Using Conversational Agents in multimodal applications on mobile platforms

Ricardo Miguel Almeida da Silva, Nº 58569
Instituto Superior Técnico, Lisboa, Portugal
ricardo.da.silva@ist.utl.pt

Resume - The inclusion of Conversational Virtual Agents in applications allows for a more natural experience for the user, successfully mimicking human-to-human communications and increasing the range of interaction between the users and the system. Over the last few years, the creation of Conversational Virtual Agents was made having a desktop-oriented installation in mind. However, there has been an upraise in mobile development, raising the issue of a possible portability of these agents towards the mobile paradigm. This work proposes the migration of the Conversational Virtual Agents tool, Ei'VA, to the mobile paradigm. This tool was originally intended to desktop computers, not contemplating performance and network restrictions that exist on mobile devices. Another objective of this thesis was to create a mobile application whose contents are loaded based on the location and characteristics of the devices. To achieve the objectives outlined, we created a rendering of explanations that consist in agents and other media contents, such as videos, images and texts. We developed an interface for the editing of these explanations and a mobile application capable of their interpretation and the rendering of a compatible representation, depending on the device that is running the application. The resulting prototypes had the Monserrate Palace, in Sintra, as a domain. Performance tests were made in order to evaluate the mobile application, which included the new synchronization system for the agent's speech, as well as tests using real users. Test results revealed a loss in the agent's acceptance due to the poor performance of mobile devices when compared to desktop computers. However, gains were achieved in terms of the agent's proximity to the user.

Keywords: Conversational Virtual Agents; Mobile Devices; Template-based applications; Distributed Spoken Dialogue system

1 Introduction

Nowadays, mobile devices have a defining role in the technological world. The breakthroughs made in micro technology which allowed for the making of smaller
and more ergonomical devices contributed heavily to this fact. Consequently, a vast market of applications emerged and made these devices useful in a wide range of ways [1].

Parallel to this evolution, it’s clear that the major technology companies invested heavily in new means of human-computer interaction, in which voice played a major role. As such, applications controlled by voice emerged, bringing new contents to the interface. Conversational Virtual Agents are a part of these new contents, embracing a natural way of communicating and providing a metaphor for human dialogues [2].

There are many reasons that lead to the inclusion of Conversational Virtual Agents on interfaces. Firstly, the enhancement of expressive power on the display of the contents; the order in which the components are related can be easily achieved by the agent pointing at them and lastly the entertainment factor and the emotional function associated with the character, which plays a vital role in keeping users more aware and involved in the application. This last factor makes the agents a valuable help in the learning process, without distracting or distancing the users from the learning experience [3], which leads to their use in numerous educational applications [4].

Virtual agents with real-time animations are conceived using game engines, which represent a setback in the creation of the applications that use them, being that these engines are directed to deal with objects in a three-dimensional environment, making it harder to create the interfaces.

2 Objectives

In order to allow for the introduction and configuration of Conversational Virtual Agents in multimodal applications, several frameworks appeared, one of them being EI²VA [5]. This presents itself as a complete tool, aggregating the configurations and technologies in well-defined modules, allowing for it to be ported to several devices. This tool is mainly targeted at desktop computers, not considering the performance and network restrictions of mobile devices.

Therefore, the main objective of this work was the migration of the Conversational Virtual Agents tool, EI²VA, to the mobile paradigm, targeting devices with the Android operating system. There was also the objective of creating a mobile application capable of loading different types of explanations, which would vary according to the device’s location and specifications. It was intended that the users received contents that fitted its environment and device. The explanations were sequences of multimedia elements composed by images, text, videos and agents.

3 Contributions

The introduction of Conversational Virtual Agents with real-time animations in mobile devices opens up a new world of interface content customization on Android devices. Using real-time animated agents brings many advantages, mainly the realism and credibility of the agents, speech customization and the agent’s knowledge base.
In terms of the modifications made to the virtual agents, the separation achieved on the agents in relation to where it’s being executed constitutes an evolution of the EI²VA tool, seeing that this tool was depending on a benchmark made to the host machine in order to calculate the ideal animation timings. Therefore, a new mechanism was developed in order to assure the approximation of the animation and audio timings, which works in various devices.

The mobile application and the interface for the editing of explanations present a number of contributions, mainly the simplicity of creating explanation sequences with several multimedia elements.

Another contribution was the possibility to include an agent in these explanations, enriching the interface of the mobile devices.

## 4 State of the art

Engine for Immersive Interaction with Virtual Agents, EI²VA, is a framework based on virtual agents integrated in a Spoken Dialogue System in order to achieve an immersive and multimodal journey, where realism and interactivity are baselines. The structure of this tool presents a clear division into three main blocks: Block Application Engine Animation and Spoken Dialogue System. The Application block corresponds to a standard multimedia support that interfaces to the procedures of the agents; Block Engine Animation is responsible for grant realism to agents by creating facial animations, emotional and bodily; Spoken Dialogue System ensures interaction between knowledge domains.

