nature of a mobile device. Mobile devices provide an array of sensors that can be used in virtual human systems, including orientation, position and microphone sensors. The use of such sensors represents one of the main differentiators between mobile and desktop applications. For example, a mobile device can have a location sensor, a gyroscopic sensor, an accelerometer sensor, and a camera sensor.

We simplify access to various sensors through the use of the Interface and Sensor API. While not part of our mobile platform, we anticipate that easy access to a camera-based sensors that reports facial expressions would also be

useful. Such access would allow, for example, the acquisition of facial expressions from the mobile user in real time.

Language Processing. Natural Language Understanding (NLU) and Dialogue Management (DM) form the focal point where the different sensor inputs come together with the speech recognition output and the system decides how to respond to the user's actions.

There are many NLU approaches ranging from simple keyword spotting to detailed semantic parsing [20]. The main goal is the same – it is to ingest the text of the user's speech and produce some sort of machine readable representation that reflects the speech content. There is always a compromise when selecting an appropriate NLU strategy: a simple technique may not provide sufficient information for meaningful interaction, while a more complex approach requires extensive knowledge engineering and can be rather brittle in the presence of ASR mistakes. Another dimension where a virtual human system designer has to make a choice is whether to have the NLU system select one of several predefined utterance meanings or generate one on the fly. The former approach guarantees a predictable result coming from the NLU component, while limiting the number of things the character can understand to whatever is stored in the system.

In our system we use the statistical text classification approach [12]. This approach assumes that all possible system responses are known and stored in a database at the system construction stage. It relies on a set of linked sample utterances and response pairs to "translate" the incoming user's utterance into a query that it uses to search the database of responses. The algorithms returns a ranked subset (possibly empty) of the system responses. While this approach is limited by the number of things the virtual human can understand, it is fast, robust to the ASR errors, and it does not require extensive knowledge engineering beyond providing sample utterances that should trigger individual system responses. It can also combine both verbal and non-verbal features, e.g., sensor data, in one classification step. It has been shown to perform successfully in a variety of applications [12].

The second part of virtual human language processing is the Dialogue Management (DM) module that takes the result from the NLU, considers the current state of the interaction, and selects the system response. Here also a number of approaches exist that range from a set of simple rules to a sophisticated multi-level reasoning process that includes modeling of the character knowledge, goals, and emotions [19]. For our platform we use a rule-based decision approach, where the character designer specifies a number of rules triggered by the specific NLU outputs, sensor events, or system timers. The designer can use the scripting language to model and maintain the dialogue state, keep track of the interaction history, and combine the DM state with the system events into complex behavior strategies. A general purpose dialogue management script is included with the system sufficient for constructing question-answering characters [12]. We also provide a tool box of script-based functions for the character designer

to extend and modify the dialogue strategy, instead of relying on a separate tool to configure the NLU and DM.

Communication between mobile devices or between a mobile device and a server is a common part of many virtual human apps. Mobile applications frequently require communication that extends beyond the device itself, and potentially to other devices or servers. For example, a mobile application might communicate with a server to collect or retrieve data. Standard mobile platform communication mechanisms can be used over standard TCP/IP networks.

We include an asynchronous communication protocol called the Virtual Human Messaging System (VHMSG) that allows the easy communication between virtual human applications. Thus, only one line of script code is needed to connect, and one line of script code for each message is necessary. In this way, we allow numerous mobile devices to communicate with each other without requiring a separate protocol to be defined and managed.

4 Architecture

The Mobile Virtual Humans framework consists of a set of code, scripts, configuration files and processes to build and develop a mobile application on an Android operating system platform. The app author configures the mobile virtual human by writing scripts that access five Application Programming Interfaces (APIs); the Virtual Human API, the Interaction and Sensor API, the Rendering API, the Dialogue Management API, and the Communication API. The five APIs, in turn, communicate with the animation engine, rendering engine, mobile os platform, and sensors. The rendering component interacts with the animation engine, mobile OS and app code, as shown in Figure 1.

Application authors can leverage the application, called the "vanilla app", by associating data and process. Thus, a mobile virtual human platform author does not need to compile a new application, only to modify the control scripts and data in order to have a functioning mobile virtual human app.

Virtual Human API The Virtual Human API handles the setup and control of the virtual human characters. The API includes methods to create a virtual human, configure its behaviors, and respond to user input. The API leverages an animation system SmartBody [18] to construct and configure characters and the environment. Lip syncing to speech is automatically performed and matched to the character's facial rigs [22]. Complex configuration elements, such as configuring the facial rig, and behavioral control are abstracted via interface scripts, and thus the construction of characters can be done with a single line of code.

Interface and Sensor API The Interface and Sensor API manages the user interaction with the mobile device, such as touch and widget interfaces, as well as access to the various mobile sensors, such as the gyroscope, microphone, location and camera. In addition, it allows access to device-specific information, such as the device id, IP address and name of the user interacting with the device.

