1 Introduction

The number and quality of smartphones on the market has dramatically raised lately. Researchers and developers are, thus, more and more pushed to bring algorithms and techniques from desktop environments to mobile platforms. One of the biggest constraints in mobile applications is the fine control of computing power and the relative power consumption. Although smartphones’ manufactures are offering better computing performance and longer battery life, the mobile architecture is not always powerful enough. Furthermore, nowadays, the touch-less interaction (e.g., the usage of voice commands) on mobile devices is particularly attractive. The device can also possibly answer to our questions (e.g., Siri-Speech Interpretation and Recognition Interface, which according to Apple is “the intelligent personal assistant that helps you get things done just by asking”). The use of talking avatars can improve the quality of the interaction and make it more useful and pleasant. Since avatars are static models, but the interaction requires dynamics, it is almost obliged to introduce avatars’ animations.

Animation in desktop environment [Orvalho et al. 2012] has been studied for decades. However transferring the same techniques to the mobile environment requires to reinterpret them due to the aforementioned constraints. This is especially true when we want to achieve real-time animations. The goal is more complicated when the animation of a 3D model are related to facial expressions. A real-time realistic animation has an heavy computational cost than pre-calculated animation. Due to this reason, in our work, we have analyzed a possible methodology to simulate the speech of a 3D-avatar in the most fluid and realistic possible way.

2 Our approach

We centred our approach on animating the eyes and the mouth of the avatar to simulate speech. Using a mid-res model (3000 faces and 1500 vertices), the animation would require the identification of some feature point to be used as anchors to realize the facial expression. The problems related to processing power, as thoroughly analyzed in [Boi et al. 2011], are still present, since we need to move approximately 300 vertices per frame, with an heavy computational effort. We started studying the usage of two poses for each phoneme and calculating the Δshift for each vertex obtaining a list of pairs [vertex, shift]. These data are pre-computed and loaded into the app to allow the reading when necessary. With this approach, our model was poorly performant, the code heavy and not very robust. We, thus, decided to use warping to simulate the human speech, working on the texture of the 3D object. This time, the quality of the results was low since we used a static geometry of the model. Our final solution is described in the rest of the section. We decided to split the model in 3D submeshes. We use one mesh for the inside of the mouth and another one for the outside: one for the hair; four meshes for the eyes and the eyelids; and the last one for the rest of the head (see Figure 1). After this we edited the submeshes to represent the movements frame by frame. We then generated three meshes for the eyes (look-right, look-center, look-left), sharing the same texture and with the same number of vertices and connectivity, but differing in the spatial coordinates of each vertex and the associated normals. To simulate the movement of the eyes we need to preload in memory the three meshes (about 50 points per mesh) and then work on the visibility of a single one of them; for labial animation we, instead, preload the meshes for each pronounce, ten for each phoneme (about 400 faces and 200 vertices for mesh). We use only three phonemes: a, f and c (see Figure 2). We show them randomly to produce a realistic and language-independent speech. The ten meshes for each phoneme, represent different degrees of openness, that is, nine different lip movements between neighbour meshes.

We developed our app on an Android 2.3 platform testing it on several medium-high level terminals and we got good results in terms of time: it takes five seconds to load the basic model and all the meshes used for the animation. See Figure 3 for an example of the final result. The results presented here can be improved by increasing the number of meshes in the set used for animation. Using some techniques like eye blinking and look around simulation the final effect is more pleasant and realistic.

![Figure 1: Subdivision of the original mesh, from left to right: eyes, mouth (internal and external), and the whole head.](image)

![Figure 2: A subset of the ‘A’ phoneme, frames 1, 5, and 10.](image)

![Figure 3: Screenshot taken from running App.](image)

Acknowledgements

The developed application uses Min3D Framework (based on OpenGL/ES 1.0) and the text-to-speech engine offered by the Android platform. To generate the face models we used Face-Gen.

References