There are limitations related to the blocks described above, mainly with animation engine block, since it is not prepared to run on devices with limited capabilities. This block has mechanisms that allow synchronization between sound and agents animations, which are prepared to run on desktop computers only. Considering another block, the spoken dialog system, has also limitations regarding the migration to the mobile paradigm, assuming the presence of the microphone in the same machine as the agent and does not provide restrictions on sending audio via network.

INCA is a conversational agent that plays the role of personal assistant [6], using Natural Language as input and output to interact with the user. It was created taking into account the limited computing resources and constraints with audio capture in mobile devices. The SmartKom Mobile is an application that takes into account the limited resources of mobile devices and corresponding to a distribution of multimodal applications with agents for these devices to have essentially the onboard computers in cars and PDAs [7]. The application Carletto is the practical application of a virtual agent, in this case a cartoon of a spider, which is intended to guide users in a palace (Palazzo Chiabelese in Torim) [8].

A project of Limburgs Universitair Centrum was to create a language for describing interfaces can be used for various mobile and embedded platforms, operating in heterogeneous environments and able to make decisions at runtime [9]. Was considered the need to provide the designer interface a powerful tool able to edit the interfaces at design time and after generation, its definition is interpreted by various platforms taking into account the constraints of each.
5 EI²VA in the mobile paradigm

The Animation Engine block of the EI²VA system is included in the application with the agents, as the SDF is mapped in the server. Although there is this separation, the application has a few responsibilities towards the SDF block, namely the coordination of the recognition timings, seeing that the controls are available in the application’s interface, and verifying periodically if there is a result for the recognition on the server. If it’s the case, then it also delivers the VHML with the results of the Animation Engine block.

The presence of this local server favors the conversion of the EI²VA tool to the mobile paradigm, seeing that there is a need for the presence of a server on the network. On this paradigm change, the application resides in a mobile device hit the microphone while the server is in a different machine.

In order for the agent to speak correctly, there have to be mechanisms that assure the correct synchronism between the animation and the audio. Achieving an absolute synchronization between these two elements constitutes a hard task, seeing that these are being played in two different pieces of hardware, one in the graphics card and the other one on the sound card. However, there is a way of using approximations, which make the delays unseen by the human eye.

On the EI²VA tool, when an agent receives a VHML with the description of the visemes and timings associated with the speech, these are mapped to animations of the agent. At the beginning of the speech and after all of the structures are filled, the Animation module authorizes the audio to be played, followed by the execution of the agent’s animations, each one taking the time described in the VHML. In case everything goes well, the animation sequence ends before the audio. In order to bridge this gap, a delay was introduced on the animations, multiplying each animation time by a Div factor.

However, the Div factor constituted an impediment to the EI²VA’s portability and so a robust solution was developed in order to improve this functionality. During the configuration process of the tool, a benchmark needed to be made in order to determine the ideal value for this variable in each device. This process is subjective and prone to errors, seeing as it’s made using only human perception.

The solution to the problem was to use numerous Divs for each viseme, seeing that the delays differed according to the each viseme. The mapping between each viseme and the last ten Divs is stored in a structure inside an XML file, which is updated after each animation. The ideal Div values corresponding to each viseme and the device are stored in this file, so it is expected that these values vary between different devices.

\[
\text{newDiv} = \text{usedDiv} + \frac{\text{audioTime} - \text{visemaDelay}}{\text{audioTime}}
\]  

(1)

In formula 1, the variable usedDiv is associated with the average value of the last Divs and is used to delay the timing of the viseme. audioTime refers to the ideal animation time, set in the VHML file corresponding to the speech and visemaDelay refers to the animation’s actual time. With these variables, newDiv is calculated and then inserted into the Div structures associated with the viseme. If the structure has any 0 values, one of these is replaced with newDiv, if not then the oldest Div is
replaced. With this formula, there is a subtle reduction in delays and the values are optimized to future executions.

\[
\text{accumulatedDelay} = \sum_{i=1}^{n} \text{visemaDelay}(i)
\]  

In formula 2, accumulatedDelay refers to the total delay of the animation towards the audio. If the value of accumulatedDelay is over 0.1, which is a significant difference between the audio and the animation, the Animation module enters containment mode. During this mode, the previous formula isn’t used to calculate the value of newDiv. Instead, there are abrupt increases and decreases in the audio or video, depending on which one is delayed, resulting in effective corrections. This is, however, an exceptional case, so this is rarely used. When the Animation module detects a recovery in the delays, it resumes normal behavior.