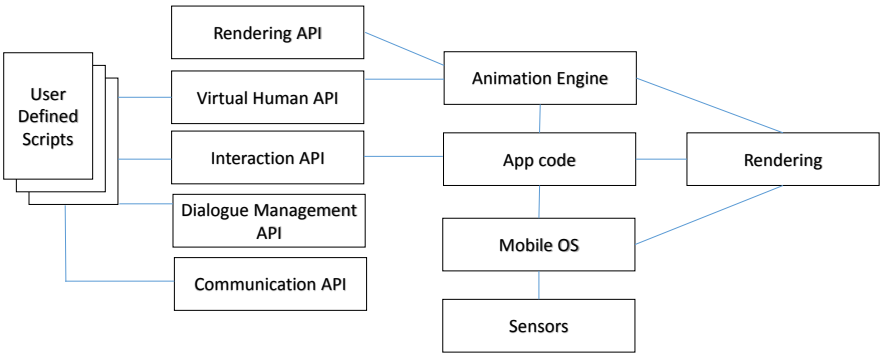

Fig. 1. Components of Mobile Virtual Human architecture. The authoring interface is primarily through a set of scripting interfaces that allow control of the Virtual Human API, the Interface API, the Rendering API, and the Dialogue Management API.

Rendering API The Rendering API manages the display elements such as cameras and lighting. The platform draws 3D content using OpenGL ES [14].

Dialogue Management API The Dialogue Management API handles control of the speaking turn and the virtual human responses to user questions. The NPC Editor [12] is used to provide a question-and-answer interface.

Communication API The Communication API allows easy access to all the capabilities in the system. The platform leverages the Virtual Human Messaging System (VHMSG) which transmits asynchronous messages across a TCP/IP network.

5 Applications

We demonstrate a number of applications that can be generated through our architecture and describe the key control elements needed in Figure 2.

6 Conclusion

In this work, we have presented a platform for the development of virtual human systems on mobile devices. Unlike many 3D frameworks and game engines, our framework is focused on the interaction between a virtual human and a user. Our framework also differs from desktop-based virtual human frameworks in that it allows convenient access to capabilities that are commonly found on mobile devices, such as the gyroscopic, microphone and location sensors. Modification of an application developed by our platform can be done in a data-driven way by modifying the scripts that control the virtual human, interface and rendering.

All software architecture designs differ in which aspects can be exposed to the platform author and which aspects will be hidden. Any 3D platform, such as a game engine, can be used to design virtual human systems. Our design choices

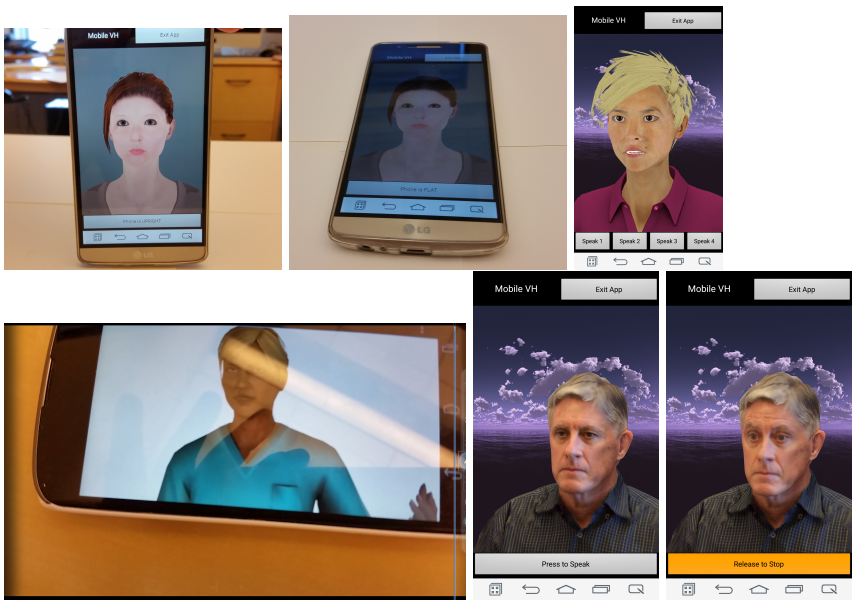

Fig. 2. (Top Left and Top Center) Using the accelerometer to determine the orientation of the mobile device. (Top Right) An example using text-to-speech and proper lip sync useful for dynamic conversations. (Bottom Left) Use of a virtual human in the medical domain. A virtual nurse queries the users for health issues. (Bottom Center and Bottom Right) A realtime render showing a photorealistic virtual human backchannelling in response to a user's speech.

make the development of a 3D conversational virtual human require very little configuration or expertise. For example, a virtual human can be displayed and made to speak with only a few lines of code, and without even going through the configuration process of building an app. By contrast, a game engine would be a superior platform for the development of large-scale 3D worlds that include various particle effects, such as water, dust and so forth.