On the original version of the EI²VA tool, the application had not only all of the responsibilities towards the agents, but also a few of the SDF. Specifically, it warned the server to start the capture of audio via the microphone, followed by the download of the results and delivery of results to the components responsible for the playback of audio and video. On migrating to the mobile paradigm, this application went on to play a new role, the role of recording audio from the microphone of the mobile device and sending it to the server, replacing the server in terms of manipulating the input coming from the microphone.

6 System

The Mobile Application is designed for smartphones and tablets (mobile devices) with the Android operating system, corresponding to the system interface. The server is planned to interact with several devices simultaneously having the Mobile Application installed, being all connected to the same Wi-Fi network, having a range of interaction of a few dozens of devices to a single server. The Mobile Application features elementary structures central to its operation, keeping largely distributed resources on the server. The domain used in the prototype system corresponded to the Palace of Monserrate in Sintra. This domain may be changed, once the system has been designed considering future uses.

A Timeline corresponds to a sequence of Templates, which together form a room explanation. For a given explanation several Timelines may exist, differing in their explanatory degree and order. The explanatory degrees correspond to Basic, Advanced and Expert, where Basic degree is a basic explanation and Expert corresponds to a deeper explanation.

The interface of the Mobile Application is provided with mechanisms that allow the user to freely navigate in the Timeline. When the end of the explanation is reached, the user can choose to hear the explanation again, go back to the monument plant or play a more advanced explanation of the room, if there is one. Each room can have at most three Timelines, yet only one degree is sufficient to ensure the correct functioning of the application. In the case that there are several degrees of explanations, these are played in the order stated earlier (Basic, Advanced and
Expert). The user can interrupt the explanatory sequences at any time, returning to the menu with the plant of the monument.

Figure 1 – a) Relational order between Timeline, Template e Elemento; b) Relation with Aplicação Móvel interface

A Timeline is described as a sequence of Templates, where a Template corresponds to an interface formed by multimedia components, arranged to transmit a message to the user. These components are called Elements, differing in size and characteristics. The presentation of templates can be processed automatically, since the application has mechanisms that implement this functionality.

Not all Timelines are played the same way in the Mobile Application. This application is able to determine the dimensions of the screen and recognize the type of device (tablet or mobile phone). If it corresponds to a mobile device, the elements are divided so that some of the Templates are divided into more than one screen, thus providing a vertical dimension of the navigation application.

We found it necessary to define a structure able to represent interfaces textually, in this case the Timelines. This need is due to the setting of the interfaces, since the elements have different characteristics.

The Server provides Web Services capable of returning the content of Timelines. For that only the name of the room is needed as well as the type of explanation you want. When a Timeline referring to a specific room is required, the Mobile Application Server inquires the explanations available, following a request by the most basic explanation, in this case the Basic Timeline grade, and if it does not exist it returns a higher grade one. Upon receiving the textual representation of Timeline, the mobile application provides interpreters able to convert this textual representation in a visual representation.

In the process of creating the visual component of a Timeline, the memory device does not support the loading of all elements, since its storage is limited, and therefore could lead to the stopping of the application. As such we have decided to create loading windows, making the loading of the elements most likely to be played. This window can have any desirable size, depending on the memory available for the device. However, a value of three templates per window was defined. This corresponds to the minimum value that ensures the functioning of the application, as it loads the Template elements that are to be shown in the interface in the memory, as well as the elements of the template above and below.
The Timeline Analyzer module of the Mobile Application aims to extract all information from XML representation of Timelines. This module receives an XML file, parses it, and converts it to the format of the Mobile Application, filling important structures that represent the Timelines and respective Templates. These are the structures which are to be iterated in the presentation of the explanatory sequence.

The Timeline Splitter module is only used when the Global Manager detects the presence of a mobile phone. This module is responsible for subdividing the timeline, so that the elements can be seen on screens with reduced dimensions, using the result of the Timeline Analyzer and sending it to this partition. The division process makes it present only the element with highest priority on each template and the remaining elements in another screen accessible via a "More" button, adding depth to each of the templates. The rules by which to govern the division process use the priorities of the elements, resizing them and seeking new positions so there are no gaps in their display on the interface.

The Presentations Manager module is primarily responsible for the visual representation of Timelines, ie it is able to read the structures from the Analyzer modules and Timeline Splitter, converting these structures in multimedia elements visible to the user. The content of some elements such as images and videos are available in the Network Manager module, where the Presentation Manager will fetch the contents when needed. It is this module that knows the various representations associated with the various types of elements, and arrangements in the respective interface.

Creating Timelines manually would be time consuming and error prone. As such, this interface enables non-programmers, but which nevertheless present some training on the system, to configure the content that is made available to the Mobile Applications. Based on the motivation described, we created a Timeline Management Interface, which is composed of three main menus: the Settings, the Template Editor and the Timeline Editor.