References

1. Baldassarri, S., Cerezo, E., Seron, F.J.: Maxine: A platform for embodied animated agents. Computers & Graphics 32(4), 430–437 (2008)
2. Bickmore, T., Mauer, D.: Modalities for building relationships with handheld computer agents. In: CHI'06 Extended Abstracts on Human Factors in Computing Systems. pp. 544–549. ACM (2006)
3. Cassell, J., Vilhjálmsson, H.H., Bickmore, T.: Beat: the behavior expression animation toolkit. In: Life-Like Characters, pp. 163–185. Springer (2004)
4. Gutierrez, M., Vexo, F., Thalmann, D.: Controlling virtual humans using pdas. In: The 9th International Conference on Multi-Media Modeling (MMM" 03). pp. 150–166. No. VRLAB-CONF-2007-028 (2003)

5. Hartholt, A., Traum, D., Marsella, S.C., Shapiro, A., Stratou, G., Leuski, A., Morency, L.P., Gratch, J.: All together now. In: Intelligent Virtual Agents. pp. 368–381. Springer (2013)
6. Heloir, A., Kipp, M.: Real-time animation of interactive agents: Specification and realization. Applied Artificial Intelligence 24(6), 510–529 (2010)
7. Klaassen, R., Hendrix, J., Reidsma, D., et al.: Elckerlyc goes mobile enabling technology for ecas in mobile applications. In: UBICOMM 2012, The Sixth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies. pp. 41–47 (2012)
8. Kopp, S., Krenn, B., Marsella, S., Marshall, A.N., Pelachaud, C., Pirker, H., Thórisson, K.R., Vilhjálmsson, H.: Towards a common framework for multimodal generation: The behavior markup language. In: Intelligent virtual agents. pp. 205–217. Springer (2006)
9. Lee, J., Marsella, S.: Nonverbal behavior generator for embodied conversational agents. In: Intelligent virtual agents, pp. 243–255. Springer Berlin Heidelberg (2006)
10. Lee, J., Marsella, S.C.: Predicting speaker head nods and the effects of affective information. Multimedia, IEEE Transactions on 12(6), 552–562 (2010)
11. Leuski, A., Gowrisankar, R., Richmond, T., Shapiro, A., Xu, Y., Feng, A.: Mobile personal healthcare mediated by virtual humans. In: Proceedings of International Conference on Intelligent User Interfaces (2014)
12. Leuski, A., Traum, D.: NPCEditor: Creating virtual human dialogue using information retrieval techniques. AI Magazine 32(2), 42–56 (2011)
13. Marsella, S., Xu, Y., Lhommet, M., Feng, A., Scherer, S., Shapiro, A.: Virtual character performance from speech. In: Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation. pp. 25–35. ACM (2013)
14. Munshi, A., Ginsburg, D., Shreiner, D.: OpenGL ES 2.0 programming guide. Pearson Education (2008)
15. Poggi, I., Pelachaud, C., de Rosis, F., Carofiglio, V., De Carolis, B.: Greta. a believable embodied conversational agent. In: Multimodal intelligent information presentation, pp. 3–25. Springer (2005)
16. Rincón-Nigro, M., Deng, Z.: A text-driven conversational avatar interface for instant messaging on mobile devices. Human-Machine Systems, IEEE Transactions on 43(3), 328–332 (2013)
17. Sanner, M.F., et al.: Python: a programming language for software integration and development. J Mol Graph Model 17(1), 57–61 (1999)
18. Shapiro, A.: Building a character animation system. In: Motion in Games, pp. 98–109. Springer (2011)
19. Traum, D.: Talking to virtual humans: Dialogue models and methodologies for embodied conversational agents. In: Wachsmuth, I., Knoblich, G. (eds.) Modeling Communication with Robots and Virtual Humans, pp. 296–309. Springer (2008)
20. Traum, D., Swartout, W., Gratch, J., Marsella, S., Kenney, P., Hovy, E., Narayanan, S., Fast, E., Martinovski, B., Bhagat, R., Robinson, S., Marshall, A., Wang, D., Gandhe, S., Leuski, A.: Dealing with doctors: Virtual humans for non-team interaction training. In: Proceedings of the 6th annual SIGDIAL Conference. Lisbon, Portugal (September 2005)
21. van Welbergen, H., Reidsma, D., Ruttkay, Z.M., Zwiers, J.: Elckerlyc. Journal on Multimodal User Interfaces 3(4), 271–284 (2009)
22. Xu, Y., Feng, A.W., Marsella, S., Shapiro, A.: A practical and configurable lip sync method for games. In: Proceedings of Motion on Games. pp. 131–140. ACM (2013)