7 Results and Validation

The first set of tests targeted the new functionalities of synchronization between audio and agent animations. With these tests, the objective was to prove the advantages of the new calculation method, exercising it on various platforms and comparing it to the original method. On the second part of the tests, the measurements made to the mobile application loading mechanisms are described, more specifically the one that allows the loading of the timeline elements by windows. The third set of tests was based on the usage of the mobile application in tablets and smartphones, determining its usability and measuring eventual losses related to the porting of the agents to the mobile paradigm.

These tests included the participation of numerous users which responded to surveys with questions related to the contents’ relevance and application usability.
For the Div-related tests, thirty speeches were created to the agents, more specifically thirty VHML files with different sizes. Of the thirty VHMLs, ten had about 20 words, ten had 50 words and the final ten had about 85 words. The speeches consisted on current news and the application was tested with the agents in three different platforms: one tablet, one smartphone and a desktop computer.

![Bar chart showing average time between audio and animations using dynamic Div](image)

**Figure 2 - Average time between audio and animations using dynamic Div**

It’s correct to conclude that the dynamic Div showed positive results in all platforms, presenting differences in both that didn’t compromise the synchronization between the audio and the animations. On the various devices, the differences between the dynamic Divs with ideal and non-ideal values aren’t significant, showing that the time it takes for the Divs to be determined doesn’t affect the agent’s normal behavior. EI²VA’s static Div is more precise in desktop computers, when rigorously determined, presenting negligible differences between the audio and animations. However, the process of benchmarking is time consuming, which leaves to ponder whether it’s viable to waste time configuring the agent and the precision that is wanted to the synchronization. The results of the delays between the audio and animations proved that the incorporation of the Div with dynamic values was an important evolution of the EI²VA tool.

The second test had the purpose of determining the efficiency of the loading of elements by the mobile application. The memory usage rate is measured with an auxiliar software, which was being run at the same time as the mobile application, measuring every one second.

The values obtained by this test aren’t totally deterministic, seeing that besides the mobile application and the measurement software there are other applications and system services being run, which also consume memory. The usage of the 3 element windows had better results than the 5 element windows, but still the latter showed an acceptable usage of memory which leads to conclude that both methods could be used in tablets.

The loading of all the elements isn’t recommended due to the oscillations that this causes in the usage of memory by the device, potentially compromising the application’s acceptable behavior.

The user-oriented tests had a total of 30 subjects, in order to capture a significant amount of impressions in terms of the application’s usability. The object of interest was the usability differences between a Timeline shown on a tablet and a Timeline divided to fit a smartphone. The tests included the usage of the application on a tablet
and a smartphone followed by the filling of surveys. To determine the usability of the application on the various devices, the SUS - System Usability Scale [10] was used, which included a survey made of ten redundant questions that uses a specific scale that allows for a more universal view of the system’s usability. Parallel to the survey answers, small knowledge evaluations were made after each usage of the devices. At the end of the usage in each device, the opinions about the interface and the agent were collected.

Summarizing, the tests started with the users visiting the Music Room on the mobile application on one of the mobile devices, followed by a SUS survey about the application on that device and a small test about the room. Following that, the other device was used and a new SUS survey was made, as well as the same test about the room’s contents. In the end, a general appreciation survey was made about the mobile Application.

Through the analysis of the results, it was proven that the mobile application was usable in both devices, seeing that the SUS results were similar and showed high scores. In short, although there were losses in the agent’s performance due to the porting to the mobile devices, especially in terms of visual impact and audio quality, these agents still provide quite satisfactory results, being their integration in a mobile application a relevant matter, contributing to the explanation of the various explanations that compose the timelines.

In terms of the results obtained with the evaluation of the mobile application, these show that the splitting mechanism used in timelines for smaller devices had a positive impact, seeing that there was a significant number of people that preferred to use the application on a smartphone.

8 Conclusions

Through this assignment, we successfully included Virtual Conversational Agents in mobile devices. Even though there were a few losses due to the paradigm shift, motivated by the low performance of mobile devices when compared to desktop computers, there were several positive points, such as an enhanced proximity with the end-user, seeing that the agent is no longer in a kiosk but in the user’s hand or pocket.
In this system, we also proved that it’s possible to combine agents and other multimedia contents in the same application, working in harmony and sending a clear message to the user, regardless of its location or device. It is then allowed for agent-based applications to be created, which present a high degree of dynamism on the provided contents. An important breakthrough in this dynamism was the creation of a textual representation of the Timelines, which enabled the creation of agents and other elements. This representation can also be used in other systems that present a similar purpose. It was also proven possible to use interfaces with similar contents on mobile devices which dimensions are different, through splitting some of the interface’s components, making possible for the application to be shown on numerous devices whilst also presenting a high level of usability.

